א טיפער איבערבליק אויף אידיש – די פולע, לענגערע אויסגאבע

YouTube 7 ווידעאס: AI, מאשין-לערנען, AI אגענטן, Pixel 9a און Veo 3.1

טעקע 3 פון 3: טייל 4, צווייטע העלפט – פון 26:32:13; און טייל 5, 6, 7
אלעס וואס ווערט געזאגט און געוויזן אין די ווידעאס, כמעט שריט ביי שריט
ארום 83 שעה ווידעא

אקטאבער 2026

אינהאלט

טייל 4 (המשך): דער גאנצער Agentic AI קורס 2027 – Simplilearn 6

לופ-אינזשענירינג: פון פראמפט צו א פראצעס 6

לופ-אינזשענירינג: פלאנירן, אויספירן און אפשאצן 11

לופ-אינזשענירינג און LangGraph אין דער פראקטיק 15

פון פיוטשער-פרופינג ביז Pydantic מיט LangGraph 20

פון Pydantic מיט LangGraph צו CrewAI 25

LangGraph און די אינסטאלאציע פון CrewAI 29

א CrewAI פראיעקט: סטרוקטור, לויפן און אגענט-דיזיין 32

CrewAI: טאסקס, crew.py און Flows 36

LangGraph: זכרון, דיבאגינג און דער ערשטער פראיעקט 39

לאגיק, געצייג, טעותים און דער אנהייב פון Autogen 43

ערשטער AutoGen אגענט מיט Ollama און מולטי-אגענט שמועסן 47

פון AutoGen דעפלוימענט ביז MCP 51

בויען א MCP-סערוואר מיט Firecrawl אין Cursor 55

MCP: פון ערשטן סערוואר ביז א וואלוטע-אגענט 59

א וועב-קראולער פראיעקט מיט Cursor AI און Firecrawl MCP 63

פון MCP-דעמא צו מולטי-אגענט סיסטעמען 66

קאמוניקאציע, ארכיטעקטורן און מוסטערן פון מולטי-אגענט סיסטעמען 71

בויען א מולטי-אגענט סיסטעם מיט CrewAI 76

CrewAI: באגריף, אינסטאלאציע און ערשטע אגענטן 80

CrewAI פראיעקט פארענדיקט און הקדמה צו Playwright 85

אינסטאלירן Playwright און אנהייבן א Todo פראיעקט 88

Playwright אויטאמאציע און RAG אפשאצונג 92

אפשאצן RAG סיסטעמען: מעטריקס, מכשירים און ארבעטס-גאנג 95

אפשאצן RAG, LLM בענטשמארקס און דאטא-עטיק 100

עטיק פון דאטא: פריוואטקייט, זיכערהייט און יושר 105

דאטא-עטיק: פרינציפן, געזעצן און באניצער-צוטרוי 111

דאטא-עטיק: פרינציפן, וויכטיגקייט און ביישפילן 114

עטישע פראבלעמען: פריוואטקייט און דורכזיכטיגקייט 117

עטישע דאטא-אנאליטיק: פרינציפן, שווערigkייטן און געזעצן 121

רעכטלעכע ראמען, אויסגאבעס און בעסטע פירונגען פאר דאטא-עטיק 124

דאטא-עטיק: געזעצן, קעיס-סטודיס און שוץ פון פערזענלעכע דאטא 127

דאטא-פריוואטקייט, עטיק און א פראקטישע איבונג 131

פון עטיק צו אגענטישן RAG און ראוטער 135

Router-מאטאר און Tool Calling אין אגענטישן RAG 139

אגענט-קאמפאנענטן, קאנטראל און רעזאנירונגס-לופ 143
א מולטי-דאקומענט אגענט און דער סוף פון RAG-טייל 147
א פארשונג-אגענט מיט N8N: יסודות און אנהייב 152
n8n אויטאמאציע: ריסוירטש-אגענטן און מיטינגען 155
פארקויפן אן n8n אגענט און א נייע לעקציע 159
AI-ארבעטס-פלוס אויטאמאציע: יסודות און מכשירים 162
מכשירים פאר אויטאמאציע און א Zapier ביישפיל 166
Zapier-אגענטן און דער יסוד פון AI-אגענטן 170
אגענטן מיט Azure OpenAI, LangChain און CSV דאטא 174
אגענט וואס רעדט מיט א SQL דאטא-באזע 178
דאטאבאַזע-אגענטן: SQL, function calling און Assistants API 183
א Zapier אגענט פאר סאשעל-מידיא 188
אויטאמאטישע פאסטן און א זעלבסט-לערנענדיקער אגענט 192
בויען דעם AI-אגענט אין n8n מיט Airtable 195
פארענדיקן דעם n8n אגענט און יסודות פון AI אגענטן 198
בויען אן AI קול-אגענט מיט Vapi 202
א שטים-אגענט אין פראקטיק, און די יסודות פון AI-אגענטן 205
צען AI-אגענט פראיעקטן און דער ערשטער בוי-שריט 210
אינסטאלירן די געצייג און ערשטע FastAPI אפליקאציע 215
FastAPI: Swagger, CRUD און Pydantic 218
FastAPI פארענדיקט, און Streamlit 222
Streamlit: פון אינסטאלירן ביז דאטא-אויספארשונג 226
א קליינע Streamlit דעשבאָרד און ארויסשטעלן אויף דער קלאוד 230
Docker: קאנטיינערס קעגןווירטועלע מאשינען 234
Docker: מעלות, אינסטאלאציע און ארכיטעקטור 237
דאקער: רעדזשיסטרי, קאנטעינערס, Compose און Swarm 241
דעמא: Docker אויף אן Ubuntu סערווער 244
א אייגענער Docker אימעדזש פאר Nginx מיט PHP 247
LLMOps: פאַרוואלטן LLM-אפליקאציעס פון אנהייב ביז סוף 251
LLMOps מעטריקס און די יסודות פון DevOps און CI/CD 256
Jenkins און א CI/CD פייפליין 260
Jenkins-פייפליין, MLOps און אגענט-אינטערוויו-פראגעס 262
אינטערוויו-פראגעס: יסודות פון אגענטישן AI 266
אינטערוויו-פראגעס: מולטי-אגענטן, זיכערהייט און אדוואנסירטע טעמעס 272
אינטערוויו-פראגעס און א וועג-קארטע פאר אגענטיק AI 277
וואס איז אגענטישע AI און דער וועג צו לערנען 282
א וועג-קארטע צו לערנען Agentic AI 287
דער סוף: פראבלעם, ניצן, צוקונפט און וועג-מאפע 291

טייל AI :5 אינזשעניר קורס 2026, לייוו – Simplilearn 295
וועגן דעם ווידעא 295
הקדמה צום קורס און ווי אזוי צו באהערשן AI געצייג 297
פראקטישע AI געצייג: פון ChatGPT ביז ElevenLabs 303
פון קרעאטיווע געצייג ביז Generative AI 309
געזעראטיווע AI אנווענדונגען און NLP יסודות 314
NLP-אנווענדונגען, AI-ווידעא און פראמפט-אינזשעניריע 318
פראמפט-סארטן און טשעטבאטס אן קאוד 322
א פלאן ווי צו ווערן אן AI אינזשעניר 326
פון ספאם-פילטערס ביז ליניארע רעגרעסיע 331
פון 2:00:00 ביזן סוף: וואס דער עקראן ווייזט 335
לינעארע רעגרעסיע און באשלוס-ביימער (לויטן עקראן) 336
פון Decision Tree צו SVM: מאפינס קעגן קאפקעיקס (לויטן עקראן) 341
SVM: מאפין אדער קאפקעיק – פונעם קאוד ביזן גראף (לויטן עקראן) 344
K-Means קלאסטערינג: טעאריע און א Python ביישפיל (לויטן עקראן) 347
K-Means קלאסטערינג אויף אויטא-דאטא (לויטן עקראן) 351
לאגיסטישע רעגרעסיע: טעאריע און א ערשטער נאוטבוק (לויטן עקראן) 354
לאגיסטישע רעגרעסיע אויף אַ שוואלסט-דאטא, און דער אנהייב פון) LLMלויטן עקראן) 357
פון גרויסע שפראך-מאדעלן צו אן MCP סערווער (לויטן עקראן) 360
Cursor מיט MCP, און RNN נויראל-נעצן (לויטן עקראן) 363
RNN: אנווענדונגען, גראדיענט-פראבלעמען און) LSTMלויטן עקראן) 367
LSTM: פון טויערן ביז אקציע-פרייזן (לויטן עקראן) 370
LSTM מאדעל פאר Google אקציע-פרייזן (לויטן עקראן) 374
פון אקציע-פרייזן צו Keras און נויראל-נעצן (לויטן עקראן) 376
א Keras-נויראל-נעץ פון אנהייב (לויטן עקראן) 379
טריינירן און פרובירן א Keras מאדעל (לויטן עקראן) 382
טריינירן א מאדעל צו דערקענען מאסקעס (לויטן עקראן) 386
טריינירן און לויפן א מאסקע-דעטעקטאר (לויטן עקראן) 389
פון מאסקע-דעטעקציע ביז) Multimodal AIלויטן עקראן) 393
בויען אן אפליקאציע מיט) Boltלויטן עקראן) 396
פון Bolt-אפליקאציע ביז) Reinforcement Learningלויטן עקראן) 399
א טיק-טאק-טא שפיל מיט) Reinforcement Learningלויטן עקראן) 403
טיק-טאק-טאו און Q-לערנען (לויטן עקראן) 406
א Q-Learning ביישפיל אין) Pythonלויטן עקראן) 409
נויראל-נעצן, פערסעפטראן און לאגישע טויערן (לויטן עקראן) 413
נויראל-נעצן: XOR, סארטן און ווי זיי ארבעטן (לויטן עקראן) 417
גראדיענט-אראפגאנג, TensorFlow און) MNISTלויטן עקראן) 420
MNIST-האנטשריפט מיט) TensorFlowלויטן עקראן) 423

פון MNIST-טריינירן ביז אביעקט-דערקענונג איןווידעא (לויטן עקראן) 426
אביעקט-דערקענונג אין ווידעא און) CNNלויטן עקראן) 429
די שיכטן פון א CNN און דער CIFAR-10 ביישפיל (לויטן עקראן) 433
CIFAR-10 בילדער: צוגרייטן די דאטא פאר א) CNNלויטן עקראן) 437
בויען דעם CNN מיט TensorFlow הילפס-פונקציעס (לויטן עקראן) 439
טריינירן א CNN און א AI-קלאן מיט) HeyGenלויטן עקראן) 442
HeyGen און AI-פראיעקטן פאר אנהייבער (לויטן עקראן) 446
AI-פעאיקייטן פאר פרילענסן און פרנסה (לויטן עקראן) 449
AI-שקילס און מאשין-לערנען אינטערוויו-פראגעס (לויטן עקראן) 452
ענדע פון די אינטערוויו-פראגעס (לויטן עקראן) 456
טייל 6: Google Pixel 9a – א לאנג-טערמין רעצענזיע – Arran Brown 459
וועגן דעם וויידעא 459
די Pixel 9a נאך לאנגע באניצונג – ערשטער טייל 461
באגן, באטעריע און פרייז פון Pixel 9a 465
טייל 7: ווי אזוי צו באנוצן Veo 3.1 אומזיסט – Hyrexverse 469
וועגן דעם וויידעא 469
דריי אומזיסטע AI וויידעא-געצייג, אריינגערעכנט Veo 470

טייל 4 (המשך): דער גאנצער Agentic AI קורס 2027 – Simplilearn

נאמען: Agentic AI Full Course 2027 | Complete Agentic AI Course For Beginners | Simplilearn

קאנאל: Simplilearn

דאטום: 13 אקטאבער 2026

לענג: 42:37:52

לינק: https://youtu.be/36T5jhJLM8A

לופ-אינזשענירינג: פון פראמפט צו א פראצעס (26:32:13)

אפשלוס פון דעם o1-חלק (26:32:13)

דער קורס שליסט אפ דעם חלק וועגן "reasoning" מיט o1. עס ווערט איבערגעחזרט וואס מען האט דורכגעגאנגען: וואס reasoning מיט o1 איז, פארוואס עס איז וויכטיג אין היינטיגע AI-וועלט, און וואו מען קען עס ניצן. מען האט אנגעהויבן מיט די יסודות, און דערנאך זיך פראקטיש באשעפטיגט: געשריבן גוטע פראמפטס, גענאיבט לאגישע און סטרוקטורירטע טראכטן, געשריבן און פארבעסערט קאוד מיט o1, און לעצטנס פארשטאנען ווי יעדע שיכט בויט זיך אויף צו א פולע AI-אינטעראקציע.

עס ווערט באטאנט אז מען האט נישט נאר געלערנט די טעאריע, נאר עס אליין דורכגעלעבט דורך דעמאס, אויפגאבעס און ביישפילן פון דער אמת'ער וועלט. סיי ביים אויפלעזן לאגיק, סיי ביים פירן AI דורך סטרוקטורירטע טראכטן, סיי ביים פאררעכטן דעם אייגענעם פראצעס, האט מען יעצט א שטארקן אנכאפ ווי צו טראכטן "מיט" AI, נישט נאר "וועגן" AI. דער רוף איז: ווייטער עקספערימענטירן, ווייטער פרעגן.

פארוואס לופ-אינזשענירינג איז היינט וויכטיג (26:32:56)

דא הייבט זיך אן א נייע טעמע: "loop engineering". דער קורס עפנט מיט א ביישפיל: מען ארבעט מיט אן AI-קאוד-געצייג און מען זאגט עס: "בוי א login-בלאט פאר דער אפליקאציע". דאס געצייג שרייבט קאוד, אבער דער בלאט ארבעט נישט ריכטיג. מען באמערקט א טעות, מען ערקלערט דאס פראבלעם, מען בעט די AI עס צו פאררעכטן, מען לאזט לויפן די אפליקאציע נאכאמאל, מען טרעפט נאך א פראבלעם, און מען גייט צוריק צו דער AI. די AI העלפט, אבער דער מענטש פירט נאך אלץ דעם גאנצן פראצעס.

יעצט שטעלט מען זיך פאר א צווייטן צוגאנג: מען לייגט אוועק דעם ציל, און די סיסטעם ארבעט אליין דורך די אויפגאבע – זי קוקט איבער וואס זי האט געשאפן, געפינט וואס איז נישט ריכטיג, מאכט ענדערונגען, טעסט נאכאמאל, און גייט אזוי ווייטער ביז זי דערגרייכט א באשטימטן רעזולטאט. דער איבערגאנג פון שטענדיג נאכאמאל געבן פראמפטס פאר א AI, צו פלאנירן דעם פראצעס ארום איר – דאס הייסט לופ-אינזשענירינג.

די דעפיניציע: לופ-אינזשענירינג איז א נייע פראקטיק, וואו פראגראמירער פלאנירן "workflows" וואס לאזן AI-סיסטעמען האנדלען, באמערקן די רעזולטאטן, אפשאצן זייער ארבעט, און ווייטער איבערחזרן צום ציל

וואס דער באניצער האט באשטימט – מיט ווייניגער האנט-איינמישונג. IBM באשרייבט לופ-אינזשענירינג ווי די נייע פראקטיק הינטער AI-קאוד-סיסטעמען ווי Claude Code און OpenAI Codex.

פארוואס איז דאס באזונדערס שייך אין 2026? ווייל דער שמועס ארום AI גייט שוין ווייטער פון פשוט שאפן אינהאלט. פראגראמירער ארבעטן מער און מער מיט קאוד-אגענטן און אנדערע סיסטעמען וואס קענען טון עטליכע אקציעס, אנשטאט ענטפערן אויף איין באפעל אויפאמאל. למשל, GitHub-דאקומענטאציע באשרייבט אן "agent loop", וואו אן "orchestrator" קען מאכן עטליכע רופן צום מאדעל און ווייטער ניצן געצייג ביז די אויפגאבע ווערט פארטיג.

דערפאר ענדערט זיך די וויכטיגע פראגע: אנשטאט פרעגן "ווי שרייב איך א בעסערן פראמפט?", פרעגט מען יעצט "ווי פלאניר איך א סיסטעם וואס זאל ווייטער ארבעטן צום רעזולטאט וואס איך וויל טאקע?" – און דאס איז דאס פראבלעם וואס לופ-אינזשענירינג פרובירט צו לייזן.

וואס מיינט לופ-אינזשענירינג געניו (26:34:46)

פשוט געזאגט, לופ-אינזשענירינג מיינט פלאנירן אן איבערחזרנדיגן ארבעטס-פראצעס ארום א AI-סיסטעם, אז זי זאל קענען טון אן אקציע, נאכקוקן די רעזולטאטן, מאכן א באשלוס לויט וואס זי געפינט, און ווייטער ארבעטן ביז זי דערגרייכט א באשטימטן ציל.

דאס וויכטיגע ווארט איז "loop". דער קורס ברענגט א משל: ווען מען לייזט א שווער פראבלעם, ווייסט מען נישט שטענדיג דעם פערפעקטן ענטפער פון פריער. מען פרובירט ערשט, מען קוקט וואס איז געשען, מען געפינט וואס איז שיף געגאנגען, מען ענדערט דעם צוגאנג, און מען פרובירט נאכאמאל. ווייכווארג-אנטוויקלונג ארבעט פונקט אזוי: א פראגראמירער שרייבט קאוד, לאזט לויפן טעסטן, זעט א דורכפאל, ענדערט דעם קאוד, לאזט לויפן די טעסטן נאכאמאל, ביז די ווייכווארג פירט זיך אויף ווי געהעריג.

לופ-אינזשענירינג ברענגט דעם איבערחזרנדיגן פראצעס אריין אין א AI-workflow . אנשטאט ערווארטן אז א AI-סיסטעם זאל שאפן א פערפעקטן ענטפער מיטן ערשטן פרואוו, איז די סיסטעם געבויט צו ארבעטן דורך עטליכע ראונדן: זי הייבט אן מיט א ציל, באשליסט וואס מען דארף טון, טוט אן אקציע, קוקט איבער דעם רעזולטאט, און באשטימט וואס דארף קומען דערנאך.

IBM דעפינירט לופ-אינזשענירינג אלס די פראקטיק פון פלאנירן אגענט-workflows וואס פירן AI-סיסטעמען שריט ביי שריט צו ציל, וואס דער באניצער באשטימט, מיט מינימאלער מענטשליכער איינמישונג. דער קורס ווארנט: דאס מיינט נישט אז די AI ווערט ווי א כישוף פערפעקט, אדער אז מענטשן ווערן אראפגענומען פונעם פראצעס. עס מיינט אז דער workflow איז אזוי געבויט, אז די נוצליכע קאנטראלן און באשלוסן געשען צווישן איין פרואוו און דעם צווייטן.

א ביישפיל: אויב מען בעט א AI-קאוד-סיסטעם צו פאררעכטן א "bug" אין ווייכווארג, איז דער נוצליכער פראצעס נישט פשוט שאפן נייעם קאוד. די סיסטעם דארף אפשר:

- איבערקוקן די שייכותדיגע פיילס
- ענדערן דעם קאוד
- לאזן לויפן די טעסטן
- לייענען די טעסט-רעזולטאטן
- באשטימען צי די ענדערונג האט געארבעט

• מאכן נאך אן ענדערונג אויב נויטיג

דער לופ איז דאס וואס פארבינדט אלע די אקציעס צוזאמען. די פשוטסטע וועג עס צו פארשטיין: אנשטאט נאר צו באהאנדלען די AI אליין, פלאנירט מען דעם גאנצן פראצעס ארום דער AI.

פארוואס דער אלטער שטייגער ארבעטן מיט AI ענדערט זיך (26:36:24)

לאנגע צייט האט ניצן AI געמיינט פירן א שמועס: מען שרייבט א באפעל, באקומט אן ענטפער, קוקט עס איבער, און באשליסט וואס צו פרעגן ווייטער. דער צוגאנג איז זייער נוצליך פאר אסאך טעגליכע אויפגאבעס – אויב מען וויל צוזאמענפאסן אן ארטיקל, איבערשרייבן אן email, ערקלערן א פראגראמירונג-באגריף, אדער שאפן ערשטע אידייען, איז א פשוטע אינטעראקציע אפשר גענוג.

דאס פראבלעם קומט וועז די אויפגאבע ווערט לענגער און פארלאנגט עטליכע פארבונדענע אקציעס. דער ביישפיל: מען בעט א AI-סיסטעם אויספארשן א וועכווארג-פראבלעם אין א גרויסער אפליקאציע. די אויפגאבע קען פארלאנגען: פארשטיין די סטרוקטור פונעם פראיעקט, געפינען די שייכותדיגע פיילס, לייענען דעם עקזיסטירנדיגן קאוד, איבערמאכן דאס פראבלעם, געפינען די סיבה, ענדערן דעם קאוד, לאזן לויפן טעסטן, נאכקוקן צי די רעזולטאטן זענען ריכטיג, און אפשר איבערחזרן דעם פראצעס עטליכע מאל.

אויב א מענטש וואלט געגעבן א באפעל פאר יעדע איינע פון די אקציעס, איז די AI נאך אלץ נאר טיילווייז אויטאמאטיזירט – דער מענטש ווערט דער וואס פארבינדט יעדן שריט. לופ-אינזשענירינג ענדערט דאס דורך אריבערטראגן א טייל פון דער קאארדינאציע אין דער סיסטעם אליין. מען גיט די AI א ציל און א סעט כללים ווי צו ארבעטן דערצו, און די סיסטעם באשליסט וועלכע אקציע צו טון, באמערקט וואס איז געשען, און באשטימט וואס דארף קומען ווייטער.

דאס איז ענג פארבונדן מיט ווי מאדערנע קאוד-אגענטן ארבעטן. IBM באמערקט אז מאדערנע קאוד-געהילפן קענען אליין אויסארבעטן שריט, לאזן לויפן באפעלן, אפשאצן רעזולטאטן, און איבערחזרן איידער זיי שטעלן פאר דעם לעצטן רעזולטאט.

אבער דער קורס באטאנט: דאס מיינט נישט אז יעדע AI-סיסטעם זאל ארבעטן אן מענטשליכער אויפזיכט. אדרבה, מענטשליכע איבערקוקונג קען זיין זייער וויכטיג וועז די אויפגאבע האט קאנסעקווענצן פון זיכערהייט, גלעלט, געזעץ אדער "production". די וויכטיגע ענדערונג איז אז מענטשן דארפן נישט דווקא פירן מיט די הענט יעדן קליינעם שריט. די סיסטעם קען באהאנדלען דעם איבערחזרנדיגן ציקל, אבער דער מענטש באשטימט דעם ציל, שטעלט גרענעצן, קוקט איבער וויכטיגע באשלוסן, און נעמט קאנטראל וועז נויטיג.

פון פראמפט-אינזשענירינג צו לופ-אינזשענירינג (26:38:22)

כדי גוט צו פארשטיין לופ-אינזשענירינג, קוקט דער קורס אויף דער אנטוויקלונג פון AI:

• פראמפט-אינזשענירינג: עס איז געווארן וויכטיג וועז מענטשן האבן געכאפט אז די וועג ווי מען רעדט מיט שפראך-מאדעלן קען שטארק באאיינפלוסן די קוואליטעט פון זייערע ענטפערס. אנשטאט פשוט בעטן "שרייב קאוד", קען א פראגראמירער ערקלערן די פראגראמירונג-שפראך, די פאדערונגען, באגרעניצונגען, דעם ערווארטעטן רעזולטאט און ביישפילן. אבער די אינטעראקציע איז נאך אלץ געווען צענטרירט ארום איינצלנע באפעלן.

• RAG (retrieval augmented generation): דער נעקסטער גרויסער שריט איז געווען געבן AI צוטריט צו נאך אינפארמאציע. RAG האט דערלויבט סיסטעמען אראפצוברענגען שייכותדיגע אינפארמאציע פון

דרויסנדיגע מקורות איידער זיי שאפן אן ענטפער – באזונדערס נוצליך ווען AI דארף צוטריט צו פירמע-דאקומענטן, וויסנס-באזעס אדער אנדערע וויכטיגע אינפארמאציע.

- AI-אגענטן: סיסטעמען וואס קענען מער ווי נאר שאפן טעקסט – ניצן געצייג, צוקומען צו אינפארמאציע, ארבעטן מיט ווייכווארג, און טון עטליכע אקציעס.

דאס האט געשאפן א נייע פראבלעם: אז AI קען טון עטליכע אקציעס, ווי באשליסט מען וואס זי זאל טון וויטער? ווי וויסט מען צי אן אקציע איז געלונגען? ווי האלט מען זי אפ פון וויטער ארבעטן ווען זי האט שוין דערגרייכט דעם ציל? און ווי ראטעווט מען זיך ווען עפעס גייט שיף? לופ-אינזשענירינג קאנצענטרירט זיך אויף די פראגעס, און אנשטאט קוקן נאר אויף דעם איינצלנעם פראמפט וואס מען שיקט דעם מאדעל, קוקט זי אויף דעם גאנצן workflow ארום דעם מאדעל.

IBM מאכט אן ענליכן אונטערשייד: פראמפט-אינזשענירינג קאנצענטרירט זיך אויף שאפן אן ווירקזאמען באפעל, בשעת לופ-אינזשענירינג פלאנירט אן אויטאמאטישע סיסטעם וואס קען געבן פראמפטס, נאכקוקן, און פירן דעם אגענט איבער און איבער ביז דער באשטימטער ציל ווערט דערגרייכט.

מען קען טראכטן פון דער אנטוויקלונג ווי א באווענונג: פון בעסערע באפעלן, צו בעסערן צוטריט צו אינפארמאציע, צו בעסערער פעאיגקייט צו טון אקציעס, און יעצט צו בעסערע סיסטעמען פאר פירן איבערחזרנדיגע אקציעס און "feedback". דערפאר איז לופ-אינזשענירינג נישט פשוט אן ערזאץ פאר פראמפט-אינזשענירינג – פראמפטס זענען נאך אלץ א טייל פונעם פראצעס. דער חילוק איז אז פראגראמירער טראכטן יעצט ווען דעם וואס געשעט נאך דעם ערשטן ענטפער.

וואס געשעט אינעווייניג אין א לופ (26:40:22)

א לופ-אינזשענירינג workflow הייבט זיך אן מיט א ציל, אבער דאס וויכטיגע איז וואס געשעט נאך דער ערשטער אקציע. דער ביישפיל: א פראגראמירער גיט א AI-קאוד-סיסטעם די אויפגאבע: "פארריכט די טעסטן וואס פאלן דורך אין דעם פראיעקט". די סיסטעם דארף ערשט פארשטיין וואס דער ציל פארלאנגט. זי קען איבערקוקן דעם פראיעקט, געפינען די טעסטן וואס פאלן דורך, און באשטימען וועלכע טיילן פונעם קאוד זענען פארבונדן מיט די דורכפאלן. דערנאך טוט זי אקציעס – למשל, ענדערט א שטיקל קאוד.

אבער דער לופ האלט דא נישט אויף. די סיסטעם דארף באווייזן צי די ענדערונג האט טאקע געארבעט. זי קען לאזן לויפן די טעסטן און איבערקוקן די רעזולטאטן. אויב די טעסטן גייען דורך, האט די סיסטעם באווייז און די אויפגאבע איז פארטיג. אויב די טעסטן פאלן דורך, ווערט דער רעזולטאט א "feedback" וואס זאגט דער סיסטעם אז מען דארף נאך א ראונד. די סיסטעם קען דאן איבערקוקן דעם דורכפאל, מאכן נאך אן ענדערונג, און טעסטן נאכאמאל.

דאס איז דער גרונט-געדאנק פון אן אגענט-לופ: האנדלען, באמערקן, באשליסן, און ווייטער גיין. GitHub באשרייבט איר Copilot agent loop ווי אן orchestrator וואס מאכט איינס אדער מער רופן צום מאדעל און פירט ווייטער דעם פראצעס פון ניצן געצייג ביז די אויפגאבע איז פארטיג.

דער וויכטיגער פונקט: דער לופ דארף האבן א וועג צו באשטימען וואס איז טאקע געשען. אן דעם קען די סיסטעם נישט פארלעסלעך באשליסן וואס צו טון ווייטער. דער קורס ברענגט א משל: שטעלט אייך פאר מען בעט עמיצן ווייטער פארריכטן א מאשין, אבער מען לאזט אים קיינמאל נישט טעסטן צי די מאשין ארבעט. ער קען מאכן הונדערטער ענדערונגען אן וויסן צי ער קומט נענטער צו דער לייזונג אדער נישט. דאס זעלבע פראבלעם עקזיסטירט ביי AI. א נוצליכער לופ דארף האבן אקציעס און באמערקונגען, אבער אויך א קלארע

דעפיניציע פון הצלחה. דערפאר איז לופ-אינזשענירינג נישט פשוט "לאז די AI ווייטער פרובירן", נאר פלאנירן א קאנטראל-פראצעס וואו יעדער ראונד גיט נוצליכע אינפארמאציע פאר דעם נעקסטן.

די הויפט-באשטאנדטיילן פון א לופ (26:42:12)

א גוט-געפלאנטער לופ דארף מער ווי נאר א AI-מאדעל. ער דארף א קלארן ציל, צוטריט צו דער ריכטיגער אינפארמאציע, די פעאיגקייט צו טון אקציעס, א וועג צו באמערקן וואס איז געשען, און כללים פאר באשליסן וואס קומט ווייטער:

- ציל: די סיסטעם דארף וויסן וואס זי פרובירט צו דערגרייכן, און ווי א געלונגענער סוף זעט אויס. אויב דער ציל איז אומקלאר, קען דער לופ ווייטער גיין אן א קלארן ציל-פונקט.
- קאנטעקסט: די סיסטעם דארף שייכותדיגע אינפארמאציע כדי צו מאכן נוצליכע באשלוסן. אין א קאוד-workflow קען דאס אריינרעכענען די פראיעקט-פיילס, דאקומענטאציע, קאנפיגוראציע, פריערדיגע ענדערונגען, און טעסט-רעזולטאטן.
- אקציע: די סיסטעם דארף צוטריט צו געצייג וואס לאזן איר טאקע טון עפעס. אין א קאוד-סביבה קענען די געצייג זיין א terminal, א פייל-סיסטעם, version control, א טעסט-framework, אדער אנדערע אנטוויקלונגס-געצייג.
- באמערקונג (observation): נאך אן אקציע דארף די סיסטעם א וועג צו פארשטיין וואס איז געשען – א טעסט-רעזולטאט, א טעות-מעלדונג, א זוך-רעזולטאט, א מעסטונג פון פערפארמענס, אדער א feedback פון א באניצער.
- באשלוס-מעכאניזם: לויט דער באמערקונג – זאל זי ווייטער מאכן נאך אן ענדערונג, פרעגן א מענטש, אדער אפשטעלן?

די באשטאנדטיילן מאכן פון א פשוטער AI-אינטעראקציע אן ארבעטנדיגן לופ. דאס ערקלערט אויך פארוואס פשוט פארבינדן א שפראך-מאדעל מיט געצייג שאפט נישט אויטאמאטיש א פארלעסלעכע לופ-אינזשענירינג סיסטעם – דער workflow ארום איז וויכטיג. IBM ערקלערט אז לופ-אינזשענירינג באשטייט פון פלאנירן סיסטעמען וואס געבן פראמפטס, קוקן נאך, און פירן AI-אגענטן, אנשטאט פשוט געבן דעם מאדעל באפעלן איין מאל נאכן אנדערן.

וועו די באשטאנדטיילן זענען פארבונדן, קענען די סיסטעמען גיין דורך איבערחזרנדיגע ציקלען. אבער עס איז דא איין טייל פון לופ וואס פארדינט באזונדערע אויפמערקזאמקייט, ווייל ער באשטימט צי די סיסטעם מאכט טאקע פארשריט – דאס איז די אפשאצונג (evaluation).

פלאנירונג און אויספירונג (26:43:57)

דער קורס גייט איבער צו פלאנירונג און אויספירונג. איידער א AI-סיסטעם קען פארענדיגן א קאמפליצירטע אויפגאבע, דארף זי באשטימען ווי מען קען צוגיין צו דער אויפגאבע – און דארט ווערט פלאנירונג אזוי וויכטיג. דער משל: עמיצער זאגט אייך "בוי אן אנליין געשעפט". דער באפעל באשרייבט דעם לעצטן ציל, אבער ער זאגט נישט יעדע אקציע וואס מען דארף טון כדי אהין צו קומען. מען וואלט ערשט געדארפט פארשטיין די פאדערונגען, באשטימען וואס מען דארף בויען, און באשליסן וואס קומט צו ערשט.

לופ-אינזשענירינג: פלאנירן, אויספירן און אפשאצן (26:44:27)

פלאנירן און אויספירן אין א לופ (26:44:27)

א AI-סיסטעם וואס ארבעט אינעווייניג אין א לופ שטייט פאר דער זעלבער שווערקייט ווי א מענטש וואס דארף צעטיילן א גרויסע ארבעט אין קלענערע טאסקס און דערנאך זיי דורכארבעטן. דער סיסטעם דארף צעטיילן א גרעסערן ציל אין קלענערע שריט, און באשליסן וועלכער שריט קומט ווייטער.

דער קורס ברענגט א ביישפיל פון א סאפטוועיר-פראיעקט: דער סיסטעם קען צום ערשט אדורכקוקן דעם repository, דערנאך אויסגעפינען די פיילס וואס זענען שייך צו דער ארבעט, מאכן אן ענדערונג, און לויפן די פאסיגע טעסטס.

עס ווערט ערקלערט אז פלאנירן מיינט נישט דווקא צו שאפן איין פערפעקטן פלאן פונעם אנהייב און דערנאך אים נאכגיין אן קיין שום ענדערונג. אין א נוצליכן לופ קען דער פלאן זיך ענדערן ווען עס קומט אויף נייע אינפארמאציע. אויב א טעסט ברענגט ארויס אן אומגעריכטן טעות, קען דער סיסטעם דארפן איבערטראכטן דעם ערשטן צוגאנג. דאס איז א וויכטיגער אונטערשייד, ווייל אין דער פראקטישער וועלט גייען טאסקס זעלטן פונקט ווי מען האט גערעכנט.

דער שריט פון אויספירן (execution) פארוואנדלט דעם פלאן אין ווירקליכע ארבעט. דער AI קען ניצן געצייג (tools) צו לייענען פיילס, שרייבן קאוד, לויפן קאמאנדעס, זוכן אינפארמאציע, אדער ארבעטן מיט אן אנדער סיסטעם. מאדערנע קאודינג-געהילפן זענען געבויט אויף דער קאמבינאציע פון א AI-מאדעל מיט געצייג וואס לאזן אים ארבעטן אינעווייניג אין זיין סביבה. IBM באשרייבט די געצייג אלס הכרחדיג, ווייל א מאדעל אליין קען נישט עפענען פיילס און ארבעטן אין א טערמינאל.

די קאמבינאציע פון פלאנירן און אויספירן לאזט דעם סיסטעם אריבערגיין פון א געדאנק צו א ווירקליכער אקציע. אבער אויספירן אן אקציע איז נאך נישט דאס זעלבע ווי פארענדיגן א טאסק: דער סיסטעם דארף נאך וויסן צי דאס וואס ער האט געטון איז ריכטיג. דאס פירט צום וויכטיגסטן טייל פונעם לופ: אפשאצונג און פידבעק.

אפשאצונג און פידבעק (26:45:27)

אפשאצונג (evaluation) איז דאס וואס זאגט א AI-סיסטעם צי זיין לעצטע אקציע איז געווען נוצליך אדער נישט. אן אפשאצונג איז א לופ בלויז א רייע אקציעס, איינע נאך די אנדערע. מיט אפשאצונג קען דער סיסטעם לערנען פונעם רעזולטאט פון יעדער אקציע און באשליסן וואס צו טון ווייטער.

דער קורס ברענגט א פשוט'ן קאודינג-ביישפיל: מען בעט א AI-סיסטעם צו פאררעכטן א באג. ער ענדערט דעם קאוד און לויפט דערנאך די טעסטס וואס זענען שוין פאראן.

- אויב אלע שייכותדיגע טעסטס גייען דורך, איז דאס א באווייז אז די ענדערונג האט מעגליך געלייזט דעם פראבלעם.
- אויב די טעסטס פאלן דורך, גיט דער דורכפאל אינפארמאציע וואס קען אנווייזן דעם קומענדיגן פרואוו. דער סיסטעם קען אונטערזוכן דעם טעות, פארשטיין וואס האט זיך געענדערט, און פרובירן אן אנדער לייזונג.

דאס איז א סך נוצליכער ווי בעטן דעם AI נאכאמאל און נאכאמאל צו שאפן קאוד אן קאנטראלירן צי דער קאוד ארבעט בכלל.

אפשאצונג קען האבן פארשידענע פארמען, לויט דעם טאסק:

- ביי סאפטוועיר-אנטוויקלונג קען זי זיין אויטאמאטיש: compiler-טעותים, static analysis, אדער קאנטראלן פון פערפארמענס.
- ביי א שטיצע-סיסטעם פאר קאסטומערס (customer support) קען זי מיינען צו קאנטראלירן צי דעם קאסטומער'ס פראבלעם איז געלייזט געווארן, אדער צי מען דארף אריבערגעבן דעם שמועס צו א העכערע מדרגה (escalation).
- ביי דאטא-אנאליז קען זי מיינען צו באשטעטיגן חשבונות, אדער פארגלייכן רעזולטאטן מיט באקאנטע באדינגונגען.

דער עיקר איז אז אפשאצונג דארף ארויסגעבן באווייזן. דער סיסטעם דארף עפעס קאנקרעטס וואס זאל אים העלפן באשליסן צי ער זאל ווייטער גיין אדער נישט.

דער מענטש אינעווייניג אין לופ (26:46:59)

דא קען אויך א מענטשליכע איבערקוק ווערן א טייל פונעם לופ. אויב דער סיסטעם קומט צו א באשלוס וואס פארלאנגט אורטייל, קען ער זיך אפשטעלן און בעטן א מענטשליכע באשטעטיגונג אנשטאט ווייטער גיין אויטאמאטיש. IBM באשרייבט "human in the loop" סיסטעמען אלס וואָרקפלאוס וואו מענטשן נעמען אנטייל אין דער ארבעט, אין דער השגחה אדער אין באשליסן, כדי צו פארבעסערן די געוויקייט, די זיכערהייט, די אחריות און עטישע באשלוסן.

דעריבער מיינט אפשאצונג נישט שטענדיג אז דער AI משפט זיך אליין. זי קען אויך זיין אויטאמאטישע קאנטראלן, אנדערע אויסערליכע באווייזן, אדער א מענטשליכער באקוקער. ווען דער סיסטעם באקומט דעם פידבעק, קען ער באשליסן צי דער טאסק איז פארענדיגט אדער צי מען דארף נאך א רונדע (iteration).

פארוואס אפשאצונג איז דער צענטער (26:47:29)

דער קורס שטעלט א וויכטיגע פראגע: פארוואס איז אפשאצונג אזוי צענטראל צום גאנצן געדאנק פון לופ-אינזשענירינג? דאס ווארט "לופ" קען מאכן דעם אײַנדרוק אז דער AI חזר'ט בלויז איבער דעם זעלבן טאסק נאכאמאל און נאכאמאל, אבער איבערחזרן אליין ברענגט נישט קיין פארבעסערונג.

דער משל: שטעלט אייך פאר אז מען בעט א AI-סיסטעם צו שרייבן דעם זעלבן שטיקל קאוד 10 מאל, אן קאנטראלירן קיין אייניגע ווערסיע. מען וועט האבן 10 פרואוון, אבער נישט דווקא א בעסערע לייזונג. דער אונטערשייד צווישן בלויז איבערחזרן און א נוצליכער רונדע איז די אפשאצונג: זי גיט דעם סיסטעם אינפארמאציע וואס איז געשען, און צי דער לעצטער פרואוו האט אים דערנענטערט צום ציל.

דעריבער דארף א גוט-אויסגעטראכטער לופ האבן א קלארע דעפיניציע פון הצלחה:

- ביים בויען א וועבזייטל קען הצלחה מיינען אז די נויטיגע זייטלעך ארבעטן, וויכטיגע טעסטס גייען דורך, און די אפליקאציע ערפילט באשטימטע פאדערונגען.
- ביים אנאליזירן דאטא קען הצלחה מיינען אז די חשבונות זענען ריכטיג, און די מסקנות זענען געשטיצט פון דער דאטא.
- ביים באהאנדלען קאסטומער-פארלאנגען קען הצלחה מיינען אז דער פראבלעם איז געלייזט, אדער אויף א פאסיגן אופן אריבערגעגעבן ווייטער.

וואס קלארער די קריטעריעס פון הצלחה, אלס גרינגער ווערט עס אריינצובויען נוצליכע קאנטראלן אינעם לופ. דאס איז באזונדערס וויכטיג ווייל AI-סיסטעמען קענען ארויסגעבן רעזולטאטן וואס זענען אויס איבערצייגנד אבער אנטהאלטן טעותים. א לופ וואס פרעגט בלויז דעם זעלבן AI-סיסטעם "איז דיין ענטפער ריכטיג?" קען נישט זיין גענוג. א בעסערע אפשאצונג קען אריינענעמען אומאפהענגיגע טעסטס, כללים, מעסטונגען אדער א מענטשליכע איבערקוק. ביי סאפטוועיר-אנטוויקלונג, למשל, קענען אויטאמאטישע טעסטס געבן שטארקערע סיגנאלן ווי בלויז פרעגן דעם מאדעל צי זיין קאוד זעט אויס ריכטיג.

IBM'ס אנווייזונגען וועגן קאודינג מיט אגענטן באטאנען: באשטימען קלארע קריטעריעס פון הצלחה, צושטעלן דעם ריכטיגן קאנטעקסט, און פירן א רעקארד (logging) פון דעם אגענט'ס באשלוסן און tool-אויפרופן, כדי מען זאל קענען פארשטיין דעם וואָרקפלאו און אים דיבאגן. דער אמת'ער כח פון לופ-אינזשענירינג איז נישט וואס דער AI קען פרובירן נאכאמאל, נאר וואס דער סיסטעם קען ניצן באווייזן פון איין פרואוו כדי דער קומענדיגער פרואוו זאל זיין מער אינפארמירט.

לופ-אינזשענירינג קעגן פראמפט-אינזשענירינג (26:49:31)

דא קומט א פארגלייך, ווייל אסאך מענטשן פארמישן די צוויי באגריפן. זיי זענען ענג פארבונדן, אבער זיי באהאנדלען פארשידענע שטאפלען פון א AI-וואָרקפלאו.

פראמפט-אינזשענירינג באשעפטיגט זיך בעיקר מיט ווי מען גיט איבער א טאסק צו א AI-מאדעל: די אנווייזונגען, דער קאנטעקסט, ביישפילן, באגרעניצונגען און דער געוואונטשענער רעזולטאט, אויסגעשטעלט אזוי אז דער מאדעל זאל קענען געבן א נוצליכן ענטפער. למשל, ווען מען בעט א AI -מאדעל צו שרייבן א Python-פונקציע, ערקלערט מען וואס די פונקציע זאל טון, וועלכע inputs זי זאל אננעמען, וואס זי זאל צוריקגעבן, און וועלכן פראגראמירונג-סטיל מען ערווארט. דאס איז פראמפט-אינזשענירינג.

לופ-אינזשענירינג פרעגט א סך א גרעסערע פראגע: וואס געשעט נאכדעם וואס דער AI שאפט דעם קאוד? היט דער סיסטעם אפ דעם קאוד? לויפט ער די טעסטס? וואס געשעט אויב א טעסט פאלט דורך? ווער אדער וואס זאל אפשאצן דעם רעזולטאט? זאל דער סיסטעם פרובירן נאכאמאל? ווען זאל ער אויפהערן? ווען זאל ער בעטן הילף ביי א מענטש?

פראמפט-אינזשענירינג קאנצענטרירט זיך אויף פארבעסערן די אנווייזונג, לופ-אינזשענירינג אויף אויסטראכטן דעם פראצעס וואס רינגלט ארום די אנווייזונג. IBM מאכט דעם אונטערשייד בפירוש: פראמפט-אינזשענירינג שאפט אן עפעקטיוו אנווייזונג, בשעת לופ-אינזשענירינג פלאנירט אויטאמאטישע סיסטעמען וואס קענען שיקן פראמפטס, קאנטראלירן, און פירן אגענטן דורך איבערגעחזר'טע רונדעס.

דאס מיינט אז פראמפט-אינזשענירינג פארשווינדט נישט. פראמפטס קענען בלייבן א טייל פונעם לופ, און א גוט-אויסגעטראכטער לופ קען אנטהאלטן עטליכע פראמפטס, יעדער פאר אן אנדער שטאפל פונעם פראצעס. דער אונטערשייד איז אז דער פראגראמירער שרייבט שוין נישט מיט דער האנט יעדן פראמפט ביי יעדן שריט; דער וואָרקפלאו באשטימט וועלכע אינפארמאציע מען גיט דעם AI, און ווען מען דארף נאך א רונדע. בקיצור: פראמפט-אינזשענירינג איז צו באקומען א בעסערן ענטפער פונעם AI; לופ-אינזשענירינג איז צו בויען א פראצעס וואס פארענדיגט די ארבעט.

לופ-אינזשענירינג און AI-אגענטן (26:51:02)

AI-אגענטן און לופ-אינזשענירינג זענען פארבונדן, אבער נישט דאס זעלבע. א AI-אגענט איז א סיסטעם וואס קען אויספירן טאסקס אין נאמען פונעם באניצער אדער פון אן אנדער סיסטעם: ער קען ניצן געצייג, מאכן

באשלוסן, און דורכפירן עטליכע אקציעס. לופ-אינזשענירינג קאנצענטרירט זיך אויף ווי מען פלאנירט דעם איבערחזרנדיגן וואָרקפלאו וואס פירט אוא אגענט צום ציל.

א נוצליכער משל: דער אגענט איז דער ארבעטער, און דער לופ איז דער ארבעטס-פראצעס ארום אים. דער אגענט קען שרייבן קאוד, זוכן אינפארמאציע אדער ניצן געצייג, אבער דער לופ באשטימט וואס געשעט פאר און נאך די אקציעס.

א ביישפיל: א AI-קאודינג-אגענט באקומט א טאסק צו פאררעכטן א פראבלעם וואס פאלט דורך. ער קען אדורכקוקן דעם repository, ענדערן פיילס און לויפן קאמאנדעס. אבער לופ-אינזשענירינג פרעגט נאך פראגעס: וואס זאל דער אגענט קאנטראלירן פארן מאכן ענדערונגען? וועלכע טעסטס זאל ער לויפן נאכדעם? וועלכע באווייזן באשטימען צי דער טאסק איז פארענדיגט? וויפיל פרואוון זאל ער מאכן? וואס געשעט ווען ער בלייבט שטעקן? זאל ער בעטן א מענטש'ס באשטעטיגונג? דאס זענען פראגעס וועגן דעם וואָרקפלאו, נישט וועגן דעם מאדעל אליין. IBM באשרייבט לופ-אינזשענירינג אלס א וואקסנדיגע פראקטיק וואס שטייט אונטער AI-קאודינג-אגענטן ווי Claude Code און OpenAI Codex.

דער אונטערשייד ווייזט פארוואס עס איז נישט גענוג בלויז צו האבן א AI-אגענט. אן פאסיגע קאנטראלן און גרעניצן קען ער מאכן אומנויטיגע ענדערונגען, איבערחזרן טעותים, און ווייטער ארבעטן נאכדעם וואס די נוצליכע ארבעט איז שוין געענדיגט. לופ-אינזשענירינג גיט די באדינגונגען ווען דער אגענט זאל אגירן, אפשאצן, ווייטער גיין, אויפהערן, אדער אריינרופן א מענטש. AI-אגענטן געבן סיסטעמען די פעאיקייט אויסצופירן טאסקס; לופ-אינזשענירינג פלאנירט דעם פראצעס וואס העלפט זיי עס טון אויף א פארלעסליכן אופן.

א ביישפיל פון AI-קאודינג (26:53:06)

סאפטווער-אנטוויקלונג איז איינע פון די קלארסטע ביישפילן. א פראגראמירער באקומט א טיקעט: "פאררעכט דעם authentication-פראבלעם". עס הערט זיך פשוט, אבער א פראיעקט קען אנטהאלטן הונדערטער און טויזנטער פיילס, פארשידענע סערוויסעס, קאנפיגוראציע-פיילס, טעסט-אפהענגיגקייטן און ביזנעס-כללים. דער אגענט דארף פארשטיין דעם פראיעקט פאר ער קען מאכן א נוצליכע ענדערונג.

דער וואָרקפלאו גייט אזוי:

- דער אגענט קוקט אדורך די שייכותדיגע טיילן פונעם codebase און פארשטייט דעם פראבלעם.
- ער גענפינט אויס וועלכע פיילס דארפן זיך מעגליך ענדערן, און פירט אויס א לייזונג.
- אנשטאט בלויז צו וועזן דעם געענדערטן קאוד צום פראגראמירער, לויפט דער סיסטעם טעסטס און אנדערע קאנטראלן.
- אויב די טעסטס גייען דורך, איז דאס געוואונטשענע פירונג באשטעטיגט און דער טאסק קען ווערן באטראכט אלס פארענדיגט.
- אויב זיי פאלן דורך, ווערט דער דורכפאל פידבעק: דער אגענט אונטערזוכט דעם טעות, פאררעכט די אויספירונג און לויפט ווידער די קאנטראלן, אפשר עטליכע רונדעס.

מאדערנע AI-קאודינג-געצייג ווערן אלץ מער געבויט ארום אזא פירונג. IBM ערקלערט אז קאודינג-אגענטן קענען אויסשטעלן שריט, לויפן קאמאנדעס, אפשאצן רעזולטאטן, און איבערארבעטן זייער רעזולטאט פארן עס וועזן דעם באניצער. GitHub באשרייבט ענליך אן "agent loop" וואו אן orchestrator מאכט עטליכע אויפרופן צום מאדעל און פירט ווייטער דעם ציקל פון tool-באנוץ ביז דער טאסק איז פארענדיגט.

אבער מען דארף נישט לאזן דעם אגענט ענדערן אלעס אן גרעניצן. דער פראגראמירער מוז באשטימען צו וואס דער אגענט האט צוטריט, וועלכע קאמאנדעס ער מעג לויפן, וועלכע טעסטס באשטימען הצלחה, און ווען מען דארף א מענטשליכע באשטעטיגונג. דא ווערט לופ-אינזשענירינג אן אמת'ע אינזשעניר-דיסציפלין, אנשטאט בלויז לאזן א AI-געצייג לויפן נאכאמאל און נאכאמאל: מען פלאנירט די באדינגונגען וואס מאכן דעם לופ נוצליך, מעסטבאר, און מעגליך זיך צו ערהוילן פון טעותים.

אנווענדונגען אין דער פראקטישער וועלט (26:55:09)

לופ-אינזשענירינג קען זיין נוצליך וואו נאר א טאסק האט עטליכע אקציעס, מעסטבארע רעזולטאטן און געלעגנהייטן צו פארבעסערן:

- קאסטומער-שטיצע: א געוועלכער AI-chatbot ענטפערט א פראגע און ענדיגט. א מער פארגעשריטענער וואָרקפלאו קען פארשטיין דעם פראבלעם, ארויסנעמען די שייכותדיגע אינפארמאציע וועגן דעם אקאונט אדער פראדוקט, פארשלאגן א לייזונג, קאנטראלירן צי דער פראבלעם איז געלייזט, און אריבערגעבן דעם שמועס ווייטער אויב דער קאסטומער דארף נאך הילף. IBM באשרייבט AI-אגענטן אין קאסטומער-סערוויס וואס לייזן טיקעטס, רעדן מיט קאסטומערס, אנאליזירן אינפארמאציע, פערזענליכן די שמועסן, און גיבן איבער קאמפליצירטע פראבלעמען צו מענטשליכע אגענטן.
- דאטא-אנאליז: אנשטאט בעטן איין באריכט אין איין שריט, קען א וואָרקפלאו ארויסנעמען די נויטיגע דאטא, זי אנאליזירן, קאנטראלירן חשבונות, אויסגעפינען אומגעווענליכע רעזולטאטן, פאררעכטן די אנאליז, און דערנאך צוגרייטן דעם לעצטן באריכט.
- פארשונג: א סיסטעם קען זוכן אינפארמאציע, פארגלייכן מקורות, אויסגעפינען וואס פעלט, פארשארפן די פארשונג-פראגע, און ווייטער גיין ביז די נויטיגע באווייזן זענען צוזאמגענומען.
- ביזנעס-אפעראציעס: ענליכע לופס פאר מאניטארירן, אויסגעפינען אויסנאמען, און באשליסן וואס זאל געשען ווייטער.

לופ-אינזשענירינג איז נישט צוגעבינדן צו איין אינדוסטריע; דער מוסטער פאסט וועו די ארבעט גייט נאטירליך לויטן ציקל פון אקציע, באאבאכטונג און פארבעסערונג. אבער נישט יעדער טאסק דארף א לופ. וועו מען בעט א AI צו ערקלערן א פשוט'ן באגריף, קען א קאמפליצירטער וואָרקפלאו שאפן אומנויטיגע לאסט. לופ-אינזשענירינג איז נוצליכער וועו דער טאסק איז לאנג, איבערחזרנדיג, שוואר צו קאנטראלירן מיט דער האנט, אדער פארלאנגט עטליכע פארבונדענע אקציעס. דער ציל איז נישט צוצולייגן מער קאמפליצירטקייט, נאר דעם ריכטיגן שיעור סטרוקטור פאר דעם פראבלעם.

דערנאך גייט דער קורס איבער צו די טעכנאלאגיעס מיט וועלכע מען קען בויען אזעלכע וואָרקפלאוס, און באטאנט אז לופ-אינזשענירינג איז נישט קיין איינציגער סאפטוועיר-פראדוקט אדער framework, נאר א צוגאנג.

לופ-אינזשענירינג און LangGraph אין דער פראקטיק (26:57:12)

וועלכע טעכנאלאגיעס שטיצן לופ-אינזשענירינג (26:57:12)

דער קורס ערקלערט אז עס זענען פארן פארשידענע וועגן ווי צו פלאנירן AI ארבעטס-פראצעסן, און מען קען ניצן פארשידענע געצייג כדי זיי אויסצופירן. אינמיטן פון אזא סיסטעם שטייט געוועלך אן AI מאדעל וואס

קען פארשטיין אנווייזונגען און ארויסגעבן באשלוסן אדער אקציעס. ארום דעם מאדעל לייגן די פראגראמירער צו געצייג וואס לאזן די סיסטעם האנדלען מיט דער דרויסנדיגער וועלט.

ביי א קאוד-ארבעט קענען די געצייג זיין: דער פייל-סיסטעם, דער טערמינאל, דאס ווערסיע-קאנטראל סיסטעם, דאקומענטאציע, און דער "issue tracker" (א ליסטע פון פראבלעמען און באגס). ביי אן אנדער אפליקאציע קענען די געצייג זיין דאטא-באזעס, API's, זוך-סיסטעמען, ביזנעס-ווייכווארג אדער רעקארדס פון קונים.

מאדערנע קאוד-סיסטעמען, ווי Claude Code אדער OpenAI Codex, זענען ביישפילן פון געצייג וואס ארבעטן אינערהאלב דעם ברייטערן אגענט-לופ צוגאנג. עס ווערט אנגעוויזן אז IBM רעכנט ספעציפיש Claude Code און OpenAI Codex צווישן די AI קאוד-סיסטעמען וואס זענען געבויט אויף לופ-אינזשענירינג.

פרעימווארקס קענען אויך העלפן פראגראמירער בויען אזעלכע פראצעסן. LangGraph, למשל, קען מען ניצן כדי צו ארגאניזירן שריט וואס האלטן א "סטעיט" (דעם מצב פון דער ארבעט) און וואס ניצן עטליכע אגענטן. אנדערע אגענט-פרעימווארקס האבן אנדערע וועגן ווי צו קאארדינירן אקציעס און באשלוסן.

די וויכטיגע זאך צו געדענקען איז אז דער געצייג אליין שאפט נישט אויטאמאטיש א גוטן לופ. דער פראגראמירער מוז נאך אלץ באשליסן:

- וואס איז דער ציל;
- צו וועלכע אינפארמאציע די סיסטעם זאל האבן צוטריט;
- וועלכע געצייג זי זאל ניצן;
- וואס רעכנט זיך פאר הצלחה;
- ווי מען באהאנדלט טעותים;
- ווען א מענטש דארף זיך אריינמישן.

אנדערש געזאגט: די טעכנאלאגיע גיט די בוי-שטיינער, אבער דער פלאן פון דעם ארבעטס-פראצעס באשטימט ווי די בוי-שטיינער ארבעטן צוזאמען. דערפאר קענען צוויי סיסטעמען וואס ניצן דעם זעלבן AI מאדעל זיך פירן גאר אנדערש. איינע קען האבן קלארע קריטעריעס פאר הצלחה, שטארקע טעסטן, גוטן קאנטעקסט און פארשטענדליכע תנאים וועו אפצושטעלן. די אנדערע קען פשוט ווייטער ארויסגעבן רעזולטאטן אן קיין באדייטנדע קאנטראל.

די צוקונפט פון לופ-אינזשענירינג (26:58:43)

לופ-אינזשענירינג איז נאך א נייער באגריף וואס האלט אין אנטוויקלען זיך, דערפאר איז שווער צו זאגן פונקטליך ווי די פראקטיק וועט זיך אנטוויקלען אין די קומענדיגע יארן. איין זאך איז אבער שוין קלאר: AI סיסטעמען גייען צו לענגערע און מער קאמפליצירטע ארבעטס-פראצעסן. פראגראמירער ארבעטן מער און מער מיט סיסטעמען וואס ניצן געצייג, טוישן ווייכווארג, לויפן טעסטן און ארבעטן ווייטער איבער עטליכע שריט.

IBM באשרייבט היינט לופ-אינזשענירינג ווי די נייע אגענטישע AI אינזשענירינג-פראקטיק, און פארבינדט עס גלייך מיט מאדערנע קאוד-סיסטעמען ווי Claude און Codex. לויט דעם קורס ווייזט דאס אן אז עס איז א וויכטיגע פעאיגקייט פאר פראגראמירער: צו קענען פלאנירן די סביבה אין וועלכער דער AI ארבעט. אנשטאט נאר שרייבן אנווייזונגען פאר יעדן איינצלנעם געשפרעך, וועלן פראגראמירער מעגליך דארפן פארברענגען

מער צייט מיט: באשטימען פאדערונגען, שאפן טעסטן, איינשטעלן ערלויבענישן, פלאנירן מעכאניזמען פאר רעאקציע, אויפהאלטן דעם קאנטעקסט פון פראיעקט, און באשליסן וואו מען דארף א מענטשליכע באשטעטיגונג.

דהיינו, די ראלע פון פראגראמירער קען זיך מער ריקן צו פלאנירן די סיסטעמען וואס פירן דעם AI. דאס מיינט אבער נישט אז די געווענליכע פראגראמירונג פארשווינדט, גאר פארקערט. וואס מער קאוד די AI סיסטעמען קענען ארויסגעבן, אלס וויכטיגער ווערט די פעאיגקייט צו באשטימען פאדערונגען, פארשטיין די ארכיטעקטור, טעסטן ווי די סיסטעם פירט זיך, איבערקוקן ענדערונגען, און אויפהאלטן פארלעסליכע סיסטעמען. לופ-אינזשענירינג קען דעריבער ווערן נאך א שיכט פון ווייכווארג-אינזשענירינג, נישט אן ערזאץ דערפאר.

מען קען אויך ערווארטן אז מען וועט מער אכטונג געבן אויף געביטן ווי: א סטעיט וואס בלייבט אפגעהיטן לאנגע צייט, אפשאצונג (evaluation), מאניטארינג, קאנטראל פון קאסטן, זיכערהייט, און זיך ערהוילן פון פעלערן. וואס מער אומאפהענגיג אן AI סיסטעם קען האנדלען, אלס וויכטיגער ווערן די אינזשענירינג-קאנטראלן. דער הויפט-געדאנק איז פשוט: אויב AI וועט ארבעטן אויף לענגערע אויפגאבעס, דארף מען בעסערע וועגן צו פירן די ארבעט צווישן דער ערשטער אנווייזונג און דעם ענדגילטיגן רעזולטאט. דאס איז דער פראבלעם וואס לופ-אינזשענירינג פרובירט צו לייזן.

סך-הכל: די הויפט-פונקטן (27:00:15)

דער קורס ברענגט צוזאמען אלע געדאנקען. לופ-אינזשענירינג טוישט ביים יסוד ווי מען פלאנירט AI ארבעטס-פראצעסן. אנשטאט טראכטן נאר וועגן דעם פראמפט וואס מען גיט דעם AI, טראכט מען וועגן דעם גאנצן פראצעס וואס לאזט די סיסטעם ארבעטן צו א ציל. מען באשטימט וועלכן ציל מען דארף דערגרייכן, מען גיט דער סיסטעם די אינפארמאציע און די געצייג וואס זי דארף, מען לאזט זי טון אקציעס, מען קוקט וואס איז געשען, און מען ניצט די רעזולטאטן צו באשליסן וואס דארף קומען ווייטער.

דער וויכטיגסטער געדאנק: איבערחזרן אליין איז נישט גענוג. א נוצליכער לופ דארף אפשאצונג און רעאקציע (feedback), אזוי אז יעדע רונדע האט א תכלית. דאס איז אויך וואס אונטערשיידט לופ-אינזשענירינג פון פשוט בעטן דעם AI "פרוביר נאכאמאל".

פראמפט-אינזשענירינג קאנצענטרירט זיך אויף פארבעסערן די אנווייזונג וואס מען גיט דעם AI, בשעת לופ-אינזשענירינג קאנצענטרירט זיך אויף פלאנירן דעם בריטערן פראצעס ארום דעם מאדעל. AI אגענטן געבן די מעגליכקייט צו טון אקציעס, און לופ-אינזשענירינג גיט א וועג צו ארגאניזירן און פירן די אקציעס איבער עטליכע רונדעס.

ווייכווארג-אנטוויקלונג איז איינס פון די קלארסטע ביישפילן: אן AI קאוד-סיסטעם קען דורכקוקן א פראיעקט, מאכן אן ענדערונג, לויפן טעסטן, אונטערזוכן דעם רעזולטאט, פארריכטן פראבלעמען, און ווייטער ארבעטן ביז די באשטימטע קריטעריעס פאר הצלחה ווערן דערגרייכט. דער זעלבער אלגעמיינער געדאנק פאסט אויך פאר פארשונג, קונים-סערוויס, דאטא-אנאליז און אנדערע פראצעסן מיט עטליכע שריט.

צוגלייך ברענגט לופ-אינזשענירינג ארויס שוועריגקייטן ארום פארלעסליכקייט, קאסטן, זיכערהייט, ערלויבענישן, אפשאצונג און מענטשליכע השגחה. דער ציל איז נישט אוועקצונעמען דעם מענטש פון פראצעס, נאר צו בויען בעסערע סיסטעמען וואו מענטשן באשטימען די צילן און די גרענעצן. בשעת AI סיסטעמען גייען אריבער פון ענטפערן אויף פראגעס צו פארענדיגן לענגערע אויפגאבעס, קען דאס פארשטיין ווי צו פלאנירן אזעלכע לופס ווערן א סך א וויכטיגערער טייל פון מאדערנער AI אינזשענירינג. דער

הויפט-פונקט: די צוקונפט איז נישט נאר בויען קליגערע AI מאדעלן, נאר אויך בויען קליגערע סיסטעמען ארום זיי.

דיבאגן קאמפליצירטע ארבעטס-פראצעסן (27:02:24)

דא גייט דער קורס איבער צו א נייעם טייל, וועגן סיסטעמען געבויט מיט LangGraph. וועלן סיסטעמען ווערן מער פארגעשריטן, ווערט שווערער צו באמערקן פראבלעמען. מולטי-אגענט סיסטעמען שאפן קאמפליצירטע פראצעסן, און ווען עס זענען פאראן אסאך שריט, קען אפילו א קליינער טעות ארויסרופן א קייט פון טעותים. געווענליכע פראבלעמען:

- אומריכטיגע ענדערונגען פונעם סטעיט, ווען די געמיינזאמע אינפארמאציע ווערט פאלש געטוישט;
- אומענדליכע לופס, ווען די סיסטעם חזרט איבער שריט און קומט קיינמאל נישט צו א ריכטיגן סוף;
- דורכפאלן פון געצייג אדער API's, ספעציעל ווען דער פראצעס איז אפהענגיג אין דרויסנדיגע סערוויסעס.

מאניטארינג-געצייג ווי LangSmith העלפן צו זען "execution traces", דאס הייסט, זיי ווייזן ווי דער פראצעס גייט פון איין שריט צום צווייטן. אזוי איז גרינגער צו געפינען וואו עפעס איז שיף געגאנגען. דיבאגן איז נישט נאר פארריכטן טעותים, נאר אויך פארבעסערן די ענויקייט, פארלעסליכקייט און סטאביליטעט. אין עכטע פראיעקטן איז דאס זייער וויכטיג, ווייל א פראצעס וואס ארבעט נאר אמאל איז נישט באמת נוצליך.

פארבעסערן די לייסטונג (27:03:25)

דער קורס גיט עצות ווי צו מאכן די סיסטעמען מער עפעקטיוו:

- רעדוצירן איבעריגע רופן צום מאדעל: אויב די זעלבע סארט בקשה ווערט געמאכט נאכאמאל און נאכאמאל אן א גוטן סיבה, פארשלעפט דאס די ארבעט און הייבט אויף די קאסטן.
- ניצן קעשינג (caching) פאר איבערגעחזרטע פראגעס: קעשינג היט אפ פריערדיגע רעזולטאטן, כדי די סיסטעם זאל נישט דארפן יעדעס מאל טון די זעלבע ארבעט.
- אפטימירן דעם באניץ פון טאקענס, ווייל גרויסע אינפוטס און אוטפוטס ווערן גיך טייער און פאמעלעך.
- לויפן אומאפהענגיגע אגענטן פאראלעל: אויב צוויי אויפגאבעס זענען נישט אפהענגיג איינע אין דער אנדערער, קענען זיי לויפן צו דער זעלבער צייט, וואס שפארט צייט.
- מאניטארן די לאטענסי (פארזאמונג) און די קאסטן פון דער סיסטעם: א סיסטעם קען זיין זייער גענוי, אבער אויב זי איז צו פאמעלעך אדער צו טייער, איז שווער זי צו ניצן אויף א גרויסן פארנעם.

לייסטונג-אפטימיזאציע איז באמת א פראגע פון באלאנס: מען וויל גוטע קוואליטעט, אבער אויך שנעלקייט און עפעקטיווקייט.

פראקטישע באניצן (27:03:55)

אגענטישע AI סיסטעמען ווערן גענוצט:

- ביי פארשונג-הילף, וואו די סיסטעמען העלפן זאמלען, ארגאניזירן און פארשטעלן אינפארמאציע;
- ביי אויטאמאטישע ראפארט-שאפונג, וואו אינפארמאציע ווערט געזאמלט, ארגאניזירט און פארוואנדלט אין ראפארטן מיט וויניגער האנט-ארבעט;

- ביי קלוגע זוך-סיסטעמען, וואו דער ציל איז נישט נאר צו געפינען אינפארמאציע, נאר די נוצליכסטע אינפארמאציע לויטן קאנטעקסט;
- ביי אויטאמאטישן קונים-סערוויס, ספעציעל וואו עס זענען פאראן אסאך איבערגעחזרטע פראגעס און ארגאניזירטע פראצעסן;
- ביי געצייג פאר שאפן און דיבאגן קאוד, וואו זיי העלפן שרייבן, קאנטראלירן און פארבעסערן קאוד.

וואס פארבינדט אלע ביישפילן איז אז זיי נעמען איריין מער ווי איין פשוטן שריט: זיי נעמען איריין סעריעס פון שריט, קאנטראלן, קאנטעקסט און אקציעס.

LangGraph אין פארגלייך מיט אנדערע פרעימווארקס (27:04:57)

- LangGraph פאסט מערסטנס פאר קאמפליצירטע מולטי-אגענט פראצעסן; דאס איז איר הויפט-שטארקייט.
- AutoGen קאנצענטרירט זיך מער אויף געשפרעכן צווישן עטליכע אגענטן.
- CrewAI איז באקאנט פאר אגענטן וואס ארבעטן ווי א טיעם, וואו פארשידענע אגענטן ארבעטן צוזאמען כמעט ווי אויסגעטיילטע ראלעס אין א גרופע.
- LangChain ווערט אפט גענוצט פאר מאדעל-פייפליינס, וואו דער פאקוס איז צו פארקייטלען אויפגאבעס און בויען ארגאניזירטע פראצעסן ארום דעם ארבעט מיטן מאדעל.

די פרעימווארקס זענען נישט פונקט די זעלבע; יעדער האט אן אנדער הויפט-באניץ. וועלכן צו קלייבן, וועגדט זיך אין וואספארא סיסטעם מען וויל בויען. וועז דער פראצעס איז קאמפליצירט און האט עטליכע פארבונדענע שריט און אגענטן, ווערט LangGraph ספעציעל נוצליך.

וועז נישט צו ניצן LangGraph (27:05:28)

א שטארקער פרעימווארק איז נישט אלעמאל די ריכטיגע תשובה. עס זענען פאראן פעלער וואו LangGraph איז נישט נויטיג:

- וועז אן אויפגאבע פאדערט נאר איין רוף צום מאדעל, ווי עס ווערט ערקלערט אין די פריערדיגע סליידס. דארט איז א גרויסער פראצעס-אויפבוי איבעריג.
- וועז אן אפליקאציע דארף זייער נידעריגע לאטענסי, דאס הייסט גאר שנעלע ענטפערס; דעמאלט קען די עקסטערע ארקעסטראציע ווערן א משא.
- ביי קליינע פראטאטייפן אדער פשוטע chatbot סיסטעמען, ווייל דער אויפבוי קען צולייגן מער סטרוקטור ווי דער פראיעקט דארף.
- וועז די ארקעסטראציע פון פראצעס אליין לייגט צו איבעריגע קאמפליצירטקייט; דעמאלט איז אפט בעסער צו בלייבן ביי פשוטע זאכן.

דער פראקטישער לימוד: א גוטער סיסטעם-פלאן מיינט נישט ניצן דעם גרעסטן געצייג, נאר דעם ריכטיגן געצייג.

אינסטאלירן: קלאוד קעגן אויף-פלאץ (27:06:29)

א וויכטיגער באשלוס אין עכטע פראיעקטן איז צי מען זאל אינסטאלירן די סיסטעם אין קלאוד אדער אויף-פלאץ (on-premise). טיעמס באטראכטן געווענליך: קאנטראל, קאסטן, זיכערהייט, אויפהאלטונג און ווי

שנעל מען קען עס אויפשטעלן. קלאוד גיט אפט מער פלעקסיביליטעט און גרינגער צו וואקסן אין פארנעם; אויף-פלאץ גיט מער דירעקטע קאנטראל און קען פאסן פאר סביבות מיט שטרענגע אינערליכע פאדערונגען.

צוגלייך ווערט נאכאמאל געווארנט: עס זענען פאראן פעלער וואו LangGraph איז בכלל נישט נויטיג, ווי פשוטע אויפגאבעס מיט נאר איין מאדעל-רוף, אפליקאציעס וואס דארפן זייער נידעריגע לאטענסי, קליינע פראטאטייפן אדער פשוטע chatbots, און פעלער וואו ארקעסטראציע לייגט צו איבעריגע קאמפליצירטקייט. איידער מען קלייבט צווישן קלאוד און אויף-פלאץ, דארף מען פרעגן צי דער פראצעס דארף באמת אזא שטאפל פון סטרוקטור. די אינסטאלאציע-ברירה איז וויכטיג, אבער די ברירה פונעם געצייג איז נאך וויכטיגער און קומט פריער.

מאניטארן און דערהיינטיגן (27:07:30)

בויען די סיסטעם איז נאר איין טייל פון דער ארבעט. שטענדיגע מאניטארינג העלפט האלטן די סיסטעם פארלעסליך: וועו א פראצעס לויפט, דארף מען אים רעגלמעסיג אויפפאסן כדי צו כאפן פראבלעמען פרי. די הויפט-מאסן צו פאלגן:

- באניץ פון טאקענס, ווייל דאס ווירקט סיי אויף לייסטונג סיי אויף קאסטן;
- לאטענסי, ווייל מענטשן באמערקן גיך פארשפעטיגונגען;
- די געוויקייט פון די באשלוסן פון אגענט, ווייל דער פראצעס דארף נישט נאר לויפן, נאר מאכן פארשטענדליכע באשלוסן;
- דער פראצענט פון הצלחה ביי לויפן געצייג, ווייל דרויסנדיגע געצייג און סערוויסעס מוזן ארבעטן קאנסיסטענט כדי דער פראצעס זאל בלייבן סטאביל.

געצייג ווי LangSmith געבן טרעיסינג, אנאליטיקס און דיבאג-אינפארמאציע, וואס העלפט טיעמס פארשטיין וואס די סיסטעם טוט און וואו מען דארף פארבעסערן. מאניטארינג איז וואס מאכט פון א דעמא וואס ארבעט א פארלעסליכע סיסטעם.

צוגרייטן פראיעקטן פאר דער צוקונפט (27:08:30)

"פיוטשער-פרופינג" מיינט פלאנירן סיסטעמען אזוי אז זיי זאלן קענען וואקסן און זיך צופאסן מיט דער צייט. בויען סיסטעמען וואס קענען וואקסן הייבט זיך אן מיט מאדולארע אגענט-ארכיטעקטורן: וועו יעדער טייל האט א קלארע ראלע, ווערט די סיסטעם גרינגער צו פארברייטערן און אויפהאלטן. עס העלפט אויך אפצוטיילן די אגענטן פון די דרויסנדיגע געצייג, כדי ענדערונגען אין די געצייג זאלן נישט צעברעכן דעם גאנצן פראצעס. נאך א קלוגער שריט איז צוגרייטן די סיסטעם פאר נייע מאדעל-פארבעסערונגען, ווייל געצייג און מאדעלן וועלן זיך ווייטער אנטוויקלען. און מען דארף אויך איינפירן מאניטארינג און feedback-לופס.

פון פיוטשער-פרופינג ביז Pydantic מיט LangGraph (27:09:01)

צוגרייטן זיך פאר דער צוקונפט און מקורות פאר ווייטער לערנען (27:09:01)

דער קורס ענדיגט דעם געדאנק פון "פיוטשער-פרופינג" (future proofing): א סיסטעם דארף זיך כסדר פארבעסערן לויט ווי באניצער נוצן עס באמת. אויך דארף מען האבן א סקאלירבארע אינפראסטרוקטור, ווייל וואקסן איז נישט נאר א שאלה פון לאגיק, נאר אויך פון דער אונטערשטיצונג וואס שטייט אונטער דער

סיסטעם. דער עיקר פון פיוטשער-פרופינג איז נישט צו בויען עפעס וואס ארבעט נאר היינט, נאר עפעס וואס וועט נאך ארבעטן מארגן.

דערנאך וועט ערקלערט וואו מען קען ווייטער לערנען נאך דער פרעזענטאציע:

- די אפיציעלע LangGraph דאקומענטאציע: איינס פון די בעסטע ערטער אנצוהייבן, ווייל זי ערקלערט דעם פריימווארק גלייך פונעם מקור.
- די LangChain GitHub רעפאזיטאריס: נוצלעך ווייל מען זעט דארט עכטע ביישפילן און ווי מען ארבעט אקטיוו אויפן פראיעקט.
- AI-פראגראמירער קהילות אויף פלאטפארמעס ווי Discord און Reddit: מען לערנט דורך דיסקוסיעס, און מען טיילט זיך ווי צו לייזן פראבלעמען.
- אפן-קאוד (open source) ביישפיל-פראיעקטן: זיי זענען ספעציעל נוצלעך ווייל מען זעט ווי אידייען ארבעטן אין דער פראקטיק.
- AI פארשונג-בלאגס און טעכנישע לעקציעס: זיי צעלייגן נייע אנטוויקלונגען און ווייזן ווי מען נוצט זיי אין דער אמת'ער וועלט.

אלע צוזאמען העלפן זיי פראגראמירער באהערשן agentic AI סיסטעמען טריט נאך טריט. דער שליסל איז נישט צו פארלאזן זיך אויף איין מקור. מען לערנט שטארקער וען מען שטעלט צוזאמען דאקומענטאציע, ביישפילן, קהילות און אייגענע פראקטישע פרובירונגען.

בלייבן מעודכן אין א שנעל-טוישנדיגן פעלד (27:10:02)

דער קורס באטאנט אז דאס איז א געביט וואס טוישט זיך גאר שנעל, און גיט עטליכע געוואוינהייטן:

- נאכפאלגן פארשונג-פובליקאציעס, כדי צו פארשטיין וואוהין דאס פעלד באוועגט זיך.
- דורכזוכן GitHub רעפאזיטאריס און אפן-קאוד פראיעקטן, ווייל אסאך פראקטישע אידייען באווייזן זיך דארט צום ערשט.
- נאכפאלגן נייע פריימווארקס און געצייג, ווייל דאס בילד טוישט זיך כסדר און עס קומען אויף רעגולער נייע מעגליכקייטן.
- זיך באטייליגן אין פראגראמירער-קהילות, וואו מען לערנט פון עכטע איבערלעבונגען און נישט נאר פון טעאריע.
- איינס פון די בעסטע וועגן: פרובירן מיט פראקטישע פראיעקטן, ווייל וואס מען טוט מיט די הענט בלייבט ביי אייך.

דער כלל איז פשוט: בלייבט נייגעריג, בלייבט פראקטיש, און לערנט ווייטער אין קליינע, כסדר'דיגע טריט.

דער ערשטער Pydantic מאדעל (27:10:32)

איצט הייבט זיך אן א פראקטישער טייל. איידער מען בויט אן agentic workflow מיט LangGraph, דארף מען פארשטיין ווי סטרוקטורירטע דאטא ארבעט. Pydantic העלפט דעפינירן א ריינע סטרוקטור פאר דער דאטא, און עס קאנטראלירט (validates) אויך צי די דאטא איז ריכטיג.

דער קאוד גייט אזוי:

- אין דער ערשטער שורה וועט אימפארטירט BaseModel פון pydantic. דאס איז א וויכטיגע שורה, ווייל יעדער Pydantic מאדעל וועט געווענליך געשאפן דורך ירשענען (inherit) פון BaseModel.
- מען דעפינירט א קלאס User, וואס שטעלט פאר די סטרוקטור פון דער דאטא וואס מען וויל היטן.
- אינעווייניג שטייט name: str, דאס הייסט דער נאמען מוז זיין א טעקסט (string), און age: int, דאס הייסט דער עלטער דארף זיין א גאנצע צאל (integer).
- דערנאך שאפט מען אן אביעקט user, גיט אריין די אמת'ע ווערטן, און מען דרוקט אויס דעם רעזולטאט.

ווען מען לאזט לויפן דעם קאוד, זעט מען אז Pydantic האט געגעבן א ריינע, קאנטראלירטע סטרוקטור אנשטאט א צופעליגן, אומארגאניזירטן דיקשאנערי. דאס איז זייער נוצלעך אין AI workflows, ווייל אגענטן גיבן אפט איבער דאטא פון איין טריט צום צווייטן, און אויף יענע דאטא מוז מען קענען זיך פארלאזן.

צולייגן כללים צו פעלדער מיט Field (27:12:04)

ביז איצט האט מען נאר קאנטראלירט דעם טיפ פון דער דאטא. אין עכטע אפליקאציעס דארף מען אבער אפט שטרענגערע כללים. למשל, א פראגע (query) טאר נישט זיין צו קורץ, און א פריאריטעט מוז בלייבן אין א געוויסן ראם. דא קומט אריין די פונקציע Field.

- אין דער ערשטער שורה אימפארטירט מען סיי BaseModel און סיי Field. Field לאזט דעפינירן נאך כללים פאר יעדן פעלד.
- אינעם קלאס AgentTask איז דא א פעלד query. עס ווערט באשטימט אז עס מוז האבן כאטש 10 אותיות און מאקסימום 200 אותיות. דאס איז נוצלעך, ווייל א גאר קורצע פראגע איז אפשר נישט גענוג באדייטפול פאר אן אגענט צו באארבעטן.
- priority איז א גאנצע צאל, און עס מוז זיין גרעסער פון 0 און ווייניגער אדער גלייך צו 5.
- agent_type האט אויך באגרעניצונגען אויף דער לענג.
- דערנאך שאפט מען א גילטיגן task אביעקט און מען דרוקט אים אויס.

דער פשט פון דעם קאוד איז אז Pydantic קאנטראלירט נישט נאר דעם טיפ פון דער דאטא, נאר אויך די ביזנעס-כללים. אין agentic AI איז דאס וויכטיג, ווייל יעדע אויפגאבע וואס גייט דורך דעם workflow דארף האלטן א גילטיגע סטרוקטור. ווען מען לאזט עס לויפן, זעט מען אין רעזולטאט די query און די priority.

באהאנדלען פאלשע אינפוט מיט ValidationError (27:13:36)

דער טייל ווייזט וואס עס פאסירט ווען א באניצער גיט אריין פאלשע אינפוט. דאס ווערט אנגערופן איינס פון די וויכטיגסטע טיילן פון דער לעקציע, ווייל אין דער עכטער וועלט איז אינפוט נישט שטענדיג ריין. באניצער קענען געבן קורצע פראמפטס, פאלשע צאלן אדער אומפולשטענדיגע ווערטן, און Pydantic העלפט כאפן די פראבלעמען פרי.

- אין דער אימפארט-שורה וועט אויך אימפארטירט ValidationError . דאס איז דער פעלער וואס Pydantic גיט ווען די דאטא האלט נישט די כללים וואס מען האט דעפינירט.
- מען דעפינירט דעם זעלבן AgentTask מאדעל ווי פריער.
- ערשט ווייזט מען א גילטיגן ביישפיל, כדי צו זען אז ריכטיגע אינפוט ארבעט פיין. דערנאך דרוקט מען אויס א מעלדונג אז איצט פרובירט מען אומגילטיגע אינפוט.

- מען נוצט א try בלאק, און אינעווייניג שאפט מען בכוונה א שלעכטן task: די query איז צו קורץ, די priority איז ארויס פונעם ערלויבטן ראם, און agent_type איז אויך צו קורץ.
- ווען די אינפוט איז אומגילטיג, כאפט דער except בלאק דעם ValidationError און דרוקט אויס די פולע פעלער-מעלדונג.

דער מעלה איז אז אנשטאט דורכפאלן שטילערהייט, זאגט Pydantic פונקטליך וואס איז געווען פאלש. אין agentic AI סיסטעמען איז דאס א גרויסע מעלה, ווייל מען וויל כאפן פעלערן איידער זיי גייען טיפער אריין אינעם גראף. ווען מען לאזט לויפן דעם קאוד, זעט מען ווי די אומגילטיגע אינפוטס ווערן געכאפט.

דיפאלט ווערטן און default_factory (27:15:08)

דער טייל ערקלערט ווי צו שאפן ווערטן אויטאמאטיש ווען דער באניצער גיט זיי נישט. אין עכטע AI סיסטעמען דארף מען נישט יעדן ווערט אריינגעבן מיט דער האנט. עטליכע פעלדער קענען האבן פארשטענדליכע דיפאלט ווערטן, און עטליכע, ווי task IDs, קען מען שאפן אויטאמאטיש.

- אין דער ערשטער שורה אימפארטירט מען uuid4 פון Python'ס uuid מאדול, וואס שאפט אן איינציגארטיגן ID.
- דערנאך אימפארטירט מען BaseModel און Field פון pydantic.
- אינעם קלאס AgentTaskWithDefaults איז דא א פעלד task_id וואס נוצט default_factory. דאס הייסט אז יעדעס מאל ווען מען שאפט א נייעם אביעקט, ווערט אויטאמאטיש געשאפן א נייער ווערט. אין דעם פאל איז עס אן איינציגארטיגער ID.
- query ווערט דעפינירט מיט באגרעניצונגען אויף דער לענג.
- priority האט א דיפאלט ווערט פון 1, און status האט א דיפאלט ווערט פון "pending".

ווען מען שאפט task2, גיט מען נאר אריין די query. כאטש מען האט נישט געגעבן keinen task_id, priority אדער status, פילט Pydantic זיי אויס אליין. דאס איז נוצלעך אין agentic AI , ווייל געוויסע ווערטן אין א workflow דארפן ווערן געשאפן דורך דער סיסטעם, אנשטאט זיי אריינצוגעבן מיט דער האנט יעדעס מאל.

נעסטעד מאדעלן פאר סטרוקטורירטע בקשות (27:16:11)

דא גייט מען איבער פון א פשוטן, פלאכן מאדעל צו א מער רעאליסטישער סטרוקטור וואס ליגט איינס אין צווייטן (nested). אין אן עכטער AI סיסטעם ארבעט מען געווענליך נישט מיט נאר איין קליין שטיקל דאטא. מען האט אפט צוזאמען פרטים פון דער אויפגאבע, די קאנפיגוראציע פונעם מאדעל, און אנדערע איינשטעלונגען. דערפאר זענען נעסטעד מאדעלן וויכטיג.

- ערשט דעפינירט מען AgentTaskWithDefaults, די task-סטרוקטור וואס מען קען שוין.
- דערנאך שאפט מען א מאדעל ModelConfig פאר איינשטעלונגען פונעם מאדעל. ער האט model_name און temperature. דאס temperature פעלד האט גרענעצן צווישן 0 און 1.
- דערנאך שאפט מען א נייעם מאדעל AgentRequest, וואס אנטהאלט צוויי זאכן: task און config. פשוט געזאגט, מען לייגט איין מאדעל אריין אין אן אנדערן מאדעל, און דערפאר הייסט עס א נעסטעד סטרוקטור.
- ווען מען שאפט א request אביעקט, גיט מען נאר אריין די query אינעם task.

וועו מען לאוט לויפן דעם קאוד, שאפט Pydantic אויטאמאטיש די דיפאלט ווערטן פארן task, און אויך די דיפאלט קאנפיגוראציע פונעם מאדעל, ווי מען זעט אין רעזולטאט. דאס איז וויכטיג אין agentic AI, ווייל אן עכטער workflow טראגט נישט נאר אינסטרוקציעס, נאר אויך די איינשטעלונגען וואס באשטימען ווי דער אגענט זאל זיך אויפפירן.

רייניגן אינפוט מיט א custom validator (27:17:41)

דער טייל ווייזט אז Pydantic קען טון מער ווי נאר קאנטראלירן אינפוט. עס קען זי אויך רייניגן און נארמאליזירן איידער דער רעשט פון דער סיסטעם נוצט זי. דאס איז נוצלעך, ווייל אין עכטע אפליקאציעס שרייבן באניצער אפט א געמישטן טעקסט מיט איבעריגע ספעיסעס און פונקטואציע.

- אין דער אימפארט-שורה אימפארטירט מען field_validator צוזאמען מיט BaseModel און Field.
- מען שאפט דעם מאדעל ValidatedAgentTask, וואס זעט אויס ענליך צו די פריערדיגע ביישפילן.
- דער וויכטיגער טייל איז די שורה @field_validator("query"). דער דעקאראטאר זאגט פאר Pydantic אז מען וויל לאזן לויפן אן אייגענע לאגיק אויף דעם query פעלד.
- די מעטאד clean_query באקומט דעם רויען ווערט פונעם באניצער. אינעווייניג נעמט זי ערשט אוועק איבעריגע ספעיסעס: מען צעטיילט דעם טעקסט (split) און מען שטעלט אים צוריק צוזאמען (join) מיט איינציגע ספעיסעס.
- דערנאך קוקט מען צי דער ריינער טעקסט ענדיגט זיך מיט א פונקט. אויב יא, נעמט מען אוועק דעם לעצטן פונקט.
- צום סוף גיט מען צוריק דעם ריינעם ווערט.

וועו מען שאפט good_task, גיט מען בכוונה אריין א געמישטע query מיט איבעריגע ספעיסעס און א פונקט אינעם סוף. נאך דער קאנטראל ווערט דער אויסגעדרוקטער רעזולטאט ריין און נארמאליזירט. ביים ערשטן לויפן איז ארויסגעקומען א פעלער. נאכדעם וואס מען האט עס איבערגעקוקט, האט מען באקומען דעם ריכטיגן רעזולטאט. דאס איז נוצלעך אין agentic AI, ווייל מען וויל שיקן ריינע אינפוט אינעם workflow, נישט רויען געמישטן טעקסט.

פארבינדן Pydantic מיט LangGraph state (27:19:12)

דאס איז דער לעצטער קאוד-טעמע, וואס פארבינדט אלעס צוריק מיט LangGraph. ביז איצט האט מען געזען ווי צו דעפינירן סטרוקטורן און קאנטראלירן דאטא מיט Pydantic , און איצט נוצט מען דעם געדאנק אין א LangGraph workflow. אין LangGraph גייט דאטא פון איין נאוד (node) צום צווייטן אלס א געמיינזאמער state. אויב דער state איז ריין און קאנטראלירט, איז דער workflow אסאך מער פארלעסלעך.

- אין דער ערשטער שורה אימפארטירט מען BaseModel פון pydantic, און דערנאך StateGraph, START און END פון langgraph.
- דערנאך שאפט מען א קלאס AgentState, וואס ווערט דער געמיינזאמער state פארן גראף. ער אנטהאלט query, status און result.
- דער דיפאלט status איז "received", און דער דיפאלט result איז א ליידיגער טעקסט.
- נאכדעם דעפינירט מען די ערשטע נאוד-פונקציע, validate_step, וואס באקומט דעם state. ווייטער ווערט עס ערקלערט אינעם קומענדיגן טייל.

פון Pydantic מיט LangGraph צו CrewAI (27:20:16)

דער LangGraph-ביישפיל ענדיגט זיך (27:20:16)

דער קורס פירט צו ענדע דעם קאוד-ביישפיל וואו Pydantic ווערט פארבונדן מיט LangGraph. די ערשטע נאוד-פונקציע, validate step, באקומט דעם איצטיגן state, דרוקט אים אויס, און גיט צוריק אן update וואס טוישט דעם status צו validated.

דערנאך ווערט דעפינירט די צווייטע נאוד-פונקציע, process step. אויך זי באקומט דעם state און דרוקט אים אויס, נאר דאס מאל טוישט זי דעם status צו completed און שאפט א רעזולטאט-מעסעדזש פון דעם query וואס איז אריינגעקומען.

נאך די נאוד-פונקציעס ווערט געשאפן דער graph builder דורך StateGraph מיט AgentState . די AgentState-סטרוקטור דינט אלס דער געמיינזאמער state פון גאנצן גראף. דערנאך:

- מען לייגט צו די צוויי נאודס אין גראף.
- מען פארבינדט דעם פלוס מיט edges: פון START צו validate step, דערנאך צו process step, און צום סוף צו END.
- מען קאמפיילט דעם גראף.
- מען רופט אים אויף (invoke) מיט אן ערשטן query.
- מען דרוקט אויס דעם רעזולטאט.

דער הויפט-געדאנק: LangGraph לאזט פירן א סטרוקטורירטן state דורך עטליכע טריט אין א workflow. Pydantic האלט דעם state ריין, און LangGraph ארגאניזירט ווי ער גייט פון איין נאוד צום צווייטן. דאס איז די יסודות'דיגע פארבינדונג צווישן Pydantic און אגענטישן AI מיט LangGraph.

א איבערבליק פון דעם Pydantic-חלק (27:21:17)

דער קורס חוז'ט איבער ווי דער קאוד-דורכגאנג איז געגאנגען:

- מען האט אנגעהויבן מיט א פשוט'ן Pydantic-מאדעל און פארשטאנען ווי סטרוקטורירטע דאטא ארבעט.
- מען האט צוגעלייגט באגרעניצונגען (constraints) מיט Field.
- מען האט באהאנדלט אומגילטיגע רעזולטאטן מיט ValidationError.
- מען האט געשאפן default-ווערטן מיט default_factory.
- מען האט געבויט nested מאדעלן פאר רעאליסטישע בקשה-סטרוקטורן.
- מען האט געריניגט דעם באניצער'ס אינפוט מיט א custom validator.
- צום סוף האט מען פארבונדן די סטרוקטורירטע דאטא מיט א LangGraph-workflow.

דער עיקר: ווען מען בויט אוואנסירטע אגענטן, טולס און סיסטעמען מיט מערסטאפיגן חשבון, דארף מען פארלעסליכע דאטא-סטרוקטורן. Pydantic דעפינירט און באשטעטיגט די סטרוקטורן, און LangGraph פירט די דאטא צווישן די נאודס אין אן אגענטישן workflow. לויט דעם קורס איז די קאמבינאציע זייער שטארק פאר בויען AI וואס מען קען זיך אויף אים פארלאזן.

וואס איז CrewAI? (27:21:47)

דא הייבט זיך אן א נייער טעמע. CrewAI איז אן open-source פריימווערק פאר בויען AI-אגענטן וואס ארבעטן צוזאמען אויף אויפגאבעס. דער קורס עפנט די אפיציעלע דאקומענטאציע פון CrewAI . דארט שרייבט CrewAI אויף זיך אז עס איז א פריימווערק פאר ארגאניזירן (orchestrating) אויטאנאמע אגענטן און קאמפליצירטע workflows, מיט צוויי יסוד-באגריפן: crews און flows.

- א crew איז א מאנשאפט פון אגענטן וואס ארבעטן צוזאמען.
- א flow איז א סטרוקטורירטער, event-driven workflow וואס קאנטראלירט ווי די ארבעט גייט פאר.

א פשוט'ער משל: איין AI-אגענט ארבעט ווי א פארשער, א צווייטער ווי א שרייבער, און א דריטער ווי א רעצענזענט. די דריי, יעדער מיט אן אנדער אויפגאבע, ווערן א מאנשאפט. CrewAI העלפט באשטימען זייערע ראלעס, זיי געבן טולס, און זיי קאארדינירן כדי צוזאמען דורכצופירן א גרעסערע ארבעט. לויט דער דאקומענטאציע קענען אגענטן ניצן טולס, האלטן זכרון (memory), ארבעטן צוזאמען מיט אנדערע אגענטן, און איבערגעבן (delegate) אויפגאבעס ווען דאס איז ערלויבט.

פארוואס ניצט מען CrewAI? דער קורס רעכנט אויס:

- ווי א פארשונג-געהילף (research assistant)
- פאר שאפן באריכטן (report generation)
- פאר קאסטומער-סערוויס workflows
- פאר דאטא-אנאליז pipelines
- פאר מערסטאפיגע ביזנעס-אויטאמאציע

אין די אלע פעלער איז CrewAI זייער שטארק. נאך א נוצליכער פונקט: CrewAI שטעלט זיך פאר אלס א Python-פריימווערק פאר production-ready מולטי-אגענט סיסטעמען, און אין זיין GitHub-רעפאזיטאריע שטייט אז עס איז געבויט אומאפהענגיג, נישט אויף LangChain.

אין קורצן: CrewAI איז א פריימווערק וואס לאזט עטליכע AI-אגענטן ארבעטן ווי א מאנשאפט, וואו יעדער אגענט האט א ראלע און זיי ענדיגן צוזאמען אן אויפגאבע.

די ארכיטעקטור פון CrewAI: state, events און control flow (27:24:23)

דער קורס ווייזט א דיאגראם פון דער CrewAI-ארכיטעקטור. מען זעט דארט code, crew און state. דאס איז א flow-סיסטעם אין CrewAI וואס גיט מער פיינע קאנטראל, באשלוס-קראפט און פינקטלעכקייט איבער די אויפגאבעס פון די אגענטן. מען נעמט דורך יעדן באגריף באזונדער.

State management: דאס לאזט דעם סיסטעם האלטן דעם מצב פון פראצעס במשך דער צייט. דער סיסטעם געדענקט דאטא, וועריעבלס אדער באשלוסן פון פריערדיגע טריט און קען זיי ניצן שפעטער. דער ביישפיל: אן אגענט העלפט באשטעלן א פראדוקט אנליין. אין ערשטן טריט זאמלט ער די באניצער'ס פרעפערענצן, למשל וועלכן סארט פראדוקט מען זוכט און אין וועלכן פרייז-ראם. אין נעקסטן טריט ניצט ער די אינפארמאציע צו פארשלאגן אפציעס. דער state האלט די פרעפערענצן אויפגעהיטן וויל דער אגענט גייט ווייטער.

Event-driven execution: א קאמפאנענט וואס לאזט אויס באשטימטע אקציעס לויט געוויסע events אדער אינפוטס וואס פאסירן במשך דעם flow. מען קען עס פארשטיין ווי א listener וואס ווארט אויף געוויסע

תנאים איידער מען גייט ווייטער. דער ביישפיל: אן אגענט אין א קאסטומער-סערוויס flow. אויב דער באניצער גיט פידבעק ווי "איך בין נישט צופרידן מיטן פראדוקט", לאזט דער event-driven סיסטעם אויס אן אקציע צו עסקאלירן די זאך צו א סופארט-אגענט. אן דעם וואלט דער flow גלאט ווייטער געגאנגען, אן רעאגירן אויף דעם קונה'ס פידבעק.

Control flow: דא דעפינירט מען די לאגיק ווי דער אגענט ווערט קאנטראלירט און ווי ער גייט דורכן פראצעס. מען ניצט תנאי-לאגיק, loops און branching צו באשליסן דעם נעקסטן טריט, אזוי ווי מען זאגט: אויב X פאסירט, טו Y, אנדערש טו Z. דער ביישפיל איז א פראדוקט-רעקאמענדאציע סיסטעם:

- Branching: אויב דער באניצער קלייבט שיך אין דער קאטעגאריע, פארשלאגט דער flow שיך. אויב ער קלייבט העמדער, גייט עס אריבער צו העמדער.
- Loops: אויב דער באניצער האט נאך דריי פארשלאגן נאך נישט געטראפן קיין פראדוקט וואס געפעלט אים, קען דער סיסטעם זיך צוריקדרייען און ווייזן אנדערע רעקאמענדאציעס.

ווי אלע חלקים פאסן זיך צוזאמען (27:27:25)

דער code איז די לאגישע טריט אדער סקריפטס געשריבן פאר יעדע סטאדיע פון פראצעס, למשל דער קאוד וואס פירט דעם רעקאמענדאציע-פראצעס, דער קאוד פאר באשלוסן, און אזוי ווייטער. די crew איז א מאנשאפט פון אגענטן אדער סוב-פראצעסן וואס ארבעטן אויף דער אויפגאבע. למשל, איין crew קען באהאנדלען די אויסוואל פון פראדוקטן, און א צווייטע דעם רעקאמענדאציע-מאטאר. דער state איז וואו מען האלט און פארוואלטעט די דאטא, אזוי אז פריערדיגע באשלוסן און אינפוטס בלייבן אויפגעהיטן איבער עטליכע טריט.

צו פארשטיין דעם גאנצן flow ברענגט דער קורס א ביישפיל פון אן AI-געטריבענעם אנליין שאפינג-געהילף:

- דער ערשטער טריט איז code פלוס crew: דער סיסטעם פרעגט וועגן די פרעפערענצן, למשל דער סארט פראדוקט און דער בודזשעט.
- אינעם state ווערן די פרעפערענצן אויפגעהיטן פאר שפעטער.
- די דורכפירונג גייט event-driven: דער סיסטעם ווייזט אפציעס, און ווען מען קליקט אויף א פריט איז דאס אן event, וואס לאזט אויס די נעקסטע אקציע, נעמליך ווייזן די פרטים פון דעם פריט.
- אויב די פרטים געפעלן נישט, דרייט זיך דער flow צוריק (loop) און ווייזט אנדערע פראדוקטן.
- אויב מען איז צופרידן, גייט מען ווייטער צום checkout-בלאט.

לויט דעם קורס גיבן די flows אינעם דיאגראם די סטרוקטורירטע פינקטלעכקייט וואס מען דארף כדי די קאמפאנענטן זאלן גלאט ארבעטן צוזאמען. דאס מאכט מעגליך א דינאמישע, אינטעראקטיווע איבערלעבונג, וואו AI-אגענטן ארבעטן עפעקטיוו צוזאמען לויט די דעפינירטע states, events און לאגיק.

CrewAI קעגן LangChain (27:29:28)

דער קורס דערקלערט דערנאך דעם חילוק צווישן CrewAI און LangChain. LangChain און LangGraph דינען אנדערע צוועקן, אבער אלע העלפן בויען AI-אגענטן וואס קענען באהאנדלען קאמפליצירטע אויפגאבעס.

CrewAI איז געמאכט פאר בויען מולטי-אגענט סיסטעמען, וואו עטליכע AI-אגענטן ארבעטן צוזאמען צו ענדיגן אויפגאבעס. עס פאסט צו ארגאניזירן אויטאנאמע אגענטן וואס פירן אויס אויפגאבעס פאראלעל אדער איינס

נאכן אנדערן. דאס איז נוצליך פאר פראקטישע אנוועכדונגען ווי קאסטומער-סערוויס באטס, פארשונג-געהילפן און אויטאמאטישע workflows. די הויפט-אייגנשאפטן:

- מולטי-אגענט צוזאמענארבעט: עס ארגאניזירט עטליכע אגענטן וואס ארבעטן פאראלעל און גיבן אויפגאבעס איינער דעם אנדערן.
- flow-באזירטע ארכיטעקטור: די אגענטן גייען לויט א סדר פון טריט וואס ווערן אויסגעלאזט דורך געווייסע תנאים אדער events.
- state management: דער אגענט געדענקט דעם קאנטעקסט אדער די דאטא צווישן טריט, וואס לאזט אים ארבעטן עפעקטיוו אין קאמפליצירטע מצבים.
- event-driven execution: עס לאזט אויס אקציעס לויט events אדער אינפוטס פון באניצער אדער פון דרויסנדיגע סיסטעמען.
- פיינע קאנטראל: מען ניצט control flows, ווי תנאי-לאגיק, loops אדער branching, צו פירן ווי די אגענטן גייען דורך די אויפגאבע.

א ביישפיל: א מאנשאפט אגענטן ארבעט אויף אויטאמאציע פון קאסטומער-סופארט. איין אגענט שאפט טיקעטס, א צווייטער באהאנדלט פראגעס, און א דריטער עסקאלירט פראבלעמען ווען נויטיג. דאס אלעס קען מען גרינג דורכפירן מיט CrewAI.

LangChain איז אויך א פריימווערק, אבער פאר בויען אפליקאציעס מיט שפראך-מאדעלן, באזונדערס גרויסע שפראך-מאדעלן (LLMs). עס פארבינדט זיך מיט דרויסנדיגע טולס און APIs צו בויען שטארקע workflows פאר LLM-אפליקאציעס, ווי chatbots, דאקומענט-ארויסנעם סיסטעמען, און נאך. די הויפט-אייגנשאפטן:

- פארבינדונג מיט LLMs: מען קען גרינג ארבעטן מיט LLMs ווי GPT-מאדעלן פאר פארשידענע אויפגאבעס.
- chains און agents: א chain איז א סעריע פראמפטס וואס ווערן איינס נאכן אנדערן געשיקט צו א מאדעל. agents זענען AI-סיסטעמען וואס קענען באשליסן אדער טון אקציעס.
- איינגעבויטע שטיצע פאר memory: דער סיסטעם געדענקט פריערדיגע שמועסן און אינטעראקציעס, כדי זיך צו פארבעסערן מיט דער צייט.
- טולס: מען קען ניצן דרויסנדיגע טולס ווי דאטאבעיסעס, web scraping, API-רופן, א קאלקולאטאר, און אזוי ווייטער, צו פארברייטערן די מעגליכקייטן.
- prompt engineering: שטארקע פארוואלטונג פון פראמפטס, צו פארמאטירן אינפוט און אויספוט און גלאט באהאנדלען א פראצעס מיט עטליכע טריט.

א ביישפיל פון LangChain איז א סיסטעם פאר ענטפערן אויף פראגעס, וואס ניצט LangChain צו זוכן אין דאקומענטן און שאפן ענטפערס פון אן LLM.

דער סך-הכל פון דער פארגלייך: LangChain קאנצענטרירט זיך בעיקר אויף בויען אינטעליגענטע סיסטעמען מיט LLMs און זיי פארבינדן מיט אנדערע טולס, אריינגערעכנט פארוואלטן פראמפט-פלוסן. עס פאסט מער פאר אויפגאבעס פון natural language processing. CrewAI, אויף דער אנדערער זייט, איז געמאכט פאר קאארדינירן עטליכע אגענטן און פאר מער קאמפליצירטע event-driven workflows.

LangGraph און די אינסטאלאציע פון CrewAI (27:34:05)

וואס איז ?LangGraph (27:34:05)

דער קורס ערקלערט אז LangGraph איז אויך א פריימווערק, וואס איז געבויט צו דערמעגליכן טראכטן און באשליסן אויפן יסוד פון א גראף, מיט שפראך-מאדעלן. מען באניצט עס צו בויען ארבעט-פלוסן מיט עטליכע טריט נאכאנאנד. דער חילוק: LangChain פירט די טריט אין א ליניע, איינס נאכן צווייטן, און LangGraph אָרגאַניזירט דעם גאנצן ארבעט-פלוס אלס א גראף. יעדער "נאוד" (node) שטעלט פאר א טריט, און די "עדזשעס" (edges), די ליניעס צווישן די נאודס, שטעלן פאר די פארבינדונגען צווישן די טריט, אדער די אקציעס וואס קומען פון זיי.

די הויפט-אייגנשאפטן פון LangGraph (27:34:37)

אין דער לעקציע ווערן אויסגערעכנט די וויכטיגסטע אייגנשאפטן:

- א ארכיטעקטור אויף א גראף: LangChain באניצט א ליניארע קייט, און LangGraph באניצט א גראף. די נאודס זענען די אויפגאבן, און די עדזשעס זענען די באציאונגען צווישן זיי. דאס מאכט עס זייער גרינג צו פירן ארבעט-פלוסן וואס צעטיילן זיך אויף פארשידענע וועגן.
- היטן דעם "סטעיט" (state): LangGraph דערלויבט צו פאלגן נאך דעם מצב פונעם ארבעט-פלוס ביי יעדן נאוד. אזוי זעט מען טיפער ווי די אויפגאבן אנטוויקלען זיך און ווי זיי ווירקן איינע אויף די אנדערע.
- א בילדלעכער גראף: LangGraph קומט אפט מיט געצייג וואס מאלט אויס דעם ארבעט-פלוס. דאס מאכט עס גרינגער צו געפינען פעלערן (debug) און צו פארבעסערן.
- קאמפליצירטע באשלוס-ביימער: LangGraph איז שטארק אין ארבעט-פלוסן וואס דארפן באשליסן לויט פריערדיגע דאטא אדער קאנטעקסט. עס דערמעגליכט קאמפליצירטע באשלוס-ביימער און לאגיק.

ווי א ביישפיל ווערט געבראכט א רעקאמענדאציע-סיסטעם וואס באניצט LangGraph צו באשליסן וועלכן פראדוקט פארצושלאגן, לויט ווי דער באניצער האט זיך באנומען. ביי יעדער רעקאמענדאציע צעטיילט זיך דער באשלוס לויט וואס דער באניצער האט פריער אויסגעקליבן.

פארגלייך צווישן CrewAI און LangGraph (27:35:39)

עס ווערט ערקלערט אז CrewAI קאנצענטרירט זיך אויף צוזאמענארבעט צווישן עטליכע אגענטן. LangGraph קאנצענטרירט זיך מער אויף אָרגאַניזירן אויפגאבן אין א גראף, מיט אסאך באשלוס-פראצעסן. LangGraph פאסט דעריבער וועו מען דארף זייער בייגעוודיגע פלוסן מיט אסאך צווייגן. CrewAI פאסט פאר אויפגאבן וואו די אגענטן דארפן צוזאמענארבעטן און זיך טיילן מיטן סטעיט פון טריט צו טריט. דאס זענען, קורץ געזאגט, די הויפט-חילוקים צווישן די דריי פריימווערקס, און מען קען זיי ווייטער אויספארשן אין זייער אפיציעלער דאקומענטאציע. נאכדעם קערט זיך דער קורס צוריק צו CrewAI, צום טריט פון אינסטאלירן.

וואס מען דארף האבן פאר אינסטאלירן: Python (27:36:40)

אויפן וועבזייטל פון CrewAI איז דא א באזונדערע זייט פאר אינסטאלאציע (Installation). פאר מען אינסטאלירט CrewAI, דארף מען האבן עטליכע זאכן אויפן קאמפיוטער. די ערשטע איז Python 3 . מען עפנט cmd און מען קאנטראלירט די ווערסיע פון Python 3. אין דער דעמא זעט מען ווערסיע Python 3.12.5. ווער עס האט נישט Python 3, גייט צום אפיציעלן וועבזייטל פון Pythonמען זוכט "Python download") און דרוקט אויף download. מען קען אראפנעמען דעם "Python manager", אדער גלייך די ווערסיע. אויפן

וועבזייטל וועט מען באקומען די לעצטע ווערסיע, 3.14. אבער 3.12 איז אויך גוט גענוג, און וועט עס האט עס, וועט עס טון די ארבעט.

אינסטאלירן UV (27:37:41)

אין נעקסטן טריט דארף מען א סיסטעם וואס פירט די "דיפענדענסיס" (די פעקלעך וואס דער פראיעקט דארף). דער קורס באטאָנט אז מען באניצט נישט pip צו אינסטאלירן CrewAI, ווייל די אפיציעלע דאקומענטאציע שלאגט דאס נישט פאר. מען דארף אנשטאט דעם האבן UV. מען זוכט "install UV", און די דאקומענטאציע פון astral איז די אפיציעלע אנווייזונג פאר UV. די וועגן צו אינסטאלירן:

- אויף Windows: מען קאפירט דעם באפעל פון דער דאקומענטאציע, מען עפנט PowerShell, און מען לויפט אים דארט.
- אויף Mac OS אדער Linux: מען באניצט curl, ממילא דארף curl זיין אינסטאלירט. דער ערשטער באפעל הייבט זיך אן מיט curl https://astral.sh- LsSf. וועט עס האט נישט curl, קען באניצן wget. מען קען אויך צולייגן א באשטימטע URL אדער ווערסיע.
- מען קען אויך אינסטאלירן UV דורך pipx, אויב מען האט עס, אדער אפילו דורך pip: pip install uv.

עס ווערט געזאגט אז די טריט זענען זייער גרינג צו פאלגן.

אויפשטעלן דעם פראיעקט אין VS Code (27:39:17)

פאר מען הייבט אן ארבעטן מיטן פריימווערק, שאפט מען א פראיעקט-סטרוקטור. די דעמא באניצט VS Code. מען מאכט א טעקע (folder) מיטן נאמען crew AI, און מען עפנט זי אין VS Code דורך "Open Folder" אין א נייעם פענצטער. נאכדעם עפנט מען דעם טערמינאל דורך View און Terminal. מען ווארט א ביסל ביז VS Code דערקענט די פראיעקט-סטרוקטור.

אינסטאלירן CrewAI און קאנטראלירן (27:40:19)

אין דער אפיציעלער דאקומענטאציע פון CrewAI שטייט דער באפעל צו אינסטאלירן:

```
uv tool install crewai
```

מען קאפירט אים און מען לויפט אים אינעם טערמינאל, אין דער טעקע פונעם פראיעקט. מען זעט ווי ער הייבט אן אינסטאלירן די אלע דיפענדענסיס. עס זענען 50 פעקלעך, און דאס נעמט א שטיק צייט. אין א געוויסן מאמענט שטייט מען ביי 42. דער קורס באטאָנט אז אלע די פעקלעך זענען וויכטיג פארן פראיעקט. וועט עס ענדיגט זיך, זענען אלע געצייג אינסטאלירט.

צו קאנטראלירן אז CrewAI איז טאקע אינסטאלירט, לויפט מען לויט דער דאקומענטאציע:

```
uv tool list
```

מען זעט אז CrewAI איז אינסטאלירט, אין ווערסיע 1.11.1.1 (אזוי ווי עס ווערט ארויסגעזאגט). דער קורס באמערקט אז וועט עס איז אנגעקומען ביז אהער, האט געמאכט דעם ערשטן טריט צו ארבעטן מיט סיסטעמען פון עטליכע אגענטן.

דערהיַנטיגן און שאפן א נייעם פראיעקט (27:42:20)

אין דער דאקומענטאציע זעט מען נאך באפעלן. צו דערהיַנטיגן CrewAI באניצט מען דעם אינסטאלאציע-באפעל צוזאמען מיט אן "upgrade". צו שאפן א נייעם CrewAI פראיעקט שרייבט מען:

;gt&project name;lt& crewai create crew

אין דער דעמא הייסט דער פראיעקט CrewAI .CRM assistant שאפט דאן אליין די גאנצע טעקע-סטרוקטור. דערביי פרעגט עס וועלכן "פראווייַדער" (פארזארגער פון מאדעלן) מען וויל אויפשטעלן. אין דער דעמא קלייבט מען OpenAI, און ווי מאדעל קלייבט מען GPT-4.

שאפן און אריינלייגן דעם OpenAI API שליסל (27:43:24)

נאכדעם פרעגט מען פארן OpenAI API שליסל, און עס ווערט געוויזן ווי צו באקומען אזא שליסל:

- מען גייט צו דער OpenAI API פלאטפארמע, און אויפן דעשבאָרד דרוקט מען "create new secret key". מען קען אויך באניצן א שליסל וואס מען האט שוין.
- אין דער דעמא שאפט מען א נייעם שליסל מיטן נאמען crew AI, כדי מען זאל אים קענען באניצן ביי יעדער דעמא.
- מען גיט דערלויבעניש (permissions), למשל צו לייענען און שרייבן. דא גיט מען אלע דערלויבעניש, ווייל עס איז נאר פאר א דעמא.
- מען קאפירט דעם שליסל, מען לייגט אים אריין אינעם טערמינאל, און מען דרוקט Enter.

דער שליסל ווערט אפגעהיטן, דער מאדעל GPT-4 איז אויסגעקליבן, און CrewAI שאפט די נויטיגע טעקעס. מען מאכט ריין דעם טערמינאל, כדי ער זאל אויסזען ריינער.

די טעקע-סטרוקטור: knowledge און agents.yaml (27:44:29)

אין דער סטרוקטור איז דא א טעקע מיטן נאמען knowledge, און דארט ליגט א ביישפיל-טעקע וועגן די "user preference", דאס הייסט וואס דער באניצער האט ליב. דארט שטייט אז דער באניצער הייסט John Doe, אז ער איז אן AI אינזשעניר, אז ער אינטערעסירט זיך אין AI אגענטן, און אז ער וואוינט אין San Francisco, California.

אינעם src אפטייל געפינט מען די טעקעס config און tools. אין config ליגן די קאנפיגוראציע-טעקעס. די ערשטע איז agents.yaml:

- דער ערשטער אגענט האט די ראלע פון א "researcher", און דווקא א "senior data researcher" פאר א געוויסן טעמע. זיין ציל (goal) איז אויפצודעקן די נייסטע אנטוויקלונגען אין דעם טעמע. דער טעמע ווערט אריינגעשטעלט דורך א וואריאבל topic, וואס קומט פון אנדערע טעקעס. ער האט אויך א "backstory", א הינטערגרונט-באשרייבונג, למשל אז ער איז אן איבערגעלעבטער פארשער מיט א טאלאנט צו אנטדעקן די לעצטע אנטוויקלונגען.
- דער צווייטער אגענט איז א "reporting analyst", מיט זיין אייגענעם ציל און הינטערגרונט.

דער קורס פאסט עס צוזאם: אין agents.yaml באקומט יעדער אגענט זיין ראלע, זיין ציל און זיין הינטערגרונט. דאס איז א קורצע באשרייבונג פון וואס יעדער אגענט וועט טון.

די טעקע tasks.yaml (27:46:01)

אין tasks.yaml ווערן באשטימט די אויפגאבן:

- research task: די באשרייבונג איז צו מאכן א גרינטלעכע פארשונג ווען דעם טעמע, מיט אן אנווייזונג וועלכעס יאר עס איז יעצט, כדי ער זאל וויסן וועלכע אינפארמאציע איז נייעס. דער ערוואַרטעטער רעזולטאט (expected output) איז א רשימה פון 10 פונקטן מיט די וויכטיגסטע אינפארמאציע ווען דעם טעמע. וועלכער טעמע עס זאל נאר זיין, באניצט דער אגענט אן LLM צו געפינען דעם ענטפער.
- reporting task: די באשרייבונג איז איבערצוקוקן דעם קאנטעקסט פון יעדן טעמע. דער ערוואַרטעטער רעזולטאט איז א פולשטענדיגער באריכט מיט די הויפט-טעמעס, יעדער טעמע מיט א פולן אפטייל אינפארמאציע.

יעדע אויפגאבע איז צוגעטיילט צו איר אגענט: די ערשטע צום researcher, און די צווייטע צום reporting analyst. דערמיט ענדיגט זיך דער איבערבליק פון די קאנפיגוראציע-טעקעס.

א CrewAI פראיעקט: סטרוקטור, לויפן און אגענט-דיזיין (27:47:29)

די custom tools טעקע און איר אינהאלט (27:47:29)

אין דער לעקציע ווערט ווייטער אדורכגעגאנגען דער CrewAI פראיעקט, און מען קוקט זיך איצט צו וועלכע tools דער פראיעקט באניצט. ערשט זעט מען א ליידיגע טעקע. עס ווערט ערקלערט אז געווענליך הייבט זיך א Python פעקל אן מיט א טעקע init, און מען קען זי לאזן ווי זי איז, אן קיין ענדערונגען.

דערנאך קומט די טעקע וואו עס זענען דעפינירט די custom tools, הייסט די אייגענע געצייג. מען זעט אז דער קאוד אימפארטירט BaseTool פון דעם crewai פריימווערק, Type פון דער typing ביבליאטעק, און BaseModel און Field פון pydantic.

- ערשט ווערט דעפינירט א קלאס וואס הייסט ווי "custom model input". ער ירשנט פון BaseModel, און זיין ציל איז צו באשטימען דעם input schema, דאס הייסט די פארעם פון דעם אריינגאנג וואס דער tool באקומט. דא ווערט דעפינירט א שטריקל-פעלד (string field) מיט א description, א באשרייבונג וואס דאס פעלד באדייט.
- דערנאך ווערט דעפינירט דער custom tool אליין. ער ירשנט פון BaseTool. מען גיט אים א name, דעם נאמען פונעם tool, און א description, וואס איז אויך א שטריקל. אין דער מוסטער-באשרייבונג שטייט, אז מען דארף קלאר שרייבן פאר וואס דער tool איז נוצלעך, ווייל דער אגענט וועט דארפן די דאזיגע אינפארמאציע כדי צו וויסן ווען און ווי אזוי אים צו באנוצן. דא דארף מען אליין אריינשרייבן א פאסיגע באשרייבונג.
- צום סוף איז דא א פונקציע (def) מיט די ארגומענטן. אין איר דארף מען אליין אימפלעמענטירן די ציל וואס מען וויל אז דער tool זאל אויספירן.

די crew.py טעקע: אגענטן, טאסקס און דעקאראטארן (27:49:02)

דערנאך ווערט געוויזן די crew.py טעקע. דא באניצט מען decorators, וואס מען זעט איבער דער גאנצער טעקע. מען דעפינירט א פראיעקט, אין דעם ביישפיל רופט ער זיך CRM assistant. אין דעם פראיעקט זענען דא אגענטן און א רשימה פון טאסקס.

- מיט דעם agent decorator ווערט דעפינירט א פונקציע וואס הייסט researcher. זי שטעלט צונויף דעם אגענט און גיט אים צוריק. דער Agent קלאס קומט פונעם CrewAI פריימווערק. ער באקומט זיין קאנפיגוראציע פון config researcher, און verbose ווערט געשטעלט אויף True.
- פונקט אזוי ווערט דעפינירט דער צווייטער אגענט, reporting analyst.

דער קורס בעט אויפצופאסן אויף די אלע ענינים וואס ווערן באנוצט אין דעם פראיעקט, און ערקלערט אז דאס איז א גאר יסודותדיגער פראיעקט. אזוי ווערן אין crew.py דעפינירט אלע אגענטן פונעם crew, און יעדער פון זיי גיט צוריק זיין קאנפיגוראציע.

- מיט א task decorator ווערן דעפינירט די טאסקס. למשל, research task גיט צוריק דעם טאסק וואס קומט פון דער tasks.yaml טעקע. אזוי איז עס אויך מיט די איבעריגע טאסקס.
- צום סוף ווערט באשאפן דער CRM assistant crew, וואס דארף אויספירן אלע די טאסקס. ער בינדט זיך צו דעם crew פריימווערק.

די main טעקע: run, train, replay, test און trigger (27:50:35)

די main טעקע איז די טעקע וואס לויפט דעם אגענט. מען גיט איר inputs:

- א topic, אין דעם ביישפיל AI LLMs.
- דאס יעצטיגע יאר, וואס ווערט גענומען פון דער יעצטיגער דאטום און צייט.

אין דעם קאוד ווערט באנוצט exception handling. דאס הייסט מען שרייבט try און except, אזוי אז אויב עס פאסירט עפעס אומנארמאלעס, ווערט עס אפגעהאנדלט, און דער קאוד בלייבט זיכער. ווייטער טוט די טעקע די פאלגנדע זאכן:

- train: מען טריינירט דעם crew פאר א געוויסע צאל איטעראציעס, מיט די inputs: topic, וואס איז AI LLMs, און די יעצטיגע צייט.
- replay: מען שפילט נאכאמאל איבער די אויספירונג פונעם crew.
- test: מען טעסט דעם crew מיט די זעלבע inputs.
- run with trigger: מען לויפט דעם פראיעקט מיט א trigger payload. אויב די לענג איז ווייניגער ווי צוויי, ווארפט מען אן exception.

דער אפיציעלער פראיעקט-סטרוקטור לויט די דאקומענטאציע (27:51:37)

נאכדעם וועגדט מען זיך צו דער אפיציעלער דאקומענטאציע. דארט ווערן ערקלערט די טעקעס פונעם פראיעקט:

- agents.yaml: וואו אלע AI אגענטן ווערן דעפינירט.
- tasks.yaml: וואו מען שטעלט אויף די טאסקס און די ארבעט פון די אגענטן.
- די .env טעקע: וואו מען קען האלטן די API שליסלען און די environment variables. עס ווערט באמערקט אז אין דער לעקציע איז דער API שליסל אריינגעלייגט געווארן גלייך אין קאוד. אבער ער ווערט רעקאמענדירט צו האלטן די שליסלען אין environment variables, ווייל דאס איז א זיכערע פראקטיק פאר א production סיסטעם.
- crew.py: די orchestration און קאארדינאציע פונעם crew.

- א tools טעקע: פאר די custom tools פון די אגענטן.
- א knowledge טעקע: פאר א knowledge base, א וויסן-באזע.

אינסטאלירן און לויפן דעם פראיעקט (27:52:39)

וועגן דער פראגע ווי אזוי לויפט מען דעם פראיעקט, ווייזט די דאקומענטאציע אז מען דארף ערשט לויפן דעם באפעל:

```
crewai install
```

אויב מען דארף נאך פעקלעך, לייגט מען זיי צו מיט uv add, מיט דעם נאמען פונעם פעקל. דערנאך לויפט מען דעם פראיעקט מיט:

```
crewai run
```

וען מען האט געלאפן crewai install, האט מען געזען א מעלדונג אז קיין פראיעקט-טעקע ווערט נישט געפונען. עס ווערט ערקלערט אז דאס איז ווייל מען האט נישט צוגעלייגט קיין דעפענדענסי-טעקע. מען גייט פשוט ווייטער. נאך crewai run האט דער אגענט ענדליך געלאפן.

עס ווערט דערציילט אז עס זענען געווען עטליכע פראבלעמען, ווייל מען האט באנוצט Python 3.12. דער טערמינאל האט געוויזן א טעות, און מען האט געדארפט צולייגן דעם פעקל lancedb. וען מען האט אים צוגעלייגט, האט זיך דער פראבלעם געלייזט.

א ווארענונג: מען מוז זיין אינעווייניג אין דער טעקע פונעם פראיעקט. אויב מען שטייט אין דער היים-טעקע און לויפט install און run, ווערט עס נישט ארבעטן.

דער רעזולטאט פונעם אגענט (27:54:41)

אין דער אויספירונג זעט מען אז די execution הייבט זיך אן, מיט דעם נאמען פונעם crew און זיין ID. דערנאך הייבט זיך אן דער טאסק research task, מיט זיין ID. דער אגענט, א "senior data researcher", האט די אויפגאבע צו מאכן א גרינטליכע פארשונג. מען זעט אויך זיין final answer, דעם לעצטן ענטפער. ער באקומט דעם ענטפער דורכן פרעגן OpenAI'ס GPT.

דער קורס רופט עס אן אן אויסגעצייכנטן פריימווערק. ווער עס איז אנגעקומען צו דעם שטאפל, האט מצליח געווען צו בויען זיין ערשטן אגענט.

אן איבונג פאר'ן לערנער (27:55:12)

מען בעט אליין דורכצומאכן די איבונג פון דער ערשטער בלאט אין דעם quickstart טייל פון דער דאקומענטאציע: "Build your first CrewAI agent". מען דארף טון די זעלבע זאכן וואס מען האט געטון אין דער לעקציע, מיט עטליכע צוגאבן:

- שרייבן די roles פאר וועלכע מען שאפט די אגענטן, זיי ענדערן און זיך שפילן מיט זיי.
- צולייגן טאסקס אין tasks.yaml און עקספערימענטירן.
- דערנאך שרייבן די פאסיגע decorators.

ווער עס קען נישט decorators, ווערט גראטן צו קוקן דעם Python קורס פונעם קאנאל, וואו מען וועט באקומען א קורצן באגריף.

ווי אזוי צו דיזיינען ווירקזאמע אגענטן (27:56:13)

א גרויסע פראגע איז ווי אזוי מען דיזיינט ווירקזאמע אגענטן. מען בעט צו קוקן אין דער אפיציעלער דאקומענטאציע, ווייל דארט געפינט מען רוב פון די געדאנקען און פראקטיקן. אבער פריער דארף מען האבן א גוטן יסוד אין Python. וועט עס איז נישט באהאוונט אין Python , וועט עס אים קומען שווער צו באנוצן CrewAI, ווייל מען דארף פארשטיין ווי אזוי מען מאכט execution flows, שרייבט conditionals , לייגט צו decorators, קאנפיגוראציע-טעקעס, און אזוי ווייטער.

פארוואס איז דער דיזיין פון אן אגענט וויכטיג? עס ווערן געברענגט די פאלגנדע פונקטן:

- די קוואליטעט פונעם רעזולטאט: ביי א וויכטיגן טאסק איז די קוואליטעט פון דעם אויספיר א גרויסער ענין.
- די ווירקזאמקייט פון צוזאמענארבעט: ווען מען שאפט עטליכע אגענטן, וועלן זיי אפט מוזן צוזאמענארבעטן, און דער דיזיין מוז דאס מאכן ווירקזאם.
- די פערפארמענס: ווי גוט דער אגענט גיט ארויס דעם רעזולטאט.
- סקאלירבארקייט: ביי enterprise לייזונגען מיט א גרויסן קונדן-באזע מוז מען אויך טראכטן וועגן scalability.

דער 80/20 כלל: role, goal, backstory (27:57:44)

דער 80/20 כלל זאגט: 80 פראצענט פון דער מי זאל גיין אויף דיזיינען דעם טאסק, און 20 פראצענט אויף דעפינירן די אגענטן.

אין agents.yaml אין דער config טעקע דארף מען קלאר שרייבן דעם role, דעם goal און דעם backstory פון יעדן אגענט. די קאנפיגוראציע מוז שטימען מיט דעם טאסק וואס מען גיט דעם אגענט. די עצות זענען:

- זיין ספעציפיש.
- זיך צופאסן צו דעם אמת'ן פראבלעם אויף וועלכן מען קאנצענטרירט זיך.
- באשטימען ווי פונקטליך דער אגענט דארף אויספירן דעם טאסק.
- אריינלייגן domain expertise, א פאכמאנישקייט אין דעם געביט. עס ווערט באמערקט אז אין דעם קאוד פון דער לעקציע זענען צוגעלייגט געווארן אזעלכע פאכלייט, וואס יעדער טוט זיין טאסק.

ביישפילן פון ווירקזאמע roles פון דער דאקומענטאציע: "senior UX researcher", "full-stack software architect", "corporate communication director".

- goal: מען דארף געבן א שארפע, צום פונקט, און פונדעסטוועגן אויסגעברייטערטע באשרייבונג פונעם ציל פונעם אגענט.
- backstory: ווי אזוי דער אגענט ארבעט אין זיין געביט, וואס איז זיין עקספערטיז, און וואס איז זיין פערספעקטיוו.

די דריי פאראמעטערס זענען גאר וויכטיג ביים דיזיינען אן ווירקזאמען AI אגענט.

אריינפיר צו טאסקס (27:59:48)

צום סוף הייבט מען אן צו ערקלערן וואס זענען טאסקס און ווי אזוי ארבעט מען מיט זיי. אין דער דאקומענטאציע, אונטער דעם core concepts טאב, געפינט מען dem טייל tasks. אין דעם CrewAI פריימווערק שטעלט א טאסק פאר א באשטימטע איינהייט ארבעט וואס אן אגענט דארף אויספירן.

CrewAI: טאסקס, crew.py און Flows (28:00:20)

וואס איז א טאסק אין CrewAI (28:00:20)

דער קורס ערקלערט אז א טאסק קען זיין פארשידן שוועריקייט: פון א פשוטע אויפגאבע ביז א קאמפליצירטע ארבעט וואו עטליכע אגענטן דארפן צוזאמענארבעטן. א טאסק איז א אויפגאבע וואס א געוויסער אגענט פירט אויס.

אינעם פראיעקט האט מען שוין געשריבן א tasks.yaml פייל, וואו מען האט דעפינירט די טאסקס. יעדער טאסק האט א ענויע באשרייבונג: די description, דער expected output, וועלכער אגענט איז פאראנטווארטליך דערפאר, און וועלכע געצייג (tools) מען דארף.

אין דער לעקציע ווערט געוויזן די קאנפיגוראציע-פייל. דארט זעט מען למשל א research task: מען שרייבט א באשרייבונג, וואס מען ערווארט אלס רעזולטאט, און וועלכער אגענט פירט עס אויס. דאס זענען די הויפט-פאראמעטערס.

צוויי סארטן טאסק-אויספירונג (28:01:21)

- Sequential: די טאסקס ווערן אויסגעפירט איינס נאכן אנדערן, אין א קלאר באשטימטן סדר.
- Hierarchical: די טאסקס ווערן צוגעטיילט צו די אגענטן לויט זייער ראלע און עקספערטיז.

אלס ביישפיל ברענגט דער קורס: פארשן א געוויסע טעמע מיט א סעט געצייג און אגענטן, למשל זאמלען אינפארמאציע ווען AI-אנטוויקלונגען פאר 2026.

די אטריבוטן פון א טאסק (28:02:00)

דער קורס רעכנט אויס די אטריבוטן וואס מען קען צולייגן צו א טאסק:

- description און expected output, וואס מען האט שוין צוגעלייגט אין דער YAML פייל.
- name, agent און tools, אלע אפציאנאל.
- context, אויך אפציאנאל.
- async execution: צי דער טאסק זאל לויפן אסינכראניש אדער נישט.

אויב מען ווייסט נישט וואס איז אן אסינכראנישער פראצעס, ווערט געגעבן א ביישפיל: ווען מען ברענגט דאטא פון אן API, שיקט מען א בקשה צום סערווער, און עס איז פאראן א וואַרט-צייט. די זאכן פאסירן נישט איינס נאכן אנדערן אן אפשטעל, מען דארף וואַרטן. מען רעכנט עס פאר אן אסינכראנישן פראצעס. וועלן דער רעזולטאט פון דעם API-ענטפער ווערט באניצט אלס אינפוט פאר א צווייטן טאסק, און עס פארמירט זיך א קייט פון אזעלכע שריט, באלאנגט עס אויך צום async פראצעס. דערפאר קען מען צולייגן דעם async execution פאראמעטער.

ווייטער קען מען צולייגן:

- human input, א פאראמעטער פאר אינפוט פון א מענטש.
- markdown, אפציאנאל.
- config, ווען דער טאסק דארף ספעציפישע קאנפיגוראציע-פאראמעטערס.
- output file: דער פייל-וועג וואו דער רעזולטאט ווערט אפגעהיטן.
- output json: דאס באשטימט ווי דער רעזולטאט זאל אויסזען, למשל אין JSON פארמאט. מען גיט דא א Pydantic מאדעל, וואס גיט א סטרוקטור פארן JSON רעזולטאט.
- guardrail: א פונקציע וואס פרובירט אויס דעם רעזולטאט איידער מען גייט ווייטער צום קומענדיגן טאסק.
- guardrail max retries: ווי פיל מאל מען מעג מאקסימום נאכאמאל פרובירן, אויב די guardrail-פרובע פעלט דורך.

ווי אזוי מען שאפט א טאסק און דער crew.py פייל (28:04:27)

מען האט שוין געזען ווי אזוי א טאסק ווערט דעפינירט, און דער קורס רעקאמענדירט צו לייענען די YAML-קאנפיגוראציעס, ווייל דאס איז זייער נוצליך.

כדי צו באניצן די YAML-קאנפיגוראציע אינעם קאוד, דארף מען שאפן א CrewBase דעקאראטאר. דאס קומט אריין אין דער crew.py פייל, וואס איז שוין אויטאמאטיש געשאפן געווארן אינעם פראיעקט.

די שריט:

- מען שטעלט אוועק דעם @CrewBase דעקאראטאר אויף דעם קלאס, למשל א קלאס פאר "latest AI developments".
- מען דעפינירט דעם אגענט וואס וועט ארבעטן: מען שאפט א מעטאד, מען באניצט דארט א lambda expression, מען באשטימט עס פאר אן אגענט, און מען גיט אים זיין קאנפיגוראציע.
- פונקט אזוי פאר א צווייטן אגענט, למשל א reporting analyst: מען דעפינירט דעם דעקאראטאר און זיין קאנפיגוראציע.
- אויף דעם זעלבן אופן דעפינירט מען יעדן טאסק מיט זיין קאנפיגוראציע.
- צום סוף לייגט מען צו א @crew דעקאראטאר. דארט שטעלט מען אריין אלע אגענטן וואס מען האט דעפינירט, למשל researcher און reporting analyst, און מען רופט אויף די טאסק-פונקציעס.
- דארט באשטימט מען אויך צי די טאסקס זאלן לויפן sequential אדער hierarchical. מען דעפינירט א ווערייעבל process און מען שטעלט אריין:

process=Process.sequential

אויף אזא אופן בויט מען אויס דעם crew.py פייל.

Flows: דער באשלוס-טייל אין CrewAI (28:07:00)

דער קורס דערמאנט אז crew flows זענען דער טייל וואס מאכט די באשלוסן ווען מען בויט א מולטי-אגענט סיסטעם מיט CrewAI . דאס איז איינע פון די וויכטיגסטע פיטשערס, ווייל עס מאכט גרינגער צו שאפן און פירן AI-ארבעטס-פלוסן (workflows).

וען מען בויט אן event-driven ארכיטעקטור, אדער מען לייזט א פראבלעם וואס איז געבויט אויף געשעענישן, דארף מען אריינברענגען flows אין דער ארכיטעקטור. עס מאכט פשוט'ער דעם אופן ווי די באשלוסן ווערן געמאכט.

דער קורס דערמאנט דעם e-commerce ביישפיל פון פריער: א באניצער זוכט א פראדוקט אין א געוויסער קאטעגאריע, למשל שיך, למשל Nike שיך. אויב דער פראדוקט איז נישטא, דארף דער פלוס זיך אומדרייען אויף אן אנדער וועג. מען דארף האבן א פלוס וואס צינדט אן די געשעענישן: אויב מען גייט אויפן וועג A, איז דאס דער רעזולטאט; אויב נישט A, גייט דער פראגראם אויף אן אנדער וועג.

flows זענען ענליך צו די תנאי-צווייגן אין Python, דאס הייסט if/else, נאר דא זענען זיי ספעציעל געמאכט פאר אגענטן.

פארוואס דארף מען עס:

- עס מאכט פשוט'ער צו שאפן א workflow.
- מען האט ריכטיגע state management, דאס הייסט פארוואלטונג פונעם צושטאנד.
- מען האט א פלעקסיבלע, event-driven ארכיטעקטור.
- מען האט א פלעקסיבלע קאנטראל-פלוס: מען קען אריינלייגן לאגיק, לופס און צווייגן אין דעם workflow.

א ביישפיל-פלוס אין קאוד (28:09:02)

אין דער לעקציע ווערט געוויזן א ביישפיל:

- מען אימפארטירט פון crewai.flow די זאכן Flow, listen און start.
- אויב מען באניצט environment variables, אימפארטירט מען load_dotenv פון dotenv.
- פון דער LiteLLM ביבליאטעק אימפארטירט מען completion.
- מען דעפינירט א קלאס ExampleFlow וואס ירשנט פון Flow. דער מאדעל וואס מען באניצט איז GPT-4o mini.
- מען לייגט א @start דעקאראטאר אויף א פונקציע generate_city, וואס איז געברענגט אין דער דאקומענטאציע. וועז מען הייבט אן א פלוס, באקומט דער state פון יעדן פלוס אויטאמאטיש זיין אייגענעם unique id.
- מען שאפט א response וועריעבל וואס רופט completion. מען גיט איר דעם מאדעל (GPT-4o mini) און א מעסעדזש אין JSON פארמאט מיט role און content, למשל: "Return the name of a random city in the world."
- דעם ענטפער שטעלט מען איין אין א וועריעבל random_city: מען נעמט פון choices דעם message content.
- מען היט אפ די שטאט אינעם state, און מען גיט צוריק random_city.
- דערנאך קומט א צווייטע פונקציע וואס הערט זיך צו generate_city . דארט רופט מען ווידער דעם self.model, מיט role user און content: "Tell me a fun fact about" די שטאט. דעם ענטפער היט מען אויך אפ אינעם state.

- צום סוף דעקלארירט מען א flow ווערייעבל פון ExampleFlow, און מען רופט plot. דאן ווערט אויסגעשטעלט א בילד פונעם פלוס: generate city, דערנאך generate random fact about the city.

אזוי אזוי פלאנירט מען א פלוס ביים בויען אגענטן. דאס איז זייער וויכטיג אויב מען וויל קאנטראלירן א געוויסן אנווענדונג-פאל, אדער ביים בויען א מולטי-אגענט סיסטעם ווי דער e-commerce ביישפיל. מען דארף גוט פארשטיין ווי דער פלוס גייט אין פראגראם.

אנווייזונגען צום סוף פון דעם CrewAI טייל (28:12:09)

דערמיט ענדיגט זיך דער טייל ווען CrewAI. דער קורס גיט עטליכע עצות:

- מען דארף האבן Python 3.x אינסטאלירט; עס ווערט אנגעזאגט צו דאונלאודן די לעצטע ווערסיע, ווייל א געוויסע ווערסיע מאכט פראבלעמען.
- מען מוז האבן אן OpenAI API key. אן דעם קען מען נישט ארבעטן מיט CrewAI. די OpenAI API keys קאסטן א ביסל געלט, אבער עס איז א זייער שטארקע זאך פארן בויען מולטי-אגענט סיסטעמען.

דערנאך גייט דער קורס איבער צו דער קומענדיגער טעמע: וואס איז LangGraph און פארוואס עס איז וויכטיג אין דער וועלט פון AI-אנטוויקלונג.

LangGraph: זכרון, דיבאגינג און דער ערשטער פראיעקט (28:13:10)

דער פראבלעם פון פארגעסנדיגע טשעטבאטס און וואס LangGraph לייזט (28:13:10)

דער קורס הייבט אן מיט א בילד וואס יעדער קען: מען רעדט מיט א טשעטבאט, מען פרעגט עפעס וועגן זיין טאג אדער מען זוכט הילף. דער טשעטבאט ענטפערט שנעל, אבער פלוצלונג פארגעסט ער וואס מען האט אים פונקט געזאגט. עס ווערט ערקלערט אז דאס איז א געוועליכער פראבלעם: טראדיציאנעלע טשעטבאטס אדער אסיסטענטן ענטפערן אויף איין איינציגע פראגע, אבער זיי קענען נישט האלטן דעם קאנטעקסט איבער עטליכע שמועסן נאכאנאנד. דאס פירט צו צעמישטע, צעריסענע שמועסן וואס פארדריסן די באניצער.

דא קומט אריין LangGraph. דאס איז א "פרעימווארק" וואס איז געבויט כדי צו געבן אן AI סיסטעם א זכרון. עס איז אויפגעבויט ארום "סטעיטפול" ארבעטס-פלוסן, ד.ה. עס העלפט דעם AI געדענקען פריערדיגע שמועסן און צופאסן די ענטפערס לויט יענע אינפארמאציע.

דער קורס ניצט א משל פון א GPS: LangGraph העלפט דעם AI געדענקען דעם וועג וואס ער איז שוין געפארן, אזוי אז ער קען אויסגלייכן זיינע קומענדיגע טריט לויט דעם וואס איז פריער פארגעקומען אין שמועס.

פארוואס איז דאס וויכטיג? עס ווארן געברענגט ביישפילן פון דער פראקטיק:

- אן HR אסיסטענט וואס געדענקט די פרטים פון א קאנדידאט דורכאויס דעם גאנצן אינטערוויו-פראצעס.
- א רייזע-פלאנער וואס געדענקט אייערע פרעפערענצן, פריערדיגע רייזעס און בודזשעט, כדי צו געבן בעסערע פארשלאגן.

דאס זענען נאר צוויי ביישפילן ווי די מעגליכקייט פון LangGraph נאכצוגיין שמועסן און זיך צופאסן איז הויפטזעכליך.

LangGraph קעגן LangChain (28:14:10)

דער קורס דערמאנט קורץ LangChain, ווייל דאס איז אן אנדער פאפולערער פרעימווארק אין AI . אבער LangGraph איז אנדערש. LangGraph דרייט זיך ארום סטעיטפול אינטעראקציעס, ווי א מאפע וואס פאלגט נאך דעם גאנצן וועג. LangChain, פון דער אנדערער זייט, איז מער ווי א סעט מכשירים וואס קייטלט צוזאם פארשידענע AI מאדעלן. עס איז גוט פאר פשוטע אויפגאבן, אבער LangGraph גיט מער קאנטראל איבער ארבעטס-פלוסן מיט עטליכע טריט און איבער קאמפליצירטע שמועסן.

דיבאגינג פון LangGraph אפליקאציעס (28:14:40)

וועו מען גייט טיפער אריין אין LangGraph, וועט מען זיכער טרעפן שוועריגקייטן און פעלערן. דער קורס ערקלערט אז דיבאגינג איז א וויכטיגער טייל פון וווייכווארג-אנטוויקלונג, און א געשיקטקייט וואס העלפט פארשטיין ווי אפליקאציעס ארבעטן, געפינען וואו עס גייט קאליע, און טרעפן לייזונגען. אין דעם קאפיטל ווערן באהאנדלט געוועניליכע פעלערן אין LangGraph, ווי אזוי זיי צו לייזן, און די בעסטע מעטאדן צו דיבאגן ארבעטס-פלוסן.

דעפיניציע: דיבאגינג איז דער פראצעס פון אויסגעפינען פארוואס עפעס ארבעט נישט ווי געריכט, און עס פארריכטן. דער משל איז א דעטעקטיוו-ארבעט: פונקט ווי א דעטעקטיוו זוכט סימנים צו לייזן א רעטעניש, דארף מען נאכגיין דעם פלוס פון דער דאטא, פונקטליך באשטימען וואו עס איז קאליע געווארן, און אויסטראכטן ווי אזוי עס צו פארריכטן. דער פראצעס העלפט פארשטיין ווי די אפליקאציע פירט זיך, געפינען "באגס", און פארבעסערן די פערפארמענס.

ביי ארבעטן מיט LangGraph קען זיך מאכן אז דער AI ענטפערט נישט ווי געריכט, אדער א "נאוד" (קנופ אין גראף) ווערט נישט ריכטיג אויסגעלייזט. דעמאלט ווערט דיבאגינג הויפטזעכליך. LangGraph גיט פארשידענע מכשירים און סטראטעגיעס וואס מאכן דאס גרינגער. (אין דער לעקציע ווערט געזאגט "דא זענען די געוועניליכע פעלערן", אבער אין דעם טייל ווערן זיי נישט באזונדער אויסגערעכנט.)

וויזואליזאציע מיט LangSmith (28:15:41)

וועו די LangGraph ארבעטס-פלוסן ווערן קאמפליצירטער, ווערט אלץ וויכטיגער צו זען ווי די דאטא פליסט דורך דעם גראף. וויזואליזאציע לאזט זען ווי די נאודס זענען פארבונדן, ווי די דאטא רירט זיך צווישן זיי, און וואו עס קען זיין א פראבלעם.

LangGraph איז פאראייניגט מיט LangSmith, א מכשיר וואס העלפט וויזואליזירן די ארבעטס-פלוסן און נאכגיין ("טרעיסן") ווי די דאטא גייט דורך דעם גראף. די גראפישע פארשטעלונג מאכט גרינגער צו כאפן פראבלעמען און צו פארשטיין פארוואס געוויסע נאודס ווערן נישט אויסגעפירט ווי געריכט. ווי דאס העלפט:

- וועו מען לויפט די LangGraph אפליקאציע מיט LangSmith, זעט מען יעדן נאוד און זיין "סטעיט" ביי יעדן טריט, און מען קען נאכגיין פונקטליך וועלכע דאטא ווערט פראצעסירט פון איין נאוד צום צווייטן.
- אויב א נאוד ארבעט נישט ריכטיג, קען מען דרוקן דערויף אין דער LangSmith וויזואליזאציע און זען פארוואס – למשל, צי דער פריערדיגער נאוד האט נישט דורכגעגעבן די ריכטיגע דאטא. דאס העלפט שנעל געפינען וואו דער פראבלעם ליגט.
- LangSmith ווייזט די היסטאריע פון סטעיט-ענדערונגען בשעת דער גראף לויפט, כדי צו באשטימען וועו און וואו דער סטעיט ווערט געענדערט, און צי עס זענען דא אומגעריכטע אויפפירונגען אויפן וועג.

בעסטע מעטאדן פאר טראבלשוטינג (28:16:42)

דיבאגינג קען זיין א שווערigקייט, אבער עס זענען דא עטליכע עצות וואס מאכן עס גרינגער און ווירקזאמער:

- אנהייבן קליין און ארויפבויען: ביים בויען א גראף ארבעטס-פלוס, הייבט מען בעסער אן מיט א פשוטן, מינימאלן גראף. מען קאנצענטרירט זיך צו מאכן איין נאוד אדער איין אויפגאבע ארבעטן פערפעקט, איידער מען לייגט צו מער קאמפליצירטקייט. נאכדעם וואס מען האט א באזיס-גראף, לייגט מען צו ביסלעכווייז מער מכשירים און פארבינדונגען. אזוי פארמינערט מען די שאנסן צו טרעפן איבערוועלטיגנדע פראבלעמען וואס זענען שווער צו דיבאגן.
- ניצן "יוניט טעסטינג": א טעכניק וואו מען טעסט איינצלנע טיילן פון קאוד באזונדער. ד.ה. מען טעסט יעדן נאוד אליין צו זען אז ער פירט זיך ווי געריכט, איידער מען פארבינדט אים מיט אן אנדערן נאוד. דאס מאכט גרינגער צו געפינען וואו עס גייט קאליע, און מען כאפט פעלערן פרי.
- צעטיילן קאמפליצירטע גראפן אין "סאב-גראפן": וועו דער ארבעטס-פלוס וואקסט, ווערט שווער נאכצופאלגן אלעס. יעדער סאב-גראף זאל פארשטעלן א קלענערן, באהערשבארן טייל פונעם פראיעקט. די מאדולארע צוגאנג לאזט דיבאגן ספעציפישע טיילן אן דורכגיין דעם גאנצן ארבעטס-פלוס. ביישפיל: ביים בויען א מערסטעפיגן אסיסטענט קען מען עס צעטיילן אין סאב-גראפן ווי באניצער-אויטענטיפיקאציע, באניצער-פרעפערענצן, און באניצער-פראגעס, און זיך קאנצענטרירן אויף איין טייל אויף אמאל.
- ניצן לאגס צו נאכגיין דאטא: לאגינג איז "דער בעסטער פריינט" ביים זוכן פראבלעמען אין גראף. דורך צולייגן לאגינג-אויסזאגן אין יעדן נאוד קען מען נאכגיין ווי די דאטא רירט זיך און וואס עס טוט זיך ביי יעדן טריט, און שנעל זען וואו עס גייט קאליע.

צום סוף ווערט באטאנט אז דיבאגינג פון LangGraph אפליקאציעס דארף נישט זיין שרעקלעך. דורך פארשטיין די געוועלטיכע פעלערן, ניצן מכשירים ווי LangSmith פאר וויזואליזאציע, און פאלגן די עצות ווי אנהייבן קליין און ניצן יוניט טעסטס, קען מען טראבלשוטן עפעקטיוו. דיבאגינג איז נישט נאר געפינען טעותים; עס איז פארבעסערן דעם גראף, אפטימירן די פערפארמענס און זיכער מאכן אז דער AI ארבעט ווי געריכט. וואס מער מען פראקטיצירט, אלץ שנעלער געפינט און לייזט מען פראבלעמען. מען זאל ווייטער עקספערימענטירן, טעסטן און פארפיינערן די פראיעקטן, און אזוי בויען קאמפליצירטערע, פארלעסליכע AI סיסטעמען וואס געדענקען קאנטעקסט און רעדן קלוג מיט באניצער.

אויפשטעלן דעם ערשטן LangGraph פראיעקט: Python און ווירטועלע סביבה (28:19:17)

אין דעם טייל ווערט דורכגעגאנגען די אינסטאלאציע, דאס אויפשטעלן די סביבה, און שאפן דעם ערשטן פראיעקט.

ערשטער טריט – אינסטאלירן Python און שאפן א ווירטועלע סביבה:

- מען דארף האבן Python 3.10 אדער העכער. אויב נישט, גייט מען צום אפיציעלן Python ווועבזייטל און מען לאדנט אראפ די לעצטע ווערסיע.
- מען דארף אויך אינסטאלירן PyCharm, וואס ווערט באצייכנט אלס איינס פון די בעסטע IDE פאר Python. ווער עס איז באקוועם מיט אן אנדער IDE מעג עס ניצן, אבער PyCharm מאכט דאס אויפשטעלן גרינגער.
- נאכן אינסטאלירן Python עפנט מען די קאמאנד-ליין און מען שאפט א נייעם דירעקטארי (פאלדער). אין דער לעקציע זעט מען אז דער דירעקטארי ווערט באשאפן.

• דערנאך שאפט מען א ווירטועלע סביבה. דער משל: א ווירטועלע סביבה איז ווי א "זאמדקאסטן" וואו מען קען אינסטאלירן Python ביבליאטעקן ספעציפיש פארן פראיעקט, אן אנרירן דעם רעשט פון סיסטעם.

• דערנאך אקטיווירט מען די סביבה. אויף Windows ניצט מען איין באפעל, און אויף Mac OS אדער Linux אן אנדערן (די באפעלן ווערן געוויזן אויפן עקראן). וועז די סביבה איז אקטיוו, זעט מען דעם נאמען פון דער סביבה (עפעס ווי langgraph_env) פאר דער קאמאנד-פראמפט, וואס צייגט אן אז מען ארבעט אינערהאלב דער ווירטועלער סביבה.

אינסטאלירן די אפהענגיגקייטן (28:20:18)

צווייטער טריט: LangGraph פארלאזט זיך אויף עטליכע דרויסנדיגע ביבליאטעקן, ווי LangChain און OpenAI API . מען לויפט די באפעלן אין טערמינאל כדי זיי צו אינסטאלירן, און מען זעט עס אין דער דעמא. די OpenAI API איז נויטיג כדי צו גענערירן AI ענטפערס, וואס מען ווען ניצן צו שאפן א דינאמישן שמועס-סיסטעם.

דער ערשטער גראף: "Hello World" (28:20:48)

דריטער טריט: נאכדעם וואס די סביבה איז אויפגעשטעלט, שאפט מען דעם ערשטן פראיעקט – א פשוטער "Hello World" גראף. מען שאפט א פייל hello_world.py אינעם פראיעקט-דירעקטארי, וואס וועט אנטהאלטן דעם גאנצן קאוד, און מען לייגט אריין דעם קאוד. אין סוף ווערט דער רעזולטאט געדרוקט, און פריער ווערט דער גראף "קאמפיילט" און "אינוואוקט". דער קורס ערקלערט די טיילן פונעם קאוד:

• די State קלאס: זי באשטימט אז דער סטעיט מוז אנטהאלטן א שליסל message פון טיפ str. דאס זיכערט אז דער סטעיט וואס ווערט ארומגעשיקט אינעם גראף פאלגט שטענדיג די סטרוקטור.

• די נאוד-פונקציע: די פונקציע hello_world נעמט דעם סטעיט אלס אינפוט און גיט צוריק א נייעם סטעיט וואו message איז געטוישט צו "Hello World". די פונקציע באשטימט ווי דער נאוד פירט זיך.

• שאפן דעם גראף: א StateGraph אביעקט ווערט באשאפן מיט דער State קלאס אלס מוסטער. דאס באשטימט די סטרוקטור פונעם גראף און ווי דער סטעיט פליסט דורך די נאודס.

• צולייגן נאודס און "עדזשעס" (פארבינדונגען): די פונקציע hello_world ווערט צוגעטיילט צום נאוד "hello". די עדזשעס פארבינדן: פון START צו "hello", און פון "hello" צו END – אזוי ווערט באשטימט דער סדר פון אויספירונג.

• קאמפיילן: די compile מעטאד גרייט צו דעם גראף, זיכערט אז אלע נאודס און עדזשעס זענען ריכטיג אויפגעשטעלט, און שאפט אן אויספירבארן גראף-אביעקט.

• אינוואוקן: די invoke מעטאד ווערט געופן אויפן קאמפיילטן אביעקט מיט אן אנהייב-סטעיט וואו message איז ליידיג. דער גראף לויפט, דער hello נאוד דערהיַנטיקט דעם סטעיט, און דער רעזולטאט ווערט געדרוקט.

וועז מען לויפט דעם פייל, זעט מען דעם אויסגאנג "Hello World " – דאס איז דער ערשטער ארבעטנדיגער LangGraph.

בויען אן AI אסיסטענט וואס געדענקט דעם נאמען (28:22:50)

ווייטער בויט מען א פשוטן רעדנדיגן אסיסטענט וואס געדענקט דעם באניצער'ס נאמען און באגריסט אים. דער קורס באטאנט אז די הויפט-אייגנשאפט פון יעדן קלוגן אסיסטענט איז זכרון: ער דארף געדענקען דעם נאמען און זיך צוריקרופן דערצו אינמיטן שמועס. אין LangGraph ווערט זכרון באהאנדלט דורך דעם סטעיט.

- ערשטער טריט: מען ענדערט די hello_world פונקציע אז זי זאל פרעגן "What is your name?" און צוריקגעבן דעם נאמען אינעם סטעיט אונטערן שליסל user_name.
- צווייטער טריט: מען שאפט א צווייטע פונקציע וואס ניצט דעם געהאלטענעם נאמען פונעם סטעיט צו באגריסן דעם באניצער ביים נאמען.
- דריטער טריט: מען פארבינדט די צוויי פונקציעס אין איין ארבעטס-פלוס. LangGraph מאכט גרינג צו פארבינדן נאודס דורך זיי לויפן איינס נאכן אנדערן. אין דער לעקציע ווערט אריינגעשטעלט א ריינערער קאוד אנשטאט דעם פריערדיגן.

מען לויפט דעם קאוד: ער פרעגט "What is your name?", מען טייפט "Alice", און דער אויסגאנג איז "Hello Alice". אזוי פרעגט דער אסיסטענט דעם נאמען, האלט אים, און באגריסט. עס איז פשוט, אבער דאס איז דער יסוד פון יעדן שמועס-AI סיסטעם וואס געדענקט קאנטעקסט.

צום קומענדיגן טריט: לאגיק און באדינגונגען (28:24:22)

כדי צו מאכן דעם אסיסטענט קליגער, דארף מען צולייגן סטעיטפול לאגיק און באדינגטע שפרונגען (conditional jumps). די באגריפן לאזן דעם אסיסטענט נעמען באשלוסן לויט דעם באניצער'ס אינפוט און האלטן זכרון איבער פארשידענע שטאפלען פונעם שמועס.

לאגיק, געצייג, טעותים און דער אנהייב פון Autogen (28:24:53)

באדינגונגס-לאגיק פאר באשלוסן (28:24:53)

דער ערשטער שריט איז באדינגונגס-לאגיק (conditional logic) פאר באשלוסן. דער קורס ברענגט א ביישפיל: דער אסיסטענט דארף פרעגן דעם באניצער צי ער וויל ממשיך זיין דעם שמועס אדער נישט. לויט דעם באניצער'ס ענטפער דארף דער אסיסטענט אדער ווייטער שמועסן אדער ענדיגן דעם שמועס. דא קומט אריין די באדינגונגס-לאגיק.

מען שאפט א פונקציע וואס פרעגט דעם באניצער צי ער וויל ווייטער גיין. אינעם קאוד ווערט אריינגעשריבן די פראגע "do you want continue the conversation", און אין קלאמערן "yes or no". אויב דער באניצער שרייבט yes, גיט די פונקציע צוריק true, אנדערש גיט זי צוריק false.

אין דער דעמא ווערט אריינגעגעבן דער נאמען Alice. דערנאך ווייזט זיך די פראגע צי מען וויל ממשיך זיין דעם שמועס, yes אדער no. וועט מען קלייבט yes, הייבט זיך ווידער אן דער זעלבער שמועס. וועט מען שרייבט no, ענטפערט די פראגראם "goodbye Alice see you later". דאס באשטעטיגט אז ביי yes קומט צוריק true, און אנדערש false. די סדר איז: מען שרייבט Alice, די פראגראם זאגט "hello Alice" און פרעגט צי מען וויל ווייטער גיין; ביי no זאגט זי "goodbye Alice, see you later".

באהאנדלען דעם באשלוס און פארבינדן אלעס (28:25:55)

דער צווייטער שריט איז באהאנדלען דעם באשלוס. מען שאפט א פונקציע וואס מאכט א באשלוס לויט וואס דער באניצער האט אריינגעגעבן. אויב דער באניצער וויל ווייטער גיין, באגריסט אים דער אסיסטענט נאכאמאל; אנדערש זאגט ער "goodbye". דער קורס באמערקט אז דער קאוד פון דעם שריט דארף ארויפגעשטעלט ווערן פאר דעם קאוד וואס איז שוין פריער לויפן געלאזט געווארן.

דער דריטער שריט איז פארבינדן אלעס צוזאמען. איצט זענען דא דריי פונקציעס: איינע פרעגט דעם באניצער'ס נאמען, איינע באגריסט דעם באניצער, און איינע באהאנדלט צי דער שמועס גייט ווייטער. מען פארבינדט זיי אלע אין איין איינציגן גראף, ווי עס ווערט געווײזן אויפן עקראן. די פשוטע באשלוס-פעאיקייט מאכט אז דער אסיסטענט זאל זיין מער לעבעדיג און אינטעראקטיוו.

אריינברענגען Pinecone און Hugging Face (28:26:26)

ביז דא האט מען געבויט א פשוטן אסיסטענט וואס געדענקט וואס דער באניצער גיט אריין און מאכט באשלוסן לויט דעם. דער קורס פרעגט: וואס אויב מען וויל אים געבן מער כח, ווי למשל זוכן אין דאקומענטן און געבן ענטפערס? דא קומען אריין געצייג ווי Pinecone און Hugging Face. אינעם זעלבן קאנטעקסט ווערן אויך דערמאנט LangGraph און LangChain.

- Pinecone איז א וועקטאר-דאטאבאזע וואס לאזט עפעקטיוו אפהיטן און זוכן אין גרויסע דאטא-סעטן.
- Hugging Face גיט די מאדערנסטע מאשין-לערנען מאדעלן, וואס קענען טון א בריייטע ריי ארבעטן, ווי שאפן טעקסט און ענטפערן אויף פראגעס.

ערשט ברענגט מען אריין Pinecone אין פראיעקט. מען דארף אינסטאלירן דעם Pinecone קליענט; דערפאר גייט מען אריין אין טערמינאל און מען שרייבט:

```
pip install pinecone-client
```

נאכן אינסטאלירן קען מען איניציאליזירן Pinecone און שאפן אן אינדעקס פאר די דאקומענטן. מען גייט צוריק אין IDE, מען עפנט א נייע טעקע, און מען שרייבט import pinecone. דער קאוד זוכט אין די דאקומענטן נאך רעלעוואנטע דאקומענטן לויט א פראגע (query) פונעם באניצער. די פונקציע פרעגט דעם אינדעקס און גיט צוריק דעם מערסט רעלעוואנטן דאקומענט.

דערנאך ברענגט מען אריין Hugging Face , כדי צו שאפן קלוגע ענטפערס. ערשט דארף מען אינסטאלירן די Hugging Face transformers ביבליאטעק; אין טערמינאל שרייבט מען:

```
pip install transformers
```

דערנאך קען מען ניצן Hugging Face מאדעלן צו שאפן ענטפערס לויט די דאקומענטן וואס Pinecone האט צוריקגעגעבן. בשעת די אינסטאלאציע לויפט, ווערט געשריבן דער קאוד. דער קאוד ניצט א question-answering מאדעל פון Hugging Face, וואס געפינט ענטפערס לויט דעם קאנטעקסט וואס איז ארויסגעצויגן געווארן פון Pinecone. אזוי קען דער אסיסטענט ענטפערן אויף פראגעס דורכן זוכן אין גרויסע דאטא-סעטן.

געווענליכע טעותים: פעלנדיגע אדער אומריכטיגע סטעיט (28:28:30)

דערנאך גייט דער קורס איבער צו געווענליכע טעותים וואס מען טרעפט, און ווי אזוי זיי צו לייזן.

דער ערשטער טעות איז א פעלנדיגע אדער אומריכטיגע סטעיט (state). איינע פון די מערסט געווענליכע טעותים אין LangGraph איז א "missing state" טעות. דאס פאסירט ווען דער גראף פרובירט צוקומען צו אינפארמאציע אין דער סטעיט וואס איז נאך נישט איניציאליזירט געווארן אדער עקזיסטירט בכלל נישט. אנדערש געזאגט: מען פרובירט ניצן דאטא וואס איז נישט צוגעשטעלט געווארן.

א ביישפיל: א נאוד (node) פרובירט צוקומען צו א שורה קאוד וואס ליינט דעם באניצער'ס נאמען, אבער דער באניצער האט נאך נישט אריינגעגעבן זיין נאמען. דער טעות קומט ווייל LangGraph ערווארט א שטיקל דאטא וואס איז נישטא אין דער סטעיט, און די אפליקאציע פאלט אומגעריכט.

כדי דאס צו פארמיידן, דארף מען זיכער מאכן אז די סטעיט איז ריכטיג איניציאליזירט אידער מען פרובירט צוקומען צו דאטא דערין. איין וועג איז צו קאנטראלירן צי די סטעיט אנטהאלט טאקע די נויטיגע דאטא. אינעם ביישפיל וואס ווערט געוויזן: אויב די סטעיט אנטהאלט נישט קיין username, בעט דער באט דעם באניצער עס אריינצוגעבן. דאס היט אפ אז דער באט זאל זיך נישט צוברעכן, און ער ארבעט גלאט אפילו ווען די דאטא פעלט.

ליידיגע אדער אומגילטיגע אינפוט, און נאודס וואס מען דערגרייכט נישט (28:29:31)

דער צווייטער טעות איז ליידיגע אדער אומגילטיגע אינפוט. א נאוד ערווארט אן אינפוט, אבער דער אינפוט איז אדער ליידיג אדער אומגילטיג. למשל, מען בעט דעם באניצער אריינצוגעבן זיין עלטער אדער זיין בילכערקייט, און ער גיט גארנישט אריין, אדער א פאלשן ווערט ווי א נעגאטיוו צאל. ווען דער אינפוט פאסט נישט צום ערווארטעטן פארמאט, אדער ער אנטהאלט אומגעריכטע דאטא, קען די סיסטעם קראכן אדער זיך פירן אויף אן אופן וואס מאכט נישט קיין שכל.

די לייזונג איז צו וואלידירן (באשטעטיגן) דעם אינפוט אידער מען גיט אים איבער צום נעקסטן שריט. דאס פארזיכערט אז די דאטא איז סיי ריכטיג סיי ניצבאר. אינעם ביישפיל פון פרעגן דעם באניצער'ס עלטער, קאנטראלירט דער קאוד צי דער עלטער איז א פאזיטיוו גאנצע צאל (positive integer); אויב נישט, בעט מען דעם באניצער אריינצוגעבן א גילטיגן ווערט. דער וואלידאציע-שריט פארזיכערט אז די סיסטעם זאל נישט קראכן און נישט ארויסגעבן פאלשע רעזולטאטן.

דער דריטער טעות איז נאודס וואס מען דערגרייכט נישט (unreachable nodes). אמאל, ביים אויפשטעלן דעם LangGraph ארבעטס-פלוס, באמערקט מען אז געוויסע נאודס ווערן קיינמאל נישט אויסגעפירט, ווייל דער פלוס פונעם גראף איז נישט ריכטיג. די עדזשעס (edges) צווישן די נאודס ארבעטן נישט ווי ערווארט, און געוויסע נאודס בלייבן אינגאנצן אויסגעלאזט פונעם פראצעס. אויב די באשלוס-לאגיק, ווי אן if אדער else סטעיטמענט, איז נישט ריכטיג אויפגעשטעלט, קען זי פארמיידן אז נאודס זאלן אויסגעלייזט ווערן. אז א נאוד איז נישט דערגרייכבאר, מיינט עס אז ס'איז דא א פעלער אינעם גראף: אין זיין סטרוקטור אדער אין דעם ווי דער פלוס ווערט קאנטראלירט.

די לייזונג: מען דארף פארזיכטיג איבערקוקן די פארבינדונגען צווישן די נאודס און זיכער מאכן אז די לאגיק וואס קאנטראלירט דעם פלוס איז ריכטיג אויפגעשטעלט. אויב מען ניצט באשלוס-לאגיק, דארף מען זיכער מאכן אז אלע מעגליכע באדינגונגען זענען באדעקט. אינעם ביישפיל ווערט אויפגעשטעלט א פשוטער באשלוס-פונקט אינעם גראף, וואו דער פלוס וועגדט זיך אויפן ענטפער: אויב דער ענטפער אנטהאלט yes, גייט דער גראף ווייטער, אנדערש שטעלט ער זיך אפ. מען דארף איבערקאנטראלירן די באדינגונגען און די עדזשעס, כדי אלע נאודס זאלן זיין דערגרייכבאר און אויסגעפירט ווערן אין דער ריכטיגער סדר.

מאדול איינס: וואס איז Autogen (28:31:32)

דא הייבט זיך אן א נייער טייל פונעם קורס, וועגן Autogen. דער ערשטער מאדול ערקלערט וואס Autogen איז.

דער קורס ברענגט א משל: מען וויל שרייבן א קליינע Python פראגראם. מען עפנט ChatGPT און מען בעט: שרייב א סקריפט וואס רעכנט אויס דעם דורכשניט פון א ליסטע. מען באקומט קאוד, מען קאפירט עס, מען לאזט עס לויפן, און עס קומט ארויס א טעות. מען גייט צוריק, מען לייגט אריין דעם טעות, מען פרעגט נאכאמאל, מען פאררעכנט, מען לאזט ווידער לויפן. מען שפילט דא די ראלע פון א פארמיטלער צווישן דעם AI און דעם קאמפיוטער. אזוי ניצן היינט רוב מענטשן AI.

דא טוישט Autogen די שפיל. אנשטאט איין AI, שאפט מען צוויי גאנץ פשוטע AI אגענטן. דער ערשטער אגענט'ס ארבעט איז נאר צו שרייבן קאוד; דער צווייטער אגענט'ס ארבעט איז נאר צו קאנטראלירן און טעסטן דעם קאוד. מען גיט זיי איין אנווייזונג, למשל: שרייב קאוד וואס רעכנט אויס דעם דורכשניט, און מאך זיכער אז ער ארבעט. דער ערשטער אגענט שרייבט דעם קאוד, דער צווייטער לאזט אים לויפן און זאגט אז עס קומט ארויס א טעות. דער ערשטער פאררעכנט, דער צווייטער טעסט נאכאמאל. זיי רעדן אזוי איינער מיטן אנדערן ביז דער קאוד לויפט סוף כל סוף אן א פראבלעם.

דאס איז דער הויפט-געדאנק הינטער Autogen. Autogen איז אן אפן-קוואל (open-source) פריימווערק, געשאפן דורך Microsoft Research, וואס לאזט שאפן אזעלכע קליינע AI אגענטן, יעדער מיט א באשטימטער ארבעט. דאס ווארט "אגענט" קלינגט קאמפליצירט, אבער עס מיינט פשוט אן AI מיט א באשטימטער ראלע: איינער שרייבט, איינער קאנטראלירט, איינער טעסט.

Autogen עקזיסטירט ווייל AI דארף שוין מער נישט ארבעטן אליין. אנשטאט צו בעטן איין AI אז ער זאל אלעס טון שלמות'דיג מיט איין שוס, לאזט מען עטליכע AIs ארבעטן צוזאמען, כאפן איינער דעם אנדערנ'ס טעותים און פארבעסערן דעם רעזולטאט. דער קורס באטאנט אז דאס איז פיל נענטער צו דעם ווי מענטשן טוען באמת ארבעט.

דער געדאנק גייט ווייט ווייטער פון קאוד-שרייבן. דאס זעלבע מוסטער ארבעט פאר צוזאמפאסן לאנגע דאקומענטן, ענטפערן אויף אימעילס, אדער אנאליזירן דאטא. איין אגענט טוט די הויפט-ארבעט, א צווייטער קוקט עס איבער, און די סיסטעם פארפיינט זיך אליין. מען גייט אריבער פון "AI גיט מיר אן ענטפער" צו "AI פירט דורך די גאנצע ארבעט". אויב מען האט אמאל געפילט אז AI איז שטארק אבער דארף נאך צופיל האלטן ביי דער האנט, איז דאס גענוי דער פראבלעם וואס Autogen פרובירט צו לייזן. אין דעם קורס וועט מען דאס נישט נאר הערן, נאר אליין בויען שריט ביי שריט פון אנהייב.

אין קורצן: אין זיין קערן איז Autogen וועגן שאפן שמועסן מיט עטליכע אגענטן (multi-agent conversation). אנשטאט איין AI וואס פרובירט אלעס צו טון, רעדן עטליכע AI אגענטן איינער מיטן אנדערן און פירן דורך די ארבעט צוזאמען. יעדער אגענט האט א קליינע אחריות, און זיי ארבעטן צוזאמען פונקט ווי א פאקטישע מאנשאפט. Microsoft Research האט דעם פריימווערק געשאפן כדי אזא צוזאמענארבעט זאל זיין מעגליך אויף א ריינעם און סטרוקטורירטן אופן.

מאדול צוויי: אינסטאלאציע (28:34:04)

נאכן פארשטיין וואס Autogen איז און פארוואס עס איז וויכטיג, גייט מען איבער צום פראקטישן טייל: אויפשטעלן אלעס אויף דער סיסטעם. דער קורס פארזיכערט אז עס איז נישט קאמפליצירט, און אז מען גייט רואיג מיט, וועט Autogen לויפן אין עטליכע מינוט.

פאר די אינסטאלאציע רעדט מען וועגן די סיסטעם-פאדערונגען. מען דארף האבן Python אינסטאלירט אויפן קאמפיוטער. Autogen ארבעט דאס בעסטע מיט Python 3.9 אדער העכער, דערי־בער דארף מען זיכער מאכן אז מען ניצט נישט קיין צו אלטע ווערסיע. מען קען עפענען דעם טערמינאל אדער command prompt און קאנטראלירן די Python ווערסיע. אין דער דעמא עפנט מען cmd, און עס ווייזט אז Python איז אינסטאלירט, ווערסיע 3.13.1. אויב Python איז נישט אינסטאלירט, קען מען עס אראפנעמען פון דעם אפיציעלן Python ווענזייטל, למשל די ווערסיע פאר Windows. די לינקס ווערן צוגעשטעלט אין דער באשרייבונג.

פאר מען אינסטאלירט Autogen, רעקאמענדירט דער קורס א קליינע אבער וויכטיגע זאך: ניצן א virtual environment, וואו מען וועט שאפן די Autogen אגענטן. א virtual environment איז פשוט א באזונדערער ארבעטס-ראום פאר דעם פראיעקט, אזוי ווי א קליינע אפגעזונדערטע קעסטל וואו אלע ביבליאטעקן פונעם פראיעקט וואוינען. אזוי וועט וואס מען אינסטאלירט דא נישט אנרירן אנדערע Python פראיעקטן אויף דער סיסטעם. מען קען עס שאפן גלייך פונעם טערמינאל אדער command prompt.

אין דער דעמא קאפירט מען דעם באפעל וואס שאפט דעם virtual environment, מען גייט אריין אין VS Code, וואו מען האט צוגעגרייט א פאלדער מיטן נאמען autogen, מען לייגט אריין דעם באפעל און מען דריקט Enter. עס קומט ארויס אז Python ווערט נישט געפונען; דערי־בער ניצט מען אנשטאט דעם דעם באפעל py. דער קורס ערקלערט אז ביי טייל באניצער ארבעט python, און ביי אנדערע דארף מען ניצן py. נאכדעם ווערט אלעס אינסטאלירט, מען זעט אלע ביבליאטעקן און טעקעס, און דער virtual environment איז געשאפן.

ערשטער AutoGen אגענט מיט Ollama און מולטי-אגענט שמועסן (28:36:38)

אויפשטעלן א virtual environment און אינסטאלירן AutoGen (28:36:38)

דער קורס ערקלערט אז דער באפעל צו שאפן א virtual environment מאכט א פאלדער מיטן נאמען autogenv, און אין דעם פאלדער ליגט די גאנצע איזאלירטע Python-סביבה. נאכדעם דארף מען עס אקטיווירן. אויף Windows לויפט מען דעם activate-באפעל וואס געפינט זיך אינעם Scripts-פאלדער. אין דער לעקציע ווערט געוויזן ווי מען קאפירט דעם באפעל פאר Windows און מען לייגט אים אריין אינעם טערמינאל.

עס ווערט ערקלערט אז דער שריט איז טעכניש נישט קיין מוז, ווייל מען קען אינסטאלירן AutoGen "גלאבאל" אויף דער גאנצער סיסטעם. פונדעסטווענ רעקאמענדירט דער קורס צו ניצן א virtual environment. דאס איז א גוטער שטייגער, ווייל עס פארמיידט קאנפליקטן צווישן דעפענדענסיס שפעטער, בפרט ווען מען הייבט אן בויען מער AI-אפליקאציעס.

נאכדעם וואס Python איז גרייט, איז אינסטאלירן AutoGen גאר פשוט. AutoGen קומט דורך pip, דער פעקל-פארוואלטער פון Python. אינעם טערמינאל שרייבט מען:

```
pip install autogen
```

אין עטלעכע סעקונדעס הייבט זיך אן דער דאונלאוד פון אלע דעפענדענסיס פון AutoGen. מען דארף נישט מאכן קיין האנט-קאנפיגוראציע, נאר איין באפעל.

דער קורס באמערקט אז לויט וועלכן שפראך-מאדעל מען וויל ניצן, למשל דעם OpenAI API אדער א לאקאלן מאדעל, קען מען דארפן אינסטאלירן נאך דעפענדענסיס. וועט עס ניצט די מאדעלן פון OpenAI דארף האבן דעם OpenAI פעקל און אן API key. דער key ווערט געהאלטן אלס environment variable, כדי דער פראיעקט זאל אים קענען ניצן אויף א זיכערן אופן. אין דעם קורס אבער בויט מען דעם פראיעקט מיט Ollama, וואס ווערט ערקלערט אינעם קומענדיגן מאדול.

נאך דער אינסטאלאציע איז כדאי גלייך צו קאנטראלירן צי אלעס ארבעט. מען שאפט א קליינעם טעסט-סקריפט וואס טוט import autogen און לויפט א פשוטן ביישפיל. אויב עס לויפט אן א טעות, איז די סביבה ריכטיג אויפגעשטעלט. אין דער לעקציע זעט מען א פאלדער מיטן נאמען autogen וואס איז שוין צוגעגרייט געווארן, און אין אים שאפט מען א טעקע main.py. אין דער טעקע לויפט מען א פראגראם וואס ווערט ערקלערט אין די וויטערדיגע מאדולן.

דער קורס עצהט צו נעמען צייט פאר דעם טייל, אפצושטעלן דעם ווידעא ווען נייטיג, און אלעס ריכטיג אויפצושטעלן. ערשט דערנאך גייט מען ווייטער צו פארשטיין ווי AutoGen ארבעט פון אינעווייניג, און צו שאפן אגענטן.

צוויי וועגן צו בויען אגענטן: API key אדער Ollama (28:39:15)

מען שאפט יעצט דעם ערשטן AI-אגענט. דער קורס ערקלערט אז עס זענען דא צוויי הויפט-וועגן צו בויען אגענטן מיט AutoGen:

- דער ערשטער וועג איז מיט אן API key . מען פארבינדט דעם אגענט צו אן אנליין מאדעל פון פירמעס ווי OpenAI און אנדערע. די סיסטעם שיקט פארלאנגען דורכן אינטערנעט, און דער מאדעל ענטפערט צוריק. דער וועג איז פשוט און שטארק, אבער ער הענגט אפ אינעם API-באנוץ און אין טאָקענס, וואס ווערן געווענליך אויסגעשעפט.
- דער צווייטער וועג איז Ollama. דאס איז א געצייג וואס לאזט לויפן גרויסע שפראך-מאדעלן (LLM) גלייך אויפן אייגענעם קאמפיוטער. אנשטאט זיך פארבינדן צו א cloud-סערווים מיט אן API key, לויפט דער מאדעל לאקאל אויף דער מאשין.

דער אונטערשייד, לויט דעם קורס, איז פשוט: מיט אן API key לויפט דער מאדעל אינעם cloud , און מיט Ollama לויפט ער אויף דער אייגענער סיסטעם. מען האט אויסגעקליבן Ollama צוליב עטלעכע מעלות:

- מען דארף נישט קיין API key, ווייל עס לויפט לאקאל.
- עס זענען נישטא קיין גרעניצן אויף טאָקענס, אויך צוליב דעם וואס עס לויפט לאקאל.
- מען קען ארבעטן offline, אן אן אינטערנעט-פארבינדונג.
- מען לויפט נישט אריין אין quota-גרעניצן אדער אין איבערריסן צוליב בילינג.

צו דאונלאודן Ollama גייט מען צו דעם וועבזייטל און מען דריקט אויפן download-קנעפל. צו קאנטראלירן צי עס איז אינסטאלירט, עפנט מען cmd און מען שרייבט דעם version-באפעל פון Ollama. אין דער לעקציע ווייזט עס שוין די ווערסיע, ווייל עס איז פריער געווען אינסטאלירט. דער קורס פארלאנגט פונעם באניצער צו

עפענען VS Code און דארט אינסטאלירן Ollama. מען קען עס אויך אינסטאלירן פונעם וועבזייטל אדער דורך א באפעל.

דער קאוד פונעם ערשטן אגענט מיט Llama 3 (28:41:22)

יעצט לויפט Ollama לאקאל און AutoGen איז אינסטאלירט. מען שאפט דעם ערשטן אגענט און מען פארבינדט אים צום מאדעל Llama 3 וואס לויפט אויף דער מאשין. דער קורס ערקלערט דעם קאוד שריט נאך שריט:

- אויבן טוט מען import פון asyncio, כדי דער פראגראם זאל באהאנדלען די ענטפערס גלאט, אן "פרירן".
- מען טוט import פון AssistantAgent פון AutoGen. דאס איז דער אגענט וואס מען שאפט, אזוי ווי א דיגיטאלער אסיסטענט וואס קען טראכטן און ענטפערן.
- מען טוט import פון Console, וואס ווייזט שיין די ענטפערס אינעם טערמינאל בשעת זיי ווערן געשאפן, אנשטאט ווארטן אויפן גאנצן ענטפער.
- מען טוט import פון א chat completion client פאר Ollama. דאס פארבינדט AutoGen מיט Ollama. אן דעם וואלט AutoGen נישט געווואוסט ווי צו רעדן מיטן Llama 3 מאדעל וואס לויפט לאקאל.

צו קריגן דעם מאדעל גייט מען צו cmd און מען שרייבט:

ollama pull llama3

(אדער יעדן אנדערן מאדעל וואס מען וויל), און עס ווערט אויטאמאטיש דאונלאודעט.

אין דער main פונקציע שאפט מען א model_client. דארט זאגט מען פאר AutoGen וועלכן מאדעל צו ניצן און ווו ער לויפט. מען גיט אן דעם נאמען פונעם מאדעל, אין דעם פאל llama3. דער נאמען מוז זיך געוי צופאסן מיטן מאדעל וואס מען האט דאונלאודעט מיט ollama pull. דערנאך גיט מען די host-אדרעס, די דיפאלט-אדרעס וואו Ollama לויפט אויף דער סיסטעם. מיט אנדערע ווערטער: יעדעס מאל וואס דער אגענט דארף אן ענטפער, שיק דעם פארלאנג צום לאקאלן Llama 3 מאדעל.

דערנאך שאפט מען דעם אגענט מיטן נאמען assistant, מען פארבינדט אים צום model_client, און מען גיט אים א system message: "You are a helpful AI assistant". דער קורס ערקלערט אז דער system message איז ווי די פערזענליכקייט אדער די אנווייזונג פונעם אגענט. אויב מען טוישט די שורה אויף עפעס אנדערש, טוישט זיך אויך דאס אויפפירונג פונעם אגענט.

צום סוף גיט מען דעם אגענט א טאסק דורך assistant.run_stream, מיט דער פראגע "What is AutoGen?". די Console-הילע זארגט אז דער ענטפער ווייזט זיך אינעם טערמינאל אין א streaming-פארמאט, אזוי אז מען זעט דעם ענטפער ווי ער ווערט געשאפן אין רעאלער צייט. אונטן לויפט מען (asyncio.run(main().

אין פשוטע ווערטער, פאסט דער קורס צוואם: מען האט פארבונדן AutoGen מיט א לאקאל-לויפנדיגן Ollama, מען האט געשאפן אן AI-אגענט, אים געגעבן א פערזענליכקייט, אים צוגעטיילט א טאסק, און געשטרימט דעם ענטפער, אלעס אן אן API key. דאס איז דער יסוד פאר אלעס וואס מען בויט ווייטער, אריינגערעכנט מולטי-אגענט סיסטעמען.

לויפן דעם פראגראם (28:44:00)

מען עפנט דעם טערמינאל און מען לויפט דעם פראגראם. דער קורס ווארנט אז ווען מען לויפט עס, דארף דער טערמינאל ווייזן אז autogenv איז אקטיוו. אויב נישט, דארף מען עס אקטיווירן. מען זעט די פראגע "What is AutoGen?", און Llama 3 ענטפערט דורך AutoGen. עס נעמט צייט, בערך צוויי דריי מינוט, און אין דער לעקציע האט עס אפילו לענגער געדויערט. צום סוף באקומט מען א פאראגראף וואס ערקלערט וואס AutoGen איז. דערמיט ענדיגט זיך דער ערשטער מאדול פון שאפן אן AI-אגענט מיט AutoGen.

מולטי-אגענט שמועסן: דער דיאגראם (28:45:02)

דער קומענדיגער מאדול באהאנדלט מולטי-אגענט שמועסן: מען שאפט מער ווי איין אגענט און מען זעט ווי זיי רעדן צווישן זיך. דער קורס ערקלערט א בילד וואס ווייזט ווי מולטי-אגענט מיטארבעט פונקציאנירט אין AutoGen.

אין צענטער פונעם דיאגראם שטייט א "problem solving task", דער הויפט-ציל אדער די פראבלעם וואס מען דארף לייזן. אין א טראדיציאנעלער AI-סיסטעם פרובירט אפט איין מאדעל צו באהאנדלען די גאנצע טאסק אליין. אין א מולטי-אגענט סיסטעם ווי AutoGen ווערט די טאסק אבער באהאנדלט דורך עטלעכע אגענטן וואס קאמוניקירן איינער מיטן אנדערן.

אויף דער לינקער זייט שטייט דער Assistant Agent, וואס שפילט די ראלע פונעם AI-אסיסטענט. ער ווערט געווענליך געטריבן פון א גרויסן שפראך-מאדעל, און ער איז פאראנטווארטליך צו שאפן לייזונגען און צו טראכטן איבער דער פראבלעם: ער קען שרייבן Python קאוד, שאפן א לייזונג אדער פארשלאגן פארבעסערונגען. למשל, אויב די פראבלעם פאדערט אנאליזירן דאטא, מאכן חשבונות אדער פראצעסירן אינפארמאציע, קען דער אסיסטענט שרייבן Python קאוד דערפאר. נאכדעם וואס דער קאוד לויפט, קען ער איבערקוקן די רעזולטאטן און פארבעסערן די לייזונג. דער אסיסטענט טראגט דעריבער דעם הויפט-טייל פונעם טראכטן און לייזן.

אויף דער רעכטער זייט שטייט דער User Proxy Agent. ער פארטרעט דעם מענטשלעכן באניצער, אדער אן אינטערפעיס וואס האנדלט אין נאמען פונעם באניצער, און ער איז א בריק צווישן מענטשן און דער AI-סיסטעם. ער באקומט מעסעדזשעס פונעם אסיסטענט און באשליסט ווי צו ענטפערן. דאס בילד באטאנט צוויי פעאיגקייטן פון אים:

- Automatic mode: ווען ער דערקענט א בלאק קאוד וואס מען קען לויפן אינעם שמועס, קען ער דעם קאוד אויטאמאטיש לויפן.
- Human input: ער קען בעטן אינפוט פונעם מענטש ווען נייטיג. למשל, ווען מען דארף א באשלוס, א באשטעטיגונג אדער נאך אנווייזונגען, קען די סיסטעם זיך אפשטעלן און פרעגן דעם באניצער.

די דאזיגע בייגעווידיגקייט לאזט AutoGen ארבעטן אדער אינגאנצן אויטאנאמיש, אדער מיט מענטשלעכע איינמישונג.

דער פייל אויפן בילד, באצייכנט "send message", ווייזט דעם פלוס פון קאמוניקאציע: די אגענטן ארבעטן צוזאמען דורך סטרוקטורירטע שמועסן. א טיפישער גאנג: דער User Proxy שיקט א באשרייבונג פון דער טאסק צום אסיסטענט. דער אסיסטענט אנאליזירט דעם פארלאנג און שאפט אן ענטפער, וואס קען אנטהאלטן חשבון-גאנג, א פארגעשלאגענע לייזונג אדער קאוד. דער User Proxy לויפט דעם קאוד, כאפט אויף די

רעזולטאטן און שיקט זיי צוריק צום אסיסטענט. לויט די רעזולטאטן פארבעסערט דער אסיסטענט די לייזונג אדער פארראכט טעותים. דער הין-און-צוריק שמועס גייט אזוי ווייטער ביז די טאסק איז געלונגען דורכגעפירט.

אונטן אויפן דיאגראם ווערן באטאנט דריי אייגנשאפטן: Conversable and customizable AI agents)אגענטן וואס קענען שמועסן און וואס מען קען צופאסן), fully automated אינגאנצן אויטאמאטיש), און adapt and improve solutions)צופאסן און פארבעסערן לייזונגען).

א writer- און reviewer-אגענט (28:48:36)

דערנאך ווייזט דער קורס א AutoGen workflow מיט צוויי אגענטן, פאר וועלכע מען שרייבט קאוד:

- א Writer Agent, וואס שאפט אן ערקלערונג, דאס הייסט ער שאפט דעם אינהאלט.
- א Reviewer Agent, וואס קוקט איבער וואס דער Writer האט געשאפן און גיט ארויס א reviewed output.

ביידע ניצן Llama 3, דעם זעלבן מאדעל וואס מען האט פריער גענוצט. מען לויפט דעם קאוד אין VS Code, אבער פריער ערקלערט מען אים. ער הייבט זיך אן מיט דעם זעלבן import פון asyncio וואס מען האט שוין באשפראכן אינעם פריערדיגן מאדול, כדי צו באהאנדלען טאסקס אסינכראניש, אזוי אז זיי זאלן קענען לויפן אין הינטערגרונט. מען שאפט אן async def main כדי צו פארבינדן דעם לאקאלן Llama מיט AutoGen.

פון AutoGen דעפלוימענט ביז MCP (28:49:38)

שרייבער און רעצענזענט: צוויי אגענטן ארבעטן צוזאמען (28:49:38)

אין דעם טייל פון קורס ווערט ווייטער ערקלערט דער קאוד פון מולטי-אגענט צוזאמענארבעט אין AutoGen. דער קאוד באניצט א llama client, און עס ווערן דעפינירט צוויי אגענטן: א "writer" (שרייבער) און א "reviewer" (רעצענזענט).

- דער שרייבער-אגענט באקומט די אנווייזונג: "דו ביסט א שרייבער, שרייב א קורצע ערקלערונג ווענן AutoGen".
- דער רעצענזענט-אגענט באקומט די ראלע: "דו ביסט א רעצענזענט". זיין אויפגאבע איז צו קריטיקירן און פארבעסערן וואס דער שרייבער האט געשריבן.
- צום סוף שרייבט מען ארויס דעם רעזולטאט.
- דער לעצטער בלאק קאוד לויפט די main פונקציע ווען מען לויפט דעם סקריפט דירעקט.

דערנאך גייט מען אריבער צו VS Code און מען זעט דעם רעזולטאט פון דעם זעלבן קאוד. דער שרייבער'ס ארויסגאבע הייבט זיך אן מיט "What is AutoGen", און דער רעצענזענט'ס ארויסגאבע הייבט זיך אן מיט "Here's rewritten version", דאס הייסט א איבערגעשריבענע ווערסיע. אויב עס דארפן זיין ענדערונגן אין דעם וואס דער שרייבער האט געשריבן, נעמט זיי דער רעצענזענט אויף זיך, אזוי אז דער אינהאלט וואס ווערט ארויסגעגעבן און וואס קומט צום מענטש איז גוט אויסגעפיילט און פארטיג.

דער קורס באטאנט אז דא איז בכלל נישטא קיין מענטשליכע אריינמישונג: די צוויי אגענטן רעדן צווישן זיך און שטעלן צוזאם דעם בעסטן סקריפט וואס מעגליך. די קאוד-טעקעס ווערן צוגעזאגט אין דער באשרייבונג פון ווידעא. מען רעקאמענדירט צו אפשטעלן דעם ווידעא ביי יעדן שריט און עס מיטמאכן האנט ביי האנט.

א פולע דעפלוימענט מיט FastAPI אין VS Code (28:51:10)

איצט ווערט געוויזן א פולע דעפלוימענט. אין VS Code מאכט מען א נייע טעקע מיטן נאמען app.py אין דעם זעלבן AutoGen פאלדער, און מען שרייבט אריין דעם קאוד. דער קאוד טוט די פאלגנדע זאכן:

- from fastapi import FastAPI – מען שאפט א FastAPI אפליקאציע. זי באהאנדלט די HTTP בקשות וואס מען וועט שיקן.
- מען באניצט דעם זעלבן chat completion client וואס מען האט פריער געזען. דער מאדעל איז TinyLlama.
- עס ווערט רעקאמענדירט צו לויפן אין cmd דעם באפעל ollama pull tinyllama. די סיבה איז דער סטארעדזש: TinyLlama איז א קליינער מאדעל, דערפאר אינסטאלירט ער זיך גרינג. אזוי פארבינדט מען זיך מיט א לאקאלן TinyLlama AI דורך AutoGen.
- מען דעפינירט דעם אגענט און וואס ער דארף טון: א AI אגענט וואס ענטפערט "hi " אויף יעדע פראגע. דער system message זאגט: "reply only with hi", נישט קיין חילוק וואס דער באניצער זאגט.
- app.get באהאנדלט די chat בקשות. עס דעפינירט אן endpoint וואס נעמט א פראגע פונעם באניצער און ענטפערט אלעמאל מיט "hi".

די שריט זענען:

- ערשטנס שאפט מען די API טעקע, אין אונזער פאל app.py.
- Ollama מוז לויפן. מען האט עס שוין פריער אינסטאלירט.
- אין VS Code קאפירט מען דעם באפעל צו לויפן דעם סערווער און מען לויפט אים פון דעם פאלדער.

מען זעט ווי אלעס לויפט אויפן פארט, מיט אלע אינפארמאציע, און צום סוף שטייט "application startup complete". מען דארף א ביסל ווארטן ביז דער פראצעס ענדיגט זיך.

דערנאך עפנט מען Chrome און מען לייגט אריין דעם לינק. אין אנהייב ווייזט עס "not found", און נאך א קורצער ווייל עפנט זיך דער FastAPI בלאט. מען גייט צום chat endpoint, מען דריקט "Try it out", מען שרייבט "hello" און מען דריקט "Execute". נאך א ביסל לאדן קומט דער ענטפער: "hi". נישט קיין חילוק וואס מען שרייבט, וועט דער ענטפער אלעמאל זיין "hi".

פארוואס FastAPI, און די ענדגילטיגע סטרוקטור (28:54:10)

דער קורס ערקלערט פארוואס מען באניצט דוקא דעם FastAPI URL. וען מען לייגט נאר אריין א localhost URL, דארף מען שאפן א single page application כדי עס זאל לויפן מיט VS Code. מען קען עס נישט לויפן דירעקט אויף א בראוזער אן FastAPI, אויב מען האט נישט קיין אפליקאציע.

למשל, אויב מען האט א פראנט-ענד וועבזייטל און מען אינטעגרירט דארט דעם אגענט, דאן וען מען עפנט דאס וועבזייטל וועט עס ארבעטן גאנץ גוט. דא האט מען עס נאר געוויזן כדי צו ערקלערן ווי די זאך ארבעט.

די ענדגילטיגע פראיעקט-סטרוקטור דארף אויסזען אזוי:

- דער הויפט-פאלדער AutoGen.
- דער virtual environment.
- main.py.

- app.py.

ווי אזוי בלייבט מען דערהיינטיגט מיט AutoGen (28:55:19)

צום סוף פון דעם טייל ווערט געבעטן צו ענדערן דעם צוגאנג: נישט נאר "לערנען AutoGen", נאר וואקסן מיט אים. פריימווערקס ווי AutoGen, וואס איז אנטוויקלט געווארן פון Microsoft, ענטוויקלען זיך גאר שנעל. מען לייגט כסדר צו נייע פיטשערס, באג-פיקסעס און אינטעגראציעס. וועט עס וויל טאקע זיין פאראויס אין דער AI וועלט, קען זיך נישט פארלאזן נאר אויף רעקארדירטע לעקציעס. מען מוז בלייבן פארבונדן מיטן מקור.

דער ערשטער און וויכטיגסטער מנהג איז צו באזוכן קביעות'דיג דעם אפיציעלן repo אויף GitHub:

- דער Releases טייל ווייזט וואס האט זיך געענדערט.
- דער Issues טאב ווייזט פאקטישע פראבלעמען וואס פראגראמירער האבן אין דער פראקטיק.
- דער Pull Requests טייל ווייזט אויף וועלכע פארבעסערונגען מען ארבעט.

אפילו נאר עטליכע מינוט א וואך דארט האלט אייך מיט די לעצטע ענטוויקלונגען. אנשטאט רעאגירן שפעט אויף דערהיינטיגונגען, ווייסט מען דערפון גלייך.

אבער דערהיינטיגט בלייבן איז נישט נאר ליינען release notes. מען דארף זיין א טייל פון דער קהילה. אסאך פריימווערקס האבן דיסקוסיע-פארומס, קהילה-שמועסן און אמאל Discord סערווערס. דארט טיילן פראגראמירער אידייען, פרעגן שאלות און וויזן זייערע פראיעקטן. וועט מען שליסט זיך אן, זעט מען ווי אנדערע באניצן AutoGen אין פראקטישע פעלער. מען לערנט נייע workflows, מען אנטדעקט אפטימיזאציעס, און מען פארשטייט ספעציעלע פעלער (edge cases) וואס מען וואלט אליין קיינמאל נישט געטראפן.

אריינפיר צו MCP: דער פלאן (28:56:46)

דא הייבט זיך אן א נייע לעקציע: MCP פאר אנהייבער. דער פלאן איז:

- א קורצע ערקלערונג וואס איז MCP. אין דער לעקציע ווערט עס אויך גערופן "multi-agent collaboration protocol".
- גרונט-פראטאקאלן און סטאנדארטן אין קאמוניקאציע.
- דאס פארבינדן MCP צו דעם פארשטאנד פון פראטאקאלן און סטאנדארטן.
- א דעמאנסטראציע: אנטוויקלען א web scraper אגענט מיט MCP. דערפאר דארף מען Node.js, Cursor AI און דעם Firecrawl API.

די לעקציע ווערט באשריבן ווי גאר באקוועם פאר אנהייבער, מיט פראקטישע שריט איינס נאכן אנדערן.

וואס איז MCP? (28:57:23)

לויט דעם וועבזייטל פון MCP, איז MCP (Model Context Protocol, דא אויך גערופן multi-agent collaboration protocol) אן אפן-קוואל פראטאקאל וואס שטעלט א סטאנדארט ווי אפליקאציעס גיבן צו קאנטעקסט פאר גרויסע שפראך-מאדעלן. ענליך ווי USB-C פאראיניגט די כלים-פארבינדונגען, גיט MCP א איינהייטליכן וועג צו אינטעגרירן AI מאדעלן מיט פארשידענע דאטא-מקורות און געצייג.

אין דער אפיציעלער דאקומענטאציע ווערט עס פארגליכן מיט n8n און אנדערע AI געצייג. למשל, פאר אן AI רייזע-אגענט (דער קאנאל האט א באזונדערע לעקציע דערויף), וואלט מען געדארפט:

- אן אגענט וואס שרייבט אימעילס.
- אן אגענט וואס זוכט לופטפעלד-קאודס.
- אן אגענט וואס זוכט האטעלן.
- אן אגענט וואס באשטעלט פליגערס.

אלע זענען פארבונדן איינער מיטן אנדערן אין א קאמפליצירטע נעץ אויף אן n8n פלאטפארמע. מיט MCP ווערט דאס פארמינערט. מען האט א סטאנדארטיזירטן צוגאנג וואס מאכט אוועק דעם boilerplate קאוד, דאס מאנועלע זוכן דאטא-באזעס, דאס מאנועלע זוכן וויסן-באזעס, און דאס מאנועלע זוכן אגענטן וואס זענען געמאכט פאר יענעם צוועק. מען ערקלערט נאר דעם פלאן, מען צוטשעפעט די API שליסלען, און MCP נעמט איבער. עס גיט א פארטיג און אויסגעארבעט פראדוקט אין קורצער צייט. דאס איז דער הויפט-געדאנק פון MCP.

אויף אן איבערבליק-בילד זעט מען די טיילן:

- MCP host.
- clients.
- servers.
- לאקאלע דאטא-מקורות.
- ווייטע סערוויסעס (remote services).

מען קען ארבעטן מיט לאקאלע דאטא-מקורות אדער א לאקאלער וויסן-באזע, און אויך מיט אפן-קוואל מקורות. אין דער דאקומענטאציע געפינט מען quick starts, גיידס, ביישפילן און לעקציעס. פאר א טיפערן אריינקוק זענען דא בלעטער וועגן:

- דער core architecture.
- resources.
- prompts.
- tools.
- transports.
- sampling.

מען קען אויך ביישטייערן פאר דער קהילה, און באקומען שטיצע און אפרוף.

סטאנדארטן און פראטאקאלן: דער משל פונעם אינטערנאציאנאלן טים (28:59:58)

אויב מען פארשטייט נאך אלץ נישט קלאר וואס MCP איז, ווערט געגעבן א משל פון סטאנדארטן און פראטאקאלן, כדי מען זאל האבן א מוסטער צו פארבינדן MCP דערצו.

שטעלט אייך פאר אז איר האט א טים מענטשן פאר א פראיעקט, און איר זענט דער מענעדזשער. דער פראבלעם: קיינער קען נישט דעם אנדערן, יעדער שטאמט פון אן אנדער לאנד, און עס איז נישטא קיין געמיינזאמע שפראך. מענטש A רעדט מאנדארין, מענטש B רעדט הינדי, מענטש C רעדט פראנצויזיש, און אזוי ווייטער.

יעדער מיטגליד איז ווי אן AI אגענט אדער געצייג. אין אזא פאל מוז מען hardcode'ן די קאמוניקאציע-פארטן און אנווייזונגען, און מען מוז גיין צו יעדן אגענט באזונדער און אים ערקלערן אויף א ספעציפישן אופן. דאס נעמט גאר לאנג. ביי מענטשן וואלט מען געדארפט רעדן מיט יעדן אין זיין שפראך, און מען קען דאך אפשר נישט קיין מאנדארין אדער פראנצויזיש, און מען בלייבט שטעקן.

אבער אויב עס איז דא איין שפראך וואס אלע קענען, למשל ענגליש, קען מען אלע צונויפנעמען, ערקלערן אלעס אין איין שמועס, און פארטיג. מען פארלירט נישט קיין צייט, עס איז נישטא קיין hardcoding און קיין מאנועלע מי.

דער משל פון וועב-סערווערס (29:01:58)

דאס זעלבע ביי א וועב-סערווער: איר זענט דער client, און עס איז דא א server. למשל, וועו מען ארבעט אויף א סאציאלע נעץ: מען פרובירט ארויפצולייגן עפעס, צו שרייבן א קאמענטאר, אדער צו "לייקן". די אקטיוויטעטן גייען דורך פיר סארטן בקשות:

- GET request.
- PUT request.
- POST request.
- DELETE request.

וועו מען איז פארבונדן צו איין סערווער, איז עס נישט אזוי שווער. אבער וועו מען פארבינדט זיך צו אסאך סערווערס, דארף מען יעדעס מאל אנדערע פראטאקאלן. דאס איז א צוזאמשטעל פון צוויי טיילן: HTTP און דער URL צו וועלכן מען וויל זיך פארבינדן.

כדי אוועקצונעמען די מי צווישן אסאך סערווערס און אסאך בקשות מיט איין מאל, האט מען איין אייגציגן... (און דא ענדיגט זיך דער טייל, און די ערקלערונג גייט ווייטער אינעם קומענדיגן אפשניט).

בויען א MCP-סערווער מיט Firecrawl אין Cursor (29:03:02)

REST API און MCP – א שטאנדארט פאר קאמוניקאציע (29:03:02)

דער קורס הייבט אן מיט א געדאנק וועגן שטאנדארטן. אין וועב-קאמוניקאציע איז דא א געצייג וואס הייסט API, און דער מערסט פאפולערער API איז דער REST API. ער גיט א שטאנדארטישן אופן ווי צו קאמוניקירן.

עס ווערט ערקלערט אז דער REST API שפילט די זעלבע ראלע ווי די ענגלישע שפראך ביי א מיטינג פון א טיעם, א משל וואס איז פריער געברענגט געווארן. וועו אלע רעדן איין שפראך, פארשטייט יעדער דעם אנדערן. אזוי אויך האט דער REST API אויפגעשטעלט א שטאנדארט. דורך אים קען יעדער וועב-קליענט רעדן מיט זיין סערווער, אדער מיט עטלעכע סערווערס אויף איינמאל, מיט איין שפראך, איין מיטל און איין שטאנדארט.

אן דעם REST API און אן די שטאנדארטן און פראטאקאלן וואלט די זאך געווען זייער צעמישט. מען וואלט געקענט אריינפאלן אין א קאמפליצירטע לאגע, און דער רעזולטאט וואלט נישט געווען וואס מען ערווארט.

דערנאך ווערט דער משל אריבערגעטראגן אויף AI. לאמיר זאגן מען בויט עטלעכע AI-אגענטן פאר איין ציל. דאן וואלט מען אויך געוואלט האבן א מוסטער ("טעמפלעיט") פאר בויען אגענטן, כדי דער פראצעס זאל זיין

ווייניגער קאמפליצירט. דאס זאל ארבעטן פונקט ווי דער REST API ביי א וועב-סערוויס, און פונקט ווי ענגליש ביי א מיטינג. דא קומט אריין MCP. MCP איז דער מוסטער פאר דעם צוועק.

ווי מען האט געארבעט אן MCP (29:04:02)

דער קורס ערקלערט ווי עס איז געווען פאר MCP. מען האט באנוצט א געצייג וואס הייסט) n8nאין די קעפשאנס "NA10"). צו אים זענען געווען צוגעבינדן אלע געצייג, און אלע LLM-ס זענען צו אים צוגעפראגראמירט געווארן.

דער פראבלעם איז געווען אז מען האט געמוזט שרייבן "hardcoded" קאוד פאר יעדן געצייג אין דער נעץ, און פאר יעדן אגענט וואס איז דערמיט פארבונדן. מען האט געמוזט ענוי געבן אנווייזונגען, אויפשרייבן די טריט, און אויסבויען א "פערסאנע" פארן AI-אגענט. עס איז געווען א סך ארבעט.

אז מען קען אראפנעמען די גאנצע קאמפליצירטקייט מיט איין געצייג, איז דאס MCP . מען דארף נאר אריינלייגן MCP און זאגן דעם ציל, און די ארבעט ווערט געטון.

די טריט צום בויען דעם פראיעקט (29:05:04)

נאך דער אלגעמיינער איבערזיכט, גייט מען בויען אן אייגענעם MCP. דער פראיעקט איז א "web crawler" אדער "web scraper", א מכשיר וואס זוכט און ציט ארויס אינפארמאציע פון וועבזייטלעך. די טריט זענען:

- אינסטאלירן NodeJS און צוגרייטן איר סביבה.
- א רעדאקטאר פאר קאוד: דער Cursor AI editor.
- א API: דער Firecrawl API.
- צום סוף, בויען דעם פראיעקט וועט אלע מיטלען זענען גרייט.

אינסטאלירן NodeJS און Cursor (29:05:36)

אין דער לעקציע ווערט געוויזן ווי מען אינסטאלירט NodeJS. מען ארבעט אויף Windows, דעריבער קלייבט מען Windows. די ארכיטעקטור איז די וואס דער קאמפיוטער באנוצט, דא x64.

ווער עס ארבעט מיט אן אנדער סיסטעם, למשל Mac OS, דארף קלייבן ARM אדער אן אנדער פאסיגע ארכיטעקטור. דאסזעלבע גילט פאר Linux.

דערנאך קלייבט מען דעם Windows installer, די MSI טעקע, און זי ווערט אראפגעלאדן.

בשעת דאס לאדט זיך אראפ, גייט מען צום נעקסטן וועבזייטל צו אראפלאדן און אינסטאלירן Cursor AI. מען דארף שאפן א חשבון אדער זיך איינלאגירן. דער זייטל גיט אויטאמאטיש די ווערסיע וואס פאסט צום אפעריירנדיקן סיסטעם, דא Windows. ווער עס האט שוין א חשבון, לאגירט זיך איין און באשטעטיגט אז ער איז דאס.

ביים איינשטעלן Cursor:

- מען קלייבט א טעמע. דא ווערט געזומען "cursor dark".
- מען זעט צו אז Cursor Tab, דער Control-קנעפל (Ctrl) און Agent זענען אנגעצייכנט.
- מען קען קלייבן די אפציע פון טיילן דאטא. דא ווערט געזומען "privacy mode", אבער ווער עס וויל קען קלייבן די אנדערע אפציע.

- מען גייט דורך עטלעכע איבערקוק-איינשטעלונגען, און מען קומט אן צום נייעם פענצטער פון Cursor.

דערווייל קוקט מען צוריק אויף דער NodeJS אינסטאלאציע. דער "wizard" (דער אינסטאלאציע-פראגראם) האט שוין אנגעהויבן אינסטאלירן. מען גייט איבער וואס איז געשען אין פענצטער: עס זענען אינסטאלירט געווארן פאקעטן וואס מען דארף, צווישן זיי Chocolatey און אן אפדעיט KB. צום לעצט איז Python, ווערסיע 3.13.3, באשטעטיגט געווארן. ביז אהער איז אלץ אין ארדענונג.

צוגיין צו די MCP איינשטעלונגען אין Cursor (29:07:41)

אין Cursor דארף מען אריינגיין אין די איינשטעלונגען צו שאפן א נייעם MCP. עס זענען דא עטלעכע וועגן:

- דורך File, דערנאך Preferences, און דערנאך Cursor Settings.
- אדער דורכן זוך-פענצטער, וואו מען שרייבט "settings" אדער "cursor settings". ביים פראבירן דעם וועג איז ארויסגעקומען א קליינע אומגענויקייט, אבער מען דארף זיך נישט זארגן. מען גייט דעם געווענליכן וועג דורך File, Preferences, Cursor Settings.

אויף דער לינקער זייט פונעם פענצטער זעט מען MCP . מען גייט אהין. ווייל דאס איז דאס ערשטע מאל אין Cursor, זענען נאך נישטא קיין MCP סערווערס.

מען דריקט אויפן בלויען קנעפל "Add new global MCP server". עס עפנט זיך א טעקע MCP.json, וואס איז אינגאנצן ליידיג.

אין דער טעקע דארף מען אריינשרייבן אביסל "hardcoded" קאוד. דער קורס באטאנט אז דאס איז נישט אזוי קאמפליצירט אין פארגלייך מיט וואס מען האט פריער געטון אין n8n. מען דארף אבער נאך עטלעכע געצייג.

דער "Awesome MCP Servers" GitHub זייטל (29:09:14)

איידער מען גייט צום Firecrawl וועבזייטל, ווייזט דער קורס א GitHub זייטל וואס הייסט "Awesome MCP Servers". דארט געפינט מען כמעט אלע אינפארמאציע וועגן MCP.

דער זייטל האט לינקס צו די פאלגנדיקע אפטיילונגען: What is MCP, Clients, Tutorials, Community, Legend, Server Implementations, Frameworks, און Tips and Tricks. וועט עס וויל מער וויסן, קען דורכגיין יעדן זייטל באזונדער.

אויפן הויפט-זייטל געפינט מען די יסודות וועט און וואס MCP איז. ווייטער ארום זענען דא "server implementations", אויסגעטיילט לויט קאטעגאריעס:

- Aggregators
- Art and Culture
- Cloud Platforms
- Code Execution
- Coding Agents
- Command Line
- Communication
- Customer Data Platforms

- Data Platforms

און נאך אסאך. מיט זיי קען מען דורכפירן פארשידענע אפעראציעס. דערנאך מאכט מען צו די פענצטער און מען גייט צום נעקסטן שטאפל.

אנמעלדן זיך ביי Firecrawl (29:10:47)

פארן web crawler פראיעקט דארף מען א API פון א פירמע, דא Firecrawl. אויף דער ליסטע געפינט מען "Firecrawl MCP", און דורך דעם לינק קען מען שאפן א API.

פריער גייט מען דורך דעם Firecrawl וועבזייטל:

- דער מוסטער-קאוד וואס מען וועט באנוצן.
- א ברייטע אויסוואל פון "crawl" אפציעס.
- "media parsing" אפציעס.
- "smart wait" אקציעס.
- דער פרינציפ "reliability first".
- פרייזן. דערווייל נעמט מען דעם בחינם פלאן (free plan).

מען שרייבט אריין די עמעיל אדרעס און א פאסווארט. דער ערשטער פאסווארט איז נישט געווען שטארק גענוג, און דער וועבזייטל האט אים נישט אנגענומען.

דער קורס זאגט אז דאס איז א גוטער סימן, אז דער וועבזייטל איז שטארק. מען דארף האבן שטארקע פאסווערטער. מען זאל קיינמאל נישט טיילן מיט אנדערע די פאסווערטער, און אויך נישט די API שליסלען וואס מען שאפט ביי Firecrawl.

נאך א לענגערן פאסווארט איז דער חשבון ערפאלגרייך געשאפן געווארן. דערנאך דארף מען קוקן אין דער עמעיל און באשטעטיגן.

די Firecrawl MCP סערווער דאקומענטאציע (29:12:20)

דערווייל גייט מען צו דער API דאקומענטאציע פון Firecrawl, צום זייטל "Firecrawl MCP Server". דארט זענען דא די אפטיילונגען Overview, Content, Tools, און אנדערע מיטלען.

דארט געפינט מען דעם מוסטער פאר דער סערווער-קאנפיגוראציע ("server configuration"). דער קורס זאגט אז מען זאל דאס היטן, ווייל מען וועט עס דארפן.

אונטער "Content" זענען דא די טריט וואס מען דארף נאכפאלגן, און די אייגנשאפטן. אונטער "Tools" געפינט מען געצייג פאר crawling, extraction און scraping וואס ארבעטן מיט Firecrawl.

שאפן און קאפירן דעם API שליסל (29:13:22)

די עמעיל איז נאך נישט געווען באשטעטיגט, דעריבער האט מען זי נאכאמאל געשיקט. מען האט זי באשטעטיגט אויפן טעלעפאן, אריינגעשריבן די פרטים, און זיך ערפאלגרייך איינגעלאגירט.

אויף דער לינקער זייט זעט מען: Overview, Playground, Extract, Activity Logs, Usage, API Keys , און Settings.

דער API שליסל איז לכתחילה באהאלטן. מען האט צוויי אפציעס, "view" (אנקוקן) אדער "copy" (קאפירן). מען קאפירט אים, כדי צו זיין זיכער. ווייל דער חשבון איז נייגעבויט, זענען נאך נישט רעגיסטרירט קיין אקטיוויטעטן.

אריינלייגן די קאנפיגוראציע אין MCP.json (29:13:52)

מען גייט צוריק צום Content זייטל צו קאנטראלירן די טריט:

- אינסטאלירן npm.
- אריינגיין אין די Cursor AI איינשטעלונגען און צולייגן א נייעם MCP סערווער. דאס האט מען שוין געטון.
- קאפירן דעם מוסטער-קאוד און אים אריינלייגן.

מען דארף זיך נישט זארגן וואו גענוי צו לייגן דעם API שליסל, ווייל אין מוסטער איז דא א באשטימט ארט פאר אים.

צוריק אין Cursor, ווערט דער טעקסט אין MCP.json פארביטן מיט דעם קאוד וואס מען האט קאפירט. א וויכטיגע וואראנונג: אויף Windows דארף מען באנוצן "c/ cmd". ווער עס ארבעט נישט אויף Windows, קען דאס אויסלאזן. אויף Windows דארף מען עס אבער אומבאדינגט.

מען מעקט אויס דעם אלטן טעקסט און מען גיט דעם געצייג דעם נאמען "firecrawl". אויף א נייער שורה שרייבט מען "command" אין טאפלטע געזעפיסלעך. דארט קלעבט מען אריין (V + Ctrl) דעם באפעל וואס מען האט קאפירט פון Firecrawl.

דער טייל "c/ cmd" (אין ווייס אויפן עקראן) איז וואס מען דארף היטן: אויף Windows איז עס נויטיג, אנדערש קען מען עס אויסלאזן.

אויב עס זענען דא ספקות וועגן דעם קאוד, דארף מען זיך נישט זארגן. מען קען צוריקגיין צום Firecrawl זייטל און פארגלייכן מיט דעם קאנפיגוראציע-מוסטער וואס זיי שטעלן צו. דארט זעט מען אויך דאס ארט פארן API שליסל, אין וועלכן מען דארף אריינלייגן דעם אייגענעם שליסל.

MCP: פון ערשטן סערווער ביז א וואלוטע-אגענט (29:15:58)

ענדיגן די Pipedream-קאנפיגוראציע און פרובירן דעם אגענט (29:15:58)

אין דער לעקציע ווערט ווייטער געוויזן ווי אזוי מען שטעלט אוועק דעם MCP-סערווער אין Cursor AI . אויפן Pipedream-פענצטער קאפירט מען דעם API key און מען לייגט אים אריין אין דעם ארט וואו עס שטייט "your API key" אינעם קאנפיגוראציע-קאוד. נאכדעם וואס מען האט איבערגעקוקט אז דער קאוד שטימט מיט דער קאנפיגוראציע, דארף מען באזונדערס אכטונג געבן אויף די טאפלטע געזעפיסלעך (double quotes).

עס ווערט אויך באטאנט דער טייל "c/ cmd": ווער עס ארבעט אויף Windows דארף עס אוודאי האבן. ווער עס ארבעט נישט אויף Windows קען עס איבערשפרינגען.

ביים לעצטן איבערקוקן ווערט געטראפן א טעות: א double quote איז געשטאנען אויפן אומריכטיגן ארט. מען נעמט עס אראפ פון יענעם ארט און מען שטעלט עס אוועק אינעם סוף, נאך דעם ווארט "MCP". מען קוקט אויך איבער דעם אפשטאנד צווישן דעם API key און די געזעפיסלעך, און דאן איז אלעס אין ארדענונג.

אפהיטן איז גאנץ פשוט: מען דריקט אויף די דריי פינטלעך, מען קלייבט Close און דערנאך Save. דערמיט איז די ארבעט געענדיגט, מען פארמאכט דעם mcp.json טעקע און מען גייט צוריק צום אגענט.

נאכן פארמאכן וועט מען זען דעם נייעם סערווער אין די Cursor Settings. אמאל באקומט מען אבער "no tool connected", "no server connected" אדער אנדערע טעותים. אין אזא פאל דארף מען איבערקוקן צי npm און NodeJS זענען ריכטיג אינסטאלירט, און צי אלע נייטיגע טעקעס זענען פאראן.

אויב אלעס איז אינסטאלירט, די ווערסיעס זענען זעבאר, און דער טעות בלייבט פונדעסטוועגן, דארף מען זיך נישט זארגן. מען פארמאכט Cursor AI און מען עפנט עס פון דאס ניי, אדער מען דריקט דעם refresh-קנעפל וואס איז גלייך דארט צוגעשטעלט.

נאכן ווידער-עפענען גייט מען דורך דעם זעלבן וועג: File, Preferences, Cursor Settings . דארט, ביי די MCP-טיילן, געפינט מען דעם סערווער.

א זוך-פראמפט: "Simplilearn" אויף Google (29:18:04)

מען גייט צום פענצטער וואו דער סערווער געפינט זיך און מען קלייבט אויס דעם מצב Agent. עס ווערט באטאנט אז מען דארף אויסקלייבן Agent און נישט Manual אדער Ask.

מען שרייבט א פשוטן פראמפט: "Search for Simplilearn on Google". מען ערווארט רעזולטאטן פון Google, למשל די היים-בלאט פון דער פירמע אדער אירע סאציאלע בלעטער ווי LinkedIn, YouTube אדער Instagram, יעדע זאך וואס פאסט צום טערמין "Simplilearn".

דער אגענט ברענגט דריי רעזולטאטן:

- דער וועבזייטל פון Simplilearn.
- דער YouTube-קאנאל פון Simplilearn.
- דער LinkedIn-בלאט.

מען דריקט אויף Cursor .simplilearn.com בעט א רשות צו עפענען, און דערנאך זעט מען די היים-בלאט. דערנאך גייט מען צוריק און מען עפנט דעם YouTube-בלאט, ווידער מיט א רשות-פראגע, און מען קומט אן אויפן YouTube-בלאט.

ביי LinkedIn ווערט געפרובירט צו זען צי מען קען עפענען אן זיך איינשרייבן. עס ווייזט זיך אז LinkedIn פארלאנגט יא א login. נאכן זיך איינשרייבן, וואס נעמט עטליכע סעקונדעס, עפנט זיך דער Simplilearn-בלאט אויף LinkedIn מיט הצלחה.

וואס איז MCP? (29:19:30)

דער קורס ערקלערט וואס איז Model Context Protocol, אדער MCP. דאס איז אן אפענער סטאנדארט, אנטוויקלט דורך Anthropic. זיין ציל איז צו מאכן איינהייטלעך ווי אזוי גרויסע שפראך-מאדעלן (LLMs) פארבינדן זיך מיט דרויסנדיגע דאטא, געצייג און סערוויסעס.

"דרויסנדיגע דאטא" קען זיין א לעבעדיגער וועבזייטל, א לעבעדיגע דאטא-שטאפל, א cloud-סערווער און ענליכס. מען בויט אן אגענט וואס האט צוטריט צו אזא וועבזייטל אדער דאטא-קוואל. דער אגענט ברענגט רעאל-צייט דאטא און ארבעט פאר אייך, גלייך ווי איר וואלט עס אליין געטון. מען אנטוויקלט אזוי אן אגענט וואס ארבעט אנשטאט אייך.

דערנאך גייט מען דורך די אפיציעלע Anthropic-דאקומענטאציע. דארט ווערט MCP באשריבן ווי אן open-source פראטאקאל וואס מאכט איינהייטלעך ווי אזוי אפליקאציעס גיבן קאנטעקסט צו LLMs. די דאקומענטאציע נוצט א משל: MCP איז ווי א USB-C פארט פאר AI-אפליקאציעס.

דער קורס לייגט צו א מוסטער. וועז מען בויט אן AI-אגענט דורך אן AI-IDE ווי Cursor AI, דארף מען געבן א סארט פראמפט וואס העלפט עס פארשטיין ווי אזוי צו בויען. אויפן זעלבן אופן איז MCP ווי א געמיינזאמער מוסטער. אויף אים קען מען בויען אן אייגענעם פראמפט, כדי צו ערקלערן דעם אגענט וואס צו טון און ווי אזוי פונקטלעך.

SDKs, GitHub און אינסטאלאציע (29:20:50)

אין דער דאקומענטאציע זענען פאראן פארשידענע SDKs (software development kits), לויט דער פראגראמיר-שפראך וואס מען פילט זיך באקוועם מיט:

- C#/.NET SDK, פאר פראגראמירער וואס ארבעטן אין .NET.
- Java SDK.
- Python, וואס מען נוצט אין דער לעקציע.
- Ruby, Swift און TypeScript.

חוץ דעם איז פאראן א quick start הקדמה צו MCP, און דאקומענטאציע פאר server developers, client developers און Claude Desktop באניצער. עס זענען אויך פאראן גרייטע מוסטער-סערווערס.

עס איז פאראן אן אפענער GitHub-בלאט פאר MCP. די קאוד דארט קען מען גלייך איבערמאכן אין אן IDE ווי Cursor AI, Visual Studio Code אדער יעדן אנדערן IDE. מען דארף נאכגיין די README-טעקעס, און עס איז דא א קורצער אינסטאלאציע-וועגווייזער.

אויפן command prompt לויפט מען:

```
pip3 install mcp
```

ביי דער דעמא איז עס שוין געווען אינסטאלירט, דעריבער זאגט דער באפעל "already satisfied". נאכן אינסטאלירן קען מען ווייטער גיין לויט די אנווייזונגען.

עס זענען פאראן צווויי README-טעקעס: איינע פון פאר דריי חדשים און איינע פון פאר צווויי וואכן. מען נעמט די נייערע, דעם MCP Python SDK, וואס פירט דורך אלע טריט.

דער ערשטער MCP-סערווער: דאלאר צו פונט (29:22:43)

דער ציל איז צו בויען א פשוטן MCP, כדי צו פארשטיין ווי עס ארבעט. נאכן פארשטיין די יסודות קען מען גיין צו די מער פארגעשריטענע.

טריט:

- מען עפנט Cursor AI) אדער יעדן IDE) און מען שאפט א נייעם פאלדער מיטן נאמען "MCP server".
- מען קלייבט אויס דעם פאלדער אלס ארבעט-ארט.
- מען שאפט א נייע Python-טעקע מיטן נאמען first_mcp.py.
- מען קאפירט אריין א גרייטן קאוד פון א Notepad-טעקע.

דער קאוד פארוואנדלט אמעריקאנער דאלארן אויף ענגלישע פונט סטערלינג. למשל, מען גיט $1 און מען באקומט זיין ווערט אין פונט. ביים אריינלייגן ווערט באמערקט א קליינער טעות אינעם נאמען פונעם משתנה exchange rate, און דאס ווערט פארראכטן.

דער קאוד ארבעט אזוי:

- ער אימפארטירט MCP און גיט דעם געצייג א נאמען.
- ער נוצט א MCP tool דעקאריורונג און דעפינירט וואס דאס געצייג טוט.
- עס ווערט באשטימט א פעסטער משתנה (דער אויסטויש-קורס). מיט אים רעכנט מען אויס דעם סכום.
- דער amount איז דער אינפוט וואס דער באניצער גיט, וויפיל דאלאר ער וויל פארוואנדלען.

מען היט אפ, און דערנאך גייט מען צו Preferences, Cursor Settings, MCP Tools. די פריערדיגע MCPs פון די פריערדיגע ביישפילן ווערן אויסגעמעקט. מען שרייבט "mcp run first_mcp.py", קאפירט דעם פולן path און לייגט אים אריין. Cursor AI שלאגט אליין פאר צו שאפן א mcp.json טעקע. מען דריקט Enter, און נאך א צווייטן פרואוו איז די JSON-טעקע פול פונקציאנירנדיג. מען היט אלעס אפ.

אויף דער היים-בלאט ווייזט זיך אז דאס געצייג איז נישט enabled. דאס טרעפט זיך אמאל. מען דארף נאר ריסטארטן, אדער דעם MCP disable און ווידער enable מאכן.

דערנאך היט מען אלעס אפ (Save All), מען גייט צום אגענט און מען שרייבט: "Can you convert $100 to pounds?". דער אגענט נוצט די פונקציע USD to GBP און ענטפערט: 79 GBP. ער לייגט צו אז אויב מען וויל זען עס אין אקציע, אדער מען דארף א סקריפט צו דרוקן דעם רעזולטאט, זאל מען אים לאזן וויסן.

נאך MCP-סערווערס דורך דער דאקומענטאציע (29:26:40)

דער נעקסטער טריט איז צו שאפן נאך אן AI-אגענט, אדער MCP-סערווער, וואס געפינט אינפארמאציע אויפן אינטערנעט וועגן א געגעבענעם טערמין. דאס איז דער ראאל-צייט נוצן-פאל.

פריער ווערט ערקלערט אן אלגעמיינער וועג פאר אנדערע סערווערס, למשל צו געפינען דעם היינטיגן ווערט פון Bitcoin אדער אן אנדערער cryptocurrency, אדער צו פארשן אקציע-פרייזן. מען נעמט די README-טעקע פון דער אפיציעלער דאקומענטאציע און מען גיט זי צו Cursor:

- מען עפנט א נייעם פענצטער און מען גייט צו Preferences, Cursor Settings.
- דארט איז פאראן די אפציע "Indexing & Docs". פריער האט מען דאס געדארפט זוכן אונטער "Features", אבער נאך דער נייער ווערסיע פון Cursor AI געפינט מען עס גלייך.
- מען דריקט "Add document", מען לייגט אריין דעם לינק מיט Ctrl+V, און מען דריקט Enter און Save.

אין דער דעמא איז די דאקומענטאציע שוין צוגעלייגט אונטערן נאמען "Simplilearn Python SDK demo". דערנאך שאפט מען א נייעם פראיעקט, "sample". אין דעם פראמפט שרייבט מען וואס מען דארף האבן, און מיט "@docs" לייגט מען צו דעם קאנטעקסט. אזוי קען מען זאגן דעם אגענט ער זאל זיך באניצן מיט דער דאקומענטאציע אלס רעפערענץ.

א וועב-קראולער פראיעקט מיט Cursor AI און Firecrawl MCP (29:28:57)

דער אגענט אין Cursor און דער ציל פונעם פראיעקט (29:28:57)

דער קורס הייבט אן מיט דערמאנען אז דעם דאקומענט וואס מען האט פריער געשאפן אין Python קען מען גלייך געבן פארן אגענט, צוזאמען מיט א באפעל. מען קען אויך אויסקלייבן וועלכן אגענט צו ניצן: מען קען לאזן אויף "automatic", וואו די סיסטעם קלייבט אליין אויס פון די פארהאנענע אגענטן, אדער מען קלייבט אליין א ספעציפישן מאדעל פון די אפציעס, למשל Sonnet, GPT-4.1, Gemini Flash, וכדומה.

אלעס וואס מען דארף טון איז בעטן דעם אגענט ער זאל שאפן א געוויסן אגענט וואס זאל טון א געוויסע ארבעט. מען קען אים אויך געבן פרטים, ווי די API-שליסלען.

דערנאך גייט מען איבער צום פראיעקט גופא: דער פראיעקט וועט זיין אויספארשן דאטא אויפן אינטערנעט פאר דעם טערמין "Simplilearn".

אינסטאלירן NodeJS און Cursor AI (29:29:58)

ווי פריער דערמאנט, הייבט מען אן מיט אינסטאלירן NodeJS אויפן אפעריר-סיסטעם. די שריט:

- מען ארבעט אויף Windows, דעריבער קלייבט מען Windows.
- די ארכיטעקטור דארף צופאסן צום קאמפיוטער; דא איז עס x64. ווער עס ארבעט אויף Mac OS דארף אפשר קלייבן ARM אדער אן אנדערע פאסיגע ארכיטעקטור; אויף Linux דאס זעלבע, לויט דעם סיסטעם.
- מען קליקט אויף דעם "Windows installer" MSI טעקע, און עס הייבט אן אראפצולאדן.

בשעת דאס לאדט אראפ, גייט מען צום נעקסטן וועבזייטל צו אראפלאדן און אינסטאלירן Cursor AI, וואס מען וועט דארפן פאר דער דעמאנסטראציע:

- מען דארף זיך איינשרייבן (שאפן א קאנטע) אדער אריינלאגירן, און די וועבזייטל גיט אויטאמאטיש די ווערסיע וואס פאסט צום אפעריר-סיסטעם. דא, ווייל מען איז אויף Windows , האט עס געגעבן די Windows ווערסיע.
- אז מען האט שוין א קאנטע, לאגירט מען אריין און מען באשטעטיגט אז דאס איז טאקע מען אליין, און דערנאך איז Cursor AI אינסטאלירט.
- מען קלייבט א "טעמע" (דעם אויסזען); דא קלייבט מען "Cursor Dark".
- מען פארזיכערט אז "Cursor Tab", דער Control-קנעפל און "Agent" זענען דארט.
- מען קען אויסקלייבן די אפציע פון טיילן דאטא; דא קלייבט מען "Privacy Mode", אבער ווער עס וויל קען קלייבן די אנדערע אפציע.
- נאך עטלעכע איבערקוק-איינשטעלונגען, קומט מען אן צו א נייעם פענצטער פון Cursor.

דערווייל קוקט מען נאך אויף דער NodeJS אינסטאלאציע: דער וויזארד האט שוין אנגעהויבן אינסטאלירן. מען גייט שנעל איבער וואס עס האט געטון אין יענעם פענצטער פון אנהייב: עס האט אינסטאלירט עטלעכע פעקלעך וואס מען וועט דארפן, צווישן זיי Chocolatey און אן אפדעיט מיטן נאמען KB299 ..., און צום סוף Python, ווערסיע 3.13.3, וואס איז באשטעטיגט. ביז אהער איז אלעס אין ארדענונג, און מען קען גיין צום נעקסטן שריט.

צוקומען צו די Cursor איינשטעלונגען און לייגן א נייעם MCP סערווער (29:32:00)

אין Cursor AI דארף מען גיין צו די איינשטעלונגען כדי צו שאפן א נייעם MCP. עס ווערט ערקלערט אז עס זענען דא עטלעכע וועגן צו דערגרייכן די איינשטעלונגען:

- מען גייט צו "File", דארט קלייבט מען "Preferences" און דערנאך "Cursor Settings".
- אדער מען ניצט דאס זוך-פענצטער און זוכט "settings" אדער "Cursor Settings".

אין דער לעקציע פרובירט מען דעם וועג פון זוכן, אבער עס ארבעט נישט גאנץ גלאט, דעריבער גייט מען צוריק צום געווענליכן וועג: File ← Preferences ← Cursor Settings.

אויף דער לינקער זייט זענען דא די אפציעס; מען קלייבט "MCP ". ווייל מען איז דאס ערשטע מאל אריין אין Cursor AI, איז נאך נישטא קיין שום MCP סערווער. מען קליקט אויף דעם בלויען קנעפל "Add new global MCP server", און עס עפנט זיך א טעקע מיטן נאמען mcp.json, וואס איז אינגאנצן ליידיג.

אין דער טעקע דארף מען אריינשרייבן עטלעכע שורות פעסטן קאוד. עס ווערט ערקלערט אז דאס איז נישט קאמפליצירט קענען די פריערדיגע זאכן וואס מען האט געארבעט מיט n8n, נאר מען וועט דארפן נאך עטלעכע מכשירים.

א בליק אויף דער "Awesome MCP Servers" רשימה (29:33:30)

איידער מען גייט צו דער Firecrawl וועבזייטל, קוקט מען אויף א GitHub לינק מיטן נאמען "Awesome MCP Servers". דארט געפינט מען כמעט אלע אינפארמאציע וואס האט צו טון מיט MCP. מען עפנט אין באזונדערע פענצטערס די פארשידענע לינקס: וואס איז MCP, טוטאריאלס, קהילה (community), "legend" (ערקלערונג פון די סימנים), סערווער-אימפלעמענטאציעס, פריימווארקס, עצות און טריקס. פאר מער אינפארמאציע קען מען גיין איבער יעדע פון די וועבזייטלעך באזונדער.

אויף דער הויפט-זייט געפינט מען די יסוד-אינפארמאציע וואס איז MCP. וועט מען שקראלט אראפ, זעט מען די רשימה פון סערווער-אימפלעמענטאציעס, איינגעטיילט אין קאטעגאריעס, למשל:

- אגרעגאטארן (aggregators).
- קונסט און קולטור.
- קלאוד-פלאטפארמעס.
- קאוד-אויספירונג (code execution).
- קאודירן-אגענטן (coding agents).
- באפעל-שורות (command line).
- קאמוניקאציע.
- קונה-דאטא פלאטפארמעס (customer data platforms) און דאטא-פלאטפארמעס, און נאך אסאך.

Firecrawl: איינשרייבן און זיך באקענען מיט דער וועבזייטל (29:35:03)

מען מאכט צו די פענצטערס און מען גייט צום נעקסטן שטאפל: מען דארף אן API. ווייל דער פראיעקט איז א "וועב-קראולער", דארף מען אן API פון איינעם פון די פראווייזערס, דהיינו דער Firecrawl API , אדער

Firecrawl MCP. מען קליקט אויף דעם לינק כדי ערשט צו שאפן אן API-שליסל, און דערפאר דארף מען זיך איינשרייבן ביי Firecrawl.

פריער גייט מען איבער די Firecrawl וועבזייטל פון אנהייב: דארט איז דא דער מוסטער-קאוד מיט וועלכן מען וועט ארבעטן, און א ברייטע אויסוואל פון אפציעס: קראולן, אנאליזירן מעדיע (media parsing), "smart wait" אקציעס, "Reliability first" א.א.וו. עס זענען אויך דא די פרייזן; דערווייל קלייבט מען דעם פרייען פלאן.

מען לייגט אריין די פרטים: אן אימעיל אדרעס און א פאסווארט. דער ערשטער פאסווארט איז נישט שטארק גענוג, און עס וועט באמערקט אז דאס איז א גוטער סימן: די וועבזייטל איז שטארק געזיכערט און פאדערט שטארקע פאסווערטער. דערצו וועט געווארנט: מען זאל נישט טיילן מיט אנדערע קיין פאסווערטער און קיין API-שליסלען וואס מען שאפט מיט Firecrawl. נאכן לייגן א לענגערן פאסווארט, וועט די קאנטע הצלחה'דיג געשאפן. דערנאך דארף מען טשעקן דעם אימעיל און באשטעטיגן.

די Firecrawl MCP סערווער דאקומענטאציע (29:36:36)

בשעת מען וואט אויף דער באשטעטיגונג, גייט מען צו אן אנדער וועבזייטל: די Firecrawl API דאקומענטאציע, וואו עס זענען דא אלע פרטים וואס מען דארף פאר שאפן דעם API-שליסל. מען זוכט דארט "Firecrawl MCP Server", און דאס איז די ריכטיגע זייט. דארט זענען דא אפטיילונגען: Overview, Content, Tools און אנדערע מקורות.

- דא געפינט מען דעם מוסטער פון סערווער-קאנפיגוראציע וואס מען וועט דארפן; מען זאל עס האלטן אין אויג.
- אין דער "Content" זייט זענען דא די שריט וואס מען דארף נאכפאלגן און אלע פיטשערס און פרטים.
- אונטער "Tools" זענען דא די מכשירים פון Firecrawl: קראולן, ארויסציען (extraction) און "scraping", צוזאמען מיט עטלעכע ערקלערונגען.

לאגירן אריין און באקומען דעם API-שליסל (29:37:38)

דער אימעיל איז נאך נישט באשטעטיגט, דעריבער שיקט מען עס נאכאמאל. נאכן באשטעטיגן אויפן טעלעפאן, לייגט מען אריין די פרטים און מען לאגירט אריין הצלחה'דיג.

אויף דער לינקער זייט זעט מען: Overview, Playground, Extract, Activity, Logs, Usage, API Keys און די איינשטעלונגען. דער API-שליסל איז פון סטאנדארט באהאלטן, און מען ווייזט אים נישט. עס זענען דא צוויי אפציעס: "View" (זען) אדער "Copy" (קאפירן); מען קאפירט אים, כדי צו זיין אויף דער זיכערער זייט. ווייל מען איז נייע אויף דער וועבזייטל, איז נאך נישטא קיין שום אקטיוויטעט.

דערנאך גייט מען צוריק צו דער "Content" זייט צו טשעקן די שריט: מען דארף אינסטאלירן npm , גיין צו די Cursor AI איינשטעלונגען און לייגן א נייעם MCP סערווער (וואס מען האט שוין געטון), און קאפירן דעם מוסטער-קאוד און אים אריינלייגן דארט. מען דארף זיך נישט זארגן וואו אריינצולייגן דעם API-שליסל; אינעם מוסטער איז דא א באשטימט ארט דערפאר.

אריינלייגן די קאנפיגוראציע אין mcp.json (29:38:39)

צוריק אין Cursor AI, ביטט מען אויס דעם ליידיגן מוסטער מיטן קאוד וואס מען האט קאפירט. א וויכטיגע וואריאנט: ווער עס ארבעט אויף Windows דארף ניצן cmd /c. ווער עס איז נישט אויף Windows קען דאס אוועקלאזן, אבער אויף Windows דארף מען עס אויף זיכער.

די שריט:

- מען מעקט אויס דעם אלטן אינהאלט.
- מען רופט דאס מכשיר "firecrawl".
- אויף א נייער שורה שרייבט מען דעם "command", אין טאפלטע געזעפיסלעך, און מען לייגט אריין (Ctrl+V) דעם קאוד וואס מען האט קאפירט פון Firecrawl.
- דער טייל cmd /c איז דער וואס מען דארף אכטונג געבן: אויף Windows דארף מען אים; אויף אנדערע סיסטעמען קען מען אים אוועקלאזן.

כדי צו זיין זיכער אז דער קאוד איז ריכטיג, קען מען צוריקגיין צו דער Firecrawl זייט און פארגלייכן מיטן מוסטער פון קאנפיגוראציע וואס זיי שטעלן צו. אויף דער זעלבער זייט זעט מען אויך דעם API-שליסל: מען קאפירט אים און לייגט אים אריין אויפן ארט וואו עס שטייט "your API key".

לעצטע קאנטראל און שפייכלען (29:40:13)

נאכן פארגלייכן, דארף מען ספעציעל טשעקן די טאפלטע געזעפיסלעך. מען פארגרעסערט דעם בילד אויפן עקראן כדי בעסער צו זען, און מען ווייזט נאכאמאל אויף c/ cmd: אויף Windows דארף מען עס, אויף אנדערע סיסטעמען קען מען עס אוועקלאזן.

ביים לעצטן קאנטראל טרעפט מען א טעות: א טאפלטער געזעפיסל איז געווען אויפן אומריכטיגן ארט. מען מעקט אים אויס פון יענעם ארט און לייגט אים צום סוף, נאך "mcp". מען טשעקט אויך דעם אפשטאנד צווישן דעם API-שליסל און די געזעפיסלעך, און דאס איז אין ארדענונג.

שפייכלען איז פשוט: מען דריקט אויף די דריי פינטלעך, "Close", און דערנאך "Save". דערמיט איז מען פארטיג, און מען קען צומאכן די mcp.json טעקע און צוריקגיין צום אגענט.

נאכן צומאכן, וועט מען זען דעם נייעם סערווער אין די Cursor איינשטעלונגען. אבער עס ווערט געווארנט אז אמאל באקומט מען א מעלדונג "no tool connected" אדער "no server connected", אדער אנדערע פעלערן; אין אזא פאל דארף מען איבערטשעקן די קאנפיגוראציע.

פון MCP-דעמא צו מולטי-אגענט סיסטעמען (29:41:48)

פארענדיקן די MCP-דעמא אין Cursor AI (29:41:48)

דער קורס פירט ווייטער די דעמא פון פריער, וואו מען האט אינסטאלירט דעם MCP-סערווער אין Cursor AI. עס ווערט ערקלערט אז מען דארף פריער באשטעטיגן אז די npm-אינסטאלאציע און די NodeJS-אינסטאלאציע זענען דורכגעגאנגען, און אז אלע נויטיגע פיילס זענען דארט. אבער אפילו ווען אלעס איז אינסטאלירט און מען זעט די ווערסיעס, קען זיין אז מען זעט נאך אלץ דעם זעלבן פעלער. דער עצה איז: נישט

זארגן. מען פארמאכט Cursor AI און מען עפנט עס צוריק, אדער מען קלאפט אויף דער "refresh"-אפציע וואס געפינט זיך גלייך דארט.

נאכן איבערשטארטן גייט מען צוריק דעם זעלבן וועג: File, דערנאך Preferences, דערנאך Cursor Settings. דארט זעט מען דעם אפטייל פון MCP, און מען געפינט דעם סערווער אויסגעשטעלט ווי עס דארף צו זיין. דערנאך פארמאכט מען די סעטינגס און מען גייט צום פענצטער וואו מען ארבעט מיטן סערווער.

- אין דעם פענצטער קלייבט מען אויס דעם מאדוס "agent". דער קורס באטאנט אז מען דארף אויסקלייבן נאר "agent" און נישט קיין אנדערע אפציע, נישט "manual" און נישט "ask".
- דערנאך שרייבט מען א פשוט'ן פראמפט: "Search for Simplilearn on Google".
- דער אגענט דארף צוריקברענגען רעזולטאטן וואס ער געפינט אויף Google: למשל די היים-בלאט, אדער א סאשעל-מידיא בלאט ווי LinkedIn, YouTube אדער Instagram, אלעס וואס איז פארבונדן מיטן ווארט "Simplilearn".

מען זעט אז דער אגענט האט צוריקגעגעבן דריי רעזולטאטן: ערשטנס, די אפיציעלע וועבזייטל פון Simplilearn; צווייטנס, דער YouTube-קאנאל פון Simplilearn; דריטנס, די LinkedIn-בלאט.

- מען קלאפט אויף דער אפיציעלער וועבזייטל. עס פרעגט ערלויבעניש, מען קלאפט "open", און מען קומט אן אויף דער היים-בלאט פון Simplilearn.
- מען גייט צוריק און מען קלייבט אויס די YouTube-בלאט. ווידער גיט מען ערלויבעניש, און מען קומט אן צום YouTube-קאנאל.
- ביי LinkedIn רעכנט מען אז עס וועט פארלאנגען א לאגין. מען פרובירט פונדעסטוועגן צו עפענען אן איינלאגירן, און טאקע, עס פארלאנגט צו לאגירן. מען לאגירט זיך גיך איין, וואס נעמט א פאר סעקונדעס, מען קלאפט "open", און די בלאט פון Simplilearn אויף LinkedIn ווערט מיט הצלחה געעפנט.

דער געדאנק הינטער מולטי-אגענט סיסטעמען (29:43:40)

דא הייבט זיך אן א נייע לעקציע. זי עפנט מיט א משל: שטעלט אייך פאר איין מענטש וואס פירט א גאנצן פראיעקט אליין: פארשונג, אנאליז, אינהאלט, קוואליטעט-קאנטראל און די לעצטע איבערגעבונג, אלעס אויף איין טעלער. ביי א קליין פראיעקט קען דאס נאך גיין, אבער וען דער פראיעקט וואקסט, פאלט דאס גיך אויסנאנד. יעצט שטעלט אייך פאר א מאנשאפט: איינער פארשט, דער צווייטער אנאליזירט, איינער פירט אויס און איינער קוקט איבער. יעדער פארמאגט איין ארבעט און טוט זי גוט. אזוי ווערן גרעסערע און קאמפליצירטערע פראיעקטן טאקע דורכגעפירט.

עס ווערט ערקלערט אז דאס איז פונקט דער יסוד פון מולטי-אגענט סיסטעמען. אנשטאט זיך צו פארלאזן אויף איין AI-סיסטעם וואס זאל טון יעדע ארבעט, שאפט מען עטליכע AI-אגענטן, און יעדער האט א באשטימטע ראלע. איין אגענט קלייבט צונויף אינפארמאציע, א צווייטער אנאליזירט זי, א דריטער מאכט באשלוסן, און נאך איינער קוקט איבער דעם ענדגילטיגן רעזולטאט. צוזאמען ארבעטן די אגענטן מיטאנאנד צו דערגרייכן א געמיינזאמען ציל.

פארוואס איז די טעמע אזוי פאפולער געווארן? ווייל AI גייט ארויס איבער פשוטע סיסטעמען וואס ענטפערן נאר אויף פראגעס. ביזנעסער זוכן היינט AI-סיסטעמען וואס קענען פלאנירן, באשליסן, ניצן געצייג און פארענדיקן גאנצע ארבעטס-פראצעדורן (workflows). דער קורס ציטירט דעם "State of AI " באריכט פון McKinsey, וואס ווייזט אז פירמעס גייען מער און מער אריבער פון AI-עקספערימענטן צו פראקטישע AI-

איינפירונג אין זייערע ביזנעס-פראצעסן. פירמעס זוכן AI-סיסטעמען וואס קענען אנפירן קאמפליצירטע ארבעטן, נישט נאר ארויסגעבן ענטפערס.

וואס איז אן AI-אגענט, און ווי אזוי ער איז אנדערש פון א טשעטבאט (29:44:57)

איידער מען גייט ווייטער, ערקלערט דער קורס דעם יסוד: אן AI-אגענט איז א סיסטעם וואס קען פארשטיין א ציל, באשליסן וועלכע טריט מען דארף צו דערגרייכן דעם ציל, ניצן די געצייג וואס שטייען צו דער הנד, און אויספירן אקציעס כדי צו פארענדיקן די ארבעט.

אסאך מענטשן פארגלייכן AI-אגענטן מיט טשעטבאטס, אבער עס איז דא א וויכטיגער אונטערשייד. א טשעטבאט קאנצענטרירט זיך מערסטנס אויף ענטפערן פראגעס, למשל "ערקלער מאשין-לערנען". ער קען געבן אינפארמאציע און ערקלערונגען. אן AI-אגענט גייט ווייטער: ער קאנצענטרירט זיך אויף פארענדיקן א ציל. למשל, ווען מען בעט אן AI-אגענט צו שאפן א מארק-פארשונג באריכט, קען ער באשליסן וועלכע טריט מען דארף, צונויפקלייבן אינפארמאציע, אנאליזירן די דאטא און צוגרייטן דעם ענדגילטיגן באריכט.

א פשוטער וועג עס צו פארשטיין איז דורך פארגלייכן מיט א פערזענליכן סעקרעטאר (personal assistant). אזא סעקרעטאר גיט נישט נאר אינפארמאציע; ער פלאנירט, ארגאניזירט און פארענדיקט ארבעטן אויף אייער חשבון. גלייכערהייט קען אן AI-אגענט פארשטיין די פאדערונגען, פלאנירן אקציעס, ניצן געצייג און ברענגען רעזולטאטן.

די אנטוויקלונג: פון כללים-סיסטעמען ביז אגענטן (29:45:57)

דער קורס באשרייבט ווי AI האט זיך אנטוויקלט פון א פשוטן טשעטבאט ביז אן אינטעליגענטן אגענט, דורך עטליכע שטאפלען:

- אנפאנגס זענען AI-סיסטעמען געווען מערסטנס באזירט אויף כללים (rule-based): זיי האבן נאכגעפאלגט פארויס-באשטימטע אנווייזונגען. דאס האט גוט געארבעט ביי פשוטע ארבעטן, אבער זיי האבן זיך געמוטשעט ווען די לאגע איז געווארן אומפאראויסזעבאר.
- שפעטער האט מאשין-לערנען דערמעגליכט פאר סיסטעמען צו לערנען מוסטערן פון דאטא. דאס האט געברענגט אפליקאציעס ווי רעקאמענדאציע-סיסטעמען, אנטדעקן שווינדל (fraud detection) און פאראויסזאגונג-אנאליז.
- דערנאך האבן שמועס-AI (conversational AI) און גרויסע שפראך-מאדעלן געטוישט דעם אופן ווי מענטשן רעדן מיט מאשינען. באניצער האבן געקענט רעדן מיט AI נאטירליך און באקומען דעטאלירטע ענטפערס.
- אבער די טראדיציאנעלע שמועס-סיסטעמען האבן זיך מערסטנס קאנצענטרירט אויף ארויסגעבן ענטפערס. זיי האבן געקענט ערקלערן ווי אזוי צו פארענדיקן אן ארבעט, אבער זיי האבן נישט געקענט אליין פלאנירן און אויספירן דעם גאנצן פראצעס. דאס האט געשאפן א באדערפעניש פאר AI-סיסטעמען וואס זאלן טאקע נעמען אקציע.
- AI-אגענטן האבן איינגעברענגט די פעאיקייט צו פארשטיין צילן, פלאנירן טריט, ניצן געצייג און פארענדיקן ארבעטן. און ווען די ארבעטן זענען געווארן קאמפליצירטער, האבן עטליכע ספעציאליזירטע אגענטן געקענט ארבעטן צוזאמען, ווי מען זעט אויפן סלייד א מאנשאפט פון AI-אגענטן.

וואס איז א מולטי-אגענט סיסטעם (29:47:00)

א מולטי-אגענט סיסטעם איז א גרופע פון עטליכע AI-אגענטן וואס ארבעטן צוזאמען צו דערגרייכן א געמיינזאמען ציל. אנשטאט אז איין אגענט זאל פרובירן אנצופירן מיט יעדער אחריות, ווערן פארשידענע אגענטן אנגעשטעלט אויף פארשידענע ראלעס, לויט זייערע שטארקייטן.

דער משל: א פירמע האט פארשידענע אפטיילונגען, ווי מארקעטינג, פינאנצן, סעיילס (פארקויף) און אפעראציעס. יעדע אפטיילונג האט אן אנדער אחריות, אבער זיי ארבעטן מיטאנאנד צו דערגרייכן דעם ציל פון דער פירמע. א מולטי-אגענט סיסטעם גייט אויפן זעלבן אופן.

א ביישפיל: שטעלט אייך פאר א AI-סיסטעם וואס שאפט א טעכנישן קורס. א פארשונג-אגענט קלייבט צונויף אינפארמאציע; א פלאנירונג-אגענט ארגאניזירט די סטרוקטור פונעם קורס; א שרייב-אגענט שאפט די ערקלערונגען; און א איבערקוק-אגענט באשטעטיגט די געניקייט און קוואליטעט. דערצו קען א פארוואלטער-אגענט (manager agent) קאארדינירן אלע די טעטיקייטן און צונויפשטעלן דעם ענדגילטיגן רעזולטאט.

דער קורס ברענגט אז IBM ערקלערט מולטי-אגענט סיסטעמען מיטן משל פון א בינען-קאלאניע: איין בין קען טון איינצלנע ארבעטן, אבער א גרופע בינען וואס ארבעטן צוזאמען קענען בויען און אויפהאלטן א גאנצן בינשטאק. גלייכערהייט קענען עטליכע AI-אגענטן וואס ארבעטן צוזאמען לייזן קאמפליצירטע פראבלעמען מער עפעקטיוו.

פארוואס דארף מען עטליכע אגענטן (29:48:20)

א שאלה: אויב איין AI-אגענט קען אנפירן מיט אסאך ארבעטן, פארוואס דארף מען בכלל עטליכע? דער ענטפער איז: קאמפליצירטקייט. אין דער ווירקליכקייט נעמען פראבלעמען געווענליך אריין עטליכע טריט און פארשידענע געביטן פון מומחיות. למשל, שאפן א ביזנעס-באריכט קען פארלאנגען פארשונג, דאטא-אנאליז, שרייבן און קוואליטעט-קאנטראל. ערווארטן אז איין אגענט זאל אלעס טון פערפעקט קען ווערן שווער.

מולטי-אגענט סיסטעמען לייזן דעם פראבלעם דורך צעטיילן די אחריותן: איין אגענט קאנצענטרירט זיך אויף פארשונג, א צווייטער אויף אנאליז, א דריטער אויף באשלוסן, און א פערטער אויף איבערקוקן דעם רעזולטאט. דאס פארבעסערט ספעציאליזאציע און ארגאניזאציע, ווייל יעדער אגענט ווערט פאראנטוואָרטליך פאר א באשטימטע ארבעט, אנשטאט צו פרובירן אנצופירן מיט אלעס.

דאס איז זייער ענליך צו ווי א מענטשליכע מאנשאפט ארבעט: א וווייכווארג-פראיעקט ווערט נישט פארענדיקט דורך איין מענטש וואס טוט אליין דעם אנטוויקלונג, טעסטינג, דיזיין און פארוואלטונג. פארשידענע ספעציאליסטן ארבעטן צוזאמען.

אבער א ווארנונג: מער אגענטן מיינט נישט שטענדיג בעסערע רעזולטאטן. דער ציל איז נישט צו שאפן אסאך אגענטן אומזיסט, נאר צו שאפן די ריכטיגע אגענטן מיט קלארע אחריותן.

איין אגענט קעגן עטליכע אגענטן (29:49:34)

א סיסטעם מיט איין אגענט איז נוצלעך ווען די ארבעט איז פשוט און פארלאנגט באגרעניצטע טריט, למשל ענטפערן פראגעס, צוזאמענפאסן דאקומענטן אדער שאפן יסודותדיגן אינהאלט. דאס קען איין אגענט אנפירן.

א מולטי-אגענט סיסטעם ווערט נוצלעך ווען די ארבעט נעמט אריין עטליכע שטאפלען און פארלאנגט פארשידענע פעאיקייטן. למשל, בויען א וווייכווארג-אפליקאציע קען פארלאנגען פלאנירונג, קאוד-שרייבן,

טעסטינג, זיכערהייט-קאנטראל און דאקומענטאציע. אנשטאט אז איין אגענט זאל אלעס טון, קענען פארשידענע אגענטן זיך ספעציאליזירן אין פארשידענע געביטן.

דער משל: איין אגענט איז ווי אן אלגעמיינער ארבעטער וואס קען טון פארשידענע ארבעטן, בשעת א מולטי-אגענט סיסטעם איז ווי א מאנשאפט ספעציאליסטן וואס ארבעטן צוזאמען. די ברירה הענגט אפ אין דער קאמפליצירטקייט פונעם פראבלעם: ווען דער פראצעס פארלאנגט מיטארבעט, עטליכע געצייג און פארשידענע סארטן מומחיות, ווערט דער מולטי-אגענט צוגאנג מער עפעקטיוו.

די הויפט-טיילן פון א מולטי-אגענט סיסטעם (29:50:36)

דער קורס רעכנט אויס פינף וויכטיגע קאמפאנענטן:

- די אגענטן אליין: יעדער אגענט דארף האבן א קלארע אחריות. למשל, א פארשונג-אגענט קלייבט צונויף אינפארמאציע, בשעת א איבערקוק-אגענט באשטעטיגט די קוואליטעט פונעם רעזולטאט.
- דער ארקעסטראטאר אדער פארוואלטער-אגענט: ער קאארדינירט דעם פראצעס, ער באשליסט וועלכער אגענט זאל אנפירן מיט וועלכער ארבעט, און ווי די פארשידענע ארבעטן זאלן זיין פארבונדן איינס מיטן אנדערן.
- געצייג (tools): אגענטן ווערן מער נוצלעך וען זיי ארבעטן מיט דרויסנדיגע מקורות, ווי דאטאבאזעס, API's, דאקומענטן און אפליקאציעס.
- זכרון (memory): זכרון העלפט אגענטן אויפהאלטן דעם קאנטעקסט און געדענקען וויכטיגע אינפארמאציע במשך דעם פראצעס.
- קאמוניקאציע: אגענטן דארפן א וועג אויסצוטוישן אינפארמאציע, ווייל דער רעזולטאט פון איין אגענט ווערט אפט דער אריינגאב פונעם צווייטן.

צוזאמען דערמעגליכן די אלע טיילן פאר עטליכע אגענטן צו ארבעטן מיטאנאנד און פארענדיקן קאמפליצירטע ארבעטן.

ווי אזוי ארבעט א מולטי-אגענט סיסטעם, שריט ביי שריט (29:51:36)

דער פראצעס הייבט זיך געווענליך אן ווען דער באניצער גיט א ציל. אנשטאט גלייך ארויסצוגעבן אן ענטפער, פארשטייט די סיסטעם פריער וואס מען דארף דערגרייכן, און צעטיילט דעם גרויסן ציל אויף קלענערע ארבעטן.

דער ביישפיל: דער ציל איז "שאף א פולשטענדיגן מארק-פארשונג באריכט פאר א נייעם פראדוקט". דאס איז נישט קיין איינציגע ארבעט; עס פארלאנגט צונויפקלייבן מארק-אינפארמאציע, אנאליזירן די קאנקורענטן, פארשטיין וואס די קונים דארפן, און צוגרייטן א געארדנטן באריכט. די מולטי-אגענט סיסטעם צעטיילט דעם גרויסן פראבלעם אויף פשוטערע טעטיקייטן און גיט יעדע טעטיקייט צום מערסט פאסיגן אגענט.

דער נעקסטער טריט איז צוטיילן די ארבעטן: א פארוואלטער- אדער אפעראטאר-אגענט באשליסט וועלכער אגענט זאל אנפירן מיט יעדער אחריות. דער פארשונג-אגענט קלייבט צונויף אינפארמאציע, דער אנאליז-אגענט לערנט די דאטא, דער שרייב-אגענט גרייט צו דעם באריכט, און דער איבערקוק-אגענט באשטעטיגט דעם ענדגילטיגן רעזולטאט. נאכן פארענדיקן זייערע איינצלנע ארבעטן, רעדן די אגענטן איינער מיטן אנדערן און טיילן אינפארמאציע, אזוי אז דער רעזולטאט פון איינעם ווערט דער אריינגאב פונעם צווייטן. צום סוף שטעלט די סיסטעם אלעס צונויף און גיט איבער דעם ענדגילטיגן לייזונג צום באניצער.

אין קורצן קען מען דעם גרונט-פראצעס פארשטיין אזוי:

- פארשטיין דעם ציל.
- צעטיילן די ארבעט אויף קלענערע טעטיקייטן.
- צוטיילן אחריותן.
- לאזן די אגענטן ארבעטן מיטאנאנד.
- שאפן דעם ענדגילטיגן רעזולטאט.

ראלעס און ספעציאליזאציע (29:53:00)

איינער פון די גרעסטע מעלות פון מולטי-אגענט סיסטעמען איז ספעציאליזאציע. אנשטאט צו שאפן איין אגענט וואס טוט אלעס, שאפט מען פארשידענע אגענטן וואס קאנצענטרירן זיך אויף באשטימטע ארבעטן. א פלאנירונג-אגענט נעמט די אינפארמאציע און שאפט א ריכטיגע סטרוקטור; א שרייב-אגענט פארוואנדלט דעם פלאן אין פארשטענדליכן אינהאלט; א איבערקוק-אגענט באשטעטיגט צי דער רעזולטאט איז גענוי און פולשטענדיג.

גלייכצרהייט, ביי ווייכווארג-אנטוויקלונג האט מען קאוד-אגענטן וואס שרייבן קאוד, א טעסט-אגענט וואס געפינט באגס (פעלערן), א זיכערהייט-אגענט וואס זוכט שוואכע פונקטן (vulnerabilities), און א דאקומענטאציע-אגענט וואס שאפט טעכנישע דאקומענטן.

די ספעציאליזאציע מאכט די סיסטעם מער ארגאניזירט, ווייל יעדער אגענט האט א קלארן ציל. עס מאכט אויך גרינגער צו פארבעסערן איינצלנע טיילן: אויב די קוואליטעט פון דער פארשונג איז שוואך, קען מען פארבעסערן דעם פארשונג-אגענט אן טוישן דעם גאנצן פראצעס.

אבער עס ווערט געווארנט אז מען מוז דעפינירן קלארע ראלעס: ווען עטליכע אגענטן האבן די זעלבע אחריות, קענען זיי טאפלען די ארבעט אדער שאפן א צומיש. א געלונגענע מולטי-אגענט סיסטעם פארלאנגט אז יעדער אגענט זאל פארשטיין פאר וואס ער איז פאראנטוואורטליך און ווי ער טראגט בייַ צום ענדגילטיגן ציל.

קאמוניקאציע צווישן די אגענטן (29:54:07)

דער קורס גייט איבער צו דער פראגע ווי די אגענטן רעדן מיטאנאנד און ארבעטן צוזאמען. קאמוניקאציע איז איינער פון די וויכטיגסטע טיילן פון א מולטי-אגענט סיסטעם, ווייל אגענטן דארפן אויסטוישן אינפארמאציע כדי צו פארענדיקן א געמיינזאמען ציל. אן קאמוניקאציע וואלטן די איינצלנע אגענטן געארבעט באזונדער, אן בייטראגן צום ענדגילטיגן רעזולטאט. דער קורס הייבט אן מיט א משל פון א מאנשאפט אין דער ווירקליכקייט: א פארשער קען נישט פשוט פארענדיקן זיין ארבעט און אויפהערן; ער דארף טיילן וואס ער האט געפונען מיט די אנדערע.

קאמוניקאציע, ארכיטעקטורן און מוסטערן פון מולטי-אגענט סיסטעמען (29:54:38)

ווי אגענטן רעדן איינער מיטן אנדערן (29:54:38)

דער קורס פירט ווייטער מיט א משל פון א מענטשליכן טיעם: א פארשער טיילט זיינע געפינסן מיט אן אנאליסט, דער אנאליסט מוז געבן זיינע איינזיכטן צום שרייבער, און דער שרייבער דארף באקומען אן אפרוף

פונעם רעוויוער. יעדעס מענטשנס ארבעט איז פארבונדן מיט עמעצן אנדערש. פונקט אזוי טיילן אגענטן אינפארמאציע, געבן איבער דערהיינטיגונגען, און פירן רעזולטאטן איבער פון איינעם צום אנדערן.

א ביישפיל: א פארש-אגענט קען איבערגעבן די געזאמלטע דאטא צו אן אנאליז-אגענט. דער אנאליז-אגענט פראצעסירט די אינפארמאציע און שיקט איינזיכטן ווייטער צו אן אגענט וואס איז פאראנטווארטלעך פאר באשלוסן.

עס ווערט ערקלערט אז די קאמוניקאציע קען פאסירן אויף פארשידענע אופנים. אין געוויסע סיסטעמען קאנטראלירט א "מענעדזשער-אגענט" די קאמוניקאציע: ער טיילט אויס ארבעטן און זאמלט צונויף די רעזולטאטן. אין אנדערע סיסטעמען רעדן די אגענטן גלייך איינער מיטן אנדערן און ארבעטן צוזאמען לויט ווי די סיטואציע פארלאנגט.

די שוועריקייט איז צו האלטן די קאמוניקאציע קלאר. אויב צופיל אגענטן רעדן אן א ריכטיגע ארגאניזאציע, ווערט דער סיסטעם קאמפליצירט. דערפאר ניצן אסאך מאדערנע מולטי-אגענט סיסטעמען א קאארדינאטאר-אגענט אדער א סופערווייזער-אגענט וואס פירט די אינטעראקציעס.

דריי הויפט-ארכיטעקטורן (29:55:30)

די ארכיטעקטור באשטימט ווי די אגענטן זענען ארגאניזירט און ווי זיי ארבעטן מיט'נאנד. פארשידענע פראבלעמען פארלאנגען פארשידענע סטרוקטורן.

- די ערשטע: די סופערווייזער- אדער מענעדזשער-ארכיטעקטור. דא שפילט איין הויפט-אגענט די ראלע פון קאארדינאטאר. דער סופערווייזער באקומט די בקשה פונעם באניצער, באשליסט וועלכער אגענט זאל באהאנדלען יעדע ארבעט, און שטעלט צונויף דעם ענדגילטיגן רעזולטאט. דער צוגאנג איז לייכטער צו פירן, ווייל עס איז דא א קלארער באשליסער. א ביישפיל: אין א קאסטומער-סופּאָרט סיסטעם קען דער סופערווייזער-אגענט באקומען קלאגעס פון קונים און זיי איבערגעבן צו פארשידענע ספעציאליסט-אגענטן, ווי פאר בילינג (חשבונות), טעכנישע הילף, אדער פירן אקאונטס.
- די צווייטע: די דעצענטראליזירטע ארכיטעקטור. דא איז נישטא קיין איינציגער קאנטראלירנדער אגענט. די אגענטן רעדן גלייך און נעמען באשלוסן צוזאמען. דאס קען זיין נוצלעך ווען בייגעוודיגקייט איז וויכטיג, אבער פירן די קאמוניקאציע ווערט שווערער.
- די דריטע: די היעראַרכישע ארכיטעקטור. דא זענען די אגענטן ארגאניזירט אין פארשידענע שטאפלען; א העכערער אגענט פירט עטליכע נידעריגערע אגענטן.

דער קורס באטאָנט אז די ריכטיגע ארכיטעקטור הענגט אָפּ אינעם פראבלעם: פשוט'ע ארבעט-פלוסן קענען גוט ארבעטן מיט א סופערווייזער-מאדעל, אבער קאמפליצירטע סיסטעמען קענען פארלאנגען א מער בייגעוודיגע סטרוקטור.

דיזיין-מוסטערן (29:56:40)

דערנאך גייט מען איבער צו איינע פון די וויכטיגסטע געביטן אין מולטי-אגענט סיסטעמען: דיזיין-מוסטערן (design patterns). דאס זענען געוועהנליכע אופנים ווי צו ארגאניזירן אגענטן כדי צו לייזן געוויסע סארטן פראבלעמען, און זיי געבן א סטרוקטורירטן צוגאנג צו בויען אזעלכע סיסטעמען.

• ערשטער מוסטער – דער סעקוועניציעלער מוסטער: די אגענטן ארבעטן איינס נאכן אנדערן. למשל, א פארש-אגענט זאמלט אינפארמאציע, א שרייב-אגענט שאפט דעם אינהאלט, און א רעוויו-אגענט קאנטראלירט דעם ענדגילטיגן רעזולטאט.

• צווייטער מוסטער – דער פאראלעלער מוסטער: עטליכע אגענטן ארבעטן גלייכצייטיג אויף פארשידענע טיילן פונעם פראבלעם. למשל, בשעת איין אגענט אנאליזירט די קאנקורענטן, שטודירט אן אנדערער אגענט די אפרופן פון קונים, און זייערע רעזולטאטן ווערן שפעטער צונויפגעשטעלט.

• דריטער מוסטער – דער ראוטער-מוסטער: א ראוטער-אגענט באשליסט וועלכער ספעציאליסט-אגענט זאל באהאנדלען א געוויסע בקשה. דאס ניצט מען אָפט אין קאסטומער-סופּאָרט סיסטעמען, ווו פארשידענע פראבלעמען פארלאנגען פארשידענע עקספערטן.

• פערטער מוסטער – דער רעוויועור- אדער קריטיקער-מוסטער: איין אגענט שאפט דעם רעזולטאט און אן אנדערער קוקט נאך טעותים און שלאגט פאר פארבעסערונגען. דאס הייבט די קוואליטעט, ווייל דער סיסטעם באקומט א צווייטע שטאפל פון באשטעטיגונג.

• פינפטער מוסטער – דער נעץ-מוסטער (network): די אגענטן רעדן גלייך איינער מיטן אנדערן אנשטאט צו גיין לויט א פעסטע סדר.

די מוסטערן ווייזן אז מולטי-אגענט סיסטעמען באשטייען נישט פשוט אין צולייגן מער AI אגענטן; דער עיקר איז צו פלאנירן די ריכטיגע סטרוקטור פון צוזאמענארבעט. דער קורס ברענגט אז LangChain באטאָנט אויך אז נישט יעדער פראבלעם פארלאנגט עטליכע אגענטן: אמאל קען איין אגענט מיט די ריכטיגע געצייג גוט דורכפירן א געוויסע ארבעט. דער ציל איז אויסצוקלייבן דעם ריכטיגן צוגאנג לויט ווי קאמפליצירט דער ארבעט-פלוס איז.

ווי מען ניצט עס אין דער פראקטיק (29:58:20)

עס ווערט ערקלערט אז מולטי-אגענט סיסטעמען כאפן אויפמערקזאמקייט ווייל אסאך פראבלעמען אין דער ריאלער וועלט פארלאנגען עטליכע טריט, פארשידענע סארטן מומחיות, און אן אנגייענדיג באשלוס-מאכן.

• קאסטומער-סופּאָרט: אנשטאט איין AI סיסטעם וואס באהאנדלט יעדע בקשה, ארבעטן עטליכע אגענטן צוזאמען. אן אגענט וואס פארשטייט דעם קונה אידענטיפיצירט דעם פראבלעם; אן אקאונט-אגענט באשטעטיגט די פרטים פונעם קונה; א פאליסי-אגענט קוקט וועלכע לייזונגען זענען פאראן; און אן ענטפער-אגענט גרייט צו דעם ענדגילטיגן ענטפער. דאס דערמעגליכט ביזנעסער צו געבן שנעלערע און מער פערזענליכע הילף.

• ווייכווארג-אנטוויקלונג: היינט ניצט מען AI נישט נאר צו שאפן קאוד, נאר אויך צו העלפן דורכן גאנצן אנטוויקלונג-ציקל. א פלאנירונג-אגענט פארשטייט די פארלאנגען פונעם פראיעקט; א קאודירונג-אגענט שאפט די אויספירונג; א טעסט-אגענט געפינט באגס; און א זיכערהייט-אגענט זוכט שוואכע פונקטן. דאס שאפט א טיעם וואס ארבעט מיט AI און קען שטיצן די פראגראמירער אין פארשידענע שטאפלען.

• פארשונג און אנאליז: א פארש-אגענט זאמלט אינפארמאציע פון פארשידענע מקורות, אן אנאליז-אגענט געפינט די וויכטיגע איינזיכטן, און א סאמערי-אגענט גרייט צו דעם ענדגילטיגן באריכט. דאס העלפט פראפעסיאנאלן פראצעסירן גרויסע מאסן אינפארמאציע מער עפעקטיוו.

• אנדערע אינדוסטריעס, ווי געזונטהייט, פינאנצן, סייבער-זיכערהייט און קליניקהאנדל, פרואוון אויך אזעלכע סיסטעמען. אין סייבער-זיכערהייט קען איין אגענט אכטונג געבן אויף חשד'דיגע אקטיוויטעטן, אן

אנדערער אויספארשן די סכנות, און א דריטער פארשלאגן מעגליכע שריט. אין פינאנצן קענען אגענטן אנאליזירן מארקעט-אינפארמאציע, געפינען מוסטערן, און שטיצן באשלוסן.

דער קורס ברענגט אז Cognizant באטאָנט אז מולטי-אגענט סיסטעמען האבן א פאטענציאל אין אינדוסטריעס ווי געזונטהייט, באנקינג, קליניקהאנדל, אינשורענס און סייבער-זיכערהייט, ווייל פארשידענע אגענטן קענען באהאנדלען ספעציאליזירטע ארבעטן בשעת זיי ארבעטן צוזאמען.

א ביישפיל: אן AI טיעם צו שאפן אינהאלט (30:00:13)

צו פארשטיין די זאך פראקטיש, נעמט דער קורס א פשוט'ן ביישפיל: א טיעם פון AI אגענטן וואס שאפט אינהאלט. דער ציל: צו שרייבן א לעקציע פאר אנהייבער אויף א טעכנישן נושא. אנשטאט צו בעטן איין אגענט ער זאל אליין טון אלעס, טיילט מען די ארבעט צווישן עטליכע ספעציאליזירטע אגענטן:

- ערשטנס – דער פארש-אגענט: זאמלט וויכטיגע אינפארמאציע וועגן דעם נושא, אידענטיפיצירט די הויפט-באגריפן, און זאמלט צונויף פאסיגע ביישפילן.
- צווייטנס – דער פלאנירונג-אגענט: נעמט די פארש-אינפארמאציע און שאפט א ריכטיגע סטרוקטור; ער באשליסט וועלכער נושא קומט ערשט און ווי די ערקלערונג זאל פליסן.
- דריטנס – דער שרייב-אגענט: פארוואנדלט דעם פלאן אין א דעטאלירטן אינהאלט וואס לערנער קענען לייכט פארשטיין.
- פערטנס – דער רעוויו-אגענט: קאנטראלירט צי עס פעלט אינפארמאציע, צי עס זענען דא אומקלארע ערקלערונגען אדער פאלשע פרטים.
- צום סוף – דער מענעדזשער-אגענט: קאארדינירט דעם גאנצן ארבעט-פלוס, טיילט אויס די ארבעטן, זאמלט די רעזולטאטן פון די פארשידענע אגענטן און שטעלט אלעס צונויף אין דעם ענדגילטיגן רעזולטאט.

דער ביישפיל ווייזט די הויפט-מעלה: יעדער אגענט קאנצענטרירט זיך אויף איין פאראנטוואָרטלעכקייט, און צוזאמען פירן זיי דורך א גרעסערע ארבעט. פריימווארקס ווי CrewAI און LangGraph ווערן אָפט גענוצט צו בויען אזעלכע ארבעט-פלוסן, ווייל זיי געבן מעגליכקייטן צו שאפן ספעציאליזירטע אגענטן, פירן די קאמוניקאציע און קאנטראלירן דעם פלוס. דער וויכטיגער פונקט: די קראפט קומט פון דער צוזאמענארבעט, נישט פשוט פון שאפן עטליכע אגענטן.

די שווערקייטן (30:01:50)

כאטש מולטי-אגענט סיסטעמען ברענגען אסאך מעלות, ברענגען זיי אויך נייע שווערקייטן. מער אגענטן זענען נישט קיין גאראנטיע פאר בעסערע רעזולטאטן; דער סיסטעם דארף ריכטיגע פלאנירונג, קאארדינאציע און אויפזיכט.

- קאמפליצירטקייט פון קאמוניקאציע: ווען מען לייגט צו מער אגענטן, פאסירן מער אינטעראקציעס. אויב די קאמוניקאציע ווערט נישט ריכטיג געפירט, קענען אגענטן איבערחזר'ן ארבעטן, נישט פארשטיין אינפארמאציע, אדער שאפן אומנייטיגע טריט.
- קלארע פראנטוואָרטלעכקייטן: יעדער אגענט האט א באשטימטע ראלע. אויב צווי אגענטן טוען די זעלבע ארבעט, אדער קיין אגענט איז נישט פראנטוואָרטלעך פאר א געוויסע זאך, ווערט דער פלוס נישט עפעקטיוו.

• האלטן ענויקייט: אפילו וועו אגענטן ארבעטן צוזאמען, קענען זיי נאך אלץ שאפן פאלשע אינפארמאציע. א רעוויו-פראצעס איז וויכטיג צו באשטעטיגן די רעזולטאטן איידער מען גיט ארויס דעם ענדגילטיגן רעזולטאט.

• קאסטן און פערפארמענס: עטליכע אגענטן פארלאנגען מער קאמפיוטער-רעסורסן ווי א פשוט'ער איינציגער אגענט. ארגאניזאציעס דארפן אפוועגן די קאמפליצירטקייט קעגן דעם ריאלן ווערט וואס עס ברענגט.

• זיכערהייט: ווייל אגענטן קענען ארבעטן מיט דרויסנדיגע געצייג און דאטא-מקורות, דארף מען ריכטיגע קאנטראלן צו פארזיכערן א זיכערע באניצונג.

דער מסקנא: א געלונגענער מולטי-אגענט סיסטעם ווערט נישט געשאפן דורכן פשוט צולייגן אגענטן; ער פארלאנגט פארזיכטיגע פלאנירונג, קלארע פאראנטוואורטלעכקייטן און אן אנגייענדיגע פארבעסערונג.

בעסטע פראקטיקן און טעותים (30:03:20)

• הייבט אן פשוט: אנשטאט צו שאפן פון אנהייב אסאך אגענטן, הייבט אן מיט א קליינע צאל ספעציאליזירטע אגענטן, און פארגרעסערט נאר וועו עס איז נויטיג.

• קלארע כללים: יעדער אגענט זאל האבן א באשטימטע פאראנטוואורטלעכקייט – א פארש-אגענט זאל זיך קאנצענטרירן אויף פארשונג, א רעוויו-אגענט אויף קאנטראלירן קוואליטעט, און א פלאנירונג-אגענט אויף ארגאניזירן די ארבעטן.

• א ריכטיגער פלוס פון קאמוניקאציע: די אגענטן זאלן וויסן ווי און וועו צו רעדן, וועלכע אינפארמאציע צו טיילן, און ווי באשלוסן זאלן געמומען ווערן.

• צולייגן אפשאצונג-טריט: א לעצטער רעוויו-אגענט אדער א מענטשליכער באשטעטיגונג-פראצעס קען כאפן טעותים איידער דער רעזולטאט דערגרייכט דעם באניצער.

דער קורס וואלט פון צוויי געוועלטיגע טעותים: ערשטנס, מיינען אז מער אגענטן ברענגען שטענדיג בעסערע רעזולטאטן – אין דער ווירקלעכקייט קענען אומנייטיגע אגענטן מאכן דעם סיסטעם פיל פאמעלעכער און קאמפליצירטער. צווייטנס, שאפן אגענטן אן א קלארן ציל. דער ציל איז נישט קאמפליצירטקייט, נאר אן ארגאניזירטער פלוס וואו פארשידענע אגענטן ארבעטן עפעקטיוו צוזאמען.

די צוקונפט און א סך-הכל (30:04:30)

לויט דעם קורס באוועגט זיך AI פון סיסטעמען וואס ענטפערן נאר, צו סיסטעמען וואס העלפן אקטיוו דורכפירן ארבעטן. מולטי-אגענט סיסטעמען זענען א וויכטיגער טריט אין דער ריכטונג, ווייל זיי לאזן AI סיסטעמען צוזאמענארבעטן, זיך ספעציאליזירן, און באהאנדלען קאמפליצירטערע פלוסן. מען קען ערווארטן אז מער ארגאניזאציעס וועלן ניצן AI טיעמס וואס ארבעטן לעבן מענטשן – נישט צו פארביטן מענטשליכע מומחיות, נאר צו שטיצן פראפעסיאנאלן דורך באהאנדלען איבערחזר'נדיגע ארבעטן, אנאליזירן אינפארמאציע שנעלער, און העלפן מיט באשלוסן.

ביישפילן: א ביזנעס-טיעם קען האבן AI אגענטן וואס שטיצן מארקעט-פארשונג, קונה-אנאליז, פינאנציעלע פלאנירונג און באריכטן; א וויכווארג-טיעם קען האבן אגענטן וואס העלפן מיט קאודירן, טעסטן און דאקומענטאציע. אבער מענטשן וועלן בלייבן וויכטיג: זיי וועלן באשטימען די צילן, נעמען וויכטיגע באשלוסן, ברענגען שעפערישקייט, און זיכער מאכן אז AI ווערט גענוצט אויף א פראנטוואורטלעכן אופן. די צוקונפט, לויט דעם קורס, איז נישט נאר קלוגערע AI, נאר בעסערע צוזאמענארבעט צווישן מענטשן און AI.

אין א סך-הכל: א מולטי-אגענט סיסטעם איז א גרופע AI אגענטן וואס ארבעטן צוזאמען צו א געמיינזאמען ציל. AI אגענטן קענען פארשטיין צילן, פלאנירן שריט, ניצן געצייג און דורכפירן ארבעטן. מען האט געזעדט וועגן וואס אן AI אגענט איז, ווי AI האט זיך אנטוויקלט פון פשוט'ע טשעטבאטס צו סיסטעמען וואס פירן דורך ארבעטן, פארוואס עטליכע אגענטן זענען נוצלעך, דעם חילוק צווישן איין אגענט און מולטי-אגענט, ארכיטעקטורן און דיזיין-מוסטערן, אנוועדונגען אין קאסטומער-סופּאָרט, ווייכווארג, סייבער-זיכערהייט, פינאנצן און פארשונג, און ווי צו פלאנירן עפעקטיווע סיסטעמען מיט קלארע ראלעס, ריכטיגע קאמוניקאציע, אפשאצונג און פראקטישע פלוסן. דער הויפט-געדאנק: א מולטי-אגענט סיסטעם איז ווי א דיגיטאלער טיעם וואו יעדער אגענט האט א באשטימטע פאראנטוואָרטלעכקייט, און צוואמען לייזן זיי א פראבלעם וואס איז שווער פאר איין AI סיסטעם אליין.

אנהייב פונעם פראקטישן דעמא מיט CrewAI (30:06:35)

דערמיט ענדיגט זיך די פרעזענטאציע, און מען גייט איבער צום פראקטישן דעמא. מען וועט בויען א פשוט'ן מולטי-אגענט סיסטעם וואו פארשידענע אגענטן ארבעטן צוזאמען ווי א קליינער טיעם וואס שאפט אינהאלט: איין אגענט פאר פארשונג, אן אנדערער פאר ארגאניזירן די אינפארמאציע, און א דריטער פאר רעוויו פונעם ענדגילטיגן רעזולטאט.

פארן דעמא ניצט מען CrewAI, א פריימווארק וואס העלפט שאפן און פירן AI אגענט-ארבעט. דער געדאנק איז פשוט: מען באשטימט פארשידענע אגענטן, גיט זיי פאראנטוואָרטלעכקייטן און ארבעטן, און לאזט זיי צוזאמענארבעטן צו דערגרייכן א פולן ציל.

איידער מען שאפט די אגענטן, גרייט מען צו די ארבעט-סביבה. מען דארף דריי זאכן; די ערשטע צוויי ווערן דא דערמאנט:

- ערשטנס, Python אינסטאלירט אויפן סיסטעם, ווייל CrewAI ארבעט נאר מיט Python.
- צווייטנס, א קאוד-עדיטאר; פאר דעם דעמא ניצט מען Visual Studio Code.

בויען א מולטי-אגענט סיסטעם מיט CrewAI (30:07:41)

צוגרייטן די ארבעטס-סביבה (30:07:41)

דער קורס ערקלערט אז צו בויען דעם פראיעקט דארף מען דריי זאכן: Python אויפן קאמפיוטער, א סביבה צו שאפן און לויפן Python פראיעקטן, און א צוטריט צו אן AI מאדעל דורך אן API שליסל. דער API שליסל איז נויטיג ווייל די אגענטן דארפן א שפראך-מאדעל כדי צו פארשטיין אנווייזונגען און ארויסגעבן תשובות.

מען הייבט אן מיט שאפן א נייעם פראיעקט-פאלדער. מען עפנט דעם טערמינאל און מען שאפט א פאלדער מיטן באפעל mkdir multi-agent-content-system. נאכדעם גייט מען אריין אינעם פאלדער מיט cd multi-agent-content-system. די צוויי באפעלן שאפן דעם פאלדער און פירן דעם טערמינאל אריין אהין, וואו די אפליקאציע וועט געבויט ווערן.

דערנאך שאפט מען א Python virtual environment מיטן באפעל m venv venv- python. עס ווערט ערקלערט אז דער באפעל שאפט אן אפגעזונדערטע סביבה פארן פראיעקט, אזוי אז די נויטיגע לייברעריס ווערן אינסטאלירט באזונדער, אן צו אנרירן קיין אנדערע Python פראיעקטן.

דערנאך אקטיווירט מען די סביבה. אויף Windows שרייבט מען venv\Scripts\activate. וועז די סביבה איז אקטיוו, ווערן אלע קומענדיגע אינסטאלאציעס געמאכט אינעווייניג אינעם פראיעקט.

אינסטאלירן CrewAI און שאפן די פיילס (30:08:44)

יעצט אינסטאלירט מען CrewAI מיטן באפעל pip install crewai. דער קורס ווארנט אז מען זאל אכטונג געבן אז קיין אות אין crewai זאל נישט זיין א גרויסער אות. דער באפעל לאדנט אראפ און אינסטאלירט דעם CrewAI פריימווערק, וואס גיט די מעגליכקייט צו שאפן אגענטן, זיי צוטיילן אויפגאבעס, און פירן זייער צוזאמענארבעט.

מען דארף אויך שטיצע פאר environment variables, דעריבער אינסטאלירט מען python-dotenv. דער פעקל דערלויבט צו היטן קאנפיגוראציע-פרטים ווי API שליסלען אויף א זיכערן אופן, אן זיי שרייבן גלייך אינעם קאוד.

דערנאך עפנט מען VS Code מיטן פאלדער multi-agent-content-system און מען שאפט דריי פיילס:

- main.py
- agents.py
- tasks.py

דער צוועק פון יעדן פייל:

- agents.py האלט די דעפיניציע פון די AI ארבעטער.
- tasks.py האלט די אחריותן וואס ווערן צוגעטיילט צו יעדן אגענט.
- main.py פארבינדט אלעס צוזאמען און פירט אויס דעם גאנצן פראצעס.

דער .env פייל און דער API שליסל (30:10:16)

אויסער די דריי פיילס שאפט מען נאך א פייל מיטן נאמען .env, וואס היט זיכער דעם API שליסל. אין דער לעקציע ווערט אריינגעלייגט א צופעליגער שליסל אין דעם פייל. עס ווערט ערקלערט אז דער פייל היט דעם אויטענטיקאציע-שליסל וואס מען דארף צו פארבינדן די אפליקאציע מיטן שפראך-מאדעל. אנשטאט צו שרייבן סענסיטיווע אינפארמאציע גלייך אינעם Python קאוד, לאדנט מען זי ארויס פונעם environment פייל.

דער ערשטער אגענט: דער פארשונג-אגענט (30:10:46)

נאכן צוגרייטן הייבט מען אן בויען דעם ערשטן AI אגענט, דעם research agent. זיין אחריות איז פשוט: צונויפקלייבן וויכטיגע אינפארמאציע ווען א געגעבענעם טעמע. אין agents.py שרייבט מען:

- מען אימפארטירט Agent פון crewai.
- מען שאפט research_agent = Agent(...) מיט role "Research Specialist".
- goal: צונויפקלייבן גענויע און נוצליכע אינפארמאציע ווען דעם געגעבענעם טעמע.
- backstory: "דו ביסט אן עקספערט פארשער וואס געפינט וויכטיגע אינפארמאציע און ארדנט אויס די הויפט-פונקטן".
- verbose=True.

עס ווערט ערקלערט: דער role באשטימט וואס דער אגענט טוט, דער goal ערקלערט דעם ציל, און דער backstory גיט צוגעלייגטן קאנטעקסט כדי דער אגענט זאל פארשטיין ווי ער דארף זיך פירן. דער פארשונג-אגענט איז נישט פאראנטוואורטליך צו שרייבן די לעצטע תשובה; זיין איינציגע אחריות איז צונויפקלייבן און אויסארדענען די אינפארמאציע.

דער צווייטער אגענט: דער שרייבער (30:11:46)

דער צווייטער אגענט פארוואנדלט די פארשונג-אינפארמאציע אין פארשטענדליכן אינהאלט. אין agents.py לייגט מען צו writer_agent = Agent(...) מיט:

- "Content Writer" role.
- goal: שאפן קלארן און פריינטליכן אינהאלט מיט דער פארשונג-אינפארמאציע.
- backstory: "דו ביסט אן עקספערט טעכנישער שרייבער וואס ערקלערט קאמפליצירטע טעמעס אויף א פשוטן אופן".
- verbose=True.

אין דער לעקציע ווערט פארריכט א טעות, וואו מען דארף פארבייטן "is" מיט "=". אנדערש ווי דער פארשונג-אגענט, קאנצענטרירט זיך דער אגענט אויף קאמוניקאציע און ערקלערונג. דאס ווייזט דעם הויפט-מעלה פון א מולטי-אגענט סיסטעם: יעדער אגענט האט א פאקוסירטע אחריות, אנשטאט צו פרובירן אויספירן יעדע אויפגאבע.

דער דריטער אגענט: דער איבערקוקער (30:12:50)

דער לעצטער אגענט קאנטראלירט די קוואליטעט פונעם רעזולטאט. גלייך אונטערן writer agent שרייבט מען review_agent = Agent(...) מיט:

- "Quality Reviewer" role.
- goal: איבערקוקן דעם אינהאלט און אויפווייזן פעלנדיגע אינפארמאציע אדער פארבעסערונגען.
- backstory: "דו ביסט אן איבערקוקער וואס קאנטראלירט די גענויקייט און קלארקייט פארן לעצטן איבערגעבן".
- verbose=True.

דער אגענט שאצט אפ דעם רעזולטאט אנשטאט אים צו שאפן. א באזונדערער איבערקוקער פארבעסערט די קוואליטעט, ווייל דער סיסטעם באקומט נאך א שטאפל פון קאנטראל. אזוי האט מען דריי אגענטן: research agent, writer agent און review agent.

דעפינירן די אויפגאבעס אין tasks.py (30:13:52)

דער נעקסטער שריט איז באשטימען די אחריותן פון יעדן אגענט. אין א מולטי-אגענט סיסטעם דארפן אגענטן קלארע אנווייזונגען, ווייל יעדער אגענט דארף וויסן געניו וואס מען דארף אויפטון. אין tasks.py:

- מען אימפארטירט Task פון crewai, און פון agents אימפארטירט מען research_agent, writer_agent, review_agent.
- research_task = Task(...) מיט description: "פארש די לעצטע אינפארמאציע און וויכטיגע באגריפן וועגן {topic}. קלייב צונויף הויפט-פונקטן, ביישפילן און פראקטישע באנוצן". expected_output: א

געארדנטע פארשונג-סומע מיט וויכטיגע אינפארמאציע און אינדוסטריע-אנווענדונגען. :agent research_agent.

• writing_task = Task(...) מיט description: "שאף א ערקלערונג פאר אנהייבער וועגן {topic} מיט דער פארשונג. מאך דעם אינהאלט פשוט, לייכט צו פארשטיין, קלאר און אינטערעסאנט". :expected_output א פולע ערקלערונג אין טוטאריאל-סטיל מיט ריכטיגן סטרוקטור און ביישפילן. :agent writer_agent.

• review_task = Task(...) מיט description: "קוק איבער דעם אינהאלט וועגן {topic}. קאנטראליר גענויקייט, קלארקייט און פעלנדיגע אינפארמאציע, און פארשלאג וואו מען דארף פארבעסערן". :expected_output א איבערקוק-באריכט מיט באמערקונגען און פארבעסערונג-פארשלאגן. :agent review_agent.

עס ווערט ערקלערט: דער description זאגט דעם אגענט וואס פאר א ארבעט מען דארף פארענדיגן, דער expected_output באשטימט וועלכע רעזולטאטן מען וויל, און דער agent פאראמעטער פארבינדט די אויפגאבע מיטן ריכטיגן ספעציאליסט. די צעטיילונג איז וויכטיג ווייל זי שאפט א קלארן ארבעטס-גאנג: אנשטאט איין AI סיסטעם זאל טון אלעס, קאנצענטרירט זיך יעדער אגענט אויף איין אחריות.

פארבינדן אלעס אין main.py (30:17:37)

מען דארף א צענטראלן פייל וואס ברענגט אלעס צוזאמען. אין main.py:

• from crewai import Crew, Process

• from tasks import research_task, writing_task, review_task

• מען שאפט crew = Crew(...) מיט tasks=[research_task, writing_task, review_task], verbose=True ,process=Process.sequential.

• result = crew.kickoff(inputs={"topic": "multi-agent systems"}), און מען דרוקט אויס דעם result.

עס ווערט ערקלערט: דער Crew אביעקט איז דער קאארדינאטאר וואס פירט די אויפגאבעס. די ליסטע פון tasks האלט אלע אחריותן. Process.sequential מיינט אז איין אגענט ענדיגט זיין אויפגאבע איידער ער גיט איבער די אינפארמאציע צום נעקסטן. די פונקציע kickoff הייבט אן דעם פראצעס און גיט אריין דעם טעמע, דא "multi-agent systems".

וואס געשעט אינעווייניג: ערשט באקומט דער פארשונג-אגענט דעם טעמע און קלייבט צונויף אינפארמאציע; דערנאך גייט זי צום שרייבער, וואס שאפט א געארדנטע ערקלערונג; צום סוף קאנטראלירט דער איבערקוקער דעם אינהאלט און גיט באמערקונגען. אזוי ארבעטן מערערע ספעציאליסטן צוזאמען אין א מולטי-אגענט סיסטעם.

לויפן די אפליקאציע (30:19:39)

פארן לויפן דארף מען זיך פארזיכערן אז די virtual environment איז נאך אקטיוו און אז אלע פיילס זענען געהיטן. מען לויפט python main.py. אינעם טערמינאל זעט מען די אגענטן ארבעטן איינס נאכן אנדערן: דער פארשונג-אגענט אנאליזירט דעם טעמע, דער שרייבער באקומט די פארשונג און שאפט די ערקלערונג, און דער איבערקוקער קוקט איבער און גיט פארשלאגן. דער לעצטער רעזולטאט ווייזט דעם צוזאמענגעשטעלטן ארויסגאב פון דער אגענט-מאנשאפט.

דער קורס באטאנט: מען האט נישט געשאפן איין גרויסע אנווייזונג וואס פאדערט אלעס פון איין AI, נאר א קליינע מאנשאפט וואו יעדער אגענט האט א באשטימטע אחריות. דאס איז דער יסוד-געדאנק פון מולטי-אגענט סיסטעמען.

ווי אזוי צו פארבעסערן דעם סיסטעם (30:20:40)

דער פראצעס איז א פשוטער ביישפיל. אין א פאקטישער אפליקאציע קען מען אים פארשטארקן מיט נאך אגענטן און בעסערע קאארדינאציע, למשל:

- א fact-checking אגענט וואס באשטעטיגט אינפארמאציע פארן פארעפנטלעכן.
- א פארשונג-פלאנירונג אגענט וואס באשליסט וועלכע קוואלן מען זאל אנאליזירן.
- א feedback אגענט וואס שאצט אפ צי דער אינהאלט פאסט פארן עולם.

נאך א פארבעסערונג איז ענדערן דעם סטרוקטור. איצט ארבעט מען sequential: פארשונג, שרייבן, איבערקוקן. אבער טייל פראבלעמען דארפן parallel ארבעט: למשל, בשעת איין אגענט פארשט קאנקורענטן, אנאליזירט אן אנדערער קונה-באמערקונגען אין דער זעלבער צייט, און די רעזולטאטן ווערן שפעטער צוזאמענגעשטעלט. דאס מאכט שנעלערע פראצעסן.

אויך קען מען צולייגן געצייג: א פארשונג-אגענט פארבונדן מיט דרויסנדיגע דאטא-קוואלן, א קאודירונג-אגענט מיט אנטוויקלונג-געצייג, און א ביזנעס-אגענט מיט פירמע-דאטאבאזעס. דער הויפט-פונקט: די כח פון מולטי-אגענט סיסטעמען קומט פון פלאנירן דעם ריכטיגן פראצעס, נישט פון פשוט פארמערן די צאל אגענטן.

דעמאנסטראציע אינעם CrewAI אינטערפעיס (30:22:10)

אין דער לעקציע עפנט מען דעם CrewAI אינטערפעיס כדי צו סימולירן דעם זעלבן רעזולטאט וואס מען וואלט באקומען אין VS Code מיט python main.py. מען לייגט אריין א פראמפט וואס טוט די זעלבע ארבעט און מען לאזט עס לויפן; מען זעט ווי דער פראצעס הייבט אן.

דערבייַ ווערט איבערגעחזרט וואס מען האט געבויט: מען האט צוגעגרייט די סביבה און אינסטאלירט CrewAI, געשאפן דריי ספעציאליזירטע אגענטן — א פארשונג-אגענט, א שרייבער-אגענט און א איבערקוקער-אגענט — דעפינירט באזונדערע אויפגאבעס פאר יעדער אחריות, און זיי פארבונדן אין א פראצעס. דער אינטערפעיס בעט אריינצושרייבן א טעמע, און מען שרייבט אריין "multi-agent".

CrewAI: באגריף, אינסטאלאציע און ערשטע אגענטן (30:23:11)

סוף פונעם דעמא און דער הויפט-לימוד (30:23:11)

דער קורס ענדיגט דא א דעמא פון א מולטי-אגענט סיסטעם. מען לאזט לויפן די אפליקאציע, און מען זעט ווי פארשידענע אגענטן ארבעטן צוזאמען כדי צו דערגרייכן איין געמיינזאמען ציל. דער הויפט-לימוד פונעם דעמא: א מולטי-אגענט סיסטעם איז געבויט אויף צוזאמענארבעט. מען שאפט נישט איין AI סיסטעם וואס טוט אלעס אליין. מען שאפט כמה ספעציאליזירטע אגענטן וואס ארבעטן צוזאמען ווי א מאנשאפט, און דאס פאסט זיך בעסטנס צו לייזן קאמפליצירטע פראבלעמען. דערמיט האט מען ערפאלגרייך דורכגעפירט אן אויטאמאציע אויף CrewAI, און דער דעמא-טייל איז פארענדיגט.

דער קורס ערקלערט אז דער גאנצער טוטאריאל איז צעטיילט אין דריי מאדולן. מען גייט שריט ביי שריט, פון די יסודות ביז א פולשטענדיגע ארבעטנדיגע סיסטעם. אבער איידער מען אינסטאלירט CrewAI אדער שרייבט קאוד, דארף מען פארשטיין וואס CrewAI איז, און פארוואס אזויפיל פראגראמירער רעדן יעצט דערפון.

א משל פון א פירמע: ווי א מאנשאפט ארבעט (30:23:56)

עס ווערט ערקלערט מיט א פשוט'ע מעשה. שטעלט אייך פאר אז איר ארבעט אין א פירמע, און דער מענעדזשער גיט אייך א ארבעט: גרייטן א פרטימדיגן באריכט אויף א געוויסן נושא. געווענליך טוט נישט איין מענטש אלעס אליין, נאר די ארבעט ווערט צעטיילט:

- איינער פאנגט אן מיט פארשן דעם נושא.
- א צווייטער אנאליזירט די אינפארמאציע.
- א דריטער פארוואנדלט די איינזיכטן אין א געארדנטן באריכט.
- צום סוף קוקט א פערטער איבער דעם רעזולטאט צו זיין זיכער אז אלעס איז אין ארדנונג.

דאס הייסט: א גרופע מענטשן מיט פארשידענע כשרונות ארבעטן צוזאמען כדי צו פארענדיגן איין גרעסערע ארבעט.

יעצט שטעלט אייך פאר די זעלבע זאך, נאר אנשטאט מענטשן האט מען AI אגענטן. איין אגענט פארשט, א צווייטער אנאליזירט, א דריטער שאפט דעם באריכט, און דער נעקסטער קוקט עס איבער אדער פארבעסערט עס. יעדער אגענט האט זיין אייגענע ראלע און אחריות, און צוזאמען פארענדיגן זיי די ארבעט. דער געדאנק פון כמה AI אגענטן וואס ארבעטן צוזאמען, דאס איז פונקט וואס CrewAI מאכט מעגליך.

פארוואס איין פראמפט איז נישט גענוג (30:24:52)

דער קורס באמערקט אז היינט ניצן אסאך מענטשן AI געצייג ווי ChatGPT: מען גיט א פראמפט און מען באקומט א תשובה. פאר קלענערע ארבעטן ארבעט דאס זייער גוט. אבער ביים בויען אמת'דיגע אפליקאציעס זענען דא געווענליך כמה שריט: מען דארף זאמלען אינפארמאציע, זי אנאליזירן, שאפן אינהאלט, און דערנאך פארפיינערן דעם רעזולטאט. אלעס דאס פירן מיט איין פראמפט ווערט גיך זייער צומישט.

CrewAI לייזט דעם פראבלעם: מען קען בויען א מאנשאפט פון AI אגענטן וואס ארבעטן אויטאמאטיש צוזאמען אינעווייניג אין א געארדנטן ארבעטס-פלוס (workflow). אנשטאט איין AI וואס טוט אלעס, שאפט מען ספעציאליזירטע אגענטן, יעדער מיט זיין באשטימטער אחריות, וואס ארבעטן צוזאמען צו א געמיינזאמען ציל.

צוויי הויפט-באגריפן: Flows און Crews (30:25:34)

ביים קוקן ווי CrewAI איז געבויט, זענען דא צוויי וויכטיגע באגריפן וואס מען טרעפט אפט: flows און crews.

- א flow קען מען זיך פארשטעלן ווי דער מענעדזשער פון דער גאנצער סיסטעם. ער קאנטראלירט דעם ארבעטס-פלוס, באשליסט וועלכער שריט לויפט ערשט און וואס קומט נאכדעם, און ווי די אינפארמאציע גייט דורך דעם פראצעס.
- א crew איז א מאנשאפט פון AI אגענטן וואס טוען בפועל די ארבעט. יעדער אגענט אין דער crew האט א באשטימטע ראלע, א ציל, און אמאל אויך צוטריט צו פארשידענע געצייג (tools) כדי צו פארענדיגן די ארבעט.

פשוט געזאגט: דער flow ארגאניזירט דעם פראצעס, און די crew פירט אויס די קלוגע ארבעט. די קאמבינאציע לאזט פראגראמירער בויען AI סיסטעמען וואס זענען פיל מער געארדנט, קענען וואקסן (scalable), און קענען לייזן קאמפליצירטע פראבלעמען.

צוליב דעם צוגאנג וועלכער CrewAI שוין געוואקסן אין אסאך אמת'דיגע אפליקאציעס: אויטאמאטישע פארשונגס-געהילפן, פייפליינס פאר שאפן AI אינהאלט, סיסטעמען פאר קונה-סערוויס, און אפילו AI געהילפן פאר פראגראמירן. אנשטאט רעדן מיט AI דורך אייניציגע פראמפטן, לאזט CrewAI בויען א פולן AI ארבעטס-פלוס וואו כמה אגענטן ארבעטן צוזאמען.

אינסטאלאציע: Python, א ווירטועלע סביבה און CrewAI (30:26:44)

נאכדעם וואס מען פארשטייט וואס CrewAI איז, איז דער נעקסטער שריט צו אינסטאלירן CrewAI און אויפשטעלן די ארבעטס-סביבה. דער קורס זאגט קלאר ווען מען ניצט Command Prompt אדער PowerShell, און ווען דעם טערמינאל אין VS Code.

ערשטער שריט: בודק זיין די Python ווערסיע. CrewAI פארלאנגט Python 3.10 אדער העכער, אבער נידריגער פון 3.14. מען עפנט Command Prompt אדער) PowerShellאויף Windows : מען דריקט דעם Windows קנעפל, מען שרייבט command prompt אדער powershell, און מען עפנט עס), און מען שרייבט א באפעל וואס ווייזט די ווערסיע פון Python. אויב די ווערסיע פאלט אריין אינעם פארלאנגטן ראם, איז אלעס אין ארדנונג. אויב נישט, לאדט מען אראפ א פאסיגע ווערסיע פונעם אפיציעלן וועבזייטל און מען אינסטאלירט זי. די לינקס ווארן געגעבן אין דער באשרייבונג פונעם ווידעא.

דערנאך עפנט מען Visual Studio Code, מען עפנט א פאלדער וואו דער פראיעקט וועט זיין, און מען עפנט דעם טערמינאל אין VS Code. דארט שאפט מען א ווירטועלע סביבה (virtual environment):

m venv venv- python

די סיבה: מען וויל האלטן אלע אפהענגיגקייטן פונעם פראיעקט באזונדער, אזוי אז די ביבליאטעקן וואס מען אינסטאלירט פאר דעם פראיעקט זאלן נישט אנרירן אנדערע פראיעקטן אויפן קאמפיוטער. נאכן שאפן אקטיווירט מען די סביבה, און מען זעט venv אינעם טערמינאל. אינעם זעלבן טערמינאל אינסטאלירט מען CrewAI:

pip install crewai

דאס אינסטאלירט אלעס וואס מען דארף כדי אנצוהייבן בויען AI אגענטן.

אויפשטעלן דעם מאדעל מיט Ollama (30:28:46)

א זייער וויכטיגער טייל: אויפשטעלן דעם AI מאדעל. CrewAI דארף א שפראך-מאדעל כדי צו "טראכטן" און שאפן תוצאות. ס'רוב מענטשן ניצן APIs ווי OpenAI, אבער דאס קאסט געלט און פארלאנגט אינטערנעט יעדעס מאל. דעריבער ניצט מען דא Ollama.

פארוואס Ollama? Ollama לאזט לויפן AI מאדעלן לאקאל אויפן אייגענעם קאמפיוטער. דאס הייסט:

- מען צאלט נישט פאר יעדן API רוף.
- די דאטא בלייבט אויפן אייגענעם מאשין, דאס הייסט בעסערע פריוואטקייט.
- מען איז נישט אפהענגיג פון אינטערנעט ווען דער מאדעל איז שוין אראפגעלאדן.

• אין אסאך פעלער גיט עס גיכערע תשובות.

פאר אנהייבער, און אפילו פאר אסאך אמת'דיגע באנוצן, איז דאס א שטארקע און ביליגע לייזונג.

שריטן:

• מען לאדט אראפ Ollama פונעם לינק אין דער באשרייבונג און מען אינסטאלירט עס.

• אין Command Prompt אדער אינעם VS Code טערמינאל שרייבט מען ollama --version, כדי צו באשטעטיגן אז עס ארבעט.

• Ollama אליין איז בלויז א "runtime", עס קומט נישט מיט קיין אינסטאלירטן מאדעל. דעריבער לאדט מען אראפ א מאדעל: ollama pull llama3. דער באפעל לאדט אראפ דעם llama3 מאדעל אויפן קאמפיוטער, כדי מען זאל אים קענען ניצן לאקאל. דאס קען דויערן א שטיק צייט, לויט דער אינטערנעט-גיכקייט, ווייל מאדעלן זענען געוועגליך גרויס.

• מען פרובירט עס אויס מיט ollama run llama3. אויב מען קען שמועסן מיטן מאדעל אינעם טערמינאל, איז אלעס ריכטיג אויפגעשטעלט.

די טעקעס פונעם פראיעקט: agents, tasks, tools, main (30:30:17)

צוריק צום VS Code פראיעקט, וואו מען הייבט אן בויען. דער פראיעקט האט כמה טעקעס: agents.py, tasks.py, tools.py, און main.py. דער קורס ערקלערט יעדע טעקע אין פרטים.

agents.py: דא באשטימט מען די אגענטן. אגענטן זענען AI ראלעס. אנשטאט איין AI וואס טוט אלעס, צעטיילט מען די ארבעט צווישן כמה אגענטן, יעדער מיט א באשטימטער אחריות. יעדער אגענט האט א role, א goal, און אמאל א backstory. די role באשטימט וואס דער אגענט איז. דער goal באשטימט וואס ער דארף דערגרייכן. די backstory גיט הינטערגרונט, כדי די תשובות זאלן זיין מער גענוי און שייך. למשל: א פארשונגס-אגענט וואס זאמלט אינפארמאציע, און א שרייבער-אגענט וואס פארוואנדלט די אינפארמאציע אין געארדנטן אינהאלט. די סיבה איז ספעציאליזאציע: אזוי ווי אין לעבן, וועלן פארשידענע מענטשן טוען פארשידענע ארבעטן, איז דער רעזולטאט פיל בעסער.

tasks.py: דא באשטימט מען וועלכע ארבעט מען דארף טון. tasks זענען אנווייזונגען וואס מען גיט די אגענטן. יעדער task באשרייבט קלאר דעם ציל און איז פארבונדן מיט א באשטימטן אגענט. למשל, איין task: "געפינען די לעצטע טרענדס אין AI", געגעבן דעם פארשונגס-אגענט. א צווייטער task: "שרייבן א קיצור באזירט אויף דער פארשונג", געגעבן דעם שרייבער-אגענט. איין אגענט ענדיגט זיין task, און דער רעזולטאט קען ווערן איבערגעגעבן צו אן אנדערן אגענט. אזוי שאפט מען א פולן ארבעטס-פלוס אנשטאט בלויז איין תשובה. דאס איז די הויפט סיבה פארוואס CrewAI איז שטארק: מען גיט נישט בלויז א פראמפט צו א AI, נאר מען בויט א מערשטופיגע סיסטעם וואו אגענטן ארבעטן צוזאמען.

tools.py: tools זענען נאך פעאיקייטן וואס מען קען געבן די אגענטן. למשל, אויב מען וויל אז דער אגענט זאל זוכן אויפן וועב, רופן אן API, אדער לויפן א פונקציע, באשטימט מען דאס דא. דער אגענט קען דאן ניצן די געצייג ביים טון זיינע tasks. דאס מאכט די סיסטעם פיל מער דינאמיש און שטארק.

main.py: דער אריינגאנג-פונקט פונעם פראיעקט, וואו אלעס קומט צוזאמען. מען אימפארטירט די אגענטן און די tasks, מען שאפט א crewדי מאנשאפט), און מען לאזט עס לויפן. וועלן מען שרייבט python main.py

אינעם VS Code טערמינאל, הייבט די טעקע אן דעם גאנצן ארבעטס-פלוס, און די אגענטן הייבן אן אויספירן זייערע tasks שריט ביי שריט.

א קיצור פון די סיבות פאר יעדן שריט: Python כדי צו לויפן אלעס. א ווירטועלע סביבה כדי צו פירן די אפהענגיגקייטן. CrewAI כדי צו בויען ארבעטס-פלוסן פון אגענטן. Ollama כדי צו לויפן AI מאדעלן לאקאל אן קאסטן. און agents און tasks כדי צו שאפן א געארדנטע קלוגע סיסטעם.

דעמאנסטראציע: שאפן די אגענטן אין agents.py (30:32:50)

יעצט גייט מען איבער צום פראקטישן טייל. אין VS Code זענען שוין אינסטאלירט אלע אפהענגיגקייטן, ווייל דאס נעמט צייט. וועט עס גייט נאך די פריערדיגע שריטן, וועט עס גרינג קענען אינסטאלירן. מען זעט א פאלדער מיטן נאמען crew AI אויפן desktop, מיט פיר הויפט-טעקעס: agent.py, main.py, task.py און tool.py.

אין agent.py אימפארטירט מען Agent פון crewai, און מען שאפט צוויי אגענטן. דער ערשטער איז senior researcher:

- role: א senior YouTube researcher, וואס פארשט דעם געגעבענעם נושא.
- goal: "explain topic simply", יעדן נושא וואס מען גיט, דארף מען ערקלערן אויף א פשוט'ן אופן.
- verbose = True: מען ווייזט אלעס אינעם טערמינאל.
- memory = False: מיט True וואלט עס געדענקט די היסטאריע; מיט False ארבעט עס יעדעס מאל פון נייעם.
- backstory: "expert in simplifying AI topics", א מומחה אין פארפשוט'ן AI נושאים.
- tools: א רשימה פון דרויסנדיגע געצייג, ווי YouTube search א.א.וו. אין דער אפיציעלער דאקומענטאציע פון CrewAI געפינט מען וואס א tool איז און וועלכע tools עקזיסטירן, למשל א Serper dev tool, file read tool, website search tool און directory tool, צוזאמען מיט אסאך אנדערע אפציעס. מען קען אויסקלייבן וואס מען וויל.
- llm: מען ניצט Ollama מיט tinyllama. מען קען דעם מאדעל אראפלאדן מיט ollama pull tinyllama.

אויפן זעלבן אופן שאפט מען דעם צווייטן אגענט, script writer:

- role: א YouTube script writer.
- goal: שרייבן קורצע סקריפטן.
- verbose = True: אויך דא ווייזט מען אלעס אינעם טערמינאל.
- memory = False.
- backstory: א מומחה אין שרייבן אינטערעסאנטע סקריפטן.
- tools: די זעלבע ווי ביים ערשטן אגענט, און אויך דער זעלבער גרויסער שפראך-מאדעל.

אזוי שאפט מען אין דער טעקע צוויי אגענטן, און מען גייט ווייטער צו דער נעקסטער טעקע.

די task.py טעקע: צוויי אויפגאבעס פאר די אגענטן (30:35:21)

דער קורס גייט ווייטער מיט דער טעקע task.py. אין אנהייב פון דער טעקע ווערט פון CrewAI אימפארטירט Task, און פון דער agents טעקע ווערן אימפארטירט די צוויי אגענטן וואס זענען פריער געשאפן געווארן: senior researcher און script writer.

די ערשטע אויפגאבע איז די research task. אלס description ווערט געגעבן: ערקלער דעם טעמע, וועלכער טעמע עס וועט נאר געפרעגט ווערן, אין דריי פשוטע bullet points. דער expected output איז: דריי קלארע bullet points וואס ערקלערן דעם טעמע. דער אגענט פאר דער אויפגאבע איז דער senior researcher.

אויף דעם זעלבן אופן איז דא די script writing task. דא ווערט די אויפגאבע דעפינירט מיט א description: ניצנדיק די פארשונג, שאף א קורצן YouTube סקריפט אויף דעם טעמע, וועלכער טעמע עס וועט געגעבן ווערן, מיט א hook line, צוויי ביז דריי שורות ערקלערונג, און מיט אן outro. דער expected output איז א קורצער און אינטערעסאנטער YouTube סקריפט אויף דעם טעמע וואס מען וועט אנווייזן. דער אגענט דא איז דער script writer, בשעת בייַ דער ערשטער אויפגאבע איז דער אגענט געווען דער senior researcher.

tools.py און main.py: צוזאמשטעלן דעם crew (30:36:21)

נאך די צוויי טעקעס agent.py און task.py קומט די טעקע tools.py. דערווייל ניצט מען נישט קיין שום tool, דעריבער ווערט די טעקע געלאזט ליידיג. אבער שפעטער, אויב מען וויל צולייגן מער tools אדער מער פונקציאנאליטעט, קען מען ניצן די טעקע. אין דער CrewAI דאקומענטאציע געפינט מען אלע tools און אלעס וואס מען דארף דערצו.

די main.py איז די טעקע וואו אלעס ווערט צוזאמגענומען, און דאס איז די טעקע וואס מען לויפט. פון crew ווערן אימפארטירט Crew און Process. פון דער agents טעקע ווערן אימפארטירט די צוויי אגענטן, senior researcher און script writer, און פון task ווערן אימפארטירט research task און script task.

דערנאך ווערט געשאפן דער crew, דאס הייסט די אגענטן, די ארבעטס-פראצעדור און אלעס אנדערע:

- agents: די צוויי אגענטן, senior researcher און script writer.
- tasks: research task און script task. איין אגענט פארשט דעם טעמע, און דער אנדערער אגענט שרייבט דערויף א סקריפט.
- process: sequential, דאס הייסט די אויפגאבעס ווערן אויסגעפירט איינע נאך דער אנדערער. קודם קומט די research task, און דערנאך די script task.
- verbose: ווערט געשטעלט אויף True.

דערנאך לויפט מען דעם crew. אין result ווערט גערופן crew.kickoff מיט אן input וואו topic איז "crew AI tutorial for beginners". דער קורס דערמאנט או אין די task באשרייבונגען שטייט דער טעמע אין געוווּנדענע קלאמערן, און אין main ווערט דער טעמע איינגעשטעלט אלס "crew AI tutorial for beginners". דאס הייבט אן די גאנצע פראצעדור, און צום סוף ווערט דער רעזולטאט אויסגעדרוקט. אט דאס זענען די פיר טעקעס פונעם פראיעקט.

לויפן דעם crew און דער רעזולטאט (30:38:23)

מען לויפט main.py, און עס דויערט א שטיק צייט. מען זעט אז די אויפגאבע הייבט זיך אן: דער senior researcher באקומט די אויפגאבע צו ערקלערן "crew AI tutorial for beginners" אין דריי פשוטע bullet points. דער אגענט פארשט און גיט א final answer אין דריי פונקטן: וואס איז CrewAI, ווי אזוי מען קומט צו דעם רעזולטאט, און א סיכום. דערנאך ווערט די אויפגאבע אנגעצייכנט אלס פארענדיקט.

דערנאך הייבט זיך אן די צווייטע אויפגאבע פונעם YouTube script writer: ניצנדיק די פארשונג, שאף א קורצן סקריפט אויף "crew AI tutorial for beginners". מען זעט אז איין אגענט האט שוין פארענדיקט זיין אויפגאבע און דער צווייטער האט אנגעהויבן. דער final output הייבט זיך אן מיט "thank you for your feedback, in this tutorial we will walk through..." און גיט א גאנצן YouTube סקריפט, וואס ענדיגט זיך מיט "regards" און דעם נאמען. דער קורס באטאנט אז דער רעזולטאט איז געקומען אן קיין שום איינמישונג פונעם באניצער: די אגענטן האבן אליין געפארשט און אליין געשריבן דעם סקריפט, און אזוי איז די ארבעט אסאך גרינגער.

וואס איז Playwright (30:39:26)

דא הייבט זיך אן א נייער טייל, וועגן Playwright. Playwright איז א פרייער open-source framework פאר אויטאמאטישן טעסטן, געשאפן פון Microsoft, און מען ניצט עס בעיקר צו אויטאמאטיזירן אפליקאציעס אין וועב-בראוזער. פשוט גערעדט, Playwright העלפט טעסטן וועבזייטלעך אויטאמאטיש, אן דארפן יעדע אקציע מיט דער האנט טון ווידער און ווידער.

א ביישפיל: אויב מען וויל טעסטן צי א login זייטל ארבעט ריכטיג, קען מען שרייבן א Playwright סקריפט וואס עפנט אויטאמאטיש דעם בראוזער, שרייבט אריין דעם username און password, קליקט אויפן login קנעפל, און טשעקט צי דער באניצער איז מצליח אריינגעקומען.

סארטן אפליקאציעס, שפראכן און בראוזערס (30:39:58)

Playwright שטיצט וועב-בראוזער אפליקאציעס, מאביל אפליקאציעס און API טעסטינג. דאס מיינט אז מען קען עס ניצן נישט נאר צו טעסטן וועבזייטלעך אויף desktop בראוזערס, נאר אויך מאביל וועב-זייטלעך און backend APIs.

עס שטיצט אויך עטלעכע פראגראמירונג-שפראכן: JavaScript אדער TypeScript מיט Node.js , ווי אויך Java, Python, און .NET. שפראכן ווי C#. לויט דעם הינטערגרונט פונעם פראגראמירער אדער לויט די פאדערונגען פונעם פראיעקט, קען מען אויסקלייבן די שפראך וואס פאסט זיך דאס בעסטע.

וואס שייך בראוזערס, שטיצט Playwright אלע מאדערנע בראוזער-מאטארן: Chromium-באזירטע בראוזערס ווי Google Chrome, Microsoft Edge, Opera און Brave; WebKit בראוזערס, וואס נעמען אריין Safari און iOS בראוזערס; און אויך Firefox. דאס איז איינע פון די גרעסטע מעלות פון Playwright: מיט איין framework קען מען טעסטן די אפליקאציע אויף אסאך בראוזערס.

די אפיציעלע וועבזייטל פון Playwright (30:41:00)

כדי צו זען Playwright בפועל, גייט מען אויפן אפיציעלן וועבזייטל, playwright.dev. מען זוכט "playwright" און קליקט אויפן אפיציעלן זייטל. דער קורס רעקאמענדירט שטארק צו לייענען די גאנצע

דאקומענטאציע איידער מען הייבט אן ארבעטן מיט Playwright . אזוי האט מען א קלארע יסוד-ידיעה, Playwright לויפט גלאטער, און אויטאמאציע ווערט גרינגער צו פארשטיין.

אויפן זייטל זענען דא אפטיילונגען פאר MCP, CLI און APIs (application programming interface). א געוויסער קנעפל ווייזט אז מען קען ארבעטן מיט Playwright אין פארשידענע שפראכן: Node.js, Python, Java און .NET. אין דעם טוטאריאל ארבעט מען מיט Node.js. אז מען קלייבט אויס Node.js און קליקט אויפן GitHub אייקאן, קומט מען צו דער Node.js אימפלעמענטאציע און צו די GitHub טעקעס פונעם גאנצן פראגראם.

דער זייטל ווייזט אויך:

- דעם באפעל וואס אינסטאלירט Playwright אויפן קאמפיוטער, פאר טעסטן.
- דעם באפעל צו אינסטאלירן Playwright CLI דורך דער command line.
- דעם באפעל פאר MCP.
- אז עס איז געבויט פאר טעסטן פאר AI אגענטן, און האט שטארקע tools.
- אז אלע פיר בראוזערס ווערן געשטיצט.
- אז רוב פירמעס קלייבן עס אויס פאר זייערע open-source פראיעקטן, דעריבער איז עס כדאי צו לערנען.

די מעלות פון Playwright (30:43:04)

Playwright קומט מיט אסאך שטארקע מעלות וואס מאכן וועב-טעסטן גרינגער און שנעלער:

- Cross browser testing: מען קען טעסטן די זעלבע אפליקאציע אויף Chromium, Firefox און WebKit, און זען צי דער וועבזייטל ארבעט ריכטיג אויף די פארשידענע בראוזערס.
- Auto waiting: Playwright ווארט אויטאמאטיש ביז די עלעמענטן זענען גרייט, איידער עס קליקט, שרייבט אדער טשעקט זיי. דאס פארמינערט "flaky" דורכפאלן פון טעסטן.
- אסאך פראגראמירונג-שפראכן: JavaScript, TypeScript, Python, .NET. און נאך, אזוי אז מען קלייבט די שפראך וואס מען פילט זיך באקוועם דערמיט.
- Parallel test execution: עטלעכע טעסטן קענען לויפן אין דער זעלבער צייט. דאס שפארט צייט, בפרט ווען די test suite ווערט גרויס.
- איינגעבויטע debugging tools: screenshots, וידיאס, traces און reports. דער קורס זאגט צו זיי צו ווייזן אין די קומענדיגע לעקציעס. זיי העלפן פארשטיין פארוואס א טעסט איז דורכגעפאלן און וואס איז די סיבה.
- Codegen: דאס שאפט טעסט-קאוד אויטאמאטיש בשעת מען טוט אקציעס אינעם בראוזער. דאס איז זייער נוצלעך פאר אנהייבער וואס לערנען אויטאמאטישן טעסטן.

בקיצור, Playwright איז שנעל, פארלעסלעך, גרינג פאר אנהייבער, און נוצלעך פאר טעסטן מאדערנע וועב-אפליקאציעס.

פאראויסזעצונגען: Node.js און npm (30:45:09)

איידער מען אינסטאלירט Playwright, דארף מען פארשטיין וואס דער קאמפיוטער דארף האבן. דאס ערשטע איז Node.js, ווייל מען אינסטאלירט Playwright דורך npm. וועז Node.js ווערט אינסטאלירט, באקומט מען npm אויטאמאטיש.

כדי צו טשעקן צי Node.js איז שוין אינסטאלירט, עפנט מען דעם command prompt און שרייבט npm -v. אין דער לעקציע איז ארויסגעקומען די ווערסיע 11.12.1, וואס מיינט אז די אינסטאלאציע איז מצליח געווען. מען קען אויך לויפן node -v. דער קורס רעקאמענדירט צו פרובירן ביידע באפעלן, כדי Playwright זאל לויפן גלאט. אינעם ביישפיל ווייזן ביידע ווערסיע-נומערן, אז ביידע tools זענען פארהאן.

אויב מען באקומט נישט קיין ווערסיע, דארף מען אינסטאלירן Node.js פונעם אפיציעלן וועבזייטל: מען זוכט "NodeJS" אינעם בראוזער, גייט אויפן אפיציעלן זייטל, קלייבט אויס Windows, macOS אדער Linux, און מען פאלגט אלע שריט וואס ווערן אנגעגעבן דארט. אין דער לעקציע איז עס שוין אינסטאלירט.

צווייטע פאראויסזעצונג: אן IDE (30:46:44)

די צווייטע פאראויסזעצונג איז אן IDE, integrated development environment. אין דעם טוטאריאל ניצט מען Visual Studio Code, ווייל עס איז פשוט, גרינג פאר אנהייבער און פראגראמירער ניצן עס אסאך. מען מעג ניצן אן אנדער IDE, אבער מען דארף זיך פארזיכערן אז עס איז ריכטיג אינסטאלירט. אויב VS Code איז נישט אינסטאלירט, נעמט מען עס אראפ פונעם אפיציעלן וועבזייטל פון Visual Studio Code.

צוויי וועגן צו אינסטאלירן Playwright, און עפענען דעם פראיעקט (30:47:15)

וועג ביידע זאכן זענען גרייט, קען מען אינסטאלירן Playwright. עס זענען דא צוויי וועגן: דורך באפעלן, אדער דורך דער VS Code extension. דער באפעל וואס מען לויפט אין CLI איז:

latest@npm init playwright

דער קורס זאגט צו צו ווייזן ביידע וועגן, איינס נאך דאס אנדערע. מען הייבט אן מיט דער אינסטאלאציע דורך באפעלן מיט npm.

דער ערשטער שריט איז צו שאפן א פאלדער און עס עפענען אין VS Code:

- אויפן C drive שאפט מען א נייעם פאלדער מיטן נאמען automation.
- אינעווייניג אין automation שאפט מען א פאלדער מיטן נאמען playwright automation.
- אין VS Code קלייבט מען אויס Open Folder, גייט צום drive, ארויין אין automation, און מען קלייבט אויס דעם פאלדער playwright automation.
- מען קליקט אויף "Yes, I trust the authors", און דער פראיעקט עפנט זיך.
- דערנאך גייט מען אין Terminal און קליקט New Terminal . מען זעט אז מען געפינט זיך אינעם פאלדער playwright automation, גרייט פאר דעם אינסטאלאציע-באפעל.

אינסטאלירן Playwright און אנהייבן א Todo פראיעקט (30:49:08)

אינסטאלירן Playwright דורך דער קאמאנד-ליניע (30:49:08)

דער קורס הייבט אן מיט די ערשטע מעגליכקייט ווי אזוי צו אינסטאלירן Playwright: מען לויפט א קאמאנד אינעם טערמינאל. דער קאמאנד איז latest@npm init playwright. דאס ווארט latest זאגט אז מען וויל אינסטאלירן די לעצטע ווערסיע פון Playwright אויפן קאמפיוטער. מען דריקט Enter.

נאכדעם שטעלט די אינסטאלאציע עטליכע פראגעס, און עס ווערט געוויזן ווי צו ענטפערן:

- צי מען וויל ניצן TypeScript אדער JavaScript. פאר דעם פראיעקט קלייבט מען JavaScript.
- וואו צו לייגן די end-to-end טעסטן. מען לאזט עס ביים פארגעשלאגענעם טעקע-נאמען tests.
- מען שטעלט true ביי דער פראגע וואס קומט דערנאך.
- צי מען וויל אינסטאלירן די Playwright בראוזערס דורך npx playwright install. מען ענטפערט true.

דערנאך אינסטאלירט דער קאמאנד אלע בראוזערס, די טעסט-פאלן און אלעס וואס מען דארף. מען דארף א ביסל ווארטן ביז ער ענדיקט אראפצולאדן די בראוזערס.

וען דער קאמאנד ענדיקט זיך, זעט מען א רשימה פון קאמאנדעס און אלע דעפענדענסיס וואס זענען אינסטאלירט געווארן. למשל npx playwright test לויפט דעם פראיעקט אויף א געוויסן בראוזער, און test example לויפט א באשטימטע טעקע. דא זענען אויך אנדערע זאכן וואס מען דארף כדי צו בויען דעם פראיעקט.

די טעקעס וואס ווערן באשאפן (30:50:30)

מען זעט או די אינסטאלאציע האט באשאפן א tests טעקע, און אין איר א טעקע example.spec.js. דארט וועלן לויפן אלע טעסטן פונעם פראיעקט, און אלע טעסטן וועלן גיין אין דעם טעקע.

די צווייטע טעקע איז package.json. דער קורס ערקלערט או דאס איז א זייער וויכטיגע טעקע וואס פירט אן דעם Node פראיעקט. זי ווייזט אלע ביבליאטעקן וואס זענען צוגעלייגט געווארן צום פראיעקט.

די דריטע טעקע איז די קאנפיגוראציע-טעקע playwright.config.js. די ענדונג js ווייזט או מען ניצט יעצט JavaScript. ווער עס וויל TypeScript קען ניצן ts. דאס איז אויך א זייער וויכטיגע טעקע, ווייל דארט מאכט מען אלע קאנפיגוראציע-איינשטעלונגען וואס דער פראיעקט דארף.

באשטעטיגן די אינסטאלאציע און זען די קאמאנדעס (30:51:40)

כדי צו באשטעטיגן או Playwright איז ריכטיג אינסטאלירט, גיט מען א קאמאנד מיט v-, למשל npx playwright -v. דער קאמאנד ווייזט די ווערסיע פון Playwright. אויפן עקראן זעט מען די ווערסיע 1.12.1 (אזוי ווי עס ווערט פארגעלייענט). דאס מיינט או Playwright איז געלונגען אינסטאלירט.

ווער עס וויל זען אלע אפציעס פון Playwright קאמאנדעס, גיט א קאמאנד מיט help-. דאן באקומט מען א רשימה פון אלע קאמאנדעס, אריינגערעכנט די פאר סקרינשאטס, פאר traces און נאך.

אינסטאלירן דורך דער VS Code עקסטענשאן (30:52:17)

די צווייטע מעגליכקייט איז צו אינסטאלירן Playwright דורך אן עקסטענשאן אין VS Code. די שריט זענען:

- מען מאכט א נייעם פאלדער אויפן C דרייוו מיטן נאמען automation exe.
- מען עפנט VS Code, גייט צו File און דערנאך Open Folder, קלייבט דעם פאלדער און עפנט אים.
- מען באשטעטיגט "Yes, I trust the authors".
- מען דריקט אויפן Extensions קנעפל, וואו מען זעט אלע עקסטענשאנס.
- מען שרייבט playwright און אינסטאלירט די Playwright עקסטענשאן פון Microsoft.
- נאך דער אינסטאלאציע גייט מען צו View און דערנאך Command Palette, און מען קלייבט Install Playwright.

אזוי האט מען געזען ביידע וועגן צו אינסטאלירן Playwright: דורך דער קאמאנד-ליניע און דורך דער עקסטענשאן. יעצט קען מען שוין גרינג אנהייבן ארבעטן מיט Playwright.

קאמאנדעס צו לויפן טעסטן (30:54:12)

דער קורס ווייזט א טאבעלע מיט קאמאנדעס און וואס יעדער קאמאנד טוט.

ערשט ווערט ערקלערט דער באגריף headless mode. דאס מיינט אז די אויטאמאציע פון Playwright ווערט נישט געוויזן אין קיין בראוזער. זי לויפט אינעם הינטערגרונט, און מען זעט גארנישט אויפן בראוזער. דער פארקערטער איז headed mode, וואו אלעס ווערט קלאר געוויזן אויף איינעם פון די בראוזערס.

- npx playwright test לויפט די טעסטן אין אלע בראוזערס אין headless mode.
- npx playwright test one.spec.js לויפט א באשטימטע טעסט-טעקע.
- npx playwright test one.spec.js two.spec.js לויפט די טעקעס וואס מען האט אנגעגעבן.
- npx playwright test one two לויפט טעקעס וואס האבן one אדער two אין זייער נאמען. וואס מען שרייבט נאך test, ווערט גענומען ווי א טייל פונעם טעקע-נאמען.
- וועט מען גיט א טיטל, לויפט מען נאר דעם טעסט מיט יענעם טיטל.
- npx playwright test --project=chromium לויפט די טעסטן נאר אויף א באשטימטן בראוזער. מען האט געלערנט דריי בראוזערס, אבער דער קאמאנד ניצט נאר איינעם.
- --headed לויפט די טעסטן אין headed mode, אזוי אז מען זעט אלעס אויפן עקראן אינעם בראוזער.
- --debug דעבאגט אלע טעסטן.
- test example.spec.js --debug דעבאגט נאר יענע באשטימטע טעקע.

דער קורס רעקאמענדירט שטארק צו נעמען א סקרינשאט פון דעם סלייד, עס דורכלייענען, לערנען און פראקטיצירן.

דער Todo אפליקאציע: א דעמא (30:56:50)

יעצט בויט מען א קליינעם פראיעקט פון דער ווירקלעכקייט: אויטאמאטיזירן א todo אפליקאציע. די אפליקאציע איז שוין געבויט און דיזיינט. מען דארף נאר נעמען דעם פארטיקן פראיעקט און אים אויטאמאטיזירן מיט Playwright אין VS Code.

מען גייט צום TodoMVC וועבזייטל און מען קלייבט React. אין דער דעמא לייגט מען צו אויפגאבעס, למשל "go on a walk", "buy clothes" און "swim", אזוי ווי מען גרייט צו א todo רשימה. וועט אן אויפגאבע איז געטון, צייכנט מען זי אפ, און זי ווערט באצייכנט אלס פארטיק.

דער באניצער קען זען דריי סארטן רשימות:

- All: אלע אויפגאבעס.
- Active: די וואס זענען נאך נישט געטון, אין דעם ביישפיל swim.
- Completed: די וואס זענען שוין פארטיק.

מען קען אויך אויסמעקן די פארטיקע אויפגאבעס מיט Clear completed. נאך דעם בלייבט אין All בלויז swim.

די שריט און צילן פונעם פראיעקט (30:58:30)

די שריט פון דער אויטאמאציע זענען:

- אינסטאלירן Playwright (שוין געטון).
- באשאפן א פראיעקט-פאלדער.
- שרייבן end-to-end טעסטן און זיי לויפן לאקאל, סיי אין headed און סיי אין headless mode.
- כאפן סקרינשאטס און ווידעא ווען א טעסט פעלט.
- אנקוקן דעם HTML באריכט.

די געצייג וואס מען דארף זענען NodeJS און Playwright, און דער עדיטאר VS Code.

די צילן פונעם פראיעקט זענען: אינסטאלירן Playwright און שרייבן טעסטן. דערנאך אויטאמאטיזירן צולייגן דריי todos, איינס אפצייכענען אלס פארטיק, און איינס אויסמעקן. נאכדעם גענעריר מען דעם Playwright HTML באריכט און מען עפנט אים. מען לייגט אויך צו assertions און מען קוקט אויף פעלערן אינעם באריכט. נאר ווען א טעסט פעלט, נעמט מען א סקרינשאט און א ווידעא, און דער קורס זאגט צו צו ערקלערן פארוואס.

באשאפן דעם פראיעקט (31:00:03)

מען מאכט א פאלדער אויפן C דרייוו מיטן נאמען todo automation אדער todo list , און מען עפנט אים אין VS Code. אינעם טערמינאל זעט מען אז מען איז אין דעם פראיעקט.

מען לויפט npm init -y. דאס y- מיינט "יא" צו אלע פראגעס, אזוי אז דער קאמאנד לויפט גלייך.

Playwright איז שוין אינסטאלירט, אבער עס ווערט רעקאמענדירט עס נאכאמאל צו אינסטאלירן פאר יעדן פראיעקט. דער טעם איז אז מען אינסטאלירט די דעפענדענסיס פאר א באשטימטן פאלדער, און נישט פאר אלעס מיט אמאל. דערפאר לויפט מען npm init playwright, ענטפערט די פראגעס (למשל true), און ער אינסטאלירט די נויטיגע ביבליאטעקן.

נאכדעם ווייזט ער די קאמאנדעס:

- npx playwright test לויפט end-to-end טעסטן.
- npx playwright test --ui הייבט אן אן אינטעראקטיוון UI מאדע.
- --project=chromium גיט אן אויף וועלכן בראוזער צו לויפן.
- npx playwright test example לויפט א באשטימטע טעקע.
- --debug לויפט אין debug מאדע.
- codegen גענערירט אויטאמאטיש טעסטן.

אלע די קאמאנדעס וועט מען זען אינעם פראיעקט.

אנהייבן שרייבן די טעסט-סקריפטן (31:02:08)

אונטער דעם todo list פאלדער זענען שוין באשאפן געווארן package.json, playwright.config.js און דער tests פאלדער מיט example.spec.js. אין דעם פאלדער האלט מען אלע טעסט-פאלן.

אינעם tests פאלדער מאכט מען א נייע טעקע todo.spec.js, און דארט שרייבט מען אלע טעסט-סקריפטן. מען גיט די ענדונג js ווייל מען ניצט JavaScript. מען קען אויך געבן ts פאר TypeScript.

דערנאך גייט מען צוריק צו דער פארטיקער TodoMVC אפליקאציע, כדי זי צו אויטאמאטיזירן.

Playwright אויטאמאציע און RAG אפשאצונג (31:03:08)

רעקארדירן טעסטן אין א todo-אפליקאציע (31:03:08)

אין דער לעקציע ווערט געוויזן א todo-אפליקאציע וואס איז געבויט מיט React. מען קליקט אויף React און מען פירט אויס אלע פונקציעס פון דער אפליקאציע: מען קען אריינשרייבן אויפגאבעס ווי run, play, eat אדער סיי וואס אנדערש. דערנאך קען מען צייכענען (tick) אן אויפגאבע, און זי גייט אריבער אונטער Completed. Active מיינט וואס איז נאך אפען (pending), און All אנטהאלט אלע דריי. דאס איז די יסוד-פונקציאנאליטעט פון א todo-אפליקאציע.

דער קורס רעקאמענדירט צו ערשט רעקארדירן די אקציעס וואס מען טוט אין דער אפליקאציע. דערפאר גייט מען צוריק צום command prompt און מען שרייבט אין טערמינאל:

npx playwright codegen

ביים ערשטן פרואוו ווערט פארגעסן צו געבן דעם URL פון דער אפליקאציע, דעריבער מאכט מען עס צו, מען לייגט אריין clear, און מען לויפט ווידער דעם באפעל, דאס מאל מיט דעם URL פון דער אפליקאציע: מען קאפירט דעם URL, מען פעיסט עס אין טערמינאל און מען דריקט Enter. דערמיט הייבט זיך אן די רעקארדירונג, און עס עפנט זיך א פענצטער מיט א סקריפט, וואו יעדע אקציע וואס מען טוט אין דער אפליקאציע ווערט אויטאמאטיש פארשריבן.

ווי די אקציעס ווערן פארוואנדלט אין טעסט-קאוד (31:04:40)

מען זעט ווי מען שרייבט run און Enter, דערנאך play און Enter, און דערנאך eat און Enter, דריי אויפגאבעס פאר א טאג. ביי יעדן Enter ווערט דער טעסט אויטאמאטיש געשאפן אין דער זעלבער סדר: run, Enter, play, Enter, eat, Enter.

דערנאך צייכנט מען אן עטליכע אויפגאבעס, און דער טעסט ווערט דערגאנצט מיט א list item filter פאר run און play, און מען זעט אז toggle איז check. נאכדעם קליקט מען אויף Active צו זען וועלכע זאכן זענען נאך אפען, און דער טעסט אויף דער רעכטער זייט ווערט דערהיינטיקט. דערנאך קליקט מען אויף Completed, וואס ווייזט אלע געענדיקטע אויפגאבעס.

אין דעם רעקארדירונג-פענצטער זענען פאראן עטליכע געצייג:

- pick locator, אויסקלייבן אן עלעמענט אויף דעם בלאט
- assert visibility, פרובירן אז אן עלעמענט איז זעבאר
- assert text, פרובירן א טעקסט
- assert value, פרובירן א ווערט
- assert snapshot

מען ווייזט ווי מען לייגט צו assert visibility: מען קליקט אויף דעם געצייג און דערנאך אויף אן עלעמענט, און אין דער לעצטער שורה פונעם קאוד ערשיינט א expect וואס פרובירט אז דער run todo-item איז to be visible. ענליך ווערט געוויזן assert text: מען קליקט אויף Active, און עס ווערט צוגעלייגט א שורה to contain text "Active". דערנאך שטעלט מען אפ די רעקארדירונג, און אלע טעסט-פעלער (test cases) זענען שוין געשריבן אין דער זעלבער סדר ווי די אקציעס זענען דורכגעפירט געווארן.

איבערטראגן דעם סקריפט און לויפן די טעסטן (31:06:13)

מען דארף זיין אין test runner מאוד. אויב עס איז נישט אזוי, גייט מען אויס Node.js און test runner. דערנאך קאפירט מען דעם גאנצן סקריפט מיט דעם copy-אייקאן, מען גייט צוריק צום פראיעקט, און מען פעיסט עס אריין אין דער טעקע וואס מען האט פריער געשאפן, todo-demo.spec.js. מען קוקט צי עס זענען פאראן גרייזן. ביז איצט זענען נישטא קיין גרייזן, און מען היט אפ די טעקע.

דער טעסט-סקריפט הייבט זיך אן מיטן ווארט import. דערנאך קומען אלע באפעלן, דער URL פון דער todo-אפליקאציע וואס מען אויטאמאטירט, און אלע קליקס און אריינשרייבונגען אין דער זעלבער סדר ווי מען האט זיי געטון.

צו לויפן די טעקע שרייבט מען:

```
npx playwright test
```

עס לויפן נײַן טעסטן מיט פיר workers. פיר workers מיינט אז עס לויפט אין פארשידענע בראוזערס, ווי WebKit, Chromium און Firefox. מען קען אויך באשטימען א געוויסן בראוזער, וואס ווערט געוויזן שפעטער.

פאררעכטן א גרייז און דער HTML-באריכט (31:07:46)

עס ווערט ארויסגעגעבן א באריכט: זעקס טעסטן זענען דורכגעגאנגען און דריי זענען דורכגעפאלן. מען קוקט וואס איז פאלש געגאנגען: עס ווייזט א timeout ביי todo-demo, און די שורה מיט uncheck און getByRole item האט גורם געווען דעם גרייז. די שורה ווערט אויסגעקאמענטירט, מען היט אפ און מען לויפט ווידער. צום סוף זענען אלע נײַן טעסטן דורכגעגאנגען מיט פיר workers.

צו זען דעם לעצטן HTML-באריכט שרייבט מען:

```
npx playwright show-report
```

דער באריכט ווייזט אז אלע טעסטן זענען דורכגעגאנגען, סיי אויף Chromium, סיי אויף Firefox , סיי אויף WebKit.

טעגס, headed מאוד און א געוויסער בראוזער (31:08:47)

אין דעם טעסט-נאמען שרייבט מען to-do list , דער נאמען פונעם פראיעקט, מיט אן @-צייכן פאר דעם טעג sanity, און מען היט אפ. דערנאך לויפט מען א וויכטיקן באפעל צו ווייזן די אויטאמאציע:

```
"sanity" grep-- headed-- npx playwright test
```

עס ווערט דערמאנט די צוויי מאודן, headed און headless. אין headed מאוד זעט מען דעם בראוזער פיזיש פאר די אויגן. ביי --grep איז וויכטיק צו שרייבן sanity אין געזונדפיסלעך. ביים ערשטן פרואוו איז געווען א טעות אין נאמען: עס דארף זיין test און נישט text. נאכן פאררעכטן לויפט די אפליקאציע, און מען זעט אויף

דעם עקראן אלע אקציעס וואס זענען דורכגעפירט געווארן. עס גייט זייער שנעל, און מען קען עס פארפאמעלעכן.

דערווייל לויפט עס אין דריי בראוזערס: WebKit, Firefox און Chromium. צו לויפן נאר אין Chromium, לייגט מען צו --project=chromium. דאן שטייט "running 1 test using 1 worker", וואס שפארט אסאך זכרון. דער קורס רעקאמענדירט צו אויסקלייבן נאר איין בראוזער פון די דריי, וועלכער עס געפעלט אייך. מיט דעם show-report באפעל קען מען ווידער עפענען דעם באריכט.

פארפאמעלעכן מיט slowMo (31:10:22)

צו פארפאמעלעכן די אויספירונג גייט מען אין דער טעקע playwright.config.js, און אונטער use שרייבט מען launchOptions מיט slowMo: 300, דאס איז 300 מיליסעקונדעס, און א קאמע צום סוף. ביים ערשטן פרואוו איז געווען א גרייז, ווייל מען האט פארגעסן צולייגן די קאמע. נאכן צולייגן די קאמע, אפהיטן און ווידער לויפן, זעט מען או עס לויפט פאמעלעכער, און מען קען זען אלע אקציעס איינס נאכן אנדערן אין סדר.

screenshot, video און UI-מאוד (31:11:23)

אין דער config-טעקע ווערן געוויזן נאך אפציעס:

- browserName: Chromium, באשטימט וואו די todo-אויטאמאציע זאל לויפן.
- screenshot: only-on-failure, אין באריכט ווערן געוויזן נאר בילדער פון די ערטער וואו עס איז געווען א דורכפאל, נישט אלע בילדער, ווייל אנדערש וואלטן געווען אזויפיל בילדער, וואס איז נישט עקאנאמיש מיט זכרון.
- video אויך נאר ביי דורכפאלן.

מען דריקט Ctrl+S, מען לויפט ווידער, און מען עפנט דעם באריכט מיט show-report. דער באריכט וואלט אריינגענומען screenshots און videos, אבער ווייל עס איז נישט געווען קיין דורכפאל, איז קיינס נישט ארויסגעקומען. וועז עס איז יא דא א דורכפאל, ערשיינען ביידע אויף דעם עקראן. מיט Ctrl+C גייט מען ארויס.

צום סוף ווערט געוויזן ווי מען לויפט אין אינטעראקטיוון UI-מאוד:

ui-- npx playwright test

עס עפנט זיך דער אינטעראקטיווער מאוד פון דער אפליקאציע.

אין א סך-הכל איבערחזרן: ווי מען רעקארדירט א טעסט, ווי מען פאררעכט גרייזן, ווי מען ברענגט אלע טעסטן צו דורכגיין, ווי מען פארפאמעלעכט, ווי מען באשטימט א בראוזער, און ווי מען נעמט screenshots און videos. Playwright האט אסאך מער ווי דאס, אבער דאס איז א לעקציע פאר אנהייבער. דער קורס מוטיקט צו אויטאמאטירן אן אנדער אפליקאציע מיט דער לעקציע ווי א ביישפיל, צו פרובירן מער באפעלן און מער פארגעשריטענע פונקציעס, און צו שרייבן א קאמענטאר ביי פראבלעמען.

די פיר שטאפלען פון א RAG-סיסטעם (31:13:57)

א נייער טעמע: אפשאצן (evaluate) א RAG-סיסטעם. צו ערשט דארף מען פארשטיין געוי ווואס פאסירט אינעווייניק אין א pipeline.

- ערשטער שטאפל, א שאלה פונעם באניצער. למשל, אן אנגעשטעלטער פרעגט אן אינערליכן AI-געהילף: וואס איז די פירמע'ס פאליסי ווענן ארבעטן פון דערהיים? ביי א געווענליכן שפראך-מאדעל וואלט דער

מאדעל געפרובירט ענטפערן בעיקר לויט דעם וואס ער האט זיך געלערנט ביים טרייניירן. דאס איז א קלארער פראבלעם: דער מאדעל האט אפשר קיינמאל נישט געזען די אינערליכע פאליסי פון דער פירמע, אדער די פאליסי האט זיך געטוישט נאכן טרייניירן.

• צווייטער שטאפל, retrieve context: די סיסטעם זוכט אין אן אויסערליכן וויסן-קוואל אינפארמאציע וואס שייכט צו דער שאלה. דער קוואל קען אנטהאלטן דאקומענטן פון דער פירמע, PDF-טעקעס, וועב-בלעטער, רעקארדס פון א דאטא-באזע און אנדערע אינפארמאציע. למשל, די סיסטעם געפינט די אפיציעלע פאליסי פון דער פירמע.

• דריטער שטאפל, generate answer: די שאלה און דער געפונענער קאנטעקסט ווערן איבערגעגעבן צום שפראך-מאדעל, וואס בויט דערפון דעם ענטפער. אידעאל דארף ער נישט אויסטראכטן א פאליסי, נאר געבן א ענטפער וואס באזירט זיך דירעקט אויף דעם געפונענעם דאקומענט.

• פערטער שטאפל, evaluate: דא הייבט זיך אן די טעמע. מען דארף באשטימען צי די געפונענע אינפארמאציע איז שייך, צי מען האט פארפעלט וויכטיקע אינפארמאציע, צי דער ענטפער איז געשטיצט פון די געפונענע ראיות, און צי דער לעצטער ענטפער איז טאקע ריכטיק.

עס זענען דא צוויי הויפט-געביטן צו אונטערזוכן: retrieval און generation. א גלאטער ענטפער איז נישט גענוג. א פארלעסלעכער RAG-ענטפער מוז זיין געשטיצט פון די ריכטיקע ראיות.

אפשאצן דעם retrieval (31:16:00)

מען הייבט אן מיט דער ערשטער העלפט פונעם pipeline, retrieval evaluation. איידער מען משפט'ט צי דער לעצטער ענטפער איז גוט, דארף מען באשטימען צי די סיסטעם האט בכלל געפונען גוטע אינפארמאציע. אויף דעם סלייד ווערן אונטערגעשטראכן פיר באגריפן: context precision, context recall, retrieval precision און retrieval recall. מען הייבט אן מיט context precision, וואס פרעגט בעיקר ווען דער געפונענער אינפארמאציע...

אפשאצן RAG סיסטעמען: מעטריקס, מכשירים און ארבעטס-גאנג (31:16:30)

קאנטעקסט-פרעציזיע און קאנטעקסט-ריקאל (31:16:30)

דער קורס ערקלערט ווייטער ווי אזוי מען שאצט אפ דעם "ריטריוועל" טייל פון א RAG סיסטעם. די ערשטע מעטריק, Context Precision, פרעגט: פון אלעס וואס די סיסטעם האט ארויסגעצויגן, ווי פיל דערפון איז טאקע שייך צו דער פראגע?

דער ביישפיל: א באניצער פרעגט ווען א פירמע'ס "refund" פאליסי, ד.ה. די כללים ווען מען באקומט צוריק געלט. די סיסטעם ציט ארויס 10 שטיקלעך טעקסט, אבער נאר דריי פון זיי רעדן טאקע ווען צוריקגעבן געלט. די אנדערע זיבן רעדן ווען שיפינג, וואראנטיס, באצאלונגען, און אפילו ווען גאנץ פרעמדע טעמעס. טעכניש האט די סיסטעם יא ארויסגעצויגן אינפארמאציע, אבער דער קאנטעקסט איז פול מיט "ראש" (noise). דא ווערט פרעציזיע וויכטיג.

די צווייטע מעטריק, Context Recall, קוקט אויף דער פראבלעם פון דער אנדערער זייט. מען פרעגט נישט ווי פיל פון דעם וואס מען האט געבראכט איז שייך, נאר: פון דער גאנצער שייכדיגער אינפארמאציע וואס געפינט זיך אין דער וויסנס-באזע, ווי פיל האט די סיסטעם געפונען? למשל, מען דארף פינף וויכטיגע שטיקלעך

אינפארמאציע כדי צו ענטפערן ריכטיג, אבער דער ריטריוועל געפינט נאר צוויי. וואס מען האט געפונען קען זיין אויסגעצייכנט, אבער דער קאנטעקסט איז נישט קאמפלעט.

Retrieval Precision און Retrieval Recall ארבעטן לויט דעם זעלבן פרינציפ, נאר ווען מען באטראכט די "chunks" (שטיקלעך) אדער דאקומענטן. Retrieval Precision זאגט צי די chunks וואס דער retriever האט צוריקגעגעבן זענען פאקוסירט און נוצליך. Retrieval Recall זאגט צי דער retriever האט געפונען אלע וויכטיגע chunks וואס האבן געדארפט ארויסקומען.

א גרינגער אופן צו געדענקען דעם חילוק:

- פרעציזיע פרעגט: ווי גוט איז דאס וואס איך האב ארויסגעצויגן?
- ריקאל פרעגט: ווי פיל פון דעם וואס איך האב געדארפט, האב איך געפונען?

דער חילוק איז וויכטיג, ווייל א שוואכער ריטריוועל שאפט פראבלעמען ווייטער אין דער קייט. אז דער קאנטעקסט איז פול מיט ראש, קען דער שפראך-מאדעל ווערן צעמישט. פעלט וויכטיגע אינפארמאציע, קען דער מאדעל געבן אן אומפולשטענדיגן ענטפער. און ווערט ארויסגעצויגן גאנץ פאלשע אינפארמאציע, קען דער מאדעל זיכער-זיך שרייבן א פאלשן ענטפער. דערפאר איז די עצה: איידער מען בארעכנט דעם שפראך-מאדעל פאר א שלעכטן RAG ענטפער, זאל מען אלעמאל קודם אונטערזוכן דעם ריטריוועל-שיכט.

אפשאצן דעם גענעראציע-טייל: געטריישאפט, שייכות, ריכטיגקייט (31:18:20)

נאכדעם וואס מען האט אפגעשאצט וואס גייט אריין אין מאדעל, גייט מען איבער צו דעם וואס קומט ארויס. נאכן ארויסציען די אינפארמאציע ניצט דער שפראך-מאדעל דעם קאנטעקסט כדי צו שרייבן דעם לעצטן ענטפער. איצט פרעגט מען נישט צי מען האט געבראכט די ריכטיגע אינפארמאציע, נאר צי דער מאדעל האט, מיט יענער אינפארמאציע, געשריבן דעם ריכטיגן ענטפער.

די ערשטע וויכטיגע מעטריק איז Faithfulness, אויך גערופן Groundedness. זי מעסט צי דער ענטפער ווערט טאקע געשטיצט פון דעם ארויסגעצויגענעם קאנטעקסט. ביישפיל: דער דאקומענט זאגט "קונים מעגן בעטן געלט צוריק ביז 30 טעג נאכן קויפן". דער מאדעל ענטפערט "קונים קענען בעטן געלט צוריק ביז 30 טעג". דער ענטפער איז געטריי, ווייל ער ווערט גלייך געשטיצט פון דעם קאנטעקסט. אבער לייגט דער מאדעל צו "פרעמיום קונים באקומען נאך 15 טעג", און די אינפארמאציע איז נישט געווען אינעם קאנטעקסט, האט דער מאדעל אריינגעברענגט אינפארמאציע אן קיין יסוד. דאס איז א פראבלעם פון "grounding" און מעגליך אויך א האלוצינאציע.

די צווייטע מעטריק איז Answer Relevance: צי דער ענטפער באהאנדלט טאקע די פראגע פון באניצער. ביישפיל: דער באניצער פרעגט ווי אזוי מען שטעלט איבער א פאסווארט, און דער אסיסטענט גיט א לאנגע ערקלערונג וועגן די בעסטע מנהגים פון סייבער-זיכערהייט. די אינפארמאציע מעג זיין נוצליך, אבער זי לייזט נישט דירעקט דעם באניצער'ס פראבלעם.

די דריטע מעטריק איז Answer Correctness: צי דער ענטפער איז פאקטיש ריכטיג, וועו מען פארגלייכט אים מיט א פארלעסליכן רעפערענץ-ענטפער אדער "ground truth".

די דריי דימענסיעס זענען פארבונדן, אבער נישט דאס זעלבע:

- א ענטפער קען זיין שייך אבער פאלש.
- ער קען זיין ריכטיג אבער שוואך באגרינדעט אינעם קאנטעקסט.

• ער קען זיין באגרינדעט אינעם קאנטעקסט און דאך נישט ענטפערן אויף א טייל פון באניצער'ס אמת'דיגער פראגע.

א קוואליטעט-סקאלע אויפן סלייד העלפט קוקן אויף דעם ווי א ספעקטרום. אויף איין עק זענען שלעכטע ענטפערס, אומשייך אדער האלוצינירט. דערנאך קומט א טיילווייז שייכדיגער ענטפער. וועז די קוואליטעט ווערט בעסער, ווערט דער ענטפער מער באגרינדעט און קאמפלעט. אויפן העכסטן שטאפל וויל מען א ענטפער וואס איז געטרייַ, שייך און ריכטיג. דער סך-הכל: ריטריוועל-אפשאצונג זאגט צי די סיסטעם האט געפונען גוטע ראיות, און גענעראציע-אפשאצונג זאגט צי דער מאדעל האט די ראיות ריכטיג גענוצט.

דער Ragas פריימווארק (31:20:40)

די נעקסטע שווערigקייט איז צו פירן אלע די מעטריקס אויף א סיסטעמאטישן אופן. איין RAG ענטפער אפשאצן מיט דער האנט איז גרינג. אבער פארשטעלט אייך אז מען דארף אפשאצן הונדערטער און טויזנטער פראגעס יעדעס מאל וואס מען טוישט דעם retriever, דעם פראמפט, דעם מאדעל, אדער אפילו די וויסנס-באזע. ביי אזא מאס דארף מען א מער סיסטעמאטישן צוגאנג.

איין פריימווארק וואס ווערט אפט גענוצט דערפאר איז Ragas. ער שאצט אפ די וויכטיגע טיילן פון א RAG פייפליין מיט פיר מעטריקס וואס זענען געוויזן אויפן סלייד: Faithfulness, Answer Relevancy, Context Precision און Context Recall. דער מעלה פון דער קאמבינאציע איז אז זי דעקט די צוויי הויפט-טיילן פון דער RAG ארכיטעקטור: Context Precision און Context Recall וויזן די קוואליטעט פון ריטריוועל, און Answer Relevance וויזט די קוואליטעט פון גענעראציע.

דער פייפליין אויפן סלייד:

• Retrieve – די סיסטעם זוכט שייכדיגע דאקומענטן.

• Augment – די ארויסגעצויגענע אינפארמאציע ווערט צוגעלייגט צום פראמפט אלס קאנטעקסט.

• Generate – דער שפראך-מאדעל שרייבט דעם ענטפער.

• Evaluate – מען שאצט אפ די רעזולטאטן. דא העלפט Ragas פארוואנדלען קוואליטאטיווע אויפפירונג אין מעסבארע סיגנאלן.

צוויי ביישפילן ווי מען לייענט די מעטריקס:

• Faithfulness איז הויך, אבער Context Recall איז נידעריג: דער שפראך-מאדעל ניצט ריכטיג די אינפארמאציע וואס ער באקומט, אבער דער retriever געפינט נישט גענוג אינפארמאציע צו ענטפערן אינגאנצן.

• פארקערט: Context Precision און Context Recall זענען שטארק, אבער Faithfulness איז שוואך: דער retriever ארבעט מסתמא ריכטיג, אבער דער שפראך-מאדעל לייגט צו אינפארמאציע אן יסוד.

דערפאר זענען באזונדערע מעטריקס אפט מער נוצליך ווי איין אלגעמיינער סקאר. דער ציל איז נישט נאר צו זאגן "מיין RAG סיסטעם האט באקומען 80%", נאר צו געפינען וועלכער טייל פון דעם פייפליין דארף פארבעסערט ווערן. Ragas גיט א מער סטרוקטורירטן וועג דערצו.

דריי צוגאנגען: אויטאמאטיש, מענטשליך, היברידיש (31:22:50)

מעטריקס און פריימווארקס זענען נאר א טייל פון דער מעשה. מען דארף אויך באשליסן ווי אזוי די אפשאצונג אליין ווערט דורכגעפירט. עס זענען פאראן דריי ברייטע צוגאנגען:

אויטאמאטישע אפשאצונג ניצט סאפטוועיר, פאראויס-באשטימטע מעטריקס, אדער אפילו שפראך-מאדעלן כדי צו געבן סקארס. איר גרעסטע מעלה איז די מאס. ביישפיל: מען האט 10,000 טעסט-פראגעס. אז א מענטש זאל אפשאצן יעדן ענטפער, וועט עס קאסטן אסאך צייט און געלט. אויטאמאטישע אפשאצונג לאזט איבערלויפן די טעסטס יעדעס מאל וואס מען טוישט די אפליקאציע, דערפאר איז זי באזונדערס נוצליך ביים אנטוויקלען און ביי "regression testing".

מענטשליכע אפשאצונג: מענטשן פארשטייען דקות'דיגע ניואנסן וואס אויטאמאטישע מעטריקס כאפן אמאל נישט. למשל, א ענטפער קען טעכניש אנטהאלטן די ריכטיגע פאקטן און דאך זיין צעמישט, פארפירעריש, שלעכט פארמולירט, אדער נישט פאסיג פארן געצילטן עולם. א מומחה אין דעם געביט קען אויך באמערקן פראבלעמען ספעציפיש פאר זיין פאך, וואס אן אלגעמיינער אפשאצער פארפעלט. דערפאר איז זי באזונדערס ווערטפול ווען קוואליטעט און ניואנסן זענען זייער וויכטיג.

היברידישע אפשאצונג קאמבינירט אויטאמאטישע סקארס מיט מענטשליכע איבערקוק. אין אסאך אפליקאציעס אין דער ווירקליכקייט איז דאס איינע פון די מערסטע פראקטישע סטראטעגיעס: מען שאצט אויטאמאטיש אפ טויזנטער ענטפערס, מען געפינט די חשד'דיגע אדער נידעריג-געשאצטע ביישפילן, און מען שיקט א קליינעם טייל צו א מומחה. אזוי האט מען די גיכקייט פון אויטאמאציע און מען באהאלט מענטשליכע אורטייל דארט וואו עס איז מערסטנס נויטיג.

דער הויפט-מסר: עס איז נישטא קיין איין פערפעקטער אופן. אויטאמאטיש גיט מאס, מענטשליך גיט ניואנסן, היברידיש גיט א באלאנס פון ביידע. די ריכטיגע ברירה הענגט אפ אינעם שטאפל פון דער אפליקאציע, קאסטן, ריזיקע, און דעם נויטיגן שטאפל פון גענויקייט.

א בילדליכע ערקלערונג און דער "טרעיד-אף" (31:24:55)

דער קורס קערט זיך צוריק צו ריטריוועל, כדי צו ווייזן פרעציזיע און ריקאל מער בילדליך. ביי Context Precision: יעדער בלאק שטעלט פאר א שטיקל ארויסגעצויגענע אינפארמאציע. טייל בלאקן זענען שייך צו דער פראגע, און טייל זענען ראש. מען פרעגט וועלכער פראפארציע פון דעם וואס איז ארויסגעצויגן איז טאקע שייך. איז כמעט אלעס נוצליך, איז די פרעציזיע הויך. ציט מען ארויס אסאך אומשייכדיגע אינפארמאציע, פאלט זי.

ביי Context Recall טוישט זיך די פערספעקטיוו: די בלאקן שטעלן פאר די גאנצע אינפארמאציע וואס דארף אידעאל זיין פאראן צו ענטפערן. ריקאל מעסט ווי פיל דערפון די סיסטעם האט געפונען. בלייבט וויכטיגע אינפארמאציע פעלן, איז דער ריקאל נידעריג. Retrieval Precision און Recall ניצן די זעלבע לאגיק לגבי די chunks וואס די זוך-סיסטעם גיט צוריק.

דאס שאפט א וויכטיגן אינזשעניר-"טרעיד-אף" (א פשרה צווישן צוויי צילן):

- מען שטעלט איין דעם retriever צו געבן נאר איין דאקומענט. ער קען זיין זייער שייך, און די פרעציזיע איז אויסגעצייכנט. אבער עס קענען זיין נאך פיר וויכטיגע דאקומענטן וואס מען האט אינגאנצן פארפעלט, און דער ריקאל איז שוואך.
- מען ציט ארויס 50 דאקומענטן. מען כאפט אפשר יעדן שייכדיגן דאקומענט, און דער ריקאל איז אויסגעצייכנט. אבער אויב רוב פון די 50 זענען אומשייך, האט מען אריינגעברענגט א ריזיגע מאס ראש, און די פרעציזיע פאלט.

דערפאר פארבעסערט נישט אויטאמאטיש א RAG סיסטעם וועט מען פארגרעסערט פשוט די צאל chunks. מען וויל אז דער retriever זאל געפינען גענוג אינפארמאציע, אן איבערשוועמקען דעם מאדעל מיט איבעריגן קאנטעקסט. דאס איז באזונדערס וויכטיג ווייל "context windows" זענען נישט קיין אומבאגרענ־יצטע רעסורס. מער קאנטעקסט קען פארגרעסערן קאסטן, פארשפעטיגונג (latency), און אפלענקונג. דער ציל איז נישט מאקסימום ריטריוועל, נאר הויכע קוואליטעט ריטריוועל. דער עיקר-לימוד אויפן סלייד: א גוטער ריטריוועל איז דער יסוד פון א גוטן RAG ענטפער.

מכשירים פאר אפשאצן RAG און LLM (31:27:13)

די נעקסטע פראקטישע פראגע איז וועלכע מכשירים קענען העלפן מעסטן. עס זענען פאראן עטליכע, און זיי לייזן נישט אלע די זעלבע פראבלעם:

- Ragas – ווי שוין ערקלערט, פאקוסירט ספעציפיש אויף RAG אפשאצונג, מיט מעטריקס פאר ריטריוועל און פאר געשריבענע ענטפערס. א נוצליכער אנהייב-פונקט פאר סטרוקטורירטע אפשאצונג ספעציפיש פאר RAG.
- TruLens – פאקוסירט אויף אפשאצן און אבזערווירן LLM אפליקאציעס. עס העלפט פראגראמירער פארשטיין ווי די אפליקאציע פירט זיך און אונטערזוכן די קוואליטעט פון די פארשידענע טיילן.
- LangSmith – באזונדערס נוצליך פאר "tracing", אפשאצן, עקספערימענטירן און מאניטארן ארבעט וואס ניצט LLMs. tracing ווערט וויכטיג ווייל א מאדערנע RAG סיסטעם קען האבן אסאך שטאפלען: א פראגע קען דורכגיין איבערשרייבן (rewriting), ריטריוועל, "reranking", בויען דעם פראמפט, גענעראציע ביים מאדעל, און "post-processing". אז עפעס פעלט, העלפט tracing פארשטיין פונקטליך וואו דער פעלער איז פארגעקומען.
- DeepEval – גיט א פריימווארק פאר אפשאצונג און טעסטינג וואס מען קען אריינלייגן אין דעם אנטוויקלונג-ארבעטס-גאנג. דאס איז נוצליך ווען א טים וויל אז אפשאצונג זאל זיך פירן מער ווי סאפטוועיר-טעסטינג.
- אייגענע (in-house) מעטריקס – זייער וויכטיג אין אפליקאציעס וואס ארבעטן שוין אין פראדוקציע. אלגעמיינע מעטריקס קענען נישט פארשטיין יעדער ארגאניזאציע'ס דעפיניציע פון הצלחה. ביישפילן: א פינאנציעלער RAG אסיסטענט קען פארלאנגען שטרענגע נומעריש גענויקייט, א לעגאלע אפליקאציע זארגט שטארק וועגן ריכטיגע ציטאטן (citations), און א קונה-סערוויס אפליקאציע זארגט וועגן האלטן זיך צו דער פאליסי.

דער ארבעטס-גאנג אויפן סלייד גייט דורך די מעגליכקייטן איינע נאך די אנדערע: מען קען אנהייבן מיט Ragas פאר די יסוד-מעטריקס, ניצן TruLens אדער LangSmith צו "דיבאגן" און אבזערווירן די אויפפירונג, ניצן DeepEval פאר סטרוקטורירטע טעסטינג, און דערנאך פארברייטערן מיט מעטריקס וואס מען שאפט ספעציעל פאר דער אייגענער אפליקאציע. דער עיקר: קיין איין מכשיר טוט נישט אלעס. מען זאל אויסקלייבן מכשירים לויט דעם וואס מען וויל מעסטן, אנשטאט צו פרובירן אז איין פריימווארק זאל לייזן יעדע אפשאצונג-פראבלעם.

א פראקטישער ארבעטס-גאנג פאר RAG אפשאצונג (31:29:30)

דער קורס ווייזט ווי RAG אפשאצונג קען אויסזען בעת דער אנטוויקלונג:

• שאפן אן אפשאצונג-דאטא-סעט – איינער פון די וויכטיגסטע שטאפלען. מען דארף א רעפרעזענטאטיוו זאמלונג פראגעס וואס באניצער וועלן מסתמא פרעגן, און וואו מעגליך אויך די ערווארטעטע ענטפערס און דעם שייכדיגן "ground truth" קאנטעקסט. די קוואליטעט פון דעם דאטא-סעט באשטימט גלייך די קוואליטעט פון דער אפשאצונג. זענען די טעסט-פראגעס נישט רעאליסטיש, קענען די סקארס אויסזען אויסגעצייכנט, בשעת די אפליקאציע אין פראדוקציע ארבעט נאך אלץ שלעכט.

• לויפן די אפשאצונג מיט מכשירים – מען קלייבט אויס די מעטריקס וואס זענען וויכטיג פאר דער אפליקאציע: פאר ריטריוועל למשל Context Precision און Context Recall , פאר גענעראציע Faithfulness, Relevance און Correctness. דערנאך לאזט מען דעם דאטא-סעט דורך אן אפשאצונג-פריימווארק.

• אנאליזירן די רעזולטאטן – דא ווערט די אפשאצונג פראקטיש. מען קוקט נישט נאר אויפן סקאר צו זען צי ער איז גוט אדער שלעכט, נאר מען זוכט מוסטערן. אפשר פעלן כסדר פראגעס וואס האבן צו טון מיט טאבעלעס; אפשר איז Faithfulness אויסגעצייכנט אבער Context Recall שוואך; אפשר ארבעט איין קאטעגאריע דאקומענטן אסאך ערגער ווי די אנדערע. אזעלכע מוסטערן העלפן געפינען די שורש-סיבות.

• פארבעסערן דעם פייפליין – איז דער ריטריוועל די פראבלעם, קען מען טוישן די chunk גרייס, דעם "embedding" מאדעל, די זוך-סטראטעגיע, "metadata" פילטערס, אדער דעם re-ranking צוגאנג. איז די גענעראציע די פראבלעם, קען מען פארבעסערן דעם פראמפט, טוישן דעם מאדעל, צופאסן די אנווייזונגען, אדער פארבעסערן דעם קאנטעקסט וואס מען גיט דעם מאדעל.

• דערנאך לויפט מען ווידער די אפשאצונג.

דערפאר איז דאס נישט קיין גלייכער פראצעס פון אנהייב ביזן סוף, נאר א קרייז וואס חזר'ט זיך: מעסטן, אנאליזירן, פארבעסערן און ווידער אפשאצן, מיט דער צייט. אזוי שאפט מען אן אנטוויקלונג-פראצעס וואס מען קען מעסטן, אנשטאט זיך צו פארלאזן נאר אויף עטליכע מיט-דער-האנט אויסגעקלויבענע ביישפילן וואס האבן צופעליג געארבעט.

צום סוף ווערט אנגעהויבן אז אפילו מיט א שטארקן ארבעטס-גאנג האט RAG אפשאצונג נאך אלץ וויכטיגע שווערי‌קייטן, און די ערשטע וואס ווערט געוויזן אויפן סלייד איז האלוצינאציע.

אפשאצן RAG, LLM בענטשמארקס און דאטא-עטיק (31:31:48)

די שווערי‌קייטן ביים אפשאצן א RAG סיסטעם (31:31:48)

דער קורס האלט ביי די הויפט-שווערי‌קייטן ביים אפשאצן א RAG סיסטעם. די ערשטע איז "האלוצינאציע": דער מאדעל קען ארויסגעבן אן ענטפער וואס קלינגט זיכער און פראפעסיאנעל, אבער ער לייגט אריין אינפארמאציע וואס שטיצט זיך נישט אויף דעם געפונענעם קאנטעקסט. עס ווערט באטאנט אז דאס איז באזונדערס געפערליך, ווייל א פאלשער ענטפער זעט נישט אלעמאל אויס פאלש.

די צווייטע שווערי‌קייט איז שוואכע זוכענ‌יש (retrieval). אויב אומפאסיגע אדער אומפולשטענדיגע אינפארמאציע קומט אריין אין דעם שטאפל פון שאפן דעם ענטפער, ווערט די קוואליטעט פונעם ענטפער זיכער ליידן. דערפאר איז אזוי וויכטיג צו אפשאצן די זוכענ‌יש באזונדער פון דעם שאפן דעם ענטפער (generation). אויב נישט, קען מען פאלש באשולדיגן דעם שפראך-מאדעל פאר א פראבלעם וואס די זוכענ‌יש האט גורם געווען.

די דריטע שווערישקייט איז א פאלשער ענטפער כאטש די זוכענישׁ איז גוט. דאס איז דער פארקערטער פאל: דער retriever געפינט פונקטליך דעם ריכטיגן דאקומענט, אבער דער שפראך-מאדעל פארשטייט אים נישט ריכטיג, איגנארירט א ווי׳כטיגן תנאי, אדער פארבינדט די אינפארמאציע אויף א פאלשן אופן. דער קורס ברענגט א ביישפיל: דער קאנטעקסט זאגט "צוריקצאלן ווערן ערלויבט ביז 30 טעג, נאר פאר פראדוקטן וואס זענען נישט געעפנט געווארן". דער מאדעל ענטפערט אבער: "אלע פראדוקטן קען מען צוריקגעבן ביז 30 טעג". די זוכענישׁ איז געווען ריכטיג, אבער ביים שאפן דעם ענטפער איז א ווי׳כטיגער תנאי פארלוירן געגאנגען.

די פערטע שווערישקייט איז פארשטעלונגען (biases) אין דער אפשאצונג אליין. מעסטונגען זענען נישט אויטאמאטיש אביעקטיוו נאר ווייל זיי געבן ארויס נומערן. עס ווערן געברענגט עטליכע ביישפילן:

- דער טעסט דאטא-סעט קען אנטהאלטן צופיל גרינגע פראגעס.
- א מאדעל וואס שאצט אפ אנדערע ענטפערס קען בעסער האלטן פון א געוויסן שרייב-סטיל.
- דער "ריכטיגער" ענטפער וואס מען האט צוגעגרייט קען אליין זיין אומפולשטענדיג.
- די קריטעריעס פון דער אפשאצונג שפיגלען אפשר נישט אפ דאס וואס אמתע באניצער דארפן טאקע.

דערפאר, ווערט ערקלערט, טאר די אפשאצונג קיינמאל נישט ווערן א בלינדע יאג נאך העכערע נומערן. די נומערן דארפן העלפן פארשטיין ווי די סיסטעם פירט זיך. דער ציל איז צו צוזאמשטעלן מעסטונגען, ביישפילן, traces) שפורן פון דעם גאנג פון דער סיסטעם), און וועג נויטיג, מענטשליכן משפט, כדי צו פארשטיין וואס עס גייט טאקע פאר.

דער פולער לעבנס-ציקל פון RAG אפשאצונג (31:33:40)

נאכן אנערקענען די שווערישקייטן, ברענגט דער קורס אלעס צוזאמען אין א לעצטן לעבנס-ציקל, וואס פאסט צוזאם דעם גאנצן געדאנק הינטער RAG אפשאצונג:

- בויען (Build): מען שאפט די RAG pipeline: די דאקומענטן, די זוך-סיסטעם, די פראמפטס, דעם מאדעל און די לאגיק פון דער אפליקאציע.
- אפשאצן (Evaluate): מען שאצט אפ סיי די זוכענישׁ און סיי דאס שאפן דעם ענטפער. מען מעסט צי די ריכטיגע אינפארמאציע איז געפונען געווארן, און צי דער מאדעל האט זי ריכטיג באנוצט.
- אנאליזירן (Analyze): דא גייט מען ווייטער פון דעם נומער. מען זוכט די לעכער, מען באטראכט די ביישפילן וואס זענען דורכגעפאלן, מען קוקט אין די traces, און מען פרובירט צו פארשטיין דעם שורש פון דער פראבלעם. מען פרעגט: איז דער פאלשער דאקומענט געפונען געווארן? איז א ווי׳כטיגער חלק (chunk) אויסגעלאזט געווארן? איז דער ריכטיגער קאנטעקסט געגעבן געווארן אבער נישט ריכטיג פארשטאנען? האט דער מאדעל צוגעלייגט אויסגעטראכטע אינפארמאציע?
- פארבעסערן (Improve): וועג מען פארשטייט די סיבה, קען מען פארבעסערן די זוכענישׁ, טוישן די פראמפטס, טוישן ווי מען צעטיילט די טעקסטן (chunking), פארבעסערן דעם דאטא-סעט, אריינלייגן re-) rankingאיבער-סדר'ן די רעזולטאטן), טוישן מאדעלן, אדער שאפן נייע שוץ-גרעניצן (guardrails). דערנאך הייבט זיך דער ציקל אן פון ניי.

עס ווערט ערקלערט פארוואס דער קרייז איז ווי׳כטיג: א RAG סיסטעם ווערט קיינמאל נישט באמת פארטיג. די ווי׳סן-באזע טוישט זיך, נייע דאקומענטן ווערן צוגעלייגט, באניצער הייבן אן פרעגן פראגעס וואס מען האט נישט פאראויסגעזען, מאדעלן ווערן דערהויבן, פראמפטס אנטוויקלען זיך, און די געשעפטליכע פאדערונגען טוישן זיך. דערפאר טאר די אפשאצונג נישט זיין נאר א לעצטער שריט נאך דער אנטוויקלונג; זי דארף זיין א

טייל פונעם אנטוויקלונג-ציקל אליין. דער הויפט-ציל, ווי עס שטייט אונטן אויפן סלייד, איז: שטענדיג מעסטן, לערנען און פארבעסערן, כדי צו געבן געוויע און פארלאזליכע ענטפערס. דאס, זאגט דער קורס, איז וואס א גוטע RAG אפשאצונג גיט: נישט בלויז א זאמלונג נומערן, נאר א געארדנטער וועג צו בויען AI אפליקאציעס וואס ווערן מיט דער צייט אלץ מער פארלאזליך.

וואס זענען LLM בענטשמארקס און ווי ארבעטן זיי (31:35:13)

דערנאך גייט דער קורס איבער צו LLM בענטשמארקס. דאס זענען סטאנדארטיזירטע געצייג צו אפשאצן ווי גוט גרויסע שפראך-מאדעלן ארבעטן. זיי געבן א געארדנטן וועג צו פרובירן LLMs אויף א געוויסער ארבעט אדער פראגע, מיט ביישפיל-דאטא און פאראויס באשטימטע מעסטונגען. זיי שאצן אפ פארשידענע פעאיגקייטן, ווי קאוד שרייבן, שכל-הישר טראכטן (common sense reasoning), און NLP ארבעטן ווי מאשין-איבערזעצונג, ענטפערן אויף פראגעס און קיצור'ן טעקסטן.

די וויכטיגקייט פון בענטשמארקס איז זייער ראלע אין פאראויסשטופן די אנטוויקלונג פון מאדעלן. זיי פאלגן נאך דעם פארשריט פון א LLM, און געבן נומערישע איינזיכטן וואו דער מאדעל איז שטארק און וואו ער דארף פארבעסערונג. די רוקמעלדונג איז קריטיש פאר דעם fine-tuning פראצעס, כדי פארשער און פראגראמירער זאלן קענען פארבעסערן דעם מאדעל. אויסערדעם געבן בענטשמארקס אן אביעקטיוע פארגלייכונג צווישן פארשידענע LLMs, וואס העלפט פראגראמירער און פירמעס אויסקלייבן דעם בעסטן מאדעל פאר זייערע באדערפענישן.

דער פראצעס איז קלאר און סיסטעמאטיש: מען גיט דעם LLM אן ארבעט, מען שאצט אפ ווי גוט ער האט עס געטון מיט געוויסע מעסטונגען, און מען גיט א נומער. די שריט:

- צוגרייטונג (Setup): דער בענטשמארק קומט מיט צוגעגרייטע ביישפיל-דאטא, ווי קאוד-אויפגאבעס, לאנגע דאקומענטן, מאטעמאטיק-פראבלעמען און אמתע שמועסן. די אויפגאבעס דעקן פארשידענע געביטן: שכל-הישר טראכטן, פראבלעם-לייזן, ענטפערן פראגעס, קיצור'ן און איבערזעצן, און אלעס ווערט פארגעלייגט פאר דעם מאדעל אין אנהייב פון דער פרובע.
- פרובירן (Testing): דער מאדעל ווערט געפרובירט אויף איינעם פון דריי אופנים. Few-shot : מען גיט דעם LLM עטליכע ביישפילן פאר מען בעט אים טון די ארבעט, וואס ווייזט ווי גוט ער לערנט פון ווייניג דאטא. Zero-shot: מען בעט דעם מאדעל טון די ארבעט אן קיין שום ביישפילן, וואס פרובירט זיין פעאיגקייט צו פארשטיין נייע באגריפן און זיך צופאסן צו אומבאקאנטע מצבים. Fine-tuned: דער מאדעל ווערט געטריינירט אויף א דאטא-סעט ענליך צו דעם פונעם בענטשמארק, כדי צו פארבעסערן זיין ארבעט אויף יענע באשטימטע אויפגאבע.
- נומערירן (Scoring): נאך דער ארבעט פארגלייכט דער בענטשמארק דעם מאדעל'ס ענטפער מיטן ערוואַרטעטן ענטפער, און גיט א נומער, געווענליך פון 0 ביז 100, וואס ווייזט ווי גענוי דער LLM האט גענארבעט.

די הויפט-מעסטונגען פאר בענטשמארקס (31:37:46)

דער קורס רעכנט אויס די מעסטונגען וואס מען באנוצט געווענליך:

- Accuracy/Precision: מעסט ווי פיל פראצענט פון די פאראויסזאגונגען פונעם מאדעל זענען ריכטיג.
- Recall) אויך גערופן sensitivity): מעסט די צאל true positives, ד"ה די ריכטיגע פאראויסזאגונגען וואס דער מאדעל האט געמאכט.

• F1 score: פאראייניגט accuracy און recall אין איין מעסטונג, מיט גלייכער וואג פאר ביידע, כדי צו באהאנדלען פאלשע positives אדער negatives . דער נומער איז פון 0 ביז 1, וואו 1 מיינט פולקאמענע precision און recall.

• Exact match: פאלגט נאך וויפיל פראצענט פון די ענטפערס שטימען פונקטליך מיטן ריכטיגן ענטפער; באזונדערס באנוצט ביי איבערזעצונג און ענטפערן פראגעס.

• Perplexity: ווייזט ווי גוט א מאדעל זאגט פאראויס דאס קומענדיגע ווארט אדער token . א נידעריגערער perplexity מיינט אז דער מאדעל פארשטייט בעסער די אויפגאבע.

• BLEU (Bilingual Evaluation Understudy): ווערט באנוצט פאר אפשאצן מאשין-איבערזעצונג, דורכן פארגלייכן n-grams, ד"ה סעריעס פון ווערטער וואס שטייען נעבן אנאנד, צווישן דעם מאדעל'ס איבערזעצונג און א מענטשליכער איבערזעצונג.

די נומערישע מעסטונגען ווערן אפט צוזאמגעשטעלט פאר א גרינטליכערע אפשאצונג. דערצו ברענגט מענטשליכע אפשאצונג קוואליטאטיווע פאקטארן, ווי צוזאמהאנג (coherence), שייכותדיגקייט און באדייטונג, וואס געבן א פיינערע אפשאצונג. אבער מענטשליכע אפשאצונג נעמט צייט און איז סוביעקטיוו; דערפאר איז וויכטיג צו האלטן א באלאנס צווישן נומערישע און קוואליטאטיווע מעסטונגען.

די חסרונות פון בענטשמארקס, און ליסטעס פון מאדעלן (31:39:17)

כאטש בענטשמארקס זענען נוצליך, האבן זיי חסרונות וואס לאזן זיי נישט פולשטענדיג פאראויסזאגן ווי א מאדעל ווערט ארבעטן אין דער אמתער וועלט:

• באגרעניצטע נומערן (bounded scoring): ווען א מאדעל דערגרייכט דעם העכסטן נומער אויף א בענטשמארק, פארלירט דער בענטשמארק זיין נוצן, און מען דארף אים דערהיינטיגן מיט שוועריגערע אויפגאבעס.

• א בריטער דאטא-סעט: בענטשמארקס שטיצן זיך אפט אויף דאטא פון פארשידענע געביטן; דער בריטער אומפאנג קען נישט גוט אפשאצן דעם מאדעל אין זעלטענע פעלער (edge cases), ספעציעלע פעלדער אדער באשטימטע באנוצונגען, וואו מען וואלט געדארפט צוגעפאסטע דאטא.

• באגרעניצטע אפשאצונג: א בענטשמארק פרובירט נאר די היינטיגע פעאיגקייטן; ווען LLMs אנטוויקלען זיך און נייע פעאיגקייטן קומען ארויס, דארף מען שאפן נייע בענטשמארקס.

• overfitting: אויב א LLM ווערט געטרייניירט אויף די זעלבע דאטא וואס דער בענטשמארק באנוצט, ארבעט ער גוט אויף דער טעסט-דאטא אבער האט שוועריקייטן מיט אמתע ארבעט, און די נומערן שפיגלען נישט אפ זיינע בריטערע פעאיגקייטן.

דערנאך ערקלערט דער קורס LLM leaderboards: זיי פארעפנטליכן א ראנגליסטע פון LLMs לויט פארשידענע בענטשמארקס, כדי נאכצופאלגן די פילע מאדעלן און פארגלייכן זיי; זייער נוצליך ביים באשליסן וועלכער מאדעל פאסט פאר אייך. אויף דער ליסטע וואס ווערט געוויזן:

• בעסטע אין multitask reasoning: OpenAI (o1) פירט, GPT-4o איז צווייטער, און דערנאך Llama 3 מיט 405B פאראמעטערס און (Claude 3.5 Sonnet) 3.5.

• בעסטע אין קאוד: OpenAI o1 פירט, אויפן צווייטן פלאץ Sonnet 3.5, און אויפן דריטן GPT-4.

• שנעלסטע מאדעלן: Llama 8B, דערנאך Llama 70B, און דריטער Gemini 1.5 Flash.

- נידעריגסטע (latencyאפהאלטונג): ווידער פירט Llama.
- ביליגסטע מאדעלן: Llama 8B פירט, צווייטער Gemini Flash 1.5, און דריטער GPT-4o mini.

צום סוף ווערט געוויזן א פארגלייך צווישן Claude 3 Opus און GPT-4 : אין אלגעמיין זענען זיי גלייך; אין reasoning פירט Claude 3 Opus; אין קאוד פירט GPT-4; אין מאטעמאטיק ווידער GPT-4; אין באנוץ פון געצייג (tool use) פירט Claude 3 Opus; און אין פיל-שפראכיגקייט פירט Claude 3 Opus.

וואס איז דאטא-עטיק (31:41:55)

א נייער טייל הייבט זיך אן מיט דאטא-עטיק: די פרינציפן וואס פירן ווי מען זאמלט, האלט, אנאליזירט, טיילט און באנוצט דאטא אויף א פאראנטוואורטליכן אופן. אין אנאליטיק איז גרינג זיך צו קאנצענטרירן נאר אויף נומערן, מאדעלן, dashboards און פאראויסזאגונגען, אבער הינטער פילע דאטא-סעטס שטייען אמתע מענטשן. א קונה-רעקארד קען אנטהאלטן איינעמ'ס נאמען, ארט, קויף-געווואוינהייטן, איינקונפט, געזונט-אינפארמאציע אדער אנליין טעטיקייט, און ווי מען באהאנדלט אזא אינפארמאציע קען האבן אמתע תוצאות.

א ביישפיל: א פירמע זאמלט קונה-אינפארמאציע כדי צו געבן אן אנליין סערוויס. צו באנוצן די אינפארמאציע פאר דעם געבעטענעם סערוויס איז ערווארטעט; אבער צו באנוצן די זעלבע אינפארמאציע פאר אן אנדער ציל, אן קלאר מודיע זיין דעם קונה, הייבט אויף אן עטישע פראגע. דאטא-עטיק העלפט פירמעס פרעגן צי דער באנוץ פון אינפארמאציע איז פאסיג, יושר'דיג, דורכזיכטיג און רעספעקטירט די מענטשן וואס די דאטא שטעלט פאר.

דער קורס (Simplilearn) רעכנט אויס די געביטן וואס זיין דאטא-עטיק לערן-פלאן באטאנט: פריוואטקייט, אינפארמירטע הסכמה, זיכערהייט, דורכזיכטיגקייט, אחריות, פארשטעלונגען (bias), יושר, עטישע באשלוסן, ווערבאליזאציע און פאראנטוואורטליכע AI. עטישע פראקטיק גילט דורכן גאנצן לעבנס-ציקל פון דאטא, פון דעם מאמענט וואס מען זאמלט די אינפארמאציע ביז מען מעקט זי אויס. דער ציל איז נישט אפצושטעלן פירמעס פון באנוצן דאטא, נאר צו מאכן זיכער אז די דאטא שאפט ווערט אן אומנויטיג צו שאדן יחידים, קהילות, קונים אדער דער געזעלשאפט.

די הויפט-פרינציפן פון דאטא-עטיק (31:43:00)

- פריוואטקייט: רעספעקטירן מענטשן'ס פערזענליכע אינפארמאציע און באגרענצן אומנויטיגע אויסשטעלונג.
- יושר (fairness): פרובירן צו פארזיכערן אז באשלוסן געבויט אויף דאטא שטעלן נישט סיסטעמאטיש צוריק געוויסע יחידים אדער גרופעס.
- דורכזיכטיגקייט: מענטשן וואס א סיסטעם ווירקט אויף זיי דארפן האבן פאסיגע אינפארמאציע ווי זייער דאטא ווערט באנוצט, און, וואו עס פאסט, ווי וויכטיגע באשלוסן ווערן געמאכט.
- אחריות (accountability): ווען א פירמע בויט אדער באנוצט אן אנאליטישע אדער AI סיסטעם, קען זי נישט אריבערשיבן די אחריות אויף אן אלגאריטם ווען עפעס גייט פאלש. איינער מוז זיין פאראנטוואורטליך פאר ווי די סיסטעם איז געבויט, באוואכט און באנוצט.
- זיכערהייט: ווען מען זאמלט סענסיטיווע אינפארמאציע, האט מען די אחריות זי צו שיצן פון אומדערלויבטן צוטריט אדער מיסברויך.

די געדאנקען שטימען נאענט מיט די אייגנשאפטן וואס NIST באצייכנט פאר פארלאזליכע (trustworthy) סיסטעמען: גילטיג און פארלאזליך, זיכער, באשיצט און ווידערשטאנדספעאיג, פאראנטוואורטליך און דורכזיכטיג, ערקלערבאר און פארשטענדליך, פריוואטקייט-שיצנדיג, און יושר'דיג מיט שעדליכע פארשטעלונגען אונטער קאנטראל. די פרינציפן זענען פארבונדן: א זייער געוויער מאדעל קען נאך אלץ זיין אומעטיש אויב ער ברעכט פריוואטקייט; א באשיצטע סיסטעם קען נאך אלץ זיין אומיושר'דיג; און א יושר'דיגער מאדעל קען נאך שאפן פראבלעמען אויב קיינער קען נישט ערקלערן ווער איז פאראנטוואורטליך פאר זיינע באשלוסן. דערפאר פאדערט עטישע דאטא-פראקטיק א באלאנס צווישן עטליכע אחריותן, אנשטאט צו פארבעסערן בלויז איין טעכנישע מעסטונג.

פריוואטקייט, הסכמה און פאראנטוואורטליך זאמלען דאטא (31:44:27)

איינס פון די וויכטיגסטע געביטן איז צו באשליסן צי מען זאל בכלל זאמלען די אינפארמאציע. פירמעס האבן אפט די טעכנישע מעגליכקייט צו זאמלען ריזיגע מאסן קונה-אינפארמאציע, אבער עטישע זאמלונג דארף האבן קלארע צילן. א ביישפיל: א פיטנעס-אפליקאציע פארלאנגט אייער עלטער, טעטיקייט-שטאפל און געניטונג-צילן כדי צו געבן רעקאמענדאציעס; די אינפארמאציע איז שייך צום סערוויס. אבער אויב די זעלבע אפליקאציע בעט צוטריט צו אומשייכותדיגע קאנטאקטן אויפן טעלעפאן אן א באדייטנדער סיבה, דארפן באניצער פרעגן פארוואס מען דארף די אינפארמאציע.

דא ווערט אינפארמירטע הסכמה וויכטיג. הסכמה מיינט נישט בלויז ארויפלייגן א לאנגע פריוואטקייט-פאליסי אויף א וועבזייטל און אננעמען אז יעדער פארשטייט עס. מענטשן דארפן קענען פארשטיין וועלכע אינפארמאציע ווערט געזאמלט, פארוואס מען דארף זי, ווי מען וועט זי באנוצן, און וועלכע ברירות זיי האבן. אינדיע'ס Digital Personal Data Protection Act באשרייבט הסכמה אלס פריי, באשטימט, אינפארמירט, אן תנאים און קלאר, מיט א קלארער באשטעטיגנדער האנדלונג, און באגרעניצט די הסכמה צו פערזענליכע דאטא וואס איז נויטיג פאר דעם באשטימטן ציל. דאס געזעץ גיט אויך יחידים די מעגליכקייט צוריקצוציען זייער הסכמה. נאך א נוצליכער פרינציפ איז data minimization: זאמלען בלויז דאס וואס איז פאסיג נויטיג פאר דעם בדעה'טן ציל.

עטיק פון דאטא: פריוואטקייט, זיכערהייט און יושר (31:45:58)

געזעצן און דער באגריף פון צווועק-באגרענעצונג (31:45:58)

דער קורס ערקלערט אז מען דארף זאמלען נאר די דאטא וואס מען דארף טאקע האבן פאר א באשטימטן צווועק. דאס פארמינערט די ריזיקע פאר פריוואטקייט, און עס קען אויך פארמינערן זיכערהייט-ריזיקעס. דערפאר הייבט זיך עטישע זאמלונג אן מיט דריי פשוטע פראגעס:

- דארפן מיר טאקע די דאטא?
- איז דער מענטש אינפארמירט געווארן?
- ניצן מיר עס נאר פאר דעם צווועק וואס ער קען גלייכערהייט ערווארטן?

עס ווערט ערקלערט אז עטיק פון דאטא און געזעצן פאר שוץ פון דאטא זענען שטארק פארבונדן, אבער זיי זענען נישט פונקט דאס זעלבע. דאס געזעץ שטעלט אוועק פארלאנגען וואס ארגאניזאציעס מוזן אפהיטן. עטיק פרעגט צי א געוויסע פראקטיק איז פאראנטוואורטלעך, אפילו ווען דאס געזעץ גיט נישט קיין פשוטן ענטפער.

איינע פון די מערסט אנערקענטע ראמען פאר פריוואטקייט איז די GDPR (General Data Protection Regulation) פון אייראפעאישן פארבאנד. זי פירט אן די באהאנדלונג און דעם שוץ פון פערזענליכע דאטא, און גיט יחידים וויכטיקע רעכטן איבער זייער אינפארמאציע. אין אינדיע איז דא דער Digital Personal Data Protection Act 2023. ער גילט פאר באשטימטע באהאנדלונג פון דיגיטאלע פערזענליכע דאטא, און ער שטעלט אוועק פליכטן וועגן מעלדונג (notice), הסכמה, צוטריט, פארריכטן, אויסמעקן, און באהאנדלען טענות. די רעגירונג האט אויך ארויסגעגעבן די Digital Personal Data Protection Rules 2025.

פאר א דאטא-אנאליסט, זאגט דער קורס, איז דער עיקר לימוד נישט אויסצולערנען יעדן געזעץ-ארטיקל. דער עיקר איז אז מען טאר נישט באהאנדלען פערזענליכע דאטא ווי אן אומבאגרענעצטע קוואל. ארגאניזאציעס דארפן וויסן:

- וועלכע דאטא זיי האלטן,
- פארוואס זיי באהאנדלען זי,
- ווער האט צוטריט צו איר,
- ווי לאנג מען דארף זי האלטן,
- און וועלכע רעכטן די יחידים האבן.

די רעגולאציעס זענען אנדערש פון לאנד צו לאנד. אבער עטישע פרינציפן בלייבן זייער שייך, ווי למשל באגרענעצונג לויטן צוועק, הסכמה, פריוואטקייט, דורכזיכטיקייט און פאראנטוואורטלעכקייט. דערפאר דארף מען באטראכטן דאס אפהיטן דעם געזעץ ווי דעם מינימום. עטיק העלפט די פאכלייט טראכטן וועגן די ברייטערע תוצאות פון זייערע באשלוסן.

זיכערהייט פון דאטא און פאראנטוואורטלעכער צוטריט (31:47:28)

דער קורס שטעלט אוועק א חילוק צווישן צוויי באגריפן. פריוואטקייט פרעגט צי מען דארף בכלל זאמלען און ניצן די אינפארמאציע. זיכערהייט פרעגט צי די אינפארמאציע איז גוט באשיצט. וועו א ארגאניזאציע זאמלט פערזענליכע אדער סענסיטיווע אינפארמאציע, נעמט זי אויף זיך די אחריות צו באשיצן די דאטא. זי דארף זי היטן פון צוטריט אן ערלויבעניש, פון ליקס, פון גנבה, פון ענדערונגען און פון שלעכטן באנוץ. דאס איז נישט נאר א טעכנישע פראגע פון סייבער-זיכערהייט, נאר אויך אן עטישע אחריות.

א באריכט פון IBM פאר 2026 וועגן די קאסטן פון א דאטא-ברעך (Cost of a Data Breach) האט געפונען אז דער גלאבאלער דורכשניט פון א ברעך האט דערגרייכט $4.99 מיליאן. דאס ווייזט ווי ריזיק די פינאנציעלע מאס פון היינטיקע דאטא-אינצידענטן איז. אבער געלט-פארלוסטן זענען נאר איין תוצאה. א ליק פון מעדיצינישע רעקארדס, פינאנציעלע אינפארמאציע, פאסווערטער אדער אידענטיטעט-אינפארמאציע קען גלייך שאטן די מענטשן וועמענס דאטא איז ארויסגעשטעלט געווארן.

דערפאר פארלאנגט פאראנטוואורטלעכע דאטא-פארוואלטונג קאנטראל איבער צוטריט. נישט יעדער ארבעטער דארף אויטאמאטיש האבן צוטריט צו יעדן דאטא-סעט, נאר ווייל ער ארבעט ביי דער זעלבער ארגאניזאציע. דער צוטריט דארף זיך באזירן אויף לעגיטימע פליכטן. מען קען אויך באשיצן דאטא דורך:

- ענקריפשן (encryption),
- אויטענטיפיקאציע,
- זיכערע אויפבאהאלטונג,

- אויפזיכט (monitoring),
- בעקאפס,
- און פאסיגע כללים ווי לאנג מען האלט די דאטא.

אן אנדער וויכטיקער פרינציפ איז צו פארמינערן איבעריקע קאפיעס פון סענסיטיווע אינפארמאציע. אז די דאטא איז איבערגעקאפירט אויף לעפטאפס, ספרעדשיטס, קלאוד-דרייווס און פריוואטע מכשירים, וועלכע פיל שוועלכער צו קאנטראלירן ווער עס האט צוטריט. NIST רעכנט זיכערהייט, ווידערשטאנדספעאיקייט (resilience) און פריוואטקייט צווישן די הויפט-אייגנשאפטן פון אן AI סיסטעם וואס מען קען זיך אויף אים פארלאזן. עטישע זיכערהייט מיינט דעריבער צו פרעגן: ווער דארף צוטריט, פארוואס דארף ער עס, און ווי באשיצן מיר די מענטשן וועלכע די דאטא שטעלט פאר?

פארורטייל און יושר אין דאטא (31:49:00)

דער קורס ערקלערט אז דאטא קען אויסזען אביעקטיוו ווייל זי באשטייט פון ציפערן. אבער די ציפערן קענען אפשפיגלען די פארורטיילן פון די סיסטעמען און געזעלשאפטן וועלכע האבן זיי געשאפן. למשל, מען בויט א מאדעל פאר אויפנעמען ארבעטער, באזירט אויף היסטארישע רעקארדס פון ארבעטער. אויב די פריערדיקע באשלוסן האבן איבערגעטריבן באפארצוגט איין גרופע, וועט דער דאטא-סעט אנטהאלטן דעם מוסטער. א מאדעל וואס מען טרייניירט אויף אזא דאטא קען זיך דאן אויסלערנען דעם זעלבן אומגלייכקייט און אים איבערחזרן. דאס מיינט אז היסטארישע דאטא איז נישט אויטאמאטיש נייטראל.

א פארורטייל קען אריינקומען אין עטלעכע שטאפלען:

- דער דאטא-סעט קען פארשטעלן געוויסע באפעלקערונגען ווייניגער ווי זיי זענען.
- די לעיבלס קענען אנטהאלטן סוביעקטיווע מענטשלעכע מיינונגען.
- די פאריאבלען וואס מען קלייבט אויס פאר דער אנאליז קענען דינען ווי פארשטעלער (proxies) פאר סענסיטיווע אייגנשאפטן.
- דער מאדעל קען ארבעטן אנדערש ביי פארשידענע גרופעס, אפילו ווען זיין אלגעמיינע געניקייט זעט אויס שטארק.

דערפאר פארלאנגט יושר אז אנאליסטן זאלן גיין ווייטער ווי איין אלגעמיינער ציפער פון פאָרשטעלונג. זיי דארפן אונטערזוכן צי די טעותים און די רעזולטאטן זענען אנדערש צעטיילט צווישן די שייכותדיקע באפעלקערונגען, און אויך זוכן פארוואס. NIST באשרייבט אן AI וואס מען קען זיך פארלאזן ווי "יושרדיק, וואו שעדלעכע פארורטיילן ווערן פארוואלטעט". זיין ראם באטאנט אז מען דארף מעסטן און פארוואלטן ריזיקעס דורכאויס דעם גאנצן לעבנס-ציקל פון AI.

א וויכטיקער פונקט: יושר מיינט נישט שטענדיק צו געבן יעדן דעם זעלבן רעזולטאט. פארשידענע פאלן קענען פארלאנגען פארשידענע דעפיניציעס און מאסן פון יושר. די אחריות פונעם אנאליסט איז צו דערקענען מעגלעכן שאדן, צו דאקומענטירן די הנחות, צו פרובירן די רעזולטאטן, און אריינצונעמען די פאסיגע באטייליגטע ווען די באשלוסן רירן אן מענטשן. עטישע אנאליטיק פרעגט נישט נאר צי דער מאדעל איז געווי. זי פרעגט אויך ווער פראפיטירט, ווער קען ליידן, און צי די חילוקים זענען געראכטפארטיקט.

דורכזיכטיקייט, ערקלערבארקייט און אחריות (31:50:33)

וועז א סיסטעם באזירט אויף דאטא האט השפעה אויף א וויכטיקן באשלוס, ווילן מענטשן גלייכערהייט פארשטיין ווי מען איז צו דעם באשלוס געקומען. דאָ וועלן דורכזיכטיקייט און ערקלערבארקייט וויכטיק. דורכזיכטיקייט מיינט צו זיין אפן וועגן די שייכותדיקע זייטן פון ווי מען ניצט די דאטא און די אנאליטישע סיסטעמען. ערקלערבארקייט מיינט צו מאכן די סיבות אדער די פאקטארן הינטער א רעזולטאט גענוג פארשטענדלעך פארן פאל וואו מען ניצט עס.

א ביישפיל: אן AI סיסטעם רעקאמענדירט אפצואווארפן א בקשה פאר א הלוואה. צו ענטפערן נאר "דער אלגאריטם האט באשלאסן" איז נישט קיין שטארקער מאדעל פון אחריות. ארגאניזאציעס דארפן פארשטיין:

- די אינפארמאציע וואס גייט אריין אין סיסטעם,
- דעם צוועק פונעם מאדעל,
- זיינע באגרענעצונגען,
- און ווער איז פאראנטוואורטלעך צו איבערקוקן זיינע רעזולטאטן.

NIST רעכנט אחריות, דורכזיכטיקייט, ערקלערבארקייט און אינטערפרעטירבארקייט צווישן די אייגנשאפטן פון אן AI וואס מען קען זיך אויף אים פארלאזן. אויך די רעגולאציע באטאנט מער און מער דורכזיכטיקייט. אונטער דעם EU AI Act זענען עטלעכע פליכטן פון דורכזיכטיקייט געווארן גילטיק פונעם 2טן אויגוסט 2026. צווישן זיי זענען פארלאנגען וועגן געווסע שמועסן מיט AI און וועגן אינהאלט וואס AI שאפט.

אחריות איז גלייך אזוי וויכטיק, ווייל ערקלערונגען אליין זענען נישט גענוג. ארגאניזאציעס דארפן קלארע אייגנטימערשאפט: ווער האט באשטעטיקט די דאטא-סעטס, ווער האט געפרואווט דעם מאדעל, ווער פאסט אויף ווי ער ארבעט, און ווער רעאגירט אויב עס קומט פאר א שאדן. פאראנטוואורטלעכע סיסטעמען האלטן דאקומענטאציע, פראצעסן פון איבערקוקן, מענטשלעכע אויפזיכט, און מעגלעכקייטן צו ווידערשפרעכן פראבלעמאטישע רעזולטאטן. דער פרינציפ איז פשוט: אויטאמאציע קען אויספירן א אויפגאבע פון באשליסן, אבער זי טאר נישט אוועקנעמען די אחריות פון דער ארגאניזאציע פאר די תוצאות.

עטישע באשלוסן און פארוואלטונג פון דאטא (31:52:04)

עטיק פון דאטא ווערט פראקטיש ווען ארגאניזאציעס בויען זי אריין אין די טאג-טעגלעכע באשלוסן. מען טאר זי נישט באהאנדלען ווי א שמועס וואס פאסירט ערשט נאכדעם וואס א פראבלעם איז שוין געשען. פארוואלטונג פון דאטא (data governance) גיט סטרוקטורן צו באשליסן ווי מען זאמלט, קלאסיפיצירט, האלט אויף, גיט צוטריט, טיילט, האלט און צום סוף מעקט אויס אינפארמאציע. זי קען באשטימען ווער פארמאגט יעדן דאטא-סעט, פראצעסן פון באשטעטיקן, זיכערהייט-פארלאנגען, סטאנדארטן פון קוואליטעט, און די פליכטן פון יעדע גרופע.

דער AI Risk Management Framework פון NIST איז אויפגעבויט אויף פיר פונקציעס: Govern, Map, Measure, Manage. Govern גילט איבערן גאנצן פראצעס. דערביי מאפן ארגאניזאציעס אויס די ריזיקעס, מעסטן זייער השפעה, און פארוואלטן זיי מיט דער צייט.

עטישע באשלוסן קען מען אנהייבן מיט א פשוטע רייע פראגעס:

- וואס איז דער צוועק פונעם פראיעקט?
- וועלכע דאטא דארף מען טאקע?

- האבן מיר א פאסיגן יסוד צו ניצן זי?
- קען מען שאטן יחידים אדער גרופעס?
- זענען דא באהאלטענע פארורטיילן?
- קען מען ערקלערן דעם רעזולטאט?
- ווער וועט איבערקוקן די סיסטעם?
- וואס טוט מען ווען עפעס גייט שיף?

מען דארף באטראכטן די פראגעס פאר מען שטעלט אוועק די סיסטעם, און נישט ערשט נאכדעם וואס די טענות קומען. פארוואלטונג איז באזונדערס נוצלעך ווען עטלעכע גרופעס ארבעטן מיט דער זעלבער אינפארמאציע. די הנחות קענען לייכט פארלוירן ווערן ווען די דאטא גייט פון זאמלונג צו אינזשעניריע, אנאליטיק, וויזואליזאציע און AI סיסטעמען. עטישע פארוואלטונג שאפט דעריבער קלארע אחריות און קאנטראלן וואס מען קען איבערחזרן. זי העלפט די גרופעס דערקענען ריזיקעס איידער זיי ווערן ריכטיגע פראבלעמען אין דער וועלט.

עטיק אין וויזואליזאציע פון דאטא (31:53:36)

די עטישע אחריות ענדיגט זיך נישט ווען די אנאליז איז פארטיק. ווי מען שטעלט פאר דעם רעזולטאט קען אויך באאיינפלוסן די באשלוסן פון מענטשן. א טשארט קען ניצן גאר ריכטיקע ציפערן און דאך שאפן א פארפירערישן איינדרוק.

דער קורס ברענגט א ביישפיל: א באר-טשארט פארגלייכט פארקויפן צווישן צוויי יאר. אין איין יאר זענען געווען 98 מיליאן, און אינעם אנדערן 100 מיליאן. אויב מען הייבט אן די ווערטיקאלע אקס ביי 97 מיליאן, קען דער חילוק אויסזען ריזיק, כאטש די ריכטיגע ענדערונג איז גאנץ קליין. אן אנדער אפטע פראבלעם איז צו ווייזן נאר די תקופה וואס שטיצט א געוויסע מעשה, און צו באהאלטן פריערדיקע דאטא וואס וואלט געענדערט ווי מען פארשטייט עס. אנאליסטן קענען אויך פארפירן דורך:

- אומפאסיגע סארטן טשארטס,
- פעלנדיקע לעיבלס,
- צעמישטע מאסן,
- איבעריקע 3D עפעקטן,
- אדער דורך פארשטעלן קארעלאציע ווי זי וואלט באוויזן א סיבה (causation).

עטישע וויזואליזאציע פארלאנגט דעריבער געניקייט, צוזאמענהאנג, קלארקייט און ערלעכקייט. דער ציל איז נישט צו מאכן דעם טשארט ווייניגער אינטערעסאנט. דער ציל איז אז דער דיזיין זאל העלפן מענטשן פארשטיין די אינפארמאציע, און נישט מאניפולירן ווי זיי זענען זי. דער לימוד-פלאן פון Simplilearn פאר עטיק פון דאטא נעמט קלאר אריין די עטישע באטראכטונגען ביי וויזואליזאציע, צוזאמען מיט פריוואטקייט, יושר און פארשטענדיקע באשלוסן.

אנאליסטן דארפן פרעגן:

- שטעלט די וויזואליזאציע פאר די דאטא אין דער ריכטיגער פראפארציע?
- פעלט וויכטיקער צוזאמענהאנג?

- קען דער טיטל מוטיקן צו א מסקנא וואס האט נישט קיין יסוד?
- זענען די אומזיכערקייט און די באגרענעצונגען צו זען ווו עס איז נויטיק?

א גוטע וויזואליזאציע גיט נישט נאר איבער שנעל. זי גיט איבער דעם אמת.

עטיק אין מאשין-לערנען און AI (31:54:37)

עטיק פון דאטא ווערט באזונדערס וויכטיק ביי מאשין-לערנען און AI, ווייל די סיסטעמען קענען מאכן אדער באאיינפלוסן באשלוסן אויף א גאר גרויסן פארנעם. א טעות אין א ספרעדשיט קען שאטן איין באריכט. א פראבלעמאטישער אויטאמאטישער מאדעל קען איבערחזרן דעם זעלבן טעות טויזנטער אדער מיליאנען מאל.

פאראנטווארטלעכע AI פארלאנגט דעריבער עטיש טראכטן דורכאויס דעם גאנצן לעבנס-ציקל: זאמלען דאטא, אנטוויקלען דעם מאדעל, אפשאצן, אוועקשטעלן, אויפפאסן, און פארבעסערן. די דאטא פאר טריינירן דארף זיין שייכותדיק, געזונג פארשטעלנדיק, און פאסיג סיי לויטן געזעץ סיי לויט עטיק. מען דארף אפשאצן דעם מאדעל נישט נאר לויט זיין אלגעמיינער געניקייט. מען דארף אויך קוקן אויף פארלעסלעכקייט, פריוואטקייט, זיכערהייט, יושר און מעגלעכע שעדלעכע רעזולטאטן. נאכן אוועקשטעלן דארפן די גרופעס וועלכע אויפפאסן אויף דער סיסטעם, ווייל די ריכטיגע דאטא און דאס פירעכץ אין דער וועלט קענען זיך ענדערן.

דער ראם פון NIST רעכנט אויס די אייגנשאפטן פון אן AI וואס מען קען זיך אויף אים פארלאזן: גילטיקייט, זיכערקייט, זיכערהייט, אחריות, דורכזיכטיקייט, ערקלערבארקייט, פריוואטקייט און יושר. אויך די רעגולאציע-סביבה ענטוויקלט זיך. רוב באשטימונגען פונעם EU AI Act זענען געווארן גילטיק דעם 2טן אויגוסט 2026. נאך פארלאנגען פאר געווסע סיסטעמען מיט הויכער ריזיקע קומען לויט באזונדערע טערמינען.

פאר אנאליסטן איז דער הויפט-לימוד אז די טעכנישע פאָרשטעלונג איז נאר איין טייל פון פאראנטווארטלעכער AI. פאר מען שטעלט אוועק א סיסטעם, דארף מען פרעגן:

- איז די דאטא פאסיג?
- איז דער מאדעל יושרדיק?
- קען מען איבערקוקן וויכטיקע באשלוסן?
- זענען די באגרענעצונגען דאקומענטירט?
- איז עמעצער פאראנטווארטלעך?

פאראנטווארטלעכע AI מיינט צו בויען סיסטעמען וואס מענטשן קענען ניצן אויף אן יושרדיקן אופן, מיט א געראכטפארטיקטן צוטרוי.

א נייע לעקציע: יסודות פון עטישע דאטא-פראקטיקן (31:56:09)

דא הייבט זיך אן א נייע לעקציע וועגן עטיק פון דאטא, אין פארעם פון א לעבעדיקן קלאס. עס ווערט ערקלערט אז "דאטא-פראקטיקן" מיינט ארבעטן מיט דאטא פאר פארשידענע צוועקן. מען קען ניצן דאטא פאר דאטא-אנאליטיק, פאר ארבעט אין data science, אדער צו טריינירן א מאדעל פון מאשין-לערנען. מען קען ניצן געזאמלטע דאטא אויך פאר צילגעריכטע (targeted) און אנדערע אויפגאבעס. דערפאר פרעגט מען: וואס דארפן זיין די עטישע פראקטיקן וועו מען ארבעט מיט דאטא אדער מיט דאטאבייס-פראיעקטן?

דער ביישפיל פון ערלויבענישן אויף א טעלעפאן (31:57:10)

דער קורס ברענגט א ביישפיל פונעם טאג-טעגלעכן לעבן. וועו מען אינסטאלירט א נייע אפליקאציע אויף א טעלעפאן, סיי iOS סיי Android, בעט די אפליקאציע ערלויבענישׂ פאר געוויסע זאכן. זי בעט צוטריט צום ארט (location), צו דער קאמערע, צום מיקראפאן, אוו אמאל אויך צו די קאנטאקטן. די אפליקאציעס בעטן די ערלויבענישן כדי צו געבן געוויסע פונקציעס וואס זיי דארפן.

דערנאך וועלט געשטעלט א פראגע צום עולם: פארוואס דארפן די אפליקאציעס באקומען די הסכמה פונעם באניצער פאר זיי ניצן געוויסע מעגלעכקייטן? וואס קען פאסירן אויב מען וואלט נישט געדארפט קיין ערלויבענישן? וואס זענען די תוצאות אויב מען שטעלט נישט אוועק אזעלכע באגרענעצונגען, אדער מען לאזט נישט וויסן דעם באניצער וועגן דעם צוטריט צו די סערוויסעס פונעם טעלעפאן?

אן ענטפער פון עולם: אויב אן אפליקאציע בעט נישט קיין ערלויבענישׂ אוו ניצט דאך די קאנטאקטן, וואו עס געפינט זיך א לאנגע רשימה, קען מען ארויסשטעלן פערזענליכע דאטא. דער קורס באשטעטיקט אז דאס איז איינע פון די תוצאות: די פערזענליכע דאטא וועט זיכער וועלן ארויסגעשטעלט אויב מען מאכט עס נישט דורכזיכטיק אוו מען לאזט עס נישט וויסן דעם באניצער.

א צווייטער ענטפער: וואס אויב די אפליקאציע פאלגט דעם ארט פונעם באניצער אן זיין הסכמה? עס וועלט ערקלערט אז וועו מען גרייכט צו אזעלכע אינפארמאציע פונעם באניצער, געהערט דאס אלץ צו עטישע דאטא-פראקטיקן. אויב אן אפליקאציע גרייכט צו דער ארט-אינפארמאציע, קען מען דאס ניצן פאר אויפזיכט (surveillance). אויב דער באניצער וויסט נישט אז די אפליקאציע גרייכט צו זיין ארט, איז דאס פאלש. מען טאר נישט נאכפאלגן קיינעם סתם אזוי. ביז מען גיט נישט קיין הסכמה, קען קיינער נישט נאכפאלגן אוו קיינער נישט אויפשטעלן קיין אויפזיכט איבער אים.

דער קורס באטאנט אז היינט זענען מיר מערסטנס ארויסגעשטעלט צו אלערליי דיגיטאלע אפליקאציעס. דערפאר איז געווארן זייער וויכטיק צו פארשטיין די תוצאות פון טיילן פערזענליכע אינפארמאציע, פונעם קוקווינקל פונעם קונה. פונעם אנדערן זייט שטעלט מען די פראגע: אויב איך בוי אן אפליקאציע, וואס זענען מיין אחריות אוו מיינע פליכטן וועו איך גרייך צו אזעלכע אינפארמאציע?

דאטא-עטיק: פרינציפן, געזעצן אוו באניצער-צוטרוי (32:00:13)

וואס מען וועט לערנען אין דער לעקציע (32:00:13)

די לעקציע הייבט אן מיט א פראגע וואס יעדע פירמע מוז זיך שטעלן: וי אזוי באהאנדלט מען די פערזענליכע אינפארמאציע פון די קונים? דאס איז דער נושא פון דער לעקציע: וואס איז דאטא-עטיק, אוו וואס איז דער ריכטיגער וועג איינצופירן עטישע דאטא-פראקטיקן אין דער דיגיטאלער וועלט.

דער קורס שטעלט אוועק דריי צילן. ביים סוף פון דער לעקציע דארף דער לערנער קענען:

- איינפירן סטאנדארט דאטא-פראקטיקן אוו לעגאלע ראמען (legal frameworks) אין א פירמע, כדי צו בויען צוטרוי ביי די קונים אוו זיך האלטן צו די רעגולאציעס. דערפאר מוז מען פריער פארשטיין וואס זענען עטישע דאטא-פראקטיקן אוו וועלכע לעגאלע ראמען מען קען איינפירן, כדי צו שאפן צוטרוי צווישן דער פירמע אוו די קונים אוו זיך האלטן צו די רעגירונגס-רעגולאציעס.
- ערקלערן פארוואס עס איז וויכטיג צו באקומען "informed consent", א באוואוסטזיניגע הסכמה, פון א מענטש אדער א גרופע איידער מען זאמלט, פראצעסירט אדער טיילט פערזענליכע דאטא. אין דער

דיגיטאלער וועלט טאר מען נישט איבערגעבן קיין פערזענליכע אינפארמאציע פון א קונה אדער יחיד צו אן אנדערן אן זיין הסכמה. דערפאר, וועז מען זאמלט אדער פראצעסירט פערזענליכע אינפארמאציע, מוז מען האבן די הסכמה פון דעם מענטש וועמען עס באטרעפט.

• שאפן לייזונגען פאר עטישע פראבלעמען. למשל, אויב עס איז פארגעקומען א טעות אין סיסטעם, אדער עס איז געווען א "data breach" (א דורכברוך וואו דאטא איז ארויסגעגאנגען), אדער א קונה הייבט אויף א פראבלעם אדער גיט איין א קלאגע: ווי אזוי באהאנדלט מען דאס, און וועלכע דאטא-פראקטיקן דארף מען איינפירן אין דער פירמע כדי זיך צו היטן בעסער.

וואס איז דאטא-עטיק (32:02:16)

דאטא-עטיק ענטפערט אויף פראגעס וועגן די עטישע פרינציפן, אנווייזונגען און סטאנדארטן וואס פירן אן ווי אזוי מען זאמלט, באניצט, האלט אפ (storage) און טיילט דאטא. דער קורס לייגט צו נאך א פינפטן ענין: ווי אזוי מען פראצעסירט די דאטא. דאס איז אויך זייער וויכטיג.

דאטא-עטיק נעמט אריין אלע פראקטיקן, סטאנדארטן, כללים און רעגולאציעס וואס פירן אן מיט דאטא: ווי מען זאמלט, ווי מען באניצט, ווי מען האלט אפ, און וועלכע כללים און סטאנדארטן גילטן וועז מען טיילט די דאטא מיט א דריטער פארטיי.

דאטא-עטיק פארלאנגט אויך פון די וואס נעמען באשלוסן אז זיי זאלן באטראכטן ווי זייערע באשלוסן ווירקן אויף דעם יחיד, אויף דער געזעלשאפט און אויף דער ארגאניזאציע. ווי מען האט שוין געלערנט אין פריערדיגע לעקציעס, טוט מען דאטא-אנאליטיקס כדי צו פארווענדלען רויע דאטא אין אינפארמאציע. מיט דער אינפארמאציע נעמט מען בעסערע באשלוסן, און דאס איז דער ענדגילטיגער ציל פון יעדער ארגאניזאציע וואס טוט דאטא-אנאליטיקס, פירט א דאטא-סייענס פראיעקט אדער זאמלט דאטא דערפאר.

אבער ביים טון דאס מוזן די ארגאניזאציעס זיך האלטן צו כללים און סטאנדארטן פון דאטא-עטיק. די כללים זארגן פאר פיר זאכן:

• פריוואטקייט (privacy): די דאטא מוז זיין זיכער אפגעהאלטן אין דער ארגאניזאציע, און קיין נישט-בארעכטיגטער באניצער טאר נישט צוקומען צו איר.

• דורכזיכטיגקייט (transparency): מען באניצט די דאטא נאר פארן ציל פאר וועלכן מען האט זי געזאמלט. אז מען זאמלט דאטא פון א מענטש פאר א געוויסער ארבעט "X", טאר מען זי באניצן נאר פאר "X" און נישט פאר אנדערע צוועקן. אין דעם גייט אריין אויך די הסכמה פון דעם קונה אדער ענד-באניצער ביים זאמלען און פראצעסירן די דאטא.

• יושר (fairness): מען באניצט די דאטא אן קיין פארנומענקייט (bias). דער קורס זאגט אז מען וועט דערוועגן שפעטער רעדן מער באריכות.

• פאראנטוואורטליכקייט (accountability): די ראלע פון דער ארגאניזאציע. אויב עס איז פארגעקומען א data breach אדער אן אומקאנסיסטענץ אינעם גאנצן פראצעס, מוז די ארגאניזאציע נעמען אויף זיך די אחריות.

לעגאלע און רעגולאטארישע ראמען (32:05:51)

זינט דיגיטאלע דאטא איז געווארן אזוי פארשפרייט און א הויפט-וועג פון טיילן און אפהאלטן אינפארמאציע, האבן פארשידענע לענדער אנטוויקלט רעגולאטארישע ראמען. אויך ארגאניזאציעס זענען דערין אריינגעמישט,

למשל די אמעריקאנער פעדעראלע רעגירונג. ארגאניזאציעס מוזן באקומען צערטיפיקאטן אדער זיך האלטן צו די כללים און רעגולאציעס וואס יענע שטעלן אוועק פאר זייערע דאטא-פראקטיקן און דאטא-פראצעסן.

די הויפט-פרינציפן פון דאטא-עטיק (32:06:21)

דער קורס רעכנט אויס פינף הויפט-פרינציפן:

- דורכזיכטיגקייט: מען מוז זיין אפן וועגן די מעטאדן פון זאמלען, אפהאלטן און טיילן דאטא. אז מען זאמלט אינפארמאציע "X" פון א באניצער, מוז מען קלאר מאכן ווי אזוי מען זאמלט זי, וואו מען וועט זי אפהאלטן, וואס מען וועט מיט איר טון, און אויב מען טיילט זי מיט א דריטער פארטיי, וועלכע פראקטיקן מען פירט דערביי. אלעס דאס דארף זיין קלאר שוין ביים זאמלען.
- יושר: מען זאל אויסמיידן פארנומענקייט און דיסקרימינאציע ביים זאמלען דאטא, ביי דער אנאליז און ביי די אלגאריטמען. דער קורס ברענגט צוויי ביישפילן.

ערשטער ביישפיל: א פירמע וויל דורכפירן א שטודיע פון "customer segmentation", צעטיילן די קונים אין גרופעס, און זאמלט דערפאר דאטא. ביים זאמלען מוז מען אנרירן יעדן טייל פון די קונים, און נישט צילן נאר אויף מענער, נאר אויף יונגע קונים אדער נאר אויף קונים פון א געוויסער דעמאגראפישער געגנט. טוט מען דאס, ברענגט מען אריין bias אין דער דאטא, און דערמיט דיסקרימינירט מען די אנדערע גרופעס וואס האבן נישט באקומען קיין חלק אין דער דאטא.

צווייטער ביישפיל: מען בויט א מאדעל פאר דורכקוקן רעזומעס. דער מאדעל פראצעסירט א רעזומע און זאגט צי דער קאנדידאט פאסט פאר אן אינטערוויו: "fit" אדער "no fit". כדי צו בויען אזא מאדעל מוז מען זאמלען דאטא גלייך פון ביידע זייטן. אויב מען זאמלט מערסטנס דאטא פון מאנספערשוינען, ברענגט מען אריין bias צוגונסטן פון מענער אקעגן פרויען. ווען מען טריינירט א מאדעל אויף פארנומענער דאטא, ווערט דער מאדעל אויטאמאטיש אויך פארנומען צו דער זעלבער זייט. דער רעזולטאט איז דיסקרימינאציע: ווען עס קומט א רעזומע פון א פרוי, וועט זי מערסטנס אריינפאלן אין "no fit". דערפאר מוז מען זיין זייער פארזיכטיג אז די דאטא ווערט געזאמלט מיט פולן יושר.

- פאראנטווארטליכקייט: זייער וויכטיג פון דער ארגאניזאציע'ס שטאנדפונקט. יחידים און ארגאניזאציעס זענען פאראנטווארטליך פאר זייערע דאטא-פראקטיקן. עס דארף זיין עמעצער צו וועמען מען קען זיך וועגן אויב עס איז פארגעקומען א breach אדער אן אומקאנסיסטענץ אין דער דאטא-אנאליז אדער ביים פראצעסירן די דאטא.
- פריוואטקייט: שיצט דאס רעכט פון א יחיד צו קאנטראלירן זיין פערזענליכע אינפארמאציע, און יעדע ארגאניזאציע און יעדער מענטש מוז דאס רעספעקטירן. אויב א קונה וויל נישט אנטפלעקן זיין פערזענליכע אינפארמאציע, קען מען אים נישט צווינגען.
- זיכערהייט (security): ווען מען האט שוין געזאמלט און אפגעהאלטן די דאטא, איז עס די פליכט פון דער ארגאניזאציע אוועקצושטעלן שיצן און זיכערהייט-מיטלען, כדי קיין נישט-בארעכטיגטער זאל נישט קענען צוקומען צו דער דאטא אדער זי מאניפולירן.

די פרינציפן זענען פארשידענע דימענסיעס פון דעם ענין, און מען מוז אלע באטראכטן מיט גלייכער וויכטיגקייט.

ביישפיל: Snapchat און פנים-דערקענונג (32:10:29)

כדי צו מאכן די פרינציפן קלאר ברענגט דער קורס א באקאנטן ביישפיל: די פאפולערע אפליקאציע Snapchat. די אפליקאציע נעמט דאס פנים פון דעם באניצער און "שפילט" זיך דערמיט: זי לייגט אויף דעם פנים פארשידענע "props", אדער טוישט עס אויף פארשידענע פנים-אויסדרוקן. זי לייגט צו קרעאטיווע פילטערס וואס פארשענערן א "selfie" מיט הילף פון facial recognition, פנים-דערקענונג-טעכנאלאגיע. דאס איז די הויפט-פונקציע וואס Snapchat איז באקאנט דערמיט.

כדי דאס צו טון מוז די אפליקאציע ערשט צוקומען צו דער קאמערע. וועז מען הייבט די קאמערע צום פנים, נעמט זי א בילד פונעם פנים, און ניצט די פנים-דאטא כדי אנצוועגדן אלע די פונקציעס: פארשענערן דעם selfie, לייגן props אויפן פנים, אדער וואס נאר. אבער לסוף מוז דער באניצער געבן ערלויבעניש צו צוקומען צו דער קאמערע, און דערנאך נעמט זי א בילד פון זיין פנים.

דער קורס פרעגט: וועלכע עטישע באטראכטונגען זענען דא ביי אזא אפליקאציע?

- דורכזיכטיגקייט: Snapchat מוז קלאר ערקלערן דעם ציל פון דער פנים-דערקענונג-טעכנאלאגיע, און געבן דעם באניצער קאנטראל איבער זיינע privacy settings. וועז מען לאזט ארויס אזא פונקציע אין אן אפליקאציע, מוז מען קלאר ערקלערן וועלכע טעכנאלאגיע מען ניצט צו באארבעטן דאס בילד פונעם פנים. דערצו מוז מען געבן דעם באניצער די מאכט צו באשליסן צי ער גיט הסכמה צו ניצן זיין פנים פאר די צוועקן. וויל דער באניצער נישט, קען ער אפזאגן דער אפליקאציע צוטריט צו דער קאמערע. דאס איז דער ענין פון קאנטראל און ערלויבענישן: וועז מען אינסטאלירט אן אפליקאציע ווי Snapchat אויפן טעלעפאן און ניצט זי דאס ערשטע מאל, פרעגט זי פאר ערלויבעניש, אז זי גייט צוקומען צו דער קאמערע, צי מען איז מסכים, און מען מוז געבן די הסכמה.
- יושר: דער צווייטער פרינציפ ביי Snapchat איז יושר. דא שטעלט זיך די לעקציע אפ אינמיטן ערקלערן אז מען דארף טעסטן דעם אלגאריטם צי ער איז פונקטליך ביי פארשידענע גרופעס. די פארזעצונג קומט אין דעם קומענדיגן חלק.

דאטא-עטיק: פרינציפן, וויכטיגקייט און ביישפילן (32:13:33)

Snapchat-פילטערס: יושר, אחריות, פריוואטקייט און זיכערהייט (32:13:33)

דער קורס פירט וויטער מיטן ביישפיל פון Snapchat און זיינע פילטערס פאר פנימער, און ערקלערט וויי יעדער פרינציפ פון דאטא-עטיק קומט צו ניץ ביי אזא אפליקאציע.

דער ערשטער פונקט איז יושר (fairness). א מאדעל מוז ארבעטן גלייך פאר מענטשן פון אלע דעמאגראפישע גרופעס, כדי עס זאל נישט זיין קיין פארנייגונג (bias) און קיין דיסקרימינאציע. וועז מען לייגט צו א פארשענערונג, א "selfie"-פילטער אדער אן אנדערן קרעאטיוון פילטער אויפן פנים, טאר עס נישט ארבעטן נאר פאר געוויסע פנימער, למשל ליכטיקע, און דורכפאלן ביי טונקעלע פנימער. וועז דער אלגאריטם פירט זיך אזוי, איז די אפליקאציע נישט יושרדיג קעגן אירע באניצער פון פארשידענע ראסן, שטאמען און דעמאגראפיעס. עס ליגט אויף די מאכער פון Snapchat די אחריות אז די אפליקאציע זאל ארבעטן גלייך גוט מיט פנימער פון אלע ראסן און מיט אלע סארטן מענטשן.

דער צווייטער פונקט איז אחריות (accountability). די פירמע מוז אויפשטעלן א באזונדערע מאנשאפט וואס זאל שנעל נאכקוקן די עטישע השלכות און ענטפערן אויף די זארגן פון די באניצער. ווי עס איז שוין פריער

דערמאנט געווארן, דארף מען האבן א קאנטאקט-פערזאן וואס זאל זיך נעמען צו יעדע דאטא-עטישע פראבלעם וועלכע עס קומט אדער א יחיד ברענגט זי ארויף.

דער דריטער פונקט איז פריוואטקייט (privacy). דער סטאנדארט איז אז דער באניצער זאל פארשטיין אין קאנטראלירן ווי מען קלייבט אים די דאטא פון זיין פנים אין וואס מען טוט דערמיט. יעדעס מאל וואס די אפליקאציע קלייבט פנים-דאטא, דארף דער באניצער גוט וויסן ווי מען קלייבט זי אין ווי מען ניצט זי. ער דארף וויסן צי מען ניצט זי נאר פאר פארשענערן דאס פנים, אדער מען וועט זי שפעטער ניצן אויף אנדערע צוועקן. Snapchat וועט פארמאגן די דאטא פון מיליאנען פנימער, און די דאטא קען מען ניצן כדי צו טריינירן אנדערע מאדעלן. דערפאר מוז דאס זיין קלאר דערקלערט פארן באניצער.

דער לעצטער פונקט איז זיכערהייט (security). Snapchat מוז שיצן די פנים-דאטא מיט שטארקע זיכערהייטס-מיטלען, כדי קיינער זאל נישט קענען צוקומען צו איר אן רשות. עס טאר נישט פאסירן אז א דריטע זייט זאל גנב'ענען די דאטא אדער אז איינער זאל זי מיסברויכן.

איבערחזר'ן די פינף פרינציפן (32:15:36)

דער קורס חזר'ט איבער די סטאנדארט-שיטות פון דער פריערדיגער זייט:

- דורכזיכטיגקייט: וועלכע מען קלייבט די דאטא, דארף מען זיין אפן מיטן באניצער.
- יושר: מען דארף ניצן די דאטא אויף א יושרדיגן אופן, און דער אלגאריטם הינטער דער אפליקאציע דארף ניצן די דאטא אויך אויף א יושרדיגן אופן.
- אחריות: עס דארף זיין א קאנטאקט-פערזאן וואס איז פארענטוואָרטליך פאר יעדע פראבלעם וואס די קונים ברענגען ארויף, און אויך וועלכע עס פאסירט א דאטא-בריטש (א ליק פון דאטא).
- פריוואטקייט: מען דארף קלייבן די דאטא אויף א גאנץ פריוואטן אופן, מיט דער הסכמה פונעם באניצער.
- זיכערהייט: די דאטא וואס מען האט געקליבן דארף מען היטן מיט פאסיגע זיכערהייטס-מיטלען, אז קיין באניצער אן רשות זאל נישט קענען צוקומען צו איר.

אזוי קומען ארויס די הויפט-פרינציפן פון דאטא-עטיק: דורכזיכטיגקייט, יושר, אחריות, פריוואטקייט און זיכערהייט.

פארוואס איז דאטא-עטיק וויכטיג? (32:16:37)

נאכדעם וועלכע ערקלערט פארוואס מען רעדט אזוי פיל וועגן די פרינציפן, די כללים און די תקנות פון דאטא-עטיק ביים ארבעטן מיט דאטא. עס זענען דא פיר סיבות:

- עס בויט אויף צוטרוי. וועלכע א באניצער פארבינדט זיך מיט אייך און טיילט מיט זיין דאטא אויף עפעס א פלאטפארמע, און איר האלט זיך צו די פרינציפן, שאפט דאס צוטרוי צווישן אייך און דעם קונה.
- עס שיצט דעם יחיד און די געזעלשאפט פון דאטא-בריטשעס. וועלכע מען קלייבט דאטא און מען האלט זי אין א דאטאבאזע, לייגט מען צו זיכערהייטס-שוץ, אז קיינער אן רשות זאל נישט צוקומען צו איר. אזוי פילט דער באניצער זיך זיכער אז זיין דאטא איז באשיצט, און ער באקומט בטחון.
- עס מוטיגט אנטוויקלען AI אויף א פארענטוואָרטליכן אופן. וועלכע מען קלייבט די דאטא אויף וואס מען טריינירט די AI-מאדעלן מיט יושר און דורכזיכטיגקייט, וועט דער מאדעל נישט האבן קיין bias און וועט נישט דיסקרימינירן קיין שום סארט מענטשן.

• עס העלפט שאפן א יושרדיגע און גלייכע דיגיטאלע צוקונפט, וואו יעדער באקומט א גלייכע געלעגנהייט ווייטער צו גיין.

דאטא-עטיק אינעם פראצעס פון דאטא-אנאליטיק (32:18:45)

דער ציל פון דאטא-אנאליטיק איז צו פארוואנדלען רויע דאטא אין נוצבארע אינפארמאציע. אבער אויף דעם וועג מוז מען קלייבן די דאטא, זי באארבעטן, זי אוועקלייגן ערגעץ וואו, און טייל מאל זי אויך טיילן מיט אנדערע. דערפאר איז דאטא-עטיק זייער וויכטיג אויך אין דעם פראצעס.

דער קורס חזר'ט איבער די דריי צילן פון דאטא-אנאליטיק:

• זי עפנט אויף אינפארמאציע און נייע געדאנקען. זי וווייזט ארויס באהאלטענע מוסטערן און טרענדס אין דער דאטא, און דאס גיט דעם מענטש די כח צו נעמען באשלוסן וואס זענען באזירט אויף אינפארמאציע.

• זי פארבעסערט פראצעסן און די ארבעט פון א סיסטעם. די אנאליז וווייזט אן וואו מען קען זיך פארבעסערן: וועלכע לעכער מען דארף אויספילן און וועלכע חסרונות מען דארף פארריכטן. מיט דעם קען מען פארבעסערן א סיסטעם וואס איז שוין דא.

• זי העלפט פאראויסזאגן די צוקונפט. וועו מען טרעפט מוסטערן, טרענדס אדער פארבינדונגען אין דער דאטא, קען מען ניצן די אינפארמאציע כדי צו טריינירן א מאדעל וואס זאגט פאראויס טרענדס און פירונגען לויט דער דאטא פון דער פארגאנגענהייט.

די עטישע פראבלעמען: bias, פריוואטקייט, דורכזיכטיגקייט (32:20:47)

וועו מען ברענגט אריין עטישע באטראכטונגען אין דעם פראצעס, קומען ארויף עטליכע פראבלעמען:

• bias און דיסקרימינאציע. וועו מען טריינירט אן אלגאריטם אדער א מאדעל אויף דאטא וואס איז נישט באלאנסירט, וועט דער מאדעל ארויסגעבן רעזולטאטן וואס זענען אויך נישט באלאנסירט, און דאס ברענגט אריין דיסקרימינאציע. אז מען האלט זיך נישט צו די פרינציפן פון דאטא-עטיק, קען מען בקיצור ענדיגן מיט א "biased" מאדעל, און אזא מאדעל ברענגט אריין דיסקרימינאציע.

• זארגן וועגן פריוואטקייט. וועו מען קלייבט און אנאליזירט דאטא אן פאראנטווארטליכקייט, קען מען פארלעצן דעם רעכט פון א מענטש צו פריוואטקייט. אין פאלן ווי איינטיילן קונים אין גרופעס (customer segmentation) אדער וועו Facebook רעקאמענדירט פריינט, מוז מען קלייבן פערזענליכע אינפארמאציע: דעמאגראפישע פרטים, דאס עלטער, אפילו פאליטישע מיינונגען, אינטערעסן און הביס. אז מען האנדלט נישט ריכטיג מיט אזעלכע אינפארמאציע, קען מען זי מיסברויכן. מען קען זיך ארויסגעבן פאר א צווייטן מענטש, דאס הייסט מען קען שאפן א דיגיטאלע פיגור וואס שפילט נאך א פאקטישן מענטש.

• א מאנגל אין דורכזיכטיגקייט. אז מען איז נישט קלאר מיט ווי מען ארבעט מיט דאטא, קען דאס אריינברענגען טעותים, שעדיגן דעם צוטרוי און שטערן די אחריות.

ווי אזוי מען האנדלט מיט bias (32:22:49)

דער קורס חזר'ט איבער אז אויב מען קלייבט די דאטא נישט מיט יושר, קומט זי מיט א bias. א מאדעל וואס מען טריינירט אויף אזא דאטא וועט אויך געבן פאראויסזאגונגען מיט א bias, און דאס ברענגט אריין דיסקרימינאציע. דערפאר שלאגט מען פאר דריי טריט:

• אויסמיידן דאטא און אלגאריטמען מיט bias. וועו מען קלייבט די דאטא, מוז מען זיך באפאסן מיטן bias אין דער דאטא און אינעם אלגאריטם, כדי צו פארמיידן דיסקרימינירנדע רעזולטאטן און ניצן די דאטא

פאראנטוואַרטליך. דאס איז באזונדערס וויכטיג ביי מאדעלן וואס מען בויט ספעציעל פאר א קהילה אדער פאר אפליקאציעס פאר דער געזעלשאפט.

- זיכער מאכן אז דער אלגאריטם איז פאראנטוואַרטליך. מען דארף שאפן מיטלען וואס זאלן האלטן דעם אלגאריטם פאראנטוואַרטליך, כדי די באשלוסן וואס זענען באזירט אויף דאטא זאלן זיין יושרדיג. דאס איז נאר מעגליך ווען מען האט געקליבן די דאטא מיט יושר, און יעדע אפטיילונג איז אויסגעקליבן געווארן צופעליג אדער באקומט א גלייכע וואג אינעם פראצעס פון בויען דעם מאדעל.
- מוטיגן פארשידנקייט אין דער דאטא-מאנשאפט. וועז א געמישטע מאנשאפט פון אנאליטיקער טרעפט אונ פארריכט די biases, איז דאס א זיכערונג פאר יושר. אויב מען זעט אז איין קאטעגאריע האט מער דאטא ווי די אנדערע, קען מען שאפן מאנשאפטן וואס זאלן צולייגן מער דאטא-פונקטן און באלאנסירן די דאטא.

ביישפיל: די געצילטע רעקלאמעס פון Facebook (32:24:20)

צום סוף ברענגט דער קורס א ביישפיל פון א פיטשער אין Facebook: געצילטע רעקלאמעס (targeted advertisement). א דיסקרימינירנדע פירונג באדייט דא אז Facebook ווייזט א געוויסע רעקלאמע נאר פאר א געוויסער גרופע מענטשן.

דער פאל: אין 2019 האט Facebook באקומען קריטיק ווייל זיין סיסטעם פון צילן רעקלאמעס האט דערלויבט דיסקרימינאציע לויט דעם מין און דעם עלטער ביי רעקלאמעס פאר דזשאבס. אן אדוואַרטייזער האט געקענט פילטערן און באשטימען אז זיין רעקלאמע זאל זיך ווייזן נאר פאר איין מין און פאר א געוויסער עלטער-גרופע. דאס האט באדייט אז די איבריגע מענטשן האבן נישט געזען די דזשאבס, זיי האבן נישט באקומען קיין געלעגנהייט זיך אנצומעלדן, און זיי האבן זיך געפילט דיסקרימינירט.

דער קורס באמערקט אז דאס איז בעצם א פיטשער פון דער אפליקאציע. וועז מען ניצט Facebook פאר רעקלאמעס, קען מען צולייגן פילטערס לויט דעם מין, לויט דעם עלטער אדער לויט א געוויסער דעמאגראפיע. אזוי ווערט די רעקלאמע געצילט פאר א באשטימטן עולם, און זי ווייזט זיך נאר פאר די באניצער וואס פאסן אריין אין די פילטערס. אבער דאס איז געווארן א דיסקרימינירנדע פירונג, ווייל אין 2019 האט מען טענה'ט אז דאס איז נישט ריכטיג און אז Facebook איז נישט יושרדיג וועז ער גיט צו אזעלכע רעקלאמעס.

עטישע פראבלעמען: פריוואטקייט און דורכזיכטיגקייט (32:26:53)

דער ערשטער פאל ענדיגט זיך: אנאנסן וואס דיסקרימינירן (32:26:53)

דער קורס ענדיגט דעם פריערדיגן ביישפיל פון "בייאס און דיסקרימינאציע". דארט האט א פלאטפארמע געלאזט אנאנסירער אויסקלייבן ווער עס זאל זען א דזשאב-אנאנס, און זיי האבן געקענט פילטערן לויט א געוויסער דעמאגראפישער גרופע. דער עטישער פראבלעם איז אז וועז מען ציילט אויף א באשטימטע דעמאגראפיע, קען מען אומגערעכט אויסשליסן מענטשן פון ארבעט-געלעגנהייטן. עס ווערט ערקלערט אז דאס איז גאר א געוועלכע זאך: ווער עס געהערט נישט צו יענער גרופע, באקומט נישט קיין גלייכע געלעגנהייט אויסגעקליבן צו ווערן, אדער אפילו זיך צו מעלדן אויף דער שטעלע.

מען רעכנט אויס די דריי עטישע פראבלעמען:

- וועז מען נוצט אויס די כלים פאר אנאנס-צילן (ad targeting), קען דאס אונטערגראבן די יושר אויפן ארבעט-מארק.

- פעלן פון דורכזיכטיגקייט ברענגט פראגעס פון פראנטווארטליכקייט און צוטרוי.
- דער באניצער, דער קונה, וועט נישט זיין צופרידן. ער קען פילן אז די פלאטפארמע האט אים דיסקרימינירט ווייל מען האט אים נישט געוויזן דעם אנאנס. דעמאלט פארלירט מען זיין צוטרוי, און מען קען אים אינגאנצן פארלירן אלס קונה.

די לעקציע דערפון: מען דארף רעגולער מאכן אן אודיט (באקוקונג) פונעם אלגאריטם כדי אראפצוברענגען די בייאסעס. מען דארף אויסגעפינען ווי אזוי מען קען עס פארבעסערן, און געבן פאררעכט צו אן עטישן אויסשטעל פון די אנאנס-כלים. אזעלכע פילטערן טארן נישט זיין צו שארף, און בעסער זאלן זיי בכלל נישט נוצן. וועו מען לאזט ארויס אן אנאנס, דארף ער גיין צו יעדן איינעם, און יעדער דארף האבן א גלייכע געלעגנהייט אים צו זען.

דער צווייטער פראבלעם: פריוואטקייט (32:28:25)

דער צווייטער עטישער פראבלעם איז פריוואטקייט. דאס האט מערסטנס צו טון מיט די פערזענליכע אינפארמאציע פון א באניצער. די בעסטע פראקטיק דא הייסט data minimization און purpose limitation, דאס הייסט: נעמען ווייניגער דאטא, און נוצן זי נאר פאר דעם באשטימטן ציל.

עס ווערט ערקלערט: וועו מען שטעלט איין דאטא-עטיק אין סיי וועלכער אפליקאציע, זאל מען שטענדיג פרובירן צו זאמלען די מינימאלע אינפארמאציע וועגן א יחיד. דאס הייסט "מינימירן די דאטא-זאמלונג". מען זאל אויך אפהאלטן נאר די מינימאלע אינפארמאציע. אינפארמאציע וואס ווערט שוין לאנג נישט געוצט, זאל מען נישט האלטן. דער גאנג זאל זיין: זאמלען, נוצן און אוועקווארפן. אזוי פארמינערט מען יעדע סכנה פון זיכערהייט אדער פון פריוואטקייט. מען זאמלט נאר דאס שטיקל אינפארמאציע וואס די אפליקאציע דארף טאקע, און נישט מער וועגן דעם באניצער.

א צווייטע פראקטיק איז צו האבן שטארקע cybersecurity מיטלען. נאר באפולמעכטיגטע באניצער זאלן קענען צוקומען צו דער דאטא, און קיין נישט-באפולמעכטיגטער זאל נישט קענען באקומען קאנטראל. דער קורס באמערקט אז דאס ליגט שוין אינדרויסן פונעם דאטא-זאמלונג פראצעס, און עס געהערט צום פלאן פון דער פירמע.

א דריטע פראקטיק זענען קלארע data breach protocols. דאס הייסט: אויב ס'קומט פאר א דורכברעך און די דאטא ווערט גע'גנב'עט, דארף מען קלאר וויסן ווי אזוי מען וועט עס באהאנדלען. וועו מען שטעלט ארויס אזעלכע פראטאקאלן אפן, העלפט דאס בויען צוטרוי און שוץ ביי די מענטשן בשעת א זיכערהייט-פאסירונג.

ביישפיל: דער דאטא-דורכברעך ביי T-Mobile (32:30:27)

אלס ביישפיל ברענגט דער קורס דעם דורכברעך ביי T-Mobile, א גרויסע מאביל-פירמע. אין יאנואר 2024 האבן האקערס באקומען נישט-באפולמעכטיגטן צוטריט צו איר API, און זיי האבן גע'גנב'עט די אינפארמאציע פון בערך 37 מיליאן קונים. עס ווערט אונטערגעשטראכן אז דאס איז א ריזיגע מאס דאטא.

דער קורס ערקלערט פארוואס עס איז אזוי ערנסט. וועו מען ווערט א קונה ביי א טעלעפאן-פירמע, גיט מען איבער א סך פערזענליכע פרטים: נאמען, פאמיליע-נאמען, אדרעס, און דערצו דעם טעלעפאן-נומער וואס איז פארבונדן מיטן נאמען. דאס אלעס הייסט PII, פערזענליך-אידענטיפיצירבארע אינפארמאציע. וועו עס כאפט די אינפארמאציע פון 37 מיליאן קונים, ווערט ממש א "קענ'ג" אויף דער דאטא-זייט, און ער קען זי נוצן פאר יעדן צוועק.

די אויסגעשטעלטע דאטא האט די קונים געשטעלט אין סכנה פון:

• identity theft, דאס הייסט אז איינער גנב'עט זיך דיין אידענטיטעט.

• phishing scams: דער קורס דערציילט אז אין אינדיע איז דאס זייער געווענליך. מענטשן רופן אן און לייגן פאר אלערליי אנגעבאטן, אדער מאל סטראשען זיי. דערנאך בעטן זיי מען זאל איבערגעבן די קאנטע-פרטים אדער דעם CVV נומער און דאס גלייכן.

• targeted marketing, דאס הייסט אנאנסן וואס צילן גראד אויף דיר.

וועו אזא וויכטיגע אינפארמאציע קומט אריין אין די הענט פון האקערס, קען מען זי נוצן פאר וועלכן צוועק מען וויל. דער פעלן פון ריכטיגע API זיכערהייט-מיטלען ברענגט אן עטישע שאלה ווי די פירמע באהאנדלט שפירעוודיגע דאטא. דער קונה פארלירט דעם צוטרוי, אוו ער וועט זיך שוין נישט פארלאזן אויף דער פירמע.

די לעקציעס פון דעם פאל:

• א פירמע מוז איינפירן שטארקע API זיכערהייט. א שוואכער אינטערפעיס לאזט אריין נישט-באפולמעכטיגטע, אוו מען פארלירט די קונים-אינפארמאציע. די זיכערהייט דארף אריינרעכענען access control, breach detection אוו א שטענדיגע אויפזיכט. וועו ס'קומט פאר א דורכברעך, זאל די סיסטעם זיך אויך קענען אפשטעלן.

• צייטיגע אנטדעקונג: דער דורכברעך איז שפעט אנטדעקט געווארן, אוו דאס ווייזט ווי דרינגלעך עס איז גיך צו כאפן אנאמאליעס, דאס הייסט אומגעווענליכע ערשיינונגען.

• דורכזיכטיגקייט: די פירמע מוז קלאר מיטטיילן די קונים וועגן דעם דורכברעך אוו וועגן די שריט וואס זי נעמט עס צו פארריכטן.

דער קורס באמערקט אז דאס איז גאנץ טיפארעטיש. מען נעמט אבער לעקציעס פון אזעלכע פאסירונגען, אוו מען פרובירט באהאנדלען די פראבלעמען. דערנאך פרעגט מען די צוהערער צי זיי קענען נאך פאלן פון דאטא-דורכברעכן, וואס איז דער עטישער פראבלעם דערביי, אוו וועלכע לעקציעס האט מען דערפון געלערנט.

דער דריטער פראבלעם: פעלן פון דורכזיכטיגקייט (32:34:04)

נאך בייאס אוו דיסקרימינאציע, אוו נאך פריוואטקייט, קומט דער דריטער פראבלעם: פעלן פון דורכזיכטיגקייט. דורכזיכטיגקייט מיינט צו געבן קלארע אינפארמאציע וועגן דריי זאכן: ווי אזוי מען זאמלט די דאטא, ווי אזוי מען אנאליזירט זי, אוו ווי אזוי מען נוצט זי. דאדורך קען מען אויך פארשטיין אוו דערקענען די פארשידענע בייאסעס. מען דארף זאגן דעם באניצער פיר זאכן: וואספארא אינפארמאציע מען זאמלט, ווי אזוי מען וועט זי אנאליזירן, פאר וואס מען נוצט זי, אוו וואו מען האלט זי אן. אזוי ווערט די סיסטעם דורכזיכטיג. וועו דורכזיכטיגקייט פעלט, ברענגט דאס ריזיקעס אוו פראבלעמען.

דריי טיילן פון דורכזיכטיגקייט ווערן ארויסגעברענגט:

• Algorithm transparency: א קלארע ערקלערונג ווי מען זאמלט, אנאליזירט אוו נוצט די דאטא. דאס העלפט פארשטיין אוו אידענטיפיצירן בייאסעס.

• מיטטיילן די ריזיקעס אוו באגרעניצונגען: מען זאל אפן אנערקענען די באגרעניצונגען אוו ריזיקעס פון דער דאטא-אנאליז, למשל בייאסעס. דאס פארשטארקט צוטרוי, אוו עס העלפט מענטשן נעמען אינפארמירטע באשלוסן.

• Data control פאר דעם באניצער: מען דארף זיכער מאכן אז דער באניצער האט די רעכט צוצוקומען צו זיין דאטא, זי צו פאררעכטן און זי אויסצומעקן. דאס היט די פריוואטקייט.

דער קורס פאסט צונויף: דער אלגאריטם דארף זיין גאר דורכזיכטיג ווי אזוי די דאטא איז געזאמלט געווארן, און צי ס'איז דא א בייאס אין איר. די באגרעניצונגען און ריזיקעס דארף מען אויך קלאר מיטטיילן. און צום סוף דארף מען געבן דעם באניצער קאנטראל: אויב ער וויל נישט, זאל ער קענען אויסמעקן, טוישן אדער פאררעכטן זיין דאטא. עס איז נישט אזוי אז ווען די דאטא איז אמאל געזאמלט, האט דער באניצער שוין קיין רעכט און קיין קאנטראל דערויף.

ביישפיל: Google Chrome און דער incognito mode (32:36:08)

א ביישפיל פון פעלן פון דורכזיכטיגקייט איז דער incognito mode פון Google Chrome. דער קורס פרעגט פארוואס מען נוצט אים, און ווי ער איז אנדערש פון געווענליכן Chrome. די צוהערער ענטפערן: מען האלט נישט אפ קיין cookies, קיין היסטאריע און קיין פאסווערטער. דער געדאנק איז אז ווען מען זוכט אדער בלעטערט אויף דער אינטערנעץ, בלייבט מען אנאנים. קיינער ווייסט נישט ווער ס'קוקט די אינפארמאציע, וויל מען שפירט נישט נאך, מען האלט נישט אפ קיין cookies אויפן היגן קאמפיוטער, קיין פאסווערטער און קיין רעקארד. אזוי פרובירט מען אפהיטן די פריוואטקייט. אבער מען שטעלט די פראגע: וואס אויב מען דערוויסט זיך אז Google רעקארדירט די אינפארמאציע אפילו בשעת מען ארבעט אין incognito mode?

מיליאנען Chrome באניצער האבן זיך פארלאזט אויף incognito mode פאר פריוואטן בלעטערן. זיי האבן געריכט אז קיין פריוואטע אינפארמאציע ווערט נישט געטיילט: נישט דער ארט וואו מען געפינט זיך, נישט די פאסווערטער, און נישט די פערזענליכע היסטאריע. אבער עס איז אריינגעגעבן געווארן א געריכט-קלאגע אז Google שפירט ווייטער נאך די באניצער-אקטיוויטעט אפילו אין דעם מאָוד, און דאס האט ארויסגערופן א מחלוקת איבער פריוואטקייט. די טענה איז געווען אז Google שפירט נאך וואס דער באניצער קוקט און פון וואו, כאטש ער האט זיך פארלאזט אז ער איז פריוואט.

די עטישע פראבלעמען:

• Deception (פארפירונג): Google האט פארפירט די באניצער ווען וואס incognito mode קען באמת. זי האט זיי נישט גוט באוואוסט געמאכט אז אויך אין דעם מאָוד ווערט געזאמלט געוויסע אינפארמאציע, און אז מען שפירט נאך געוויסע אקטיוויטעטן.

• Data privacy: די באניצער האבן ערווארט אז זייער אינפארמאציע איז אינגאנצן פריוואט, אבער דאס איז געווען פארלעצט, ווייל Google האט נאכגעשפירט זייערע אקטיוויטעטן.

• Transparency: Google'ס דורכזיכטיגקייט ווען אירע דאטא-זאמלונג פראקטיקן איז נישט געווען גוט גענוג.

די לעקציעס:

• דער צוטרוי פון באניצער הענגט אפ אין דורכזיכטיגע קאמוניקאציע, און אין דעם אז מען שטייט צו די ערווארטונגען. נאר ווען מען הייסט צו א סערוויס און מען טוט עס טאקע, קען מען ערווערבן דעם צוטרוי.

• ווען מען פארשטייט די באגרעניצונגען פון incognito mode און מען זוכט בעסערע אפציעס פאר פריוואטקייט, קען מען נעמען מער אינפארמירטע באשלוסן אויף דער אינטערנעץ. מען דארף דעריבער די באניצער באוואוסט מאכן, כדי זיי זאלן זיין בעסער אינפארמירט און זיך פירן לויט דעם.

עטישע דאטא-אנאליטיק: פרינציפן, שוועריגקייטן און געזעצן (32:40:16)

עטישע אנאליטיק פאר א בעסערער צוקונפט (32:40:16)

דער קורס ענדיגט די באהאנדלונג פון די דריי הויפט-געביטן פון דאטא-עטיק אין דאטא-אנאליטיק. וועו א באניצער גייט, למשל, אריין אין incognito mode און Google זאמלט פונדעסטוועגן זיין דאטא, איז דאס אפשר נאך אין ארדענונג. אבער די דאטא קען ווערן גענוצט פאר א צוועק וואס איז נישט ערלויבט אדער נישט טראנספארענט. די דריי געביטן זענען: ערשטנס, פארנייגונג (bias) און דיסקרימינאציע; צווייטנס, פריוואטקייט (privacy); דריטנס, טראנספארענץ.

דערנאך ווערט ערקלערט וואס מיינט "עטישע אנאליטיק", דאס הייסט עטישע דאטא-פירונגען אין דאטא-אנאליטיק פאר א בעסערער צוקונפט. עס זענען דא פינף פרינציפן פון דאטא-עטיק: אחריות (accountability), טראנספארענץ, פריוואטקייט, זיכערהייט (security) און יושר (fairness). וועו מען פירט זיי איין אין דעם פראצעס פון דאטא-אנאליטיק, ברענגט דאס פאלגנדע תועלת:

- מען בויט אויף צוטרוי אין דער טעכנאלאגיע. עטישע דאטא-פירונגען שאפן צוטרוי.
- מען שטיצט א פאראנטווארטליכע טעכנאלאגישע אנטוויקלונג, און מען פארשטארקט אריינגערעכנטקייט (inclusivity) און סאציאלע גערעכטיגקייט. וועו מען נעמט ארויס די פארנייגונגען פון דעם דאטא-פראצעס, ווערט עס יושרדיג, און דאס איז גוט פאר דער געזעלשאפט.
- מען פארעמט א פאראנטווארטליכע AI-אנטוויקלונג. וועו מען ניצט יושרדיגע דאטא ביים טריינירן א מאדעל, וועט דער מאדעל געבן יושרדיגע אורטיילן. וועו די דאטא אנטהאלט פארנייגונג, ווערט דער מאדעל פארנייגט צו א געוויסער קאטעגאריע, און דאס איז נישט גוט פאר דער געזעלשאפט. דאס איז בעיקר וויכטיג ביים אנטוויקלען מאדעלן.
- מען היט די רעכטן און פרייהייטן פון יעדן מענטש. פריוואטקייט-כללים און געגוג זיכערהייט שיצן די פערזענליכע אינפארמאציע פון א מענטש און היטן אפ זיינע רעכטן איבער דער אינפארמאציע. אויב דער באניצער איז גרייט צו טיילן זיין פערזענליכע אינפארמאציע, איז דאס אין ארדענונג. אויב ער וויל נישט, איז דאס אויך אין ארדענונג.

א קורצע איבערחזרונג מיט א פראגע (32:42:19)

דער קורס שטעלט א פראגע ווי א איבערחזרונג: וועלכער פרינציפ באטאנט מען זאל זאמלען נאר די נויטיגע אינפארמאציע פאר א באשטימטן צוועק, כדי צו פארמינערן סענסיטיוויע דאטא און איר אפהאלטן? די ברירות זענען:

- (A) דער טראנספארענץ-פרינציפ
- (B) דער פרינציפ פון דאטא-מינימיזאציע (data minimization)
- (C) דער סייבער-זיכערהייט פרינציפ
- (D) דער פרינציפ פון ערקלערבארקייט (explainability)

דער ריכטיגער ענטפער איז B. מען טאר נישט זאמלען אינפארמאציע וואס די אפליקאציע אדער דער פאל וואס מען לייזט דארף נישט. אויף דעם אופן זאמלט מען און האלט מען אפ וויניג ווי מעגליך. עס איז נישט גוט צו האלטן א גרויסע מאס סענסיטיוויע דאטא, ווייל אויב עס קומט פאר א דאטא-בראך (data breach), פארלירט מען דעם צוטרוי פון די קונים.

עטישע שווערigקייטן: דער פאל פון א לאקאציע-אפליקאציע (32:43:20)

דער קורס חזרט איבער די הויפט-פרינציפן: טראנספארענץ, יושר, אחריות און פריוואטקייט. ער חזרט אויך איבער די דריי הויפט-ענינים ביים איינפירן דאטא-עטיק אין אנאליטיק: פארנייגונג און דיסקרימינאציע, פריוואטקייט-זארגן, און א מאנגל אין טראנספארענץ. יעצט קוקט מען אויף די עטישע שווערigקייטן וואס קומען ארויס אין דער פראקטיק.

דער ערשטער פאל: א מאביל-אפליקאציע פאלגט נאך דעם באניצער'ס לאקאציע. מיט די לאקאציע-דאטא גיט זי פערזענליכע רעקאמענדאציעס און רעקלאמעס. די אפליקאציע זאמלט אלזא דעמאגראפישע אינפארמאציע וועגן דעם באניצער און ניצט זי פאר פערזענליכע פארשלאגן. די באניצער ווייסן אבער נישט ווי ווייט די דאטא-זאמלונג גייט, און זיי האבן נאר באגרעניצטע קאנטראל איבער ווי מען ניצט זייער דאטא. די אפליקאציע גיט נישט דעם באניצער דעם צוטריט אדער די קאנטראל צו ענדערן די דאטא.

די פראגע איז: וועלכע עטישע זארגן זענען דא דא, און וואס לערנט מען דערפון? דער קורס ענטפערט:

- פריוואטקייט: אז מען האט נישט קיין פולע קאנטראל ווי מען ניצט די פערזענליכע דאטא, איז דאס א פריוואטקייט-פראבלעם.
- מאנגל אין טראנספארענץ: מען ווייסט נישט ווי פיל דאטא מען זאמלט און ווי מען ניצט זי.
- זיכערהייט: מען שטעלט אויף א סארט אויפפאסונג-סיסטעם (surveillance) וואס פאלגט נאך די פערזענליכע לאקאציע.

די לעקציעס, דאס הייסט וואס מען דארף איינפירן אין דער ארגאניזאציע און אין דעם פראצעס פון דער אפליקאציע:

- א קלארע טראנספארענץ-פאליסי וואס זאגט "מיר זאמלען אזויפיל דאטא פון דיר, און די דאטא וועט ווערן גענוצט פאר אט דעם צוועק".
- א ריכטיגע פריוואטקייט-פאליסי.
- א ריכטיגער זיכערהייט-מעכאניזם צו שיצן די דאטא.
- ריכטיגע קאנטראל פאר דעם באניצער איבער דער דאטא, אז ער זאל קענען ענדערן זיין אינפארמאציע און סעטינגס ווען ער וויל.

ווי עס שטייט אין דער לעקציע, זענען די עטישע זארגן: א פארלעצונג פון פריוואטקייט-רעכטן, א מאנגל אין ערקלערטער הסכמה (informed consent), און די מעגליכקייט פון געצילטער אויפפאסונג און מאניפולאציע. די לעקציעס: מען דארף האבן קלארע און טראנספארענטע דאטא-פריוואטקייט פאליסיס, דער באניצער דארף האבן קאנטראל איבער דער דאטא-זאמלונג, און מען דארף האבן שטארקע זיכערהייט-מיטלען צו שיצן די באניצער-דאטא.

דער פאל פון א AI-געצייג פאר אויפנעמען ארבעטער (32:47:26)

דער צווייטער פאל: א פירמע ניצט א AI-געצייג צו דורכזיכן קורות-החיים (resumes) פון קאנדידאטן. דער אלגאריטם, אן אז מען ווייסט דערפון, האלט אן א פארנייגונג לויטן מין. ער באפארצוגט מענער-קאנדידאטן און פארמינערט שטארק ווי זעבאר די אפליקאציעס פון פרויען זענען. די עטישע זארגן זענען יושר, אדער פארנייגונג און דיסקרימינאציע, און אויך א מאנגל אין טראנספארענץ.

ווי עס שטייט אין דער לעקציע, זענען די זארגן: דיסקרימינאציע לויט געשיצטע אייגנשאפטן (protected characteristics), א מאנגל אין טראנספארענץ ביים אנטוויקלען דעם אלגאריטם, און די מעגליכקייט פון אומיושרדיגע אויפנאם-פירונגען. אויב דער מאדעל האט א פארנייגונג צו מענער, ווערן פרויען-קאנדידאטן אויטאמאטיש אפגעווארפן.

די לעקציעס:

- מען דארף מער יושר אין דער דאטא-זאמלונג און אין דעם באנוץ מיט דער דאטא, כדי אראפצונעמען פארנייגונג און דיסקרימינאציע.
- מען דארף מאכן יושר-קאנטראלן (fairness audits) אויף אלגאריטמען.
- דער אויפנאם-פראצעס דארף זיין טראנספארענט.
- מען דארף זיין באוואוסטזיניג פון מעגליכע פארנייגונגען אין דער דאטא, און איינפירן פירונגען וואס דערקענען אזעלכע פארנייגונגען, אפילו ווען דער מאדעל מאכט אזעלכע פאראויסזאגונגען, כדי צו באקומען אן אמת'ן אורטייל.

א קאנטראל-פראגע וועגן באניצער-דאטא (32:49:30)

דער קורס פרעגט: וואס איז א קריטישע עטישע באטראכטונג ביים זאמלען און ניצן באניצער-דאטא? די ברירות זענען:

- (A) צו זאמלען וואס מער דאטא אן הסכמה פון באניצער
- (B) צו פארזיכערן טראנספארענץ און באקומען ערקלערטע הסכמה פון באניצער
- (C) צו פארקויפן באניצער-דאטא צו א דריטער פארטיי אן אנצוזאגן
- (D) צו איגנאריר זיכערהייט-מיטלען כדי צו פארשנעלערן דאס פראצעסירן

דער ריכטיגער ענטפער איז B: מען דארף פארזיכערן טראנספארענץ, און די אינפארמאציע זאל מען באקומען מיט א ריכטיגער הסכמה פון באניצער.

לעגאלע און רעגולאטארישע ראמען (32:50:30)

דער נעקסטער טעמע זענען לעגאלע און רעגולאטארישע ראמען. די זענען זייער וויכטיג פאר יעדער ארגאניזאציע הייגע צייטן, וועו מען זאמלט אזויפיל דאטא. ווי עס ווערט ערקלערט, איז די מאס דאטא וואס מען זאמלט היינט אין איין סעקונדע גלייך צו דער גאנצער דאטא וואס מען האט געזאמלט אין דעם גאנצן יאר 1990. ביי אזא מאס און אזא גיכקייט מוז מען איינפירן דאטא-עטיק פירונגען. אויב נישט, פארלירט מען דעם צוטרוי פון די באניצער, און דער סיסטעם ווערט פארנייגט, אומיושרדיג און נישט טראנספארענט. דערפאר דארף מען האבן א רעגולאטארישן און לעגאלן ראם, כדי צו קענען זיך באהאנדלען מיט א ארגאניזאציע וואס פאלגט נישט די דאטא-עטיק פירונגען.

די דעפיניציע: דאס זענען א סכום געזעצן און כללים, און זייער ציל איז צו רעגולירן עטישע דאטא-פירונגען. דאס הייסט: ווי מען ניצט די דאטא, ווי מען האלט זי אפ, ווי מען פראצעסירט זי, און וועלכע כללים עס גילטן ביים טיילן די דאטא מיט א דריטער פארטיי. די ראמען פארזיכערן אז די עטישע סטאנדארטן ווערן איינגעפירט אין יעדער דאטא-ארגאניזאציע ביים זאמלען, אפהאלטן, ניצן אדער טיילן דאטא. אזוי האט מען סיי די הויפט-פרינציפן פון דאטא-עטיק, סיי א רעגולאטארישן ראם וואס זעט צו אז מען זאל זיי דורכפירן.

די הויפט-חלקים פון די ראמען:

- פריוואטקייט-געזעצן: די באציען זיך מערסטנס צו שיצן די דאטא פון יחידים. דער באקאנטסטער געזעץ איז GDPR (General Data Protection Regulation) אין אייראפע. נאך א געזעץ איז CCPA, דער קאליפארניער קונים-פריוואטקייט געזעץ. די צוויי געזעצן זענען בעיקר באשטימט צו רעגולירן די פריוואטקייט פון א יחיד'ס אינפארמאציע. דער יחיד קען זיין א קונה, א קליענט פון א סערוויס-פירמע, א באנק-קונה, א טעלעקאם-קונה אדער סיי וועלכער קונה.
- דאטא-שוץ געזעצן: די באשרייבן די פארשידענע שוץ-מיטלען וואס דאטא-ארגאניזאציעס מוזן איינפירן ווען זיי האלטן אפ דאטא. זיי גילטן נישט נאר ביים אפהאלטן, נאר אויך ביים טיילן די דאטא.
- טראנספארענץ און הסכמה: דאס ווערט אנגערופן ווי א זייער וויכטיגער ענין, און דער קורס הייבט אן דאס צו ערקלערן ביים סוף פון דעם חלק.

רעכטלעכע ראמען, אויסגאבעס און בעסטע פירונגען פאר דאטא-עטיק (32:55:10)

די הויפט-פונקטן פון לעגאלע און רעגולאטארישע ראמען (32:55:10)

דער קורס האלט אינמיטן ערקלערן די הויפט-פונקטן פון די געזעצן און די רעגולאטארישע ראמען וואס באהערשן ווי אזוי מען זאמלט און מען באנוצט זיך מיט דאטא. ווען א דאטא-ארגאניזאציע זאמלט דאטא פון מענטשן, באטאנען די געזעצן אז מען דארף זיך פירן מיט טראנספארענץ. ווען מען זאמלט דאטא, דארף מען זאגן גאנץ קלאר וועלכע שטיקלעך אינפארמאציע מען זאמלט און פארוואס מען זאמלט זיי.

דער ערשטער פונקט וואס ווערט דא באהאנדלט איז ווי אזוי מען באקומט די דאטא און ווי אזוי מען ניצט זי. עס דארף זיין קלאר באשטימט אז ווען מען האט צו טון מיט א קונה אדער א קליענט, ווערן געוויסע שטיקלעך אינפארמאציע געזאמלט ביים געשעפט, און אט דאס וועט זיין דער צוועק פון יענע אינפארמאציע. דאס הייסט טראנספארענץ און הסכמה (consent).

דערנאך קומט דער פונקט פון אייגנטום אויף דאטא און רעכטן. דא ווערן באשטימט די רעכטן פון דעם יחיד איבער זיין אייגענע דאטא:

- דאס רעכט צו באקומען צוטריט צו דער דאטא;
- דאס רעכט צו פארריכטן אדער דערהייַנטיקן די דאטא;
- דאס רעכט אויסצומעקן די דאטא.

עס ווערט ערקלערט: אויב א מענטש וויל נישט מער טיילן זיין אינפארמאציע, אפילו ווען ער האט זי פריער שוין געגעבן, האט ער די קאנטראל צו פארלאנגען אז מען זאל אויסמעקן יענע אינפארמאציע. דאס איז "דאטא-אייגנטום און רעכטן".

דער נעקסטער פונקט איז פאראנטוואָרטליכקייט און צופאסונג צום געזעץ (accountability and compliance). דאס איז א לעגאלער ראם וואס מעסט צי די ארגאניזאציע איז פאראנטוואָרטלעך און צי זי האלט זיך צו די כללים, ד.ה. די פאראנטוואָרטליכקייט וואס מען דארף האבן כדי איינצופירן אלע דאטא-פירונגען און פראצעדורן אינעם ארגאניזאציע.

דערנאך ווערט באהאנדלט די איבערפירונג פון דאטא איבער גרענעצן (cross-border data transfer), וואס באציט זיך צום טיילן דאטא. ווען מען גיט איבער דאטא פאר א דריטן צד, דארף מען וויסן וועלכע כללים און

תקנות עס זענען פאראן פארן טיילן, און אויף וועלכן אופן מען טיילט: אין א פארשליסלטן (encrypted) פארמאט אדער אין אן אפענעם פארמאט. עס דארף זיין אן עטישער און א פולער רעגולאטארישער ראם וואס באשטימט דעם גאנצן פראצעס.

צום סוף קומען די עטישע באטראכטונגען, וואס גייען ווייטער פון די לעגאלע פאדערונגען. דער קורס באטאנט אז דאס געזעץ קען נישט צודעקן יעדע זאך. מענטשן מוזן אויך זיין מאראליש און עטיש ווען זיי ניצן די דאטא, און זיי זאלן זיך תמיד באנוצן מיט דאטא מיט פאראנטוואַרטליכקייט. די עטישע באטראכטונגען רעדן פון יושר (fairness), פון נישט אפשיידן (no discrimination) און פון פארמיידן נייגונגען (bias).

אויפן צוזאמענפאסן ווערט געזאגט אז די הויפט-פונקטן פון יעדן לעגאלן אדער רעגולאטארישן ראם זענען זיבן:

- געזעצן פון פריוואטקייט (privacy law);
- געזעצן פון דאטא-שוץ (data protection law);
- דער ראם פון טראנספארענץ און הסכמה;
- געזעצן פון דאטא-אייגנטום;
- געזעצן פון פאראנטוואַרטליכקייט און צופאסונג;
- געזעצן פון טיילן דאטא איבער גרענעצן;
- עטישע באטראכטונגען לגבי יושר, נייגונגען און אפשיידן.

וועלטלעכע ראמען: GDPR, CCPA, PIPL, HIPAA (32:58:13)

לויט די אויבנדערמאנטע פונקטן זענען פאראן עטלעכע וועלטלעכע ראמען וואס העלפן באהערשן די דאזיקע פירונגען, און אלע דאטא-ארגאניזאציעס דארפן זיי אננעמען. דער קורס רעכנט אויס די באקאנטסטע:

- GDPR, General Data Protection Regulation. דאס איז אויסגעארבעט געווארן פון דער אייראפעאישער סיסטעם, בעיקר כדי צו באשיצן די פריוואטקייט פון דעם יחידס דאטא און זיינע רעכטן. ווען עס גייט אין דער פריוואטער אדער פערזענלעכער אינפארמאציע פון קונים און באניצער, העלפן די דאזיקע געזעצן זיי צו באשיצן.
- CCPA, California Consumer Privacy Act. דאס איז אן אמעריקאנער געזעץ וואס גיט די איינוואוינער פון קאליפארניע קאנטראל איבער זייער פערזענלעכער דאטא. "קאנטראל" מיינט דא אז יעדער יחיד קען פריי טיילן זיין אינפארמאציע, זי דערהיינטיקן, אדער זי אפילו אויסמעקן ווען עס איז נויטיק אדער ווען ער וויל.
- PIPL, Personal Information Protection Law פון כינע. דאס איז א פולשטענדיקער ראם פאר דאטא-פריוואטקייט וואס באהאנדלט אויך די PII, ד.ה. אינפארמאציע וואס קען אידענטיפיצירן א מענטש. דער קורס באמערקט אז דאטא-פריוואטקייט איז א גרויסע דאגה אין דער דיגיטאלער וועלט, און קיינער איז נישט גרייט צו טיילן זיין פערזענלעכע אינפארמאציע אן זיין ריכטיקער הסכמה.
- תקנות פאר ספעציפישע סעקטארן, ווי געזונטהייט און פינאנצן. אלס ביישפיל ווערט דערמאנט HIPAA, א ראם וואס באהערשט ווי אזוי געזונטהייט-דאטא אדער פאציענטן-דאטא ווערט געטיילט צווישן פארשידענע אינסטיטוציעס.

די דאזיקע וועלטלעכע ראמען ווערן געניצט איבער דער גאנצער וועלט כדי צו באשיצן די דאטא-עטיק, די פריוואטקייט-רעכטן און די זיכערהייט-מיטלען.

שווערקייטן ביים איינפירן דאטא-תקנות (33:00:17)

דער קורס ערקלערט אז כאטש מען האט באשטימט די ראמען און די געזעצן, זענען דא שווערקייטן ווען מען פירט איין די כללים און תקנות אין א דאטא-ארגאניזאציע:

- א רעגולאטארישע לאנדשאפט וואס ענדערט זיך כסדר. עס ווערט געבראכט א משל: אפילו ווען עס זענען פאראן געזעצן, קען מען נישט אינגאנצן אפשטעלן פארברעכן. אזוי אויך, א לעגאלער ראם מיינט נישט אז די דאטא וועט זיין זיכער אויף אייביק. האקערס און אטאקירער וועלן תמיד געפינען א וועג אויסצומיידן די סיסטעם: א שוואכקייט (vulnerability) אדער א הינטערטיר (backdoor) דורך וועלכער מען קען צוקומען צו דער דאטא. דערפאר מוז מען פון צייט צו צייט פארשטארקן די כללים, ד.ה. די לאנדשאפט מוז זיך אנטוויקלען. ווען זי ענדערט זיך, מוז מען טוישן די גאנצע פאליסי ווי מען האלט די דאטא, און אויך די דאטא-סכעמעס. דאס איז א שווערקייט ספעציעל פאר קליינע ארגאניזאציעס, ווייל ביי יעדער רעגולאטארישער ענדערונג מוז מען טוישן די גאנצע דאטאבאזע-סכעמע.
- די שטענדיקע סכנה פון דאטא-ברעכן (data breaches). האקערס זוכן תמיד נייע וועגן אריינצוברעכן אין די סיסטעמען און צו פארלעצן די פריוואטקייט.
- באלאנסירן צווישן עטישע דאטא-פירונגען און חידושים (innovation). דאס ווערט דערקלערט מיט א ביישפיל, ווי ווייטער.

דער ביישפיל פון רעקאמענדאציע-סיסטעמען (33:02:20)

א רעקאמענדאציע-סיסטעם איז א וועג צו רעקאמענדירן א שטיקל אינפארמאציע, א פראדוקט אדער אנדערע זאכן וואס פאסן צום באניצערס חשק און אינטערעס. דער ביישפיל: א באניצער האט ליב פילמען פון דעם סארט "crime thriller". דאן זאל Netflix אים רעקאמענדירן נאר אזעלכע פילמען, און אזוי קען Netflix ערווארטן אז דער באניצער וועט מער פארנומען זיין מיט דער פלאטפארמע.

רעקאמענדאציע-סיסטעמען זענען געווארן אן אינטעגראלער טייל פון יעדער סערוויס-פלאטפארמע, צי Netflix צי YouTube. דער קורס ברענגט אז ביי YouTube, ווען מען הייבט אן קוקן א געוויסן סארט ווידיאס, באקומט מען רעקאמענדאציעס פון גאנץ ענלעכע ווידיאס; און ווען מען הערט א געוויסן סארט מוזיק, באקומט מען רעקאמענדאציעס פון ענלעכע זשאנערס. אויך פלאטפארמעס ווי MakeMyTrip און TripAdvisor, ד.ה. יעדע סערוויס-באזירטע פלאטפארמע, ניצן היינט א רעקאמענדאציע-מאשין. דער ציל איז צו געבן רעקאמענדאציעס פארן קונה, אזוי אז מענטשן זאלן מער צייט פארברענגען אויף דער פלאטפארמע, און אזוי פארדינט זי מער.

יעדער באניצער האט אן אנדער טעם און אנדערע בעפארצוגונגען, דעריבער וויל מען פערזענלעכע רעקאמענדאציעס (personalized recommendations). די פלאטפארמעס פרובירן צו פארשטיין דעם באניצערס טעם, און לויט דעם מאכן זיי רעקאמענדאציעס, אזוי אז דער באניצער פארברענגט מער צייט ביי דער פלאטפארמע.

אבער דא ליגט די שווערקייט: פערזענלעכע רעקאמענדאציעס זענען נאר מעגלעך ווען די פלאטפארמע ווייסט מער ווען דעם באניצער. כדי צו וויסן מער, מוז מען זאמלען זיין פערזענלעכע אינפארמאציע: זיין מין, זיין עלטער-גרופע, זיינע הביס, וואס ער האט ליב און וואס נישט. ווען מען הייבט אן זאמלען אזעלכע אינפארמאציע, מוז מען זיין זייער זייער אפגעהיט ווי מען האנדלט מיט איר. אויב מען זאמלט זי אבער נישט, קען מען פארבלייבן הינטן מיט די פערזענלעכע רעקאמענדאציעס, ד.ה. מען בלייבט הינטן אין חידושים. די

עטישע דאטא-פירונגען זאגן אז מען זאל זאמלען וואס וויניקער אינפארמאציע, און וועט מען זאמלט וויניק, קען מען פארלירן אין חידושים. דערפאר מוז מען שטעלן א באלאנס צווישן די צוויי.

- די לעצטע שווערקייט: באגרעניצטע מיטלען ביי קלענערע ארגאניזאציעס. א קליינע ארגאניזאציע, למשל א סטארטאפ, וואס וויל איינפירן אלע עטישע דאטא-פירונגען, מוז קויפן געוויסע לייסענסעס און מיטלען, און די דאזיקע מיטלען זענען ביי איר נישט פאראן. דעריבער קענען קליינע ארגאניזאציעס פארלעצן עטלעכע עטישע דאטא-פירונגען. דאס זענען די שווערקייטן, און מען מוז זיך מיט זיי אנקומען דערגעמעס.

בעסטע פירונגען פאר עטישער דאטא-פארוואלטונג (33:06:08)

דער קורס גייט איבער צו די בעסטע פירונגען (best practices) וועט מען פירט איין א סיסטעם פון דאטא-פארוואלטונג (data governance):

- פירן א דאטא-עטיק אפשאצונג, און דאס פון צייט צו צייט. מען דארף רעגולער אפשאצן די פירונגען דורך אן עטישער ברילן. דער ביישפיל: א דאטא-ארגאניזאציע "ABC", וואס זאמלט און ניצט דאטא, מוז פון צייט צו צייט דורכגיין א געוויע אונטערזוכונג. זי מוז דורכגיין אלע סטאנדארטע עטישע דאטא-פירונגען און באווייזן אז זיי ווערן גוט אפגעהיטן ביי דער ארגאניזאציע.
- איינפירן דעם פרינציפ פון מינימום דאטא (data minimization). דאס איז זייער וויכטיק, ווייל וועט מען האלט וואס וויניקער דאטא, איז די שאנס פון א דאטא-ברעך אדער פון פארלירן דאטא קלענער. דער סטאנדארט זאגט: זאמלט און האלט נאר די נויטיקע דאטא, און האלט נישט קיין איבעריקע דאטא.
- באקומען א באוואוסטזיניקע הסכמה (informed consent). אויב מען זאמלט פערזענלעכע אינפארמאציע, מוז מען באקומען די הסכמה פונעם באניצער, און מען דארף קלאר מיטטיילן פארוואס מען זאמלט די דאטא און ווי אזוי מען וועט זי ניצן.
- דערמעגלעכן צוטריט און פארעכטונג פון דאטא. דאס גיט דעם באניצער קאנטראל, אז ער זאל קענען צוקומען צו זיין פערזענלעכער דאטא אדער זי דערהיינטיקן וועט עס איז נויטיק.
- בויען א קולטור פון דאטא-עטיק. וועט מען לערנט אויס די אנגעשטעלטע, דארף מען אריינלייגן אין זיי אז מען מוז זיך האלטן צו אלע דאטא-פירונגען, און אז דאס איז א מאראלישער חוב. מען דארף זיין עטיש פון אינעווייניק, נישט נאר ווייל א רעגולאטארישע אדער לעגאלע אפטיילונג צווינגט עס. מען זאל עס אננעמען פון גאנצן הארצן, און מען טאר נישט פאלש האנדלען מיט קיין אינפארמאציע און נישט מיסברויכן פערזענלעכע אינפארמאציע אויף קיין שום אופן.

דאס זענען די בעסטע פירונגען פאר א ראם פון עטישער דאטא-פארוואלטונג, און דערמיט ענדיקט זיך דער דאזיקער טייל.

דאטא-עטיק: געזעצן, קעיס-סטודיס און שוץ פון פערזענלעכע דאטא (33:08:56)

א פראגע וועגן GDPR און CCPA (33:08:56)

דער קורס הייבט אן מיט א פראגע צו די צוהערער: אויף וואס קאנצענטרירן זיך הויפטזעכלעך רעגולאציעס ווי GDPR און ?CCPA עס ווערן געגעבן פיר מעגלעכקייטן:

- א. צו זיכער מאכן אז סענסיטיוע אינפארמאציע ווערט ענקריפט (פארשליסלט).
- ב. צו דעפינירן די רעכטן פון יחידים בנוגע זייער דאטא.

- ג. צו באשרייבן מיטלען פאר דאטא-שוץ.
- ד. צו שאפן טראנספארענץ און הסכמה.

דער ריכטיגער ענטפער, ווי עס ווערט ערקלערט, איז ב: די דאזיגע געזעצן דעפינירן די רעכטן וואס א מענטש האט איבער זיין אייגענער דאטא. דאס הייסט, דאס זענען אלעס געזעצן פון דאטא-פריוואטקייט (data privacy).

דער פאל פון Cambridge Analytica (33:09:27)

דערנאך גייט דער קורס איבער צו קאנקרעטע פעלער פון מיסברויך פון דאטא און די געזעצליכע קאנסעקווענצן דערפון. עס ווערט באטאנט אז דאס זענען זאכן וואס זענען טאקע פארגעקומען אין דער געשיכטע, און מען האט פון זיי געלערנט א לעקציע.

דער ערשטער און באקאנטסטער פאל איז Cambridge Analytica, וואס ווערט באקאנט אלס דער Facebook פריוואטקייט-בריטש (Cambridge Analytica .(data leak איז געווען א דאטא-ארגאניזאציע, און זי האט געשפילט א גאר וויכטיגע ראלע אין די אמעריקאנער וואלן פון 2016. די וואלן האט געוואונען Donald Trump, און לויט דער ערקלערונג האט מען אנגעשטעלט א סיניאר וויצע-פרעזידענט פון יענער ארגאניזאציע, כדי צו ניצן דאטא-סייענס אלגאריטמען צו ענדערן די שטימונג פון די וויילער.

וואס האט Cambridge Analytica געטאן? זיי האבן געזאמלט די פערזענלעכע דאטא פון מיליאנען Facebook באניצער אן זייער הסכמה, און דערנאך האבן זיי גענוצט די דאטא צו שאפן געצילטע פאליטישע רעקלאמעס בעת די פרעזידענטליכע וואלן פון 2016.

דער קורס דערמאנט אז עס איז געמאכט געווארן א דאקומענטאר וועגן דעם פאל, מיטן נאמען "The Great Hack". אין דעם ווערט ערקלערט ווי אזוי די גאנצע קאנספיראציע איז דורכגעפירט געווארן, און ווי Cambridge Analytica האט געשפילט א ראלע אין ענדערן די שטימונג פון די וויילער. דער פאל איז ערשט שפעטער אנטדעקט געווארן, און אין דעם דאקומענטאר גיט אן אינטערוויו א פרוי וואס האט געשפילט א הויפט-ראלע אין פירן יענע קאמפיין. לויט דעם קורס קען מען פון דעם פאל פארשטיין די כוח פון דאטא-סייענס, און ווי גרויסן שאדן מען קען אנטון א פירמע אדער סיי וועלכן עסק וועז מען מיסברויכט דאטא.

די געזעצליכע קאנסעקווענצן (33:11:57)

וועז מען האט דאס אנטדעקט, זענען געקומען שווערע קאנסעקווענצן. ביידע פירמעס, Cambridge Analytica און Facebook, האבן באקומען ריזיגע שטראפן און זייער רעפוטאציע איז געפאלן:

- Facebook האט באקומען א שטראף פון 5 ביליאן דאלער פון דער Federal Trade Commission, פארן פארלעצן די פריוואטקייט פון באניצער און ווייל עס האט נישט באשיצט זייער דאטא.
- Cambridge Analytica איז אינגאנצן פארמאכט געווארן אלס רעזולטאט פונעם אינצידענט.

דער קורס באטאנט: פון דעם זעט מען אז די קאנסעקווענצן קענען גיין ביז אהין אז מען דארף פארמאכן די גאנצע ארגאניזאציע. דאס איז א פאל וואס איז גאר באקאנט אינעם פעלד פון דאטא-סייענס, און עס איז איינער פון די קעיס-סטודיס וואס יעדער זאל זען. דאס איז אויך דער טעם פארוואס מען רעדט וועגן איינפירן דאטא-עטיק אין דאטא-ארגאניזאציעס.

דער פאל פון Marriott (33:12:59)

דער צווייטער פאל איז די Marriott האטעל-קייט. דא איז דאס נישט געשען מיט כוונה, נאר האקערס האבן אריינגעבראכן אין Marriott'ס סיסטעם און באקומען צוטריט צו די פערזענלעכע אינפארמאציע פון בערך 5 מיליאן געסט, אריינגערעכנט זייערע פאספארט-נומערן, קרעדיט-קארטל פרטים און נאך. דער קורס רופט דאס איינער פון די גרעסטע בריטשעס פון פערזענלעכע דאטא.

אלס קאנסעקווענץ האט Marriott באקומען א שטראף פון ארום 110 מיליאן אייראָ, ארויפגעלייגט פונעם UK Information Commissioner's Office (ICO), ווייל עס האט נישט איינגעפירט געניגנדיקע זיכערהייט-מיטלען צו באשיצן די דאטא פון די געסט. דערצו האט די פירמע געהאט א לאוסוט פון די באשעדיגטע געסט, און אויך שאדן צו איר רעפוטאציע.

דער קורס זאגט אז מען קען נעמען די צוויי ביישפילן אלס קעיס-סטודיס, וואו מען זעט גרויסע דאטא-בריטשעס און מיסברויך פון פערזענלעכע דאטא, מיט די געוועלטיגע קאנסעקווענצן וואס די ארגאניזאציעס האבן געליטן.

דער "Salami Attack" (33:14:35)

דערנאך ווערט געפרעגט צי עמיצער ווייסט פון נאך אזעלכע פעלער, און עס ווערט דערציילט ווען דעם וואס ווערט גערופן "Salami attack". לויט דעם קורס איז דער אטאק דורכגעפירט געווארן אויף American Express באנק.

דער אטאקירער האט געטאן א גאר פשוטע זאך: ער האט געגנבעט ווינציגע סכומים. דער קורס ערקלערט דאס מיט אינדישע געלט: ווען אין א קאנטע ווערט צוגערעכנט אינטערעסט פון, למשל, 112 רופיס און 43 פייסע, האט ער געגנבעט די 3 פייסע און זיי אריינגעלייגט אין זיין אייגענער קאנטע. אזוי, יעדעס מאל ווען אינטערעסט איז צוגערעכנט געווארן אין סיי וועלכער קאנטע, האט ער געגנבעט די פייסע, צווישן 1 און 9 פייסע יעדעס מאל. אדער, אויב איז דער סכום געווען 860 רופיס און 67 פייסע, האט ער געגנבעט די 7 פייסע.

דאס איז געווען א מין ווירוס וואס ער האט אריינגעשטעקט אין דער באנק-סיסטעם, און נאך עטלעכע חדשים האט ער אנגעזאמלט אין זיין קאנטע ריזיגע סומעס. דא, ווי עס ווערט ערקלערט, האט די באנק נישט באשיצט איר סיסטעם, און דער מענטש האט געקענט אריינשטעקן א ווירוס, שאפן א פאלשע קאנטע און אריבערפירן אהין געלט.

דער מוסר השכל, לויט דעם קורס: אין דער פארגאנגענהייט האט מען געזען מיסברויך פון דאטא, פון פריימווערקס, פון API'ס און אלערליי סטרוקטורן, דעריבער איז גאר וויכטיג צו איינפירן דאטא-עטיק פרינציפן אין די סיסטעמען.

די Jamtara שווינדלעריין און פישינג (33:16:09)

א צוהערער האט דערמאנט די Jamtara שווינדלעריין. דער קורס ענטפערט או דאס איז ריכטיג, אבער דאס איז מער א סייבער-אטאק, נישט א פאל פון א באשטימטער דאטא-ארגאניזאציע. די Jamtara בחורים האבן פון ערגעץ געזאמלט פערזענלעכע דאטא: טעלעפאן-נומערן, נעמען, אמאל דעמאגראפישע אינפארמאציע, און אמאל אפילו קארטל-נומערן. מיט דעם האבן זיי דורכגעפירט פישינג-אטאקעס, דורך רופן מענטשן, שיקן זיי מעסעדזשעס אדער אימעילס.

דער קורס באמערקט אז אזעלכע שווינדלעריין זענען שטענדיג פאראן אין מארק, אבער דאס קומט נישט פון דעם וואס א סיסטעם האט א בריטש; זיי האבן נאר א וועג ווי צו באקומען די דאטא.

וואס דארף א דאטא-ארגאניזאציע טון (33:17:43)

דער קורס ערקלערט דא וואס דער עיקר פון דער דיסקוסיע איז: אויב איר זענט א דאטא-ארגאניזאציע, סיי וועלכער סארט, וועלכע זענען די סטאנדארט עטישע דאטא-פראקטיקן וואס מען דארף האלטן ביים זאמלען דאטא, ביים אפהיטן דאטא, ביים ניצן דאטא און ביים טיילן דאטא? און פון דער אנדערער זייט, אלס באניצער פון א דאטא-ארגאניזאציע: וועלכע רעכטן האב איך צו באשיצן מיין דאטא? אין דעם פראצעס האט מען שוין געזען די הויפט-פרינציפן, די שווערִיגקייטן, און עטלעכע סטאנדארט קעיס-סטודיס.

באשיצן פערזענלעכע דאטא (33:18:45)

דער נעקסטער טעמע איז באשיצן פערזענלעכע דאטא. פערזענלעכע דאטא איז אייער נאמען, אייער דעמאגראפיע, טעלעפאן-נומער, געשלעכט וכדומה. מען דארף זיך באשעפטיגן מיט זיכערהייט, פריוואטקייט און הסכמה אין צוזאמענהאנג דערמיט. ארגאניזאציעס מוזן נעמען דעם שוץ זייער ערנסט, אבער ווען זיי באקומען די דאטא, שטייען זיי אויך פאר געוויסע שווערִיגקייטן.

דער שוץ ווערט באטראכט פון צוויי פערספעקטיוון:

- פון יחיד'ס פערספעקטיוו: מען דארף זיין זייער פארזיכטיג ביים טיילן סיי וועלכע פערזענלעכע אינפארמאציע: לאקאציע, טעלעפאן-נומער, נאמען, געשלעכט, פאליטישע מיינונג, אדער סיי וואס וואס אידענטיפיצירט א מענטש. מען זאל נישט אנטפלעקן א פאספארט-נומער, א קרעדיט-קארטל נומער; דאס אלעס פאלט אונטער PII, Personal Identifiable Information.
- פון ארגאניזאציע'ס פערספעקטיוו: די ארגאניזאציע מוז באשיצן די אינפארמאציע.

דער יחיד דארף האבן די רעכט:

- צוטריט צו זיין פערזענלעכער דאטא וואס איז אפגעהיטן ביי דער ארגאניזאציע;
- פאררעכטן אומגענויקייטן אין דער דאטא;
- אויסמעקן זיין דאטא אונטער געוויסע אומשטענדן;
- זיך קעגנשטעלן צו פראצעסירן זיין דאטא פאר א באשטימטן צוועק.

דאס זענען קאנטראל-פונקטן וואס מען דארף איינפירן, און די ארגאניזאציע דארף דערפאר אויפשטעלן א פעסטן מעכאניזם פאר דריי זאכן: דאטא-זיכערהייט, פריוואטקייט, און הסכמה-פארוואלטונג. ווען מען האט געזאמלט די דאטא, מוז זי זיין באשיצט, אן אומבארעכטיגטן צוטריט; מען מוז האלטן די פריוואטקייט-געזעצן; מען זאל נישט זאמלען איבעריגע פריוואטע אינפארמאציע, נאר דעם מינימום וואס איז נויטיג; און ביים זאמלען מוז מען באקומען די הסכמה פונעם באניצער.

זיכערהייט און מינימיזאציע (33:21:17)

דער קורס ערקלערט אז זיכערהייט מיינט איינפירן אלע זיכערהייט-מיטלען וואס באשיצן די דאטא פון אומבארעכטיגטן צוטריט, אנטפלעקונג, ענדערונג אדער פארניכטונג. למשל: א האקער וואס קען נישט באקומען צוטריט, קען גאנץ פשוט לויפן א delete באפעל און אויסמעקן די דאטא; דאס זאל אויך נישט זיין מעגלעך.

דער צווייטער פונקט איז מינימיזירן דאס זאמלען און אפהיטן פערזענלעכע דאטא, וואס איז א גאר וויכטיגע פראקטיק: מען זאל זאמלען נאר וואס איז נוצלעך און נויטיג פאר א באשטימטן צוועק. דורך זיכערהייט און

מינימיזאציע פארמיידט מען בריטשעס און מען פארקלענערט דעם שאדן אויב א בריטש פאסירט יא; עס ווערט דעמאלט אנטפלעקט בלויז מינימאלע אינפארמאציע.

דאטא-פריוואטקייט (33:22:18)

נאך זיכערהייט קומט דאטא-פריוואטקייט. דאס איז דער פונדאמענטאלער רעכט פון א יחיד צו קאנטראלירן זיין דאטא: ער האט א פולע רעכט צו וויסן ווי אזוי די דאטא ווערט געזאמלט, ווי זי וועט געניצט ווערן, און ווי זי וועט אפגעהיטן ווערן. עס דארף זיין טראנספארענט ווער עס האט צוטריט צו דער דאטא, און פאר וועלכן צוועק מען ניצט זי.

דאטא-פריוואטקייט, עטיק און א פראקטישע איבונג (33:23:20)

די רעכטן פון באניצער און "אינפארמירטע הסכמה" (33:23:20)

דער קורס שליסט אפ די באהאנדלונג פון דאטא-פריוואטקייט מיט די פיר יסודות'דיגע רעכטן וואס א באניצער האט איבער זיין פערזענליכע דאטא:

- די רעכט צו וויסן ווי אזוי די דאטא ווערט געזאמלט, ווי אזוי זי ווערט געניצט און ווי אזוי זי ווערט אויפגעהיטן.
- די רעכט צו וויסן ווער עס וועט האבן צוטריט צו דער דאטא.
- די רעכט צו פארשטיין צו וועלכן ציל די דאטא ווערט געניצט.
- די רעכט צו באקומען צוטריט צו דער דאטא, זי צו פארריכטן, צו ענדערן און אויסצומעקן, וועו נאר מען דארף אדער וועו דער באניצער האלט אז עס פאסט.

דערנאך ווערט ערקלערט דער באגריף "אינפארמירטע הסכמה" (informed consent). אן ארגאניזאציע מוז זיכער מאכן אז יעדער מענטש גיט א קלארע, אינפארמירטע הסכמה פארן ניצן זיין פערזענליכע דאטא. מען דארף רעספעקטירן דעם יחיד'ס זעלבסטשטענדיגקייט: אויב ער וויל נישט טיילן, איז דאס זיין רעכט און זיין באשלוס. אבער אלעמאל וועו מען זאמלט דאטא, מוז מען נעמען א ריכטיגע הסכמה. אויף אזא אופן בויט מען צוטרוי און דורכזיכטיגקייט, מען באשיצט מענטשן פון אויסנוצונג און שאדן, און מען האלט זיך צו עטישע און פאראנטוואורטליכע דאטא-פירונגען. דער קורס באטאנט אז דאס איז זייער וויכטיג.

א פארלעסליכע דאטא-סיסטעם (33:24:23)

דערנאך ווערט גערעדט וועגן בויען א "פארלעסליכע דאטא-סיסטעם" (trustworthy data ecosystem), וואס שטייט אויף דריי פונקטן:

- געזעצליכקייט: מען מוז זיך האלטן צו די געזעצן אויף דער גאנצער וועלט, סיי געזעצן פאר א געוויסער בראנזשע, סיי פריוואטקייט-געזעצן, זיכערהייט-געזעצן אדער געזעצן איבער טיילן דאטא. די באארבעטונג פון דאטא מוז זיך שטיצן אויף הסכמה, אויף א קאנטראקט אדער אויף געזעצליכע פליכטן.
- יושר: מען מוז באארבעטן די דאטא אויף א גערעכטן אופן, אן פארט-אייגענשאפט (bias), אן דיסקרימינאציע און אן פאראורטיילן.

• פאראנטוואַרטליכקייט: די ארגאניזאציע מוז נעמען פולע אחריות פאר ווי זי פירט זיך מיט דאטא. אויב עס קומט פאר א דאטא-ליק (breach) אדער אן אנדער פראבלעם, מוז דארט זיין איינער וואס נעמט די אחריות און וואס מען קען אים פאדערן חשבון.

צו א פאראנטוואַרטליכער דאטא-צוקונפט (33:25:26)

דער קורס ערקלערט אז צו בויען א פאראנטוואַרטליכע דאטא-צוקונפט, דארף מען א געמיינזאמע אנשטרענגונג פון אלע זייטן:

• רעגירונגען מוזן איינשטעלן שטארקע דאטא-באשיצונג תקנות און זיכער מאכן אז מען פירט זיי טאקע דורך.

• אינדוסטריעס דארפן אנטוויקלען און איינפירן "בעסטע מעטאדן" (best practices), וואס שטעלן עטישע באהאנדלונג פון דאטא אין פארגרונט.

• טעכנאלאגישע חידושים קענען צושטעלן שטארקע געצייג, כדי זיך צו פירן מיט דאטא אויף א פאראנטוואַרטליכן אופן.

• אויפקלערונג פונעם ציבור: לערנען מענטשן וועגן זייערע רעכטן שאפט א מער אינפארמירטע און באמעכטיגטע "דיגיטאלע בירגערשאפט".

• אינטערנאציאנאלע צוזאמענארבעט: מען דארף צוזאמגלייכן די פריוואטקייט-תקנות צווישן לענדער, כדי צו שאפן א מער אייניהייטליכע און פעסטע סטרוקטור.

וועו די אלע קומען זיך צוזאם, רעגירונגען, דער חינוך-סעקטאר, די אינדוסטריע און אינטערנאציאנאלע קערפערשאפטן, קען מען בויען א פאראנטוואַרטליכע דאטא-צוקונפט.

א פראגע צום איבערחזרן (33:26:27)

דער קורס שטעלט א פראגע: וועלכע יסודות'דיגע רעכטן האט א יחיד איבער זיין פערזענליכע דאטא, אין רעם פון דאטא-פריוואטקייט? די ברירות זענען:

• די רעכט צו באארבעטן פערזענליכע דאטא.

• די רעכט צו פארקויפן פערזענליכע דאטא צו דריטע פארטייען.

• די רעכט צו באקומען צוטריט, צו פארריעכטן און אויסצומעקן פערזענליכע דאטא.

• די רעכט צו האלטן פערזענליכע דאטא אויף אייביג.

דער ריכטיגער ענטפער איז דער דריטער: די רעכט צו צוטריט, פארריעכטן און אויסמעקן. דער געדאנק פון דאטא-פריוואטקייט איז אז דער באניצער זאל האבן פולע קאנטראל איבער זיין דאטא.

די הויפט-לעקציעס פונעם קאפיטל (33:27:28)

פארן גיין צו דער פראקטישער איבונג, חזר'ט דער קורס איבער די הויפט-נקודות:

• דורכזיכטיגקייט, יושר, פאראנטוואַרטליכקייט, פריוואטקייט און זיכערהייט זענען די פינף יסודות פון עטישע דאטא-פירונג.

• יושר, פריוואטקייט און דורכזיכטיגקייט זענען די דריי הויפט עטישע באטראכטונגען ביי דאטא-אנאליז. מען מוז זיין געריעכט ביים זאמלען, מען מוז היטן דעם יחיד'ס פריוואטקייט, און מען מוז זיין אפן ביים זאמלען דאטא, דאס הייסט קלאר דערציילן וואס פאר אינפארמאציע מען זאמלט און צו וועלכן ציל.

• מען קען אויספלאנירן עפעקטיווע סטראטעגיעס, כדי זיך צו פירן מיט עטישע דילעמעס וואס קומען ארויף ביי דאטא-אנאליז און זיי אויפצולייזן. עס זענען דא שווערigקייטן, אבער וועו מען האט סטראטעגיעס און מען האלט זיך צו די פירונגען, לייזט מען די פראבלעמען.

• עטישע באשלוסן פארגרעסערן צוטרוי, צווזאמענארבעט און רעפוטאציע. זיי פארמינערן ריזיקע, זיי זארגן אז מען האלט זיך צום געזעץ, זיי מוטיגן חידושים און זיי טראגן ביי צום וואוילזיין פון דער געזעלשאפט. אויב אן ארגאניזאציע נעמט עטישע באשלוסן, און נישט ווי אין די פעלער פון Cambridge Analytica אדער Meta, בויט זי צוטרוי ביי די באניצער און א גוטע רעפוטאציע. אין סך הכל שאפט דאס א בעסערע דאטא-צוקונפט, מיט שטארקן צוטרוי צווישן אלע פארטייען.

• געזעצליכע און רעגולאטארישע סטרוקטורן נעמען אריין געזעצן, כללים, אנווייזונגען און פליכטן פאר א פאראנטווארטליכער באהאנדלונג פון דאטא און פריוואטקייט. מען האט געלערנט דריי באקאנטע: GDPR פון אייראפע, PIPL פון כינע, און CCPA פון קאליפארניע אין אמעריקע. דערצו זענען פאראן תקנות און אויפזיכט-קערפערשאפטן פאר ספעציפישע בראנזשעס.

עס ווערן אויך אנגעווויזן נאך מקורות אין דער פרעזענטאציע, פאר ווער עס ווייל זיך ווייטער לערנען. זיי באהאנדלען די עטישע פליכטן ביים טיילן דאטא, עטיק ביי "אפענער דאטא", נייע פריוואטקייט-כללים, און ווי אזוי צו באשיצן פריוואטקייט אין א געזעלשאפט וואס לעבט פון דאטא.

פראקטישע איבונג: ווי אזוי פירט מען זיך מיט סענסיטיווע קונה-דאטא (33:31:03)

די איבונג פאדערט נישט קיין שום געצייג. זי שטיצט זיך אויף א דאקומענט איבער די יסודות פון עטישע דאטא-פירונג. דער ציל איז אנצוועגדן די פרינציפן פון דאטא-פריוואטקייט אויף סענסיטיווע און פערזענליכע אינפארמאציע אין רעאלע סצענארן. דורך א רעאליסטישן פאל זעט מען ווי די קלאס-באגריפן פאסן צו צו א פראפעסיאנעלער ארבעט. די שאלה איז: ווי אזוי וואלט איר איינגעפירט די פאליסיס אין אייער ארגאניזאציע? למשל, ווי אזוי פירט מען זיך מיט סענסיטיווע קונה-דאטא?

אן ענטפער פונעם עולם איז געווען "צו פארזיכערן די דאטא", און דער קורס פרעגט ווייטער ווי אזוי, למשל דורך פארשליסלונג (encryption). דערנאך ווערט געוויזן דער ענטפער-שליסל. דער סצענאר איז אז מען דארף ענטפערן אויף א פראגע פון א קונה און דערביי האלטן די העכסטע סטאנדארטן פון פריוואטקייט. די טריט זענען:

• אריינלאגן אין א זיכערער קונה-אינפארמאציע סיסטעם מיט אויטאריזירטע קרעדענשעלס, און געפינען דעם קונה'ס פראפיל לויט די אינפארמאציע אין זיין אימעיל. אזוי זארגט מען אז נאר באפולמעכטיגטע האבן צוטריט.

• אנשרייבן אן ענטפער אויף א זיכערער פלאטפארמע, נישט אויף סיי וועלכן פלאץ. דער ענטפער זאל זיין קורץ און צו דער זאך, אן קיין איבעריגע פערזענליכע פרטים. דאס איז דער "מינימיזאציע-פרינציפ": מען ניצט נאר די אינפארמאציע וואס מען דארף.

• א הסכמה-באשטעטיגונג: אויב די פראגע פאדערט צו טיילן נאך אינפארמאציע, בעט מען העפליך די הסכמה. מען ערקלערט קלאר פארוואס מען דארף די אינפארמאציע און ווי אזוי זי וועט גענוצט ווערן. אזוי האלט מען די דורכזיכטיגקייט.

• זיכערע קאמוניקאציע: אויב עס זענען פאראן זיכערע וועגן, למשל א פארשליסלטע מעסעדזש-סיסטעם אדער א זיכערע קונה-פארטאל, ניצט מען זיי. סענסיטיווע אינפארמאציע טאר מען נישט שיקן אין א געווענליכן אימעיל.

- א בקשה צו אויסמעקן די דאטא: אויב דער קונה בעט אויסמעקן זיין דאטא, גיט מען אים אינפארמאציע איבערן פראצעס און מען באשטעטיגט מיט אים איידער מען מעקט אויס, לויט די תקנות. וועז די ארבעט איז פארטיג, זאל מען אויסמעקן די אינפארמאציע, און אזוי באשיצט מען די סענסיטיווע קונה-דאטא.

דער קורס באטאנט אז דאס איז אלעס טעאריע און עטיק: ווי אזוי מען דארף רעאגירן אין אזעלכע סענסיטיווע פעלער.

די הויפט-פונקטן פון די געזעצליכע סטרוקטורן (33:35:06)

- פריוואטקייט-געזעצן: זיי באשיצן סענסיטיווע אינפארמאציע.
- דאטא-באשיצונג געזעצן: זיי באהאנדלען די זיכערהייט פון דאטא, דאס הייסט ווי אזוי מען האלט זי און ווי אזוי מען פירט זי איבער.
- געזעצן פון דורכזיכטיגקייט און הסכמה: מען מוז נעמען הסכמה פונעם באניצער איידער מען טיילט זיין דאטא.
- אייגנטום און רעכטן איבער דאטא: דער באניצער האט פולע קאנטראל. למשל, וועז די ארבעט איז פארטיג און דער ציל איז דערגרייכט, קען מען די דאטא אויסמעקן.
- פאראנטווארטליכקייט און צוהאלטן זיך צו די כללים: די סטרוקטור שטעלט איין אחריות פאר דער ארגאניזאציע. זי מוז האבן אינערליכע פאליסיס וואס פירן דורך די אלע טריט, און וואס מאכן זי פאראנטווארטליך אז זי האלט זיך צו די סטאנדארטן פון דאטא-עטיק. ביי איבערפירן דאטא איבער גרענעצן פון לענדער קומען צו נאך תקנות.
- עטישע באטראכטונגען: חוץ די געזעצליכע פאדערונגען זענען דא אויך יושר און נישט-דיסקרימינאציע. אויף געוויסע געביטן קען מען נישט באשטימען די כללים פעסט אדער "מאטעמאטיש", דעריבער מוז מען זיך שטיצן אויף דעם עטישן חוש: וואס איז רעכט און וואס איז אומרעכט.

די הויפט-פונקטן פון כינע'ס PIPL (33:36:41)

צום סוף וועלט געפרעגט וועלכע זענען די הויפט-פונקטן פון כינע'ס געזעץ צו באשיצן פערזענליכע אינפארמאציע (PIPL). ווייל עס איז א געזעץ פאר פערזענליכע אינפארמאציע, קען מען לאגיש פארשטיין וואס עס נעמט אריין:

- מען מוז באקומען הסכמה איידער מען נעמט די אינפארמאציע.
- כללים פאר איבערפירן דאטא.
- די רעכטן פונעם מענטש וועמענס דאטא עס איז: צוטריט, פארריכטן און אויסמעקן.
- אין דער ארגאניזאציע מוז זיין א "דאטא-באשיצונג באאמטער" (data protection officer). ער איז דער אדרעס צו וועמען זיך, וועז עס קומט פאר א פראבלעם אדער א ליק.
- א דעפיניציע פון סענסיטיווע דאטא.
- אחריות און געזעצליכע התחייבות.

דער קורס פארלאנגט אז מען זאל ווייטער פארשן די פונקטן פון דעם געזעץ.

פון עטיק צו אגענטישן RAG און ראוטער (33:38:12)

א שליסווארט וועגן דאטא-עטיק אין ארגאניזאציעס (33:38:12)

דער קורס ענדיקט א פריערדיגע לעקציע מיט א געדאנק וועגן דאטא-עטיק אין ארגאניזאציעס. עס ווערט ערקלערט אז דאס איז געווען א געפירטע איבונג, וואס ווייזט ווי אזוי מען פירט איין גוטע דאטא-פראקטיקן אין א ארגאניזאציע אין פארשידענע סצענארן. מען לייגט צו אז דער לערנער זאל אויך אליין ווייטער פארשן און לערנען.

עס ווערט דערמאנט אז היינט האט כמעט יעדע ארגאניזאציע א באאמטער וואס איז פאראנטוואורטלעך פאר אלע עטישע דאטא-פראקטיקן. דער קורס שאצט אז אין דער צוקונפט וועט מען אויך האבן א באאמטער וואס איז פאראנטוואורטלעך פאר עטישע AI-פראקטיקן, ווייל דאטא און AI ארבעטן היינט זייער ענג צוזאמען. לויט דעם קורס וועט מען אזעלכע שטעלעס זען אין גרויסע ארגאניזאציעס גאר באלד.

ווי אזוי אן אגענט ארבעט (33:39:13)

דערנאך הייבט אן א נייע לעקציע. עס ווערט געזאגט אז איינס אדער מערערע אגענטן קענען צוזאמענארבעטן כדי צו פירן און דורכפירן ארבעטן. דער קורס ערקלערט צוויי עיקר טיילן ווי אזוי עס ארבעט:

- פארשטיין און אקציע: צום ערשט באניצט זיך דער אגענט מיט זיינע שפראך-פעאיקייטן כדי צו פארשטיין די פראגע פונעם באניצער. דערנאך באשליסט ער וועלכע עקסטערע גענג ער דארף ניצן כדי אויסצופירן די ארבעט.
- פארוואלטונג פון ארבעטן: אין אסאך סיסטעמען ארבעט איין אגענט צוזאמען מיט אנדערע ספעציאליזירטע אגענטן. מען קען דאס פארגלייכן מיט א מאנשאפט, וואו יעדער מיטגליד האט זיין באשטימטע ארבעט.

א ביישפיל: מען וויל איבערזעצן א דאקומענט אין עטליכע שפראכן.

- דער הויפט-אגענט איז דער ערשטער שריט. ער באקומט די בקשה און צעטיילט די איבערזעצונג-ארבעט אויף קלענערע טיילן.
- ספעציאליזירטע אגענטן: ער גיט איבער טיילן פון דער ארבעט צו אנדערע אגענטן, יעדער פון זיי ספעציאליזירט אין א באשטימטער שפראך.
- צוזאמענארבעט: די אגענטן ארבעטן צוזאמען און שיקן צוריק אלע איבערגעזעצטע טיילן צום הויפט-אגענט, וואס שטעלט אלעס צוזאמען פארן באניצער.

בקיצור, זאגט דער קורס, איז אן אגענט מער ווי סתם א טייל פון AI. עס איז אן אקטיוואר געהילף וואס ניצט זיין אייגענע אינטעליגענץ צוזאמען מיט עקסטערע געצייג. ער פירט און ענדיקט פארשידענע ארבעטן, אפט ווען ער ארבעט ווי א מאנשאפט מיט אנדערע אגענטן.

וואס איז RAG (33:40:15)

RAG שטייט פאר Retrieval Augmented Generation. דאס איז א מעטאד וואס העלפט AI צו געבן מער גענויע ענטפערס דורכן ניצן דרויסנדיגע וויסן-קוועלער. עס פארבינדט אויפזוכן אינפארמאציע מיט ענטפערס וואס די AI שאפט אליין, און אזוי קען די AI צוקומען צו פאקטישער דאטא פון דער וועלט און זי ניצן.

מיט RAG פארלאזט זיך די AI נישט נאר אויף איר איינגעבויטן וויסן. זי זוכט אויף פאסיגע אינפארמאציע אין א דאטאבאזע אדער אין דאקומענטן איידער זי ענטפערט. די וויסן-קוועלער קענען זיין סטרוקטורירטע

דאטאבאזעס, ווי SQL אדער PostgreSQL. זיי קענען אויך זיין נישט-סטרוקטורירט, ווי PDF-פיילס וואס זענען אפגעהיטן אין וועקטאר-דאטאבאזעס. דאס העלפט די AI צו געבן מער גענויע און מער דערהיינטיקטע ענטפערס.

אגענטישער RAG און פארוואס ער איז נוצלעך (33:40:45)

אנשטאט זיך צו פארלאזן נאר אויף איין שפראך-מאדעל וואס שאפט די ענטפערס, פארוואנדלט אגענטישער RAG די מאדעלן אין קלוגע AI-אגענטן. יעדער אגענט קען ניצן עקסטערע געצייג, פונקציעס און דרויסנדיגע וויסן-קוועלער כדי צו ענדיקן זיין ארבעט. אין אן אגענטישער סיסטעם קענען עטליכע אגענטן ארבעטן צוזאמען. למשל, איין אגענט קען זיין פאראנטווארטלעך פארן אויפזוכן דאטא, און אן אנדערער קען זיך קאנצענטרירן אויפן פראצעסירן און שאפן שפראך. אזוי שאפן זיי א מאנשאפט, וואו יעדער מיטגליד האט א ספעציעלע ראלע.

דער קורס רעכנט אויס פארוואס אגענטישער RAG איז נוצלעך:

- בעסערע לייזונג פון פראבלעמען: ווייל די ארבעטן ווערן צעטיילט צווישן ספעציאליזירטע אגענטן, קען אגענטישער RAG באהאנדלען מער קאמפליצירטע און פיינערע פראבלעמען ווי איין שפראך-מאדעל אליין.
- איינבינדן געצייג: יעדער אגענט איז נישט באגרעניצט נאר צו שפראך-פראצעסירונג. ער קען ניצן עקסטערע געצייג, ווי זוך-פונקציעס, דאטאבאזעס אדער באשטימטע API-ס, כדי צו באקומען מער גענויע און דעטאלירטע אינפארמאציע. א ביישפיל פון א סיסטעם פאר שטייערן: איין אגענט קען זיך קאנצענטרירן אויפן שאפן שטייער-קארטלעך. א צווייטער קען פירן די שטייער-איינזאמלונגען און ריטורנס. א דריטער קען איבערזעצן שטייער-פראגעס אין אנדערע שפראכן, ווי שוועדיש אדער שפאניש.
- בייגזאמקייט און עפעקטיווקייט: די סיסטעם פאסט זיך צו פארשידענע ארבעטן, ווייל זי גיט יעדע ארבעט צום אגענט וואס פאסט דערפאר דאס בעסטע. צוליב דער צוזאמענארבעט ווערט יעדער טייל פון א פראבלעם, אפילו ווען ער האט אסאך זייטן, באהאנדלט דורכן פעאיקסטן אגענט. דאס ברענגט מער עפעקטיוע רעזולטאטן.

דער פארגלייך פונעם קורס: אגענטישער RAG איז ווי א מאנשאפט פון עקספערט-געהילפן. אנשטאט אז איין מענטש זאל פרובירן אליין טון אלעס, איז יעדער געהילף ספעציאליזירט. זיי טיילן זיך מיט אינפארמאציע, ניצן עקסטערע געצייג און ארבעטן צוזאמען כדי צו געבן א פולע לייזונג, אפילו פאר קאמפליצירטע פראבלעמען. דאס מאכט די גאנצע סיסטעם פיל שטארקער און מער פיעלזייטיג.

די פיר טיילן (33:43:19)

דער קורס ערקלערט די פיר הויפט-טיילן וואס ארבעטן צוזאמען כדי צו געפינען, פארבעסערן און שאפן אינפארמאציע:

- פראמפט: די פראגע אדער די אנווייזונג וואס מען גיט דער AI, ווי ווען מען פרעגט א פראגע אדער מען גיט א באפעל כדי אנצוהייבן א שמועס. ער איז וויכטיג ווייל ער שטעלט אוועק דעם קאנטעקסט פון וואס די AI דארף טון, און ער פירט דעם גאנצן פראצעס.
- דאס גרויסע שפראך-מאדעל (LLM): דער הויפט-AI-מאטאר וואס פארשטייט דעם פראמפט און שאפט אן ענטפער. מען קען אים פארגלייכן מיט א זייער קלוגן טעקסט-פראצעסער וואס האט זיך אויסגעלערנט פון אסאך ביישפילן. ער ניצט זיין טריינירן און זיינע שפראך-פעאיקייטן כדי צו שאפן אן ענטפער וואס פאסט צו דער בקשה.

• אגענט: אן AI-געהילף וואס ארבעט צוזאמען מיטן LLM. ער ווייסט ווי אזוי צו ניצן עקסטערע געצייג כדי צו געפינען מער דעטאלירטע און ספעציפישע אינפארמאציע. ער ברענגט די ריכטיגע דאטא פון פארשידענע קוועלער, כדי דער ענטפער זאל זיין פולשטענדיג און גענוי.

• וויסן-באזע: א סיסטעם וואו מען היט אפ פארשידענע סארטן אינפארמאציע, ווי א גרויסע דיגיטאלע ביבליאטעק. זי קען האלטן סטרוקטורירטע דאטא, וואס איז ארגאניזירט ווי א דאטאבאזע, למשל טאבעלעס. זי קען האלטן האלב-סטרוקטורירטע דאטא, וואס איז ארגאניזירט אויף א ווייניקער שטרענגן אופן, ווי ספרעדשיטס אדער JSON. און זי קען האלטן נישט-סטרוקטורירטע דאטא, דאס הייסט רויע אינפארמאציע ווי טעקסט-דאקומענטן, ארטיקלען אדער אפילו אימעילס. די וויסן-באזע גיט דעם אינהאלט און די פאקטן, פון וועלכע די AI קען שעפן כדי צו ענטפערן גענוי אויפן פראמפט.

צוזאמען ארבעטן די פיר טיילן ווי א מאנשאפט. דער באניצער גיט א פראמפט, דער LLM טראכט אויס אן ענטפער, דער אגענט זאמלט איין די עקסטערע פרטים וואס מען דארף, און די וויסן-באזע גיט די גאנצע פאסיגע אינפארמאציע.

פארוואס דארף מען RAG-אגענטן (33:45:21)

טראדיציאנעלע RAG-סיסטעמען האבן לעכער אין וויסן, ווייל זיי קענען נישט צוקומען צו מערערע וויסן-קוועלער. ווען מען לייגט צו AI-אגענטן, קען מען אנפילן די לעכער מיט דרויסנדיגע געצייג. אויב די AI ווייסט נישט קיין ענטפער, קען זי ברענגען דאטא פון דרויסנדיגע דאטאבאזעס און פארלעסלעכע קוועלער, און אזוי ענטפערן גענוי. דאס ניצן מערערע וויסן-באזעס צוזאמען מיט AI-אגענטן העלפט אויף עטליכע אופנים:

• ווייניקער טעותים: די AI קען פארגלייכן פארשידענע קוועלער, און דערמיט ווערן די ענטפערס מער פארלעסלעך.

• בעסער פארשטאנד: ווען די AI קומט צו צו פארשידנארטיג וויסן, קען זי גענויער ענטפערן אויף קאמפליצירטע פראגעס.

• סקאלירבארקייט: די סיסטעם קען באהאנדלען גרויסע מאסן דאטא און באדינען מיליאנען באניצער אויף אן עפעקטיוון אופן.

אזוי ווערן AI-אפליקאציעס קליגער, מער פארלעסלעך, און פעאיג צו באהאנדלען קאמפליצירטע פראגעס. דער קורס זאגט אז שפעטער וועט קומען א לעקציע וועגן דעם Router Query Engine.

דער ראוטער (33:46:23)

דער ראוטער איז די פשוטסטע פארעם פון אן אגענטישער RAG-סיסטעם. ער איז פאראנטווארטלעך צו דירעקטן די פראגעס פון באניצער צום פאסיגן retriever אדער ענטפער-מאדול, לויט א לאגיק וואס איז פריער באשטימט. ער ארבעט אין דריי שריט:

• ער באקומט די פראגע, דאס הייסט דעם אינפוט אדער דעם פראמפט פונעם באניצער.

• ער קלאסיפיצירט די כוונה: ער שטעלט פעסט צי די פראגע פארלאנגט אויפזוכן פאקטן, א סיכום, אדער א געשאפענעם ענטפער.

• ער שיקט די פראגע צו דער ריכטיגער סיסטעם: צו א retriever פון א וויסן-באזע פאר פאקטן; צו א ספעציאליזירטן LLM פאר שאפן, ווען די פראגע איז שעפעריש אדער אפן; אדער צו א קאמבינאציע פון ביידע פאר היברידע ענטפערס.

פרייזן פון OpenAI און צוגרייטונג פונעם קאוד (33:47:26)

פאר די פרייזן פון OpenAI API קען מען גיין גלייך צום וועבזייטל openai.com/api/pricing. וועט מען סקראלט אראפ, געפינט מען דעטאלירטע פרייזן פאר אלע לעצטע מאדעלן, מיט די פרייזן פאר input, cached input און output. אין דעם קורס ניצט מען GPT-3.5 Turbo, און עס ווערט דערמאנט אז דער פרייז איז $0.50 פאר input און $1.50 פאר output. מען קען אויסקלייבן יעדן API וואס פאסט פארן פראיעקט.

די שריט אין קאוד:

- מען אימפארטירט דעם OpenAI-שליסל דורך א helper-פונקציע.
- מען לויפט nest_asyncio. דער טעם איז אז Jupyter לויפט אן event loop הינטער די קוליסן, און אסאך מאדולן ניצן async. כדי async זאל גוט ארבעטן מיט Jupyter Notebook, דארף מען דאס אימפארטירן.
- מען לאדנט דעם דאקומענט. מען ניצט דעם MetaGPT -ארטיקל, וואס איז לעצטנס אנגענומען געווארן ווי אן oral paper ביי ICLR 2024, וועגן א נייער multi-agent framework . מען קען לויפן יעדע צעל אויך מיט Shift+Enter.
- מען אימפארטירט SentenceSplitter פון LlamaIndex, מען באשטימט א chunk size, און מען צעטיילט די דאקומענטן אין שטיקלעך.
- א שריט וואס מען מוז נישט טון: מען באשטימט דעם LLM און דאס embedding-מאדעל דורך א גלאבאלע קאנפיגוראציע, Settings. דורכאויס דעם קורס ניצט מען GPT-3.5 Turbo און text-embedding-ada-002. דאס גיט א יסוד צו אריינלייגן אייגענע מאדעלן און embeddings. מען אימפארטירט Settings, OpenAI און OpenAIEmbedding.

אינדעקסן, query engines און דער ראוטער (33:50:00)

מען שטעלט אויף צוויי אינדעקסן: SummaryIndex און VectorStoreIndex. מען אימפארטירט די צוויי פשוטע מאדולן און מען איניציאליזירט זיי.

דערנאך פארוואנדלט מען די אינדעקסן אין query engines און נאכדעם אין query tools. יעדער query engine איז דער גאנצער אינטערפעיס פאר פראגעס איבער דער דאטא וואס שטייט אין דעם אינדעקס. ער פארבינדט א retriever מיט דער סינטעז פונעם LLM. יעדער query engine איז גוט פאר א באשטימטן סארט פראגע, און דאס איז פונקט א גוטער פאל פאר א ראוטער, וואס קען דינאמיש אויסקלייבן צווישן זיי. א query tool איז פשוט א query engine מיט metadata, ספעציעל א באשרייבונג פון וועלכן סארט פראגע דאס געצייג קען ענטפערן.

אין קאוד שאפט מען summary_query_engine פון summary_index.as_query_engine() , און vector_query_engine פון vector_index.as_query_engine(). יעדער ווערט אפגעלייטעט פון זיין אייגענעם אינדעקס.

נאכדעם אימפארטירט מען RouterQueryEngine פון llama_index.core.query_engine. מען איניציאליזירט אים מיט LLMSingleSelector.from_defaults() אלס selector, מיט די summary_tool און vector_tool וואס מען האט פריער געבויט, און מיט verbose=True:

```
query_engine = RouterQueryEngine(selector=LLMSingleSelector.from_defaults(),
query_engine_tools=[summary_tool, vector_tool], verbose=True)
```

מען פרעגט "What is the summary of the document?" און מען דרוקט אויס דעם ענטפער מיט print(str(response)). דער ענטפער איז א סיכום פונעם MetaGPT-דאקומענט: דער דאקומענט שטעלט פאר MetaGPT, א meta-programming framework וואס פארבעסערט די לייזונג פון פראבלעמען.

צום סוף קוקט מען ווי פיל קוועלער-שטיקלעך מען האט באניצט, מיט print(len(response.source_nodes)), אזוי ווי מען האט שוין פריער געזען. דער רעזולטאט איז 34.

Router-מאטאר און Tool Calling אין אגענטישן RAG (33:52:59)

ענדיגן דעם Router Query Engine (33:52:59)

אין דער לעקציע ווערט ווייטער געוויזן ווי מען ניצט דעם query engine. מען גרייט צו א שאלה פארן query engine, מען לאזט עס לויפן, און דערנאך דרוקט מען אויס דעם ענטפער מיט print(str(response)).

די שאלה איז געווען: ווי אזוי טיילן אגענטן אינפארמאציע מיט אנדערע אגענטן? דער ענטפער וואס דער סיסטעם האט געגעבן: אגענטן טיילן אינפארמאציע דורך א געמיינזאמען "message pool". אין דעם pool קענען זיי פארעפנטליכן סטרוקטורירטע מעלדונגען. זיי קענען זיך אויך אבאנירן אויף מעלדונגען וואס זענען פאסיג פאר זיי, לויט זייערע פראפילן. דער קורס באמערקט או דאס איז דער ריכטיגער ענטפער.

דערנאך ווערט אלעס צוואמגעשטעלט. מען אימפארטירט די פונקציע get_router_query_engine און מען גיט איר אריין דעם MetaGPT PDF, פונקט ווי פריער. די נייע שאלה איז: "דערצייל מיר וועגן די רעזולטאטן פון דער ablation study". מען לאזט עס לויפן און מען דרוקט אויס דעם ענטפער.

דער ענטפער זאגט או די רעזולטאטן פון דער ablation study ווייזן או MetaGPT לייזט עפעקטיוו די שווערigקייטן ארום ניצן דעם קאנטעקסט, ארום קאוד, און נאך.

דערמיט ענדיגט זיך דער טייל וועגן דעם query engine router. ווייטער וועט מען ארומרעדן tool calling. דער ציל איז צו ווייזן ווי א LLM קען נישט נאר אויסקלייבן וועלכע פונקציע מען זאל אויספירן, נאר אויך אליין ארויסרעכענען וועלכע ארגומענטן מען זאל איר איבערגעבן.

וואס איז Tool Calling אין אגענטישן RAG (33:54:36)

עס ווערט ערקלערט או ביים בויען אגענטישע RAG-אפליקאציעס (Retrieval Augmented Generation) שפילט tool calling א קריטישע ראלע. דורך דעם קענען די אגענטן געפינען אינפארמאציע, זי פראצעסירן, און ארויסגעבן גוטע ענטפערס.

"Tool calling" באדייט דא די פעאיגקייט פון אן אגענט צו רופן דינאמיש אויסערליכע געצייג, APIs אדער דאטא-באזעס. דאס פארבעסערט זיינע באשלוסן און די ענטפערס וואס ער שאפט.

קורץ געזאגט, tool calling לאזט AI-אגענטן אין א RAG-pipeline טון די פאלגנדע זאכן:

- ברענגען אויסערליכע דאטא, למשל פון א וועב-זוך, פון דאטא-באזעס אדער פון APIs.
- פראצעסירן סטרוקטורירטע אינפארמאציע, למשל דורכפירן חשבונות אדער פרעגן vector stores.
- ארבעטן מיט אנדערע סיסטעמען, למשל רופן Python-פונקציעס, SQL-פראגעס אדער cloud-סערוויסעס.
- קאנטראלירן די ענטפערס איידער מען גיט ארויס דעם לעצטן ענטפער.

אויף דעם אופן גייט דער מאדעל אריבער די גרענעצן פון דעם וואס איז געווען אין זיין טרייניר‌ונג-דאטא. ער ווערט מער צופאסבאר און מער גענוי.

מען קען בויען א "multi-agent RAG". דארט האט יעדער אגענט א ספעציאליזירטע ראלע, און tool calling לאזט זיי דינאמיש ארבעטן מיט פארשידענע קוואלן פון וויסן.

די פיר טריט פון אן אגענטישן RAG-pipeline (33:55:36)

דער קורס שטעלט אויס די טריט פון א טיפישן אגענטישן RAG-pipeline מיט tool calling.

ערשטער טריט: פראצעסירן די פראגע פונעם באניצער. דער pipeline הייבט זיך אן וועו דער באניצער שיקט איר‌יין א פראגע. דער סיסטעם אנאליזירט קודם די כוונה און דעם קאנטעקסט פון דער פראגע, כדי צו באשטימען צי מען דארף נאך אויסערליכע אינפארמאציע.

אין דעם שטאפל באשליסט דער אגענט צי מען קען ענטפערן מיטן אינערליכן וויסן פונעם מאדעל, אדער מען דארף זיך וועגדן צו אויסערליכע קוואלן, ווי APIs אדער דאטא-באזעס. אויב די פראגע פאדערט דאטא פון יעצט (realtime), סטרוקטורירטע אינפארמאציע, אדער א פארשטארקטע זוכענ‌יש, גרייט זיך דער סיסטעם צו רופן אויסערליכע געצייג.

דער אגענט קען אויך איבערפארמולירן די פראגע כדי די זוכענ‌יש זאל זיין בעסער. למשל, ער קען פארוואנדלען א שאלה אין פשוטער שפראך אין א סטרוקטורירטע SQL-פראגע אדער אין א זוך-באפעל. אויב מען דארף נישט קיין אויסערליכע זוכענ‌יש, גייט דער סיסטעם גלייך ווייטער צו שאפן דעם ענטפער פונעם שפראך-מאדעל.

צווייטער טריט: געפינען און צולייגן (retrieval and augmentation). אויב מען דארף אויסערליכע דאטא, קלייבט דער סיסטעם אויס דעם בעסטן מעכאניזם צו געפינען זי:

- איינער פון די הויפט-וועגן איז "semantic search" דורך vector-דאטא-באזעס, ווי Pinecone, FAISS אדער Weaviate. זיי געפינען די מערסט פאסיגע דאקומענטן לויט embeddings.
- אויב מען דארף סטרוקטורירטע דאטא, ווי פינאנציעלע באריכטן אדער פרטים פון קונים, בויט דער אגענט SQL- אדער NoSQL-פראגעס און פירט זיי אויס אויף דער דאטא-באזע אדער אויף א knowledge graph.
- וועו מען דארף אינפארמאציע פון יעצט, ווי וועטער, פרייזן פון אקציעס אדער פרישע נייעס, רופט דער אגענט APIs פון אויסערליכע קוואלן, ווי OpenWeather, Google Search אדער אנדערע LangChain-געצייג.
- retrievers פון פריימווערקס ווי LangChain קענען אויך ברענגען איינסן פון א knowledge base, אפטע פראגעס (FAQs) אדער רעפערענץ-דאקומענטן.

וועו מען האט געפונען די פאסיגע אינפארמאציע, הענגט מען זי צו דער פראגע. אזוי ווערט דער קאנטעקסט פונעם מאדעל בריי‌טער, און דער ענטפער ווערט געניויער.

דריטער טריט: פראצעסירן און קאנטראלירן. וועו די זוכענ‌יש איז פארטיג, פראצעסירט דער סיסטעם די דאטא כדי זיכער צו מאכן אז זי איז געגוי און פאסיג, איידער מען ניצט זי אינעם לעצטן ענטפער. דער טריט נעמט איר‌יין:

• אויב מען האט געפונען עטלעכע דאקומענטן אדער דאטא-פונקטן, שטעלט זיי דער סיסטעם אין א ריי לויט ווי פאסיג זיי זענען. ער קען אויך ניצן סוממירונג-טעכניקן, כדי צו פארקלענערן גרויסע מאסן טעקסט אין קורצע, פארדייבארע פונקטן.

• וועו מען דארף נומערישע חשבונות אדער קאמפליצירטע לאגישע טראכטונג, קען דער אגענט אויספירן Python-פונקציעס כדי אויסצורעכענען די וועריאבלן. דער ביישפיל פונעם קורס: אויב א באניצער פרעגט "וויפיל איז 15% פון 1,200 דאלאר?", רופט דער סיסטעם א פונקציע וואס מאכט דעם חשבון.

• ביים ברענגען אינפארמאציע פון יעצט, ווי אקציע-פרייזן אדער נייעס, קאנטראלירט דער סיסטעם די timestamps, כדי דער ענטפער זאל שפיגלען די נייסטע דאטא.

• אויב מען געפינט איבעריגע אדער סותר'דיגע אינפארמאציע פון עטלעכע קוואלן, פילטערט דער סיסטעם און נעמט ארויס די טאפלטע (deduplicate), איידער ער גייט ווייטער.

פערטער טריט: שאפן דעם לעצטן ענטפער. דער שפראך-מאדעל פאראייניגט זיין אינערליכן וויסן מיט די געפונענע און אויסגערעכנטע דאטא, און ער שאפט א צווזאמגעהאלטענעם, גוט-סטרוקטורירטן ענטפער. דער טריט נעמט אריין:

• דער ענטפער וועלט פארמאטירט אויף א באקוועמען אופן פארן באניצער, קלאר און קורץ.

• אויב פארשידענע קוואלן געבן פארשידענע מיינונגען, קען דער אגענט ציטירן רעפערענצן אדער פארשטעלן עטלעכע שטאנדפונקטן.

• דער סיסטעם פארפיינט די שפראך כדי דער ענטפער זאל זיין לייכטער צו לייענען, און ער פארפשוט'ט קאמפליצירטע דאטא וועו נויטיג.

• אויב דער באניצער וויל נאך פרטים, קען מען אנהייבן א "feedback loop". דארט וועלט די פראגע פארפיינט, אדער מען זוכט נאכאמאל.

דער סטרוקטורירטער pipeline זארגט אז אן אגענטישער RAG-סיסטעם זאל באשטימען וועו אן ווי אזוי צו ברענגען אויסערליכן וויסן, צו פראצעסירן אינפארמאציע, און ארויסצוגעבן פארלעסליכע ענטפערס.

tool calling פארגרעסערט שטארק די פעאיגקייטן פון AI. עס לאזט ניצן דאטא פון יעצט, מאכן חשבונות, און קאנטראלירן אינפארמאציע דינאמיש. אלס רעזולטאט וועלט דער סיסטעם מער זעלבסטשטענדיג, מער באוואוסטזינ'יג פונעם קאנטעקסט, און ער קען באהאנדלען קאמפליצירטע פראגעס פול מיט אינפארמאציע. דאס פאסט אין פילע געביטן, למשל פארשונג, קונים-שטיצע און business intelligence.

דעמא: א פשוטע Tool Calling מיט LlamaIndex (34:00:13)

יעצט וועלט אלעס געוויזן אין פראקטיק:

• קודם שטעלט מען איין דעם OpenAI API key.

• דערנאך אימפארטירט מען nest_asyncio און מען לאזט עס לויפן.

דער ציל פון דעם ערשטן טייל איז צו געבן א יסוד-פארשטאנד פון tool calling. מען וועלט וויזט וי אזוי מען שאפט אן tool-interface פון א Python-פונקציע. דער LLM שטעלט דערנאך אליין אריין די פאראמעטערס, לויט דער סיגנאטור פון דער Python-פונקציע. דאס וועלט געמאכט מיט די abstractions פון LlamaIndex.

מען דעפינירט צוויי שפיל-פונקציעס פון א קאלקולאטאר:

- אן add-פונקציע, וואס לייגט צוזאמען x און y.
- א mystery-פונקציע, וואס רעכנט (x + y) * (y + x).

די צענטראלע abstraction אין LlamaIndex איז FunctionTool. זי וויקלט ארום סיי וועלכע Python-פונקציע, אזוי אז מען קען זי צושטעלן פארן LLM. מען באמערקט אז ביידע פונקציעס האבן type annotations פאר די משתנים x און y, און אויך א docstring.

FunctionTool ארבעט נאטירליך צוזאמען מיט די function-calling פעאיגקייטן פון פילע LLMs, אריינגערעכנט OpenAI. כדי צו געבן די געצייג פארן LLM, אימפארטירט מען דעם LLM-מאדול, אין דעם פאל OpenAI מיט דעם מאדעל GPT-3.5 Turbo. דערנאך רופט מען predict_and_call אויבן אויפן LLM.

predict_and_call נעמט א סך געצייג און דעם אריינגאנג-פראמפט, אדער א ריי chat-מעלדונגען. דערנאך באשליסט עס אליין וועלכן tool צו רופן, רופט דעם tool אליין, און ברענגט צוריק דעם לעצטן ענטפער.

מען דרוקט אויס דעם ענטפער מיט (print(str(response)). אין די צווישן-טריט זעט מען אז עס ווערט געפרופן די פונקציע mystery מיט די ארגומענטן x=2 און y=9. דאס ווייזט אז דער LLM האט געפרופן דעם ריכטיגן tool, און אויך ארויסגערעכנט די ריכטיגע פאראמעטערס. דער רעזולטאט איז 121, ווייל 11 מאל 11 איז 121. מען האט באקומען דעם ריכטיגן ענטפער.

צעטיילן דעם MetaGPT-פאפיר אין חלקים און metadata (34:02:52)

דערנאך ניצט מען ווידער דעם זעלבן MetaGPT-פאפיר, און מען גיט אכטונג אויף די חלקים (chunks). מען ניצט LlamaIndex מיטן SentenceSplitter , פונקט ווי פריער, כדי צו צעטיילן דעם דאקומענט. דער chunk_size איז 1024.

יעדער node רעפרעזענטירט איין chunk. מען קוקט אויפן אינהאלט פונעם ערשטן chunk מיט:

("all"nodes[0].get_content(metadata_mode=

דאס "all"metadata_mode= זאגט אז מען וויל אויך זען די גאנצע metadata וואס איז צוגעהעפט צום אינהאלט.

אינעם רעזולטאט באקומט מען נישט נאר א פארשטעלונג פון דעם ערשטן בלאט פונעם פאפיר, נאר מען זעט אויך די metadata גאנץ אויבן. די metadata נעמט אריין:

- page_label = 1
- דער נאמען פון דער טעקע
- דער path פון דער טעקע
- דער טיפ PDF
- די גרייס פון דער טעקע
- די דאטומס ווען די טעקע איז באשאפן געווארן און ווען זי איז געענדערט געווארן

מען גיט ספעציעלע אכטונג אויף page_label. ווען מען פרובירט אן אנדער node, באקומט מען אן אנדער בלאט-נומער.

Metadata-פילטערס און די vector_query פונקציע (34:04:25)

מען דעפיניירט א VectorStoreIndex איבער די nodes. דאס בויט דעם RAG-אינדעקס pipeline: מען לייגט צו אן embedding פאר יעדן node, און מען באקומט צוריק א query engine.

כדי צו ווייזן ווי metadata-פילטערס ארבעטן, אימפארטירט מען MetadataFilters. מען שטעלט אן א פילטער אז page_label זאל זיין 2. דערנאך פרעגט מען: "וואס זענען עטלעכע הויפט-רעזולטאטן פון MetaGPT?"

דער ענטפער: MetaGPT דערגרייכט א נייעם state-of-the-art אין קאוד-שאפונג benchmarks, מיט 85.9% און 87.7% אין 1@Pass. דאמיט שלאגט עס איבער אנדערע באליבטע פריימווערקס, ווי AutoGPT און LangChain. דאס איז דער ריכטיגער ענטפער.

דערנאך גייט מען דורך די source nodes און מען דרוקט אויס זייער metadata. מען זעט דעם path פון דער טעקע, דעם טיפ, די גרייס, דעם דאטום וועז זי איז באשאפן געווארן, און דעם דאטום וועז זי איז לעצטנס געענדערט געווארן.

דערנאך פאקט מען דאס אריין אין א Python-פונקציע וואס הייסט vector_query. זי נעמט א פראגע (query) און א רשימה page_numbers . פאר יעדן בלאט-נומער אין דער רשימה שטעלט זי א פילטער אויף page_label. זי באשאפט א query engine איבערן vector index, און זי גיט צוריק דעם ענטפער. אזוי לאזט די פונקציע דורכפירן א vector -זוכעניש איבערן אינדעקס. דערנאך מאכט מען פון דער פונקציע א vector_query_tool.

רופן דעם tool מיטן LLM און צולייגן דעם summary tool (34:06:30)

יעצט רופט מען דעם tool מיט GPT-3.5 Turbo. מען ניצט predict_and_call אויפן vector_query_tool מיט דער זעלבער שאלה: הויפט-רעזולטאטן פון MetaGPT, אויף בלאט 2.

אינעם רעזולטאט זעט מען אז עס ווערט גערופן די פונקציע vector_query מיט די ארגומענטן ="query high" level results" און בלאט 2. וועז מען קאנטראלירט די source nodes, זעט מען אז נאר איין source node איז צוריקגעקומען, און דער page_label פון דעם node איז 2.

דערנאך באשאפט מען א SummaryIndex. מען אימפארטירט SummaryIndex פון llama_index.core און אויך QueryEngineTool. דערנאך נעמט מען דעם summary tool פונעם router-ביישפיל אין דער ערשטער לעקציע, און מען פאראייניגט אים מיטן vector tool. אזוי באקומט מען א סך געצייג.

מען רופט ווידער מיט א שאלה וועגן די פארגלייכן אין חלק 5 אויף בלאט 8. מען זעט אז דער LLM רופט דעם vector tool, מיט page_numbers גלייך 8, און ער גיט נאך אלץ דעם ריכטיגן ענטפער. דערנאך קען מען דאס באשטעטיגן דורך אויסדרוקן די source nodes.

אגענט-קאמפאנענטן, קאנטראל און רעזאנירונגס-לופ (34:08:34)

פארענדיגן די פריערדיגע דעמא: מקורות און דער summary-געצייג (34:08:34)

אין דער לעקציע ווערט פארענדיגט די פריערדיגע דעמא. מען דרוקט אויס די metadata פון די source nodes , די מקור-שטיקלעך פון וועלכע דער ענטפער איז געגומען געווארן. וועז מען לויפט עס, זעט מען די

זייטל נומער אכט), דעם file name און אלע אנדערע פרטים וואס מען דארף כדי צו קאנטראלירן page label) דעם ענטפער.

צום סוף שטעלט מען א פראגע: "וואס איז די סומע (summary) פון דעם ?paper" דער ציל איז צו באווייזן אז דער LLM קען נאך אלץ אויסקלייבן דעם summary-געצייג וועז עס איז נייטיג. עס ווערט באוויזן אז ער גיט טאקע דעם ריכטיגן ענטפער. דער קורס זאגט אז שפעטער וועט מען זען וויי אזוי מען בויט א רעזאנירונגס-לופ (reasoning loop) פאר אן אגענט.

ערשטער קאמפאנענט: דער Agent Task Orchestrator (34:09:04)

דער קורס שטעלט פאר די צוויי וויכטיגסטע קאמפאנענטן פון אן אגענט. דער ערשטער איז דער agent task orchestrator. ער ווערט פארגליכן מיט א דיריגענט פון אן ארקעסטער: ער זארגט אז יעדער טייל פון דער סיסטעם שפילט זיין ראלע אין דער ריכטיגער צייט און אין הארמאניע. אזוי ארבעט ער:

- צענטראלער קאארדינאטאר: ער פירט אן מיטן גאנצן גאנג פון די אויפגאבעס אינעם אגענט. פונקט ווי א פראיעקט-מענעדזשער טיילט אויס ארבעט צו די מיטגלידער פון זיין גרופע, אזוי טיילט דער orchestrator אויס אויפגאבעס צו די פארשידענע טיילן פון דער סיסטעם.
- פלאנירן און באשטימען פריאריטעטן: ער באשטימט אין וועלכן סדר די אויפגאבעס זאלן געטון ווערן. מען קען זיי שטעלן אין א ריי לויט וויכטיגקייט, לויט טערמינען (deadlines), אדער לויט וואס איינס הענגט אפ אויפן אנדערן. אזוי קען מען באהאנדלען אסאך אויפגאבעס אויף אן עפעקטיוון אופן, אן פארשפעטיגונגען און אן קאנפליקטן.
- פארוואלטן די מיטלען (resource management): ער האלט אן אויג אויף די מיטלען פון דער סיסטעם, ווי פראצעסירונגס-קראפט און זכרון (memory), און טיילט זיי אויס צו די אויפגאבעס לויט וואס יעדע איינע דארף. אזוי באקומט יעדע אויפגאבע וואס זי דארף כדי אויסגעפירט צו ווערן וויי עס דארף צו זיין.
- איבערוואכן די אויפגאבעס: ער וואכט איבער ווי ווייט יעדע אויפגאבע איז געקומען, אין רעאלער צייט. ער כאפט אויף פראבלעמען אדער פארשפעטיגונגען, און ער קען אויפגאבעס איבערגעבן צו אן אנדערן טייל אדער זיי צופאסן, כדי די ארבעט זאל גיין גלאט. דאס העלפט שנעל צו געפינען ענגע ערטער (bottlenecks) און דורכפאלן.
- קאארדינאציע צווישן אונטער-סיסטעמען: אין קאמפליצירטע סיסטעמען קענען זיין פארמישט עטליכע אונטער-סיסטעמען. דער orchestrator זארגט אז די קאמוניקאציע און דער אויסטויש פון דאטא צווישן זיי זאל גיין גלאט. ער פירט אן מיט די אפהענגיקייטן צווישן אויפגאבעס, אזוי אז דער רעזולטאט פון איין אויפגאבע ווערט ריכטיג איבערגעגעבן אלס אריינגאב (input) פאר דער צווייטער.
- באהאנדלען טעותים און זיך צוריק אויפשטעלן: אויב אן אויפגאבע פאלט דורך אדער טרעפט אן א טעות, קען דער orchestrator אנהייבן פאררעכטנדע שריט. ער קען די אויפגאבע איבערפלאנירן, אדער אנזאגן אנדערע טיילן פון דער סיסטעם זיי זאלן זי איבערנעמען. אזוי בלייבט די סיסטעם פארלעסלעך און פעסט.

צווייטער קאמפאנענט: Task Reasoning and Execution (34:11:07)

דער צווייטער קאמפאנענט איז task reasoning and execution. דא קומט פאר דאס אמת'ע טראכטן און טון: ער איז פאראנטווארטליך צו מאכן באשלוסן און אויסצופירן די אקציעס וואס מען דארף כדי פארענדיגן אן אויפגאבע.

- אנאליזירן און פארשטיין די אויפגאבע: ער הייבט אן מיט צעטיילן א קאמפליצירטע אויפגאבע אויף קלענערע שריט, וואס מען קען לייכטער באהערשן. דער מאדול אנאליזירט די פאדערונגען און דעם קאנטעקסט פון דער אויפגאבע, ענלעך צו פלאנירן די שריט ווי אזוי צו לייזן א פראבלעם.
- מאכן באשלוסן: מיט איינגעבויטער לאגיק און רעזאנירונגס-אלגאריטמען באשליסט ער וואס איז דער בעסטער וועג צו גיין. דאס קען אריינרעכענען אפשאצן פארשידענע סטראטעגיעס אדער וועגן, כדי צו זען וועלכער איז מער עפעקטיוו. ער קען זיך צופאסן לויט נייע דאטא אדער לויט ענדערונגען אין דער סביבה.
- אויספירן די אקציעס: ווען דער פלאן איז גרייט, פירט דער מאדעל אויס די נייטיגע אקציעס. דאס מיינט אז ער ארבעט דירעקט מיט הארדווער, סאפטווער אדער דרויסנדיגע סיסטעמען. די אויספירונג איז דינאמיש: דער מאדול קען צופאסן זיינע אקציעס אין רעאלער צייט אויב עס קומען אויף אומגעריכטע פראבלעמען.
- א פידבעק-לופ: נאכן אויספירן די אקציעס זאמלט ער פידבעק אויף די רעזולטאטן. דעם פידבעק ניצט מען צו פארבעסערן די קומענדיגע באשלוסן און ווי אזוי מען באהאנדלט אויפגאבעס. אן אויפהער לערנען פון הצלחות און דורכפאלן איז דא א הויפט-אייגנשאפט.
- צוזאמענארבעט מיטן orchestrator: דער orchestrator טיילט אויס און וואכט איבער די אויפגאבעס, און דער reasoning and execution קאמפאנענט טוט די אמת'ע ארבעט. די צוויי ארבעטן האנט ביי האנט: דער orchestrator גיט די אויפגאבעס, און דער רעזאנירונגס-מאדול באארבעט און פארענדיגט זיי. אזוי ארבעט דער אגענט סיי קלוג, דורכן רעזאנירן, סיי עפעקטיוו, דורך קאארדינירטער אויספירונג.

דער קורס פאסט צוזאמען: די צוויי קאמפאנענטן צוזאמען מאכן מעגלעך אז אן אגענט זאל אויספירן קאמפליצירטע אויפגאבעס אויף א פארלעסלעכן און קלוגן אופן. דער orchestrator זארגט אז אלעס גייט גלאט, און דער reasoning and execution מאדעל לייזט די פראבלעמען און טוט די אקציעס. אזוי איז דער אגענט סיי ארגאניזירט, סיי קלוג.

וואס מיינט קאנטראלירן אן אגענט, און פארוואס עס איז וויכטיג (34:13:40)

עס ווערט ערקלערט וואס עס מיינט צו קאנטראלירן אן אגענט:

- פירן דאס אויספירן: באווייזן און אנפירן ווי אזוי אן אגענט, למשל א קלוגער ראבאט אדער א סאפטווער-פראגראם, פירט זיך אויף אין פארשידענע מצבים.
- שטעלן צילן: מען שטעלט אויף קלארע צילן און כללים וואס דער אגענט זאל נאכפאלגן.
- איבערוואכן די אקציעס: מען האלט אן אויג אויף וואס דער אגענט טוט, כדי ער זאל בלייבן אויפן ריכטיגן וועג.

פארוואס איז דאס אזוי וויכטיג?

- זיכערקייט: עס פארמיידט אז דער אגענט זאל טון שעדלעכע אדער אומגעוואונטשענע אקציעס.
- פארלעסלעכקייט: עס זארגט אז דער אגענט זאל שטענדיג ריכטיג אויספירן זיין אויפגאבע.
- עפעקטיווקייט: עס העלפט דעם אגענט זיך צופאסן און פארבעסערן, דורך פידבעק און צופאסונגען.
- אנפאסונג צום ציל (alignment): מען זארגט אז די אקציעס פון אגענט זאלן שטימען מיטן געוואונטשענעם ציל אדער אויפגאבע.

דער פראצעס פון קאנטראלירן אן אגענט איז קריטיש כדי צו האלטן סדר, זיכערקייט און עפעקטיווקייט אין סיסטעמען וואס ארבעטן אויף זייער אייגענעם קאפ.

א פולער רעזאנירונגס-לופ מיט Function Calling Agent (34:14:41)

יעצט ווערט דאס אלעס געוויזן אין דער פראקטיק. ביז יעצט האט מען יעדע פראגע (query) באהאנדלט אין איין אייניציגן גאנג פאראויס (single forward pass): מען האט באקומען די פראגע, אנגערופן דעם ריכטיגן געצייג מיט די ריכטיגע פאראמעטערס, און צוריקבאקומען דעם ענטפער. אבער וואס טוט מען אויב דער באניצער פרעגט א קאמפליצירטע פראגע וואס באשטייט פון עטליכע שריט, אדער א פראגע וואס פאדערט א ברייטן ענטפער? אין דער לעקציע לערנט מען זיך דעפינירן א פולן רעזאנירונגס-לופ פאר אן אגענט. אנשטאט אנרופן א געצייג מיט איין שאס, קען דער אגענט רעזאנירן איבער געצייג און איבער עטליכע שריט. מען ניצט דעם function calling agent, אן אגענט וואס איז נאטירלעך פארבונדן מיט די function calling מעגלעכקייטן פון גרויסע שפראך-מאדעלן.

די שריט:

- מען שטעלט איין דעם OpenAI API key, און דערנאך אימפארטירט מען nest_asyncio און לויפט עס.
- מען ניצט ווייטער דעם MetaGPT paper, ווי אין די צוויי פריערדיגע לעקציעס.
- מען שטעלט איין דעם מאדעל GPT-3.5 Turbo מיט temperature = 0.
- פון LlamaIndex אימפארטירט מען FunctionCallingAgentWorker און AgentRunner . דעם FunctionCallingAgentWorker גיט מען צוויי געצייג: דעם vector tool און דעם summary tool. מען גיט אים אויך דעם LLM, און מען שטעלט verbose=True כדי צו זען די צווישן-רעזולטאטן.

די ערשטע פראגע: "דערצייל מיר וועגן די ראלעס פון די אגענטן אין MetaGPT, און ווי אזוי זיי קאמוניצירן איינער מיטן אנדערן." אין רעזולטאט זעט מען אז דער אגענט צעטיילט די גאנצע פראגע אויף שריט. דער ערשטער טייל פרעגט וועגן די אגענט-ראלעס אין MetaGPT, און דערויף רופט ער אן דעם summary tool. דערנאך דרוקט מען אויס דעם אינהאלט פונעם ערשטן source node. מען שטעלט דעם metadata mode אויף "all", ווייל מען וויל די metadata פון יעדן ענטפער. אזוי קען מען למשל באקומען דעם אינהאלט פונעם ערשטן source node וואס איז ארויסגעזוכט געווארן.

א שמועס מיט געשיכטע: פאלגנדע פראגעס (34:17:16)

דערנאך ניצט מען אנשטאט query דעם agent.chat. די פראגע איז וועגן די evaluation דאטא-סעטס וואס מען האט גענוצט. מען זעט אז דער אגענט לייגט צו צום באניצער-זכרון (user memory) די מעסעדזש "דערצייל מיר וועגן די evaluation דאטא-סעטס וואס מען האט גענוצט". ער רופט אן די פונקציע summary_tool_metagpt מיטן אריינגאב (input), און דער רעזולטאט פון דער פונקציע איז: די evaluation דאטא-סעטס אין MetaGPT רעכענען אריין HumanEval, MBPP און software development. דער ענטפער פונעם LLM איז די זעלבע זאך, און ער איז ריכטיג.

דערנאך פרעגט מען א פאלגנדע פראגע: וואס זענען די רעזולטאטן אויף די אויבנדערמאנטע דאטא-סעטס. דאס איז א ביישפיל פון דער פעאיקייט צו האלטן די געשיכטע פונעם שמועס, ווייל די פראגע באציט זיך אויף דער פריערדיגער. מען זעט באזונדער דעם "calling function", דעם "function output" און דעם LLM response. דער קורס באמערקט אז דער function output און דער LLM response דארפן זיין די זעלבע.

שריט נאך שריט מיטן low-level API (34:18:49)

יעצט דעפינירט מען ווידער דעם אגענט מיט FunctionCallingAgentWorker און AgentRunner, און מען גיט דעם agent worker אריין אין AgentRunner. דאס מאל ניצט מען דעם low-level API : מען שאפט א

task אביעקט מיט דער באניצער-פראגע, און מען גייט דורך די שריט איינס נאכן אנדערן, מיט דער מעגלעכקייט אפילו אריינצולייגן אן אייגענעם אריינגאב.

מען נעמט ווידער די זעלבע פראגע: "דערצייל מיר ווען די ראלעס פון די אגענטן אין MetaGPT, און ווי אזוי זיי קאמוניצירן איינער מיטן אנדערן." דערנאך לויפט מען בלויז איין שריט פון דער אויפגאבע:

agent.run_step(task.task_id)

אין רעזולטאט לייגט דער אגענט צו די מעסעדזש צום user memory , און רופט אן summary_tool_metagpt מיטן אריינגאב "agent roles in MetaGPT". דער function output איז אז די אגענט-ראלעס אין MetaGPT רעכענען אריין דעם product manager און אזוי ווייטער, און דער ענטפער איז ריכטיג. מען זעט אז דער אגענט האט אויסגעפירט נאר דעם ערשטן טייל פון דער פראגע, און דערנאך איז ער אפגעשטאנען.

- מיט get_completed_steps(task.task_id) באקוקט מען די לאגס, און מען זעט אז דער ערשטער טייל איז טאקע אויסגעפירט געווארן. מען דרוקט אויס וויפיל שריט זענען שוין פארענדיגט.
- מיט get_upcoming_steps(task.task_id) זעט מען וויפיל שריט זענען נאך פאראויס.

דערנאך לויפט מען דעם קומענדיגן שריט, מיט דער מעגלעכקייט אריינצוגעבן אן אייגענעם באניצער-אריינגאב. מען זעט אז עס איז צוגעלייגט געווארן צום user memory, און דער קומענדיגער אנרוף איז וועגן ווי אזוי אגענטן טיילן זיך מיט אינפארמאציע. דער אריינגאב איז "how agents share information in MetaGPT", און דער function output איז: אגענטן אין MetaGPT טיילן זיך מיט אינפארמאציע דורך א סטרוקטורירטן קאמוניקאציע-פראטאקאל. ביים לעצטן שריט באקומט מען דירעקט דעם LLM response מיט דעם זעלבן אינהאלט. די הויפט-ארבעט איז דערמיט פארטיג, און מען דארף נאר נאך איין לעצטן שריט כדי צוזאמענצושטעלן דעם ענטפער. צום סוף רופט מען אן finalize_response מיט דעם task ID און מען דרוקט אויס דעם ענטפער.

אן אגענט איבער עטליכע דאקומענטן (34:22:27)

אין דער קומענדיגער לעקציע ווייזט מען ווי אזוי צו בויען אן אגענט איבער עטליכע דאקומענטן. מען שטעלט איין דעם OpenAI API key און אימפארטירט nest_asyncio . די ערשטע אויפגאבע איז אויפצושטעלן א function calling agent איבער דריי papers. מען טוט דאס אזוי: מען נעמט דעם vector tool און דעם summary tool פון יעדן דאקומענט, מען לייגט זיי צוזאמען אין א ליסטע, און מען גיט די ליסטע צום אגענט. אזוי האט דער אגענט אינגאנצן זעקס געצייג.

מען האט אראפגעלאדן דריי papers: MetaGPT (דאס לעצטע מאל), LongLoRA און Self-RAG. דערנאך מאכט מען יעדן paper פאר א געצייג, מיט דער הילף-פונקציע get_doc_tools , וואס בויט אויטאמאטיש א vector index tool און א summary index tool איבער יעדן paper.

א מולטי-דאקומענט אגענט און דער סוף פון RAG-טייל (34:23:30)

דריי ארטיקלען, זעקס געצייג און דער ערשטער אגענט (34:23:30)

דער אפשניט הייבט אן אינמיטן א דעמא פון א מולטי-דאקומענט אגענט מיט llama-index. פאר יעדן וויסנשאפטלעכן ארטיקל (paper) ווערן געשאפן צוויי געצייג. דער vector tool מאכט א וועקטאר-זוך אינעם

ארטיקל. דער summary tool מאכט א סיכום פונעם גאנצן דאקומענט. מען זעט אז דער קאוד האט געשאפן די געצייג פאר אלע דריי ארטיקלען.

דערנאך ווערן אלע געצייג צוזאמגעלייגט אין איין פלאכע ליסטע.

שפעטער ווערט דעפינירט דער LLM: מען נעמט GPT-3.5 turbo פון OpenAI. עס ווערט אבער באטאנט אז מען קען נעמען וועלכן LLM מען וויל.

דערנאך ווערט געטשעקט וויפיל געצייג וועלן געגעבן ווערן צום אגענט. מען לויפט len אויף דער ליסטע initial tools, און דער ענטפער איז זעקס. דאס שטימט: עס זענען דא דריי ארטיקלען, און יעדער האט צוויי געצייג, א וועקטאר-געצייג און א סיכום-געצייג.

דער אגענט ענטפערט איבער Self-RAG און LongLoRA (34:24:34)

דער נעקסטער שריט איז צו בויען דעם גאנצן agent worker. ער באקומט אלע זעקס געצייג און דעם LLM. מיט אים קען מען שטעלן פראגעס איבער אלע דאקומענטן צוזאמען, אדער נאר איבער איין דאקומענט.

די ערשטע פראגע איז: "גיב מיר א סיכום פון ביידע, Self-RAG און LongLoRA". אינעם לאג זעט מען וויאזוי דער אגענט ארבעט שריט ביי שריט:

- ער לייגט צו דעם באניצער'ס מעסעדזש צום זכרון (memory).
- ער רופט א פונקציע פאר Self-RAG און באקומט איר רעזולטאט.
- ער רופט א פונקציע פאר LongLoRA און באקומט איר רעזולטאט.

אינעם ענטפער שטייט, צווישן אנדערע זאכן, איינער פון די עוואלואציע-דאטא-סעטן וואס זענען געוצט געווארן, און מען זעט אויך די עוואלואציע-רעזולטאטן פון די LongLoRA מאדעלן. צום סוף דרוקט מען אויס דעם ענטפער מיט ((print(str(response. מען באקומט די סיכומים פון ביידע ארטיקלען.

עלף ארטיקלען: URLs און טעקע-נעמען (34:25:36)

יעצט גייט דער קורס איבער צו א גרעסערן ביישפיל. דער קאוד דעפינירט צוויי ליסטעס:

- urls: בערך עלף לינקס צו וויסנשאפטלעכע ארטיקלען אין PDF-פארמאט, פון דעם OpenReview וועבזייטל.
- papers: די פאסיגע טעקע-נעמען, וואס שטעלן מסתמא פאר די אראפגעלאדענע ווערסיעס פון די ארטיקלען.

יעדער URL אין דער ערשטער ליסטע פאסט צו א געוויסן טעקע-נאמען אין דער צווייטער. דאס ווייזט אז די נעמען ווערן מסתמא געוונטן ווען מען שפייכלערט די אראפגעלאדענע טעקעס. עס ווערט ערקלערט אז אזא אויפשטעל איז ניצלעך צו אויטאמאטיזירן דאס אראפלאדן, צו האלטן חשבון פון די טעקעס, און צו רופן יעדן ארטיקל מיטן נאמען וואס מען האט אים געגעבן.

געצייג פאר יעדן ארטיקל מיט get_doc_tools (34:26:39)

דער נעקסטער קאוד-בלאק באארבעט די ליסטע ארטיקלען און ציט ארויס די געצייג פאר יעדן דאקומענט. ער ארבעט אזוי:

• ער אימפארטירט get_doc_tools פון דעם utils מאדול, און Path פון pathlib כדי צו באהאנדלען טעקע-נעמען.

• ער שאפט א ליידיגן דיקשאנערי, paper_to_tools_dict. דארט וועט מען אפהיטן וועלכע געצייג געהערן צו יעדן ארטיקל.

• ער גייט דורך די papers ליסטע, און ביי יעדן ארטיקל דרוקט ער אויס א מעסעדזש וועלכן ארטיקל מען באארבעט יעצט.

• ער רופט get_doc_tools און גיט איר דעם טעקע-נאמען, און אויך דעם נאמען אן דעם ".pdf" סוף.

• די פונקציע גיט צוריק צוויי געצייג. איינס איז א vector tool , וואס פארוואנדלט מסתמא דעם טעקסט אין embeddings פאר NLP-ארבעט. דאס צווייטע איז א summary tool, וואס מאכט סיכומים פונעם ארטיקל.

• די געצייג וועלכע ווערן אריינגעלייגט אינעם דיקשאנערי. דער טעקע-נאמען איז דער שליסל (key), און די געצייג זענען דער ווערט (value).

דער רעזולטאט איז אז מען קען גרינג און שנעל צוקומען צו די וועקטאריזירטע פארמען און צו די סיכומים פון יעדן ארטיקל. מען לויפט דעם קאוד, און עס ווערן געשאפן אלע אויסגאבעס.

אלע געצייג אין איין ליסטע און אן אינדעקס פון געצייג (34:28:12)

דערנאך שאפט דער קאוד א ליסטע מיטן נאמען all_tools . זי זאמלט צוזאם אלע געצייג פון paper_to_tools_dict פאר יעדן ארטיקל אין דער papers ליסטע. מען ניצט א nested list comprehension. דאס הייסט, מען גייט דורך יעדן ארטיקל, נעמט ארויס זיינע צוויי געצייג פונעם דיקשאנערי, און לייגט יעדעס געצייג צו all_tools. דער סוף איז אז all_tools ווערט איין ליסטע מיט אלע וועקטאר- און סיכום-געצייג פון אלע ארטיקלען, אנשטאט אז זיי זאלן בלייבן צעטיילט לויט ארטיקל. אזוי איז גרינגער צו באהאנדלען אלע געצייג צוזאמען.

נאכדעם שאפט דער קאוד אן אינדעקס פון אביעקטן (object index) און א retriever פאר די געצייג אין all_tools. ער ניצט VectorStoreIndex פון דער llama_index.core ביבליאטעק. די פונקציע ObjectIndex.from_objects נעמט אלע געצייג און ארגאניזירט זיי אין אן אינדעקס וואס מען קען זוכן אין אים. דער ארגומענט index_cls=VectorStoreIndex זאגט אז דער אינדעקס זאל ניצן וועקטאר-באזירטע זוכעניש. דאס הייסט, יעדעס געצייג ווערט פארוואנדלט אין א וועקטאר, כדי מען זאל קענען זוכן שנעל. אזוי קען מען שנעל געפינען די פאסיגע געצייג לויט ענלעכקייט, למשל וועג מען וויל געפינען געצייג וואס פאסן צו דעם באניצער'ס פראגע.

דער retriever מיט similarity_top_k=3 (34:29:42)

די שורה

obj_index.as_retriever(similarity_top_k=3) = object_retriever

שאפט א retriever-אביעקט פונעם אינדעקס. די מעטאד as_retriever פארוואנדלט דעם אינדעקס אין א retriever, וואס קען ברענגען רעזולטאטן לויט ענלעכקייט. דער פאראמעטער similarity_top_k=3 מאכט זיכער אז ביי יעדער פראגע וועט דער retriever צוריקגעבן די דריי רעזולטאטן וואס זענען די מערסטע ענלעך.

עס ווערט ערקלערט אז דאס איז ניצלעך ביי זוך-מאשינען, טשעט-באטס און רעקאמענדאציע-סיסטעמען, וואו עס איז זייער וויכטיג צו ברענגען די מערסטע פאסיגע אינפארמאציע.

דערנאך ניצט מען object_retriever.retrieve(...) = tools מיט א פראגע. דער retriever זוכט דורך די אינדעקסירטע דאטא און ברענגט די מערסטע פאסיגע רעזולטאטן, ביז דריי. די רעזולטאטן ווערן אפגעהיט אינעם variable tools. מען קען זיי ניצן ווייטער, למשל צו וועיזן דעם באניצער אדער צו געבן פאר אן AI מאדעל. אזא פראצעס ווערט אפט גענוצט אין AI-זוך-סיסטעמען, טשעט-באטס און אפליקאציעס וואס ענטפערן אויף פראגעס.

דערנאך קוקט מען אויף די מעטאדאטא פונעם דריטן רעזולטאט: tools[2].metadata . יעדער עלעמענט אין tools האט סיי דעם אינהאלט וואס מען האט געבראכט, סיי צוגעלייגטע מעטאדאטא. מעטאדאטא כולל געוועליך עקסטערע אינפארמאציע, ווי די קוועל, די צייט, דער מחבר, דער ID פונעם דאקומענט, אדער אנדערע אייגנשאפטן.

דער אגענט וואס ניצט דעם retriever (34:32:16)

יעצט שטעלט מען אויף אן AI-אגענט וואס ענטפערט לויט די אינפארמאציע וואס מען ברענגט פון דער זאמלונג ארטיקלען:

- מען אימפארטירט FunctionCallingAgentWorker און AgentRunner פון llama_index.core.agent.
- מען שאפט אן agent worker מיט FunctionCallingAgentWorker.from_tools . מען גיט אים דעם object retriever, וואס זוכט פאסיגע געצייג, און דעם LLM, וואס באארבעט און שרייבט די ענטפערס.
- א system prompt הייסט דעם אגענט זיך פארלאזן אויף די געבראכטע דאטא און נישט אויף זיין פריערדיגע וויסן.
- verbose=True גיט א פרטימדיגן לאג. דאס העלפט צו געפינען פעלערן, און צו זען וויאזוי דער אגענט ברענגט און באארבעט די דאטא.
- צום סוף נעמט agent = AgentRunner(agent_worker) דעם worker אריין אין א runner. אזוי איז גרינגער צו פירן און לויפן די אויפגאבעס פונעם אגענט.

עס ווערט געזאגט אז אזא אויפשטעל איז ניצלעך פאר פארשונג, אנאליז פון דאקומענטן, און אפליקאציעס וואס ענטפערן אויף פראגעס.

צוויי פראגעס: MetaGPT קעגן SWE-bench, און LongLoRA קעגן LoftQ (34:32:56)

די ערשטע פראגע פרעגט וועלכן עוואלואציע-דאטא-סעט מען האט געוונצט אין MetaGPT, און וויאזוי ער קומט אויס אין פארגלייך מיט SWE-bench. די פונקציע query שיקט די פראגע צום אגענט. דער אגענט ברענגט פאסיגע אינפארמאציע דורכן object retriever און באארבעט דעם ענטפער מיטן LLM. דער ענטפער ווערט אפגעהיט אין response, און מען דרוקט אים אויס מיט str. אזוי באקומט מען א גוט ארגאניזירטן ענטפער, באזירט אויף די דאקומענטן. מען זעט ווי דער אגענט הייבט אן שרייבן דעם ענטפער.

דער לעצטער שריט בעט דעם אגענט צו אנאליזירן און פארגלייכן צוויי ארטיקלען, LongLoRA און LoftQ. די פראגע איז פרטימדיג. זי בעט אז דער אגענט זאל ערשט אנאליזירן דעם צוגאנג פון יעדן ארטיקל באזונדער, און דערנאך זיי פארגלייכן און וועיזן די אונטערשיידן. דער ענטפער ווערט אפגעהיט אין response, און מען קען

אים אויסדרוקן אדער ניצן ווייטער. עס ווערט באטאנט אז אזוי קען מען יעפעקטיוו ארויסנעמען תוכן פון ארטיקלען. דאס איז ניצלעך פאר אקאדעמישע פארשונג, טעכנישע אנאליז, און סיכומים וואס AI מאכט.

סיכום פונעם Agentic RAG טייל (34:35:00)

דער קורס פארענדיקט דעם טייל איבער בויען agentic RAG אפליקאציעס, און גייט נאכאמאל איבער וואס מען האט געלערנט:

- אן איינפירונג: פארוואס agentic reasoning איז וויכטיג אין RAG, און וויאזוי עס העלפט AI באהאנדלען פארשידענע סארטן פראגעס.
- דער router query engine, וואס פירט יעדע פראגע צום ריכטיגן מקור, כדי די זוכעניש זאל זיין מער יעפעקטיוו.
- tool calling, וואס לאזט AI-אגענטן ארבעטן מיט דרויסנדיגע געצייג. אזוי קענען זיי באשליסן מער ווי נאר שרייבן טעקסט.
- אן agent reasoning loop: א סיסטעם וואס באארבעט אינפארמאציע אין עטלעכע רונדן, און פארבעסערט דעם ענטפער לויט נייע אינפארמאציע.
- א multi-document agent, וואס ציט אינפארמאציע פון עטלעכע מקורות און שטעלט זי צוזאמען. אזוי באקומט מען מער פולשטענדיגע און פינקטליכע רעזולטאטן.

עס ווערט געזאגט אז מען קען agentic RAG ניצן פאר א סך זאכן: פערזענליכע AI-געהילפן, אויטאמאטישע פארשונג-געצייג, און סיסטעמען וואס באשליסן לויט דעם קאנטעקסט. וועז AI אנטוויקלט זיך ווייטער, וועט די צוזאמשטעל פון שכל-ארבעט (reasoning), זוכעניש און ארבעט מיט געצייג זיין דער שליסל צו מער פעאיגע און זעלבשטענדיגע סיסטעמען. דער באניצער ווערט געמוטיגט צו פרובירן בויען אייגענע agentic RAG לייזונגען.

די פיר טיילן פון אן AI-אגענט (34:36:32)

דערנאך הייבט אן א נייער טייל. ער ערקלערט וויאזוי אן אגענט ארבעט, פון וועז ער באקומט אינפארמאציע, ביז ער טוט די ארבעט און געדענקט וואס איז געשען. שריט ביי שריט:

- Perception, דאס אריינברענגען: דער אגענט באקומט אינפארמאציע פונעם באניצער דורך א טשעט, א פארם, אדער א webhook. ער הערט וואס דער באניצער זאגט אדער טוט. דא הייבט זיך אן דער גאנצער פראצעס.
- Decision, דער מוח: נאכן באקומען די אינפארמאציע באארבעט דער אגענט די דאטא און באשליסט וואס צו טון. די באשלוסן פירן אים דורך די וויטערדיגע שריט.
- Action, די טואונג: דער אגענט ניצט געצייג צו טון די ארבעט. למשל, ער זוכט דורך APIs, ארבעט מיט שיטס (sheets), שיקט אימעילס, אדער ניצט אנדערע געצייג. דער עיקר איז אז די ארבעט ווערט געטון.
- Memory, דער זכרון: דא ווערט אלעס אפגעהיט. פונקט ווי א מענטש, דארף דער אגענט געדענקען דעם קאנטעקסט פון פריערדיגע טואונגען, כדי ער זאל קענען וויטער פירן די ארבעט. דער זכרון לאזט אים מיט דער צייט ארבעטן בעסער, ווייל ער געדענקט פריערדיגע אויפגאבעס, באשלוסן און רעזולטאטן, און דארף נישט יעדעס מאל אנהייבן פון אנפאנג.

וועו מען קוקט נענטער אויף דעם באניצער: צי דורך טשעט, צי דורך פארם, צי דורך webhook, הערט דער אגענט וואס דער באניצער בעט. די אינפארמאציע שטעלט דעם אגענט אין גאנג. דעם באשליס-טייל טרייבן LLMs ווי Gemini אדער ChatGPT. די מאדעלן אנאליזירן וואס דער באניצער האט געשריבן און באשליסן וואס איז דער בעסטער וועג: צו זאמלען דאטא, צו טון אן אויפגאבע, אדער צו ענטפערן אויף א פראגע. Gemini, למשל, פירט דעם AI וואס צו טון ווייטער. וועו דער באשלוס איז געמאכט, נעמט זיך דער אגענט צו דער ארבעט מיט search APIs, Google Sheets אדער אימעילס, און טוט וואס עס פאדערט זיך.

א פארשונג-אגענט מיט N8N: יסודות און אנהייב (34:39:06)

די פיר טיילן פון אן AI אגענט, און דער פארשונג-אגענט פראיעקט (34:39:06)

דער קורס ענדיגט די ערקלערונג פון די פיר טיילן פון אן AI אגענט. אינעם "אקציע" טייל טוט דער אגענט אויס דאס וואס איז פריער באשלאסן געווארן. דא טוט דער AI די אמת'דיגע ארבעט, פון ארויסנעמען דאטא ביז אויטאמאטיזירן ארבעטן.

דער לעצטער טייל איז דער זכרון. דער אגענט געדענקט וואס ער האט פריער געטון און היט אפ די רעזולטאטן. דאס איז קריטיש, ווייל אזוי האט דער אגענט א קאנטעקסט פאר די ווייטערדיגע אינטעראקציעס. וועו ער היט אפ אינפארמאציע, לערנט זיך דער AI פון פריערדיגע איבערלעבונגן און וועט מיט דער צייט מער עפעקטיוו און ערפאלגרייך.

דערנאך הייבט מען אן א פראיעקט פון א פארשונג-אגענט, וואס פירט דורך פארשונג מיט AI. ער האט די זעלבע סטרוקטור פון פיר טיילן. מען זעט ווי יעדער טייל ארבעט צוואמען מיט די אנדערע כדי צו זאמלען און אנאליזירן דאטא:

- פערצעפציע (אויפנעמען): דער אגענט באקומט זיין אינפוט. אין דעם פאל זאמלט ער די לעצטע AI מארקעטינג-טרענדס פון 2025. זיי קענען קומען פון נייעס-ארטיקלען, בלאגס אדער סאשעל-מעדיע קאנאלן. דער אינפוט גרייט צו די בינע פארן אגענט'ס פארשונג.
- באשלוס: דער אגענט באשליסט ווי אזוי צו גיין ווייטער מיט דער געזאמלטער אינפארמאציע. דא זוכט ער "לייוו" אויף Google און טרעפט מער פאסיגע און דערהיינטיגטע מקורות וועגן AI מארקעטינג-טרענדס. זיין ארבעט איז צו פראצעסירן דעם אינפוט און אויסגעפינען וואו צו זוכן די ריכטיגע דאטא.
- אקציע: דא נעמען איבער די Gemini מאדעלן. Gemini אנאליזירט און פאסט צוואם די גאנצע אינפארמאציע פון דער זוכעניש, און מאכט דערפון פארשטענדליכע אינזיכטן וואס מען קען אויספירן. עס איז ווי מען וואלט געהאט פארשונג-געהילפן וואס לייענען, פארשטייען און פאסן צוואם ארטיקלען אין רעאלער צייט.
- זכרון: דער אגענט היט אפ די רעזולטאטן אין א Google Sheet. דארט שטייט נישט נאר א צוואמפאסונג פון דער פארשונג, נאר אויך אינזיכטן און רעקאמענדאציעס לויט די טרענדס. דער זכרון העלפט דעם אגענט נאכפאלגן זיין ארבעט, כדי מען זאל עס קענען נוצן שפעטער און עס טיילן מיט אנדערע.

דורך די פיר שריט קען דער אגענט ערפאלגרייך זאמלן, אנאליזירן און אפהיטן פאסיגע דאטא. אזוי קען מען אויטאמאטיזירן פארשונג-ארבעט, שפארן צייט, און דאך בלייבן אויבן אויף די לעצטע טרענדס.

די וויכטיגע געצייג פונעם פראיעקט (34:41:07)

דער קורס צייגט די געצייג וואס טרייבן דעם פראיעקט, פון אויטאמאציע ביז אנאליז און אפהיטן:

- N8N: אן אפן-קוועל געצייג פאר אויטאמאציע, וואס לאזט פארבינדן פארשידענע סערוויסעס און אויטאמאטיזירן ארבעטס-געגנ. ער פירט דעם גאנצן פראצעס: ער אויטאמאטיזירט דאס זוכן פאסיגע דאטא, רעדט מיט API'ס און ארגאניזירט ארבעטן. ער איז דער רוקנביין פונעם פראצעס.
- Google Gemini: דער AI מאדעל וואס אנאליזירט און פאסט צוזאם די פארשונג-דאטא. ער פארשטייט קאמפליצירטע דאטא און גיט קלארע אינזיכטן וואס מען קען אויספירן. מען קען אים פארגלייכן מיטן "מוח" הינטער דער פארשונג, וואס באשליסט אויף וואס זיך צו קאנצענטרירן און ווי אזוי צוזאמצופאסן די רעזולטאטן.
- SerpAPI: א שטארקער געצייג וואס "סקרייפט" (ציט ארויס) רעזולטאטן פון זוך-מאשינען. ער ברענגט די מערסט דערהיינטיגטע דאטא גלייך פון Google זוך-רעזולטאטן. אינעם פראיעקט העלפט ער דעם אגענט טון לייוו זוכענישן נאך AI מארקעטינג-טרענדס, כדי צו באקומען די לעצטע אינפארמאציע פון פארלעסליכע מקורות.
- Google Sheets: נאכן פראצעסירן און אנאליזירן ווערט אלעס אפגעהיט אין א Google Sheet, מיט א דעטאלירטע צוזאמפאסונג, אינזיכטן און רעקאמענדאציעס. ער דינט ווי א צענטראלער פלאץ צו האלטן און טיילן די רעזולטאטן, אזוי אז מיטגלידער פון א טים קענען זיי לייכט זען און אויספירן.

צוזאמען שאפן די געצייג א גלאטן און עפעקטיוון ארבעטס-גאנג פאר זאמלען און פראצעסירן אינפארמאציע אין רעאלער צייט.

אנהייבן מיט N8N, און די פרייזן (34:43:09)

מען הייבט אן צו בויען דעם אגענט. מען זוכט "N8N" אויף Google און עפנט דעם ערשטן לינק. אויפן ערשטן בילד זעט מען א GitHub אינטעגראציע, א מעגליכקייט זיך איינצושרייבן, א "Get started" קנעפל, און א ערקלערונג ווי N8N ארבעט אין דעם פעלד פון אויטאמאציע. עס איז נוצליך פאר IT ops, SecOps און DevOps, פאר פארקויף, און אויך פאר אויטאמאטיזירן קליינע טאג-טעגליכע ארבעטן.

ווער עס האט נאך נישט קיין קאנטע, קליקט "Get started" און גיט אריין: נאמען, פירמע-נאמען (מען קען אויך געבן אן אויסגעטראכטן), אימעיל, פאסווארט, און א נאמען פאר דער קאנטע. מען לאגט זיך איין דורך app.n8n.cloud און הייבט אן א פרייע פראבע-צייט פון 14 טעג. ווער עס ווייל אז די אויטאמאציעס זאלן לויפן לענגער, חדשים אדער יארן, קען נעמען איינעם פון די פלענער. עס זענען דא פיר פלענער (די פרייזן ווערן געוויזן אויף א יערליכער באזע, אבער מען קען טוישן צו א חודש'ליכע רעכענונג):

- Starter: €20. אריינגערעכנט: א געטיילטער פראיעקט, 5 עקזעקוציעס אויף אמאל, אן א שיעור באניצער, 50 קרעדיטן פארן AI workflow builder, און הילף דורכן פארום.
- Pro: €50. דריי געטיילטע פראיעקטן, 20 עקזעקוציעס אויף אמאל, 7 טעג "אינסייטס", 150 AI workflow builder קרעדיטן, אדמין-ראלעס, גלאבאלע וואריאבלען, ארבעטס-גאנג היסטאריע, און זוכן אין עקזעקוציעס.
- Business: בערך €667, פאר פירמעס ביז 100 ארבעטער. אלעס פון Pro, און דערצו 6 און מער געטיילטע פראיעקטן, SSO, SAML און LDAP, 30 טעג אינסייטס, פארשידענע סביבות (environments), מעגליכקייטן צו וואקסן (scaling), ווערסיע-קאנטראל מיט Git, און הילף דורכן פארום.

• Enterprise: פאר מער ווי 100 מענטשן. מען קאנטאקטירט די פירמע. מען באקומט אן א שיעור געטיילטע פראיעקטן, וואס דער קורס האלט פאר דער בעסטער ברירה פאר פירמעס מיט 300 ביז 400 ארבעטער. דערצו: 200 און מער עקזעקוציעס אויף אמאל, 365 טעג אינסייטס, טויזנט AI workflow builder קרעדיטן, אינטעגראציע מיט א דרויסנדיגן סוד-שפייכלער, log streaming, לענגערע אפהאלטונג פון דאטא, א באזונדערע הילף מיט SLA, און באצאלן דורך אינוואיס.

די עצה פונעם קורס: אנהייבן מיטן פרייען 14-טאגיגן פראבע, דערמיט עקספערימענטירן, און שפעטער נעמען דעם Starter. פאר ביזנעס אדער גרויסע פירמעס זענען דא די אנדערע פלענער.

דער הויפט-קאנסאל פון N8N (34:46:48)

נאכן זיך איינלאגן זעט מען אן ענליכע בילד, נאר אן די פארטיגע ארבעטס-געגנ וואס ווערן געוויזן אין דער לעקציע. עס איז דא א קנעפל "Upgrade" צו קלייבן Starter, Pro אדער Enterprise. דערווייל קען מען בלייבן ביים פרייען פראבע.

אינעם הויפט-קאנסאל זענען דא: Workflows, Credentials, Executions, Variables און Data Tables. אויף דער לינקער זייט זעט מען אן איבערבליק פון די פראיעקטן: פערזענליכע פראיעקטן, און "Chat". ChatGPT איז אריינגעבויט אין N8N, אזוי אז מען דארף אים נישט פרעגן באזונדער, און מען קען פערזענליך צופאסן די ארבעטס-געגנ.

ווייטער זענען דא:

• אן אדמין-פאנעל.

• טעמפלעיטס (מוסטערן), למשל "Chat with uncensored Dolphin Mixtral" און "Generate SEO optimized blogs". מען קען נעמען א מוסטער אנשטאט בויען פון אנהייב.

• אינסייטס: ווי לאנג די עקזעקוציעס נעמען, און וויפיל מאל זיי דורכפאלן.

• הילף: quick start, דאקומענטאציע, פארום, קורסן, און מעלדן א פעלער.

• סעטינגס: פערזענליכע סעטינגס, באניצער, פראיעקט-ראלעס, דרויסנדיגע סודות (external secrets), סביבות, און טשאט-סעטינגס.

בויען א ארבעטס-גאנג, און די "נאודס" (34:48:20)

עס איז דא א קנעפל "Create workflow". מען קען אויך שאפן א credential, א וואריאבל אדער א דאטא-טאבעלע. אין דער לעקציע שאפט מען א ארבעטס-גאנג און גיט אים דעם נאמען "automation". ער ווערט פארשריבן אין די פערזענליכע פראיעקטן. דארט ווערט יעדער שריט פארשריבן, מיט וועלכע אקציע אדער זוכעניש ער דארף טון.

ביים צולייגן דעם ערשטן שריט זענען דא עטליכע ברירות:

• Trigger manually: מען קליקט, און די אויטאמאציע הייבט זיך אן.

• Webhook call: עס לויפט אויטאמאטיש. מען קען אויך באשטימען צייטן, למשל פון 9:00 אינדערפרי ביז 10:00 ביינאכט, לויט וואס מען דארף.

• נאך דעם קען מען צולייגן נאך: Google Drive דאקומענטן, ads, trackers.

• AI: "build automation", "summarize text". דער קורס קומט צוריק צו דעם שפעטער.

- Action in app: אקציעס אין אנדערע אפליקאציעס.

צו אינסטאלירן א "נאוד" קליקט מען "Install node" און קלייבט דעם מאדול. אלעס וואס מען דארף פאר אויטאמאציע געפינט זיך דארט, מען דארף עס נאר זוכן:

- Data transformation: באארבעטן, פילטרירן און איבערמאכן דאטא.
- Flow: צווייגן (branch), צוזאמשטעלן (merge), לופס, א.א.וו.
- Core: לויפן קאוד, שיקן HTTP פארלאנגען, און אויפשטעלן webhooks.
- Human in the loop: וועו מען וויל אז א מענטש זאל קומען און אנהייבן דעם לופ.
- Triggers.

דערצו זענען דא מעגליכקייטן צוצולייגן און זוכן, "sticky notes" צו פארשרייבן אינפארמאציע, א "focus panel", און N8N AI.

N8N AI בויט אליין א ארבעטס-גאנג (34:50:24)

לויטן קורס איז N8N AI די בעסטע מעגליכקייט אין N8N. מען דארף גארנישט וויסן וויי אזוי די אויטאמאציע דארף געטון ווערן. מען גיט אים א פראמפט, און ער בויט אליין די אויטאמאציע מיט אלע נויטיגע עלעמענטן.

אין דער לעקציע שרייבט מען "multi-agent research workflow". דער פראמפט ווערט אויסגעארבעט אויטאמאטיש: "Create a multi-agent AI workflow using GPT-4.1 mini". מען קען אים טוישן.

א ווארענונג: ביים ניצן N8N AI ווערן די קרעדיטן פארמינערט. אינעם פראבע זענען זיי זייער באגרעניצט. מיט א Pro אדער פערזענליכן פלאן האט מען מער, און מען קען ארבעטן מיט דער beta ווערסיע פון N8N AI. עס איז דא א "Ask" (טשאט) מאדע און א "Build" מאדע. דא ניצט מען Build.

מען זעט ווי ער זאמלט אלע נויטיגע עלעמענטן און שטעלט זיי אויס ריכטיג. עס איז אויך דא א שפאלט פאר logs: נאכן אנהייבן דעם ארבעטס-גאנג ווערט דארט געוויזן יעדער פעלער אדער ערקלערונג. נאכדעם וואס דער ארבעטס-גאנג איז געבויט, פילט ער אויס די נויטיגע אינפארמאציע.

א וויכטיגע עצה: פארן אנהייבן N8N פראיעקטן זאל מען פארבינדן N8N מיט GitHub. אזוי איז לייכטער צו שטופן און ציען (push/pull) דעם קאוד אדער די אויטאמאציע. דער קורס האלט עס פאר כמעט א מוז, ווייל טייל מעגליכקייטן זענען נאר צוטריטלעך דורך GitHub. מען גיט ארײן די פרטים פון דער GitHub קאנטע, און עס ווערט פארבונדן.

צום סוף זעט מען אז דער ארבעטס-גאנג איז שוין געבויט, און ער בעט עטליכע ערלויבענישן, מיט דער מעלדונג: "I have created your multi-agent AI research workflow".

n8n אויטאמאציע: ריסוירטש-אגענטן און מיטינגען (34:53:03)

דער ריסוירטש-וואורקפלאו מיט פיר אגענטן (34:53:03)

דער קורס ווייזט א פארטיגן וואורקפלאו אין n8n וואס איז געבויט מיט דער AI ביטא-ווערסיע. ער האט עטלעכע שריט. דער ערשטער איז א ריסוירטש-אגענט. ער זוכט אויפן אינטערנעט און קלייבט צוזאם פינף ביז צען הויפט-פאקטן פון דריי ביז פיר פארלעסליכע קוואלן. נאכדעם קומט א פאקט-טשעקינג אגענט, דערנאך א באריכט-שרייבן אגענט, א פארמאטירונג-אגענט, און צום סוף ווערט דער באריכט געשיקט דורך Gmail.

דער וואורקפלאו הייבט זיך אן מיט א "manual trigger". וועז מען קליקט דערויף, הייבט זיך אן דער וואורקפלאו. מען קען אויך קליקן אויף "execute workflow". ווייל מען האט נאך נישט אריינגעשטעלט קיין פרטים, ווייזט עס א פראבלעם. מען וועט דאס שפעטער פארריכטן.

דער קומענדיגער נאוד איז "workflow configuration", וואו מען שטעלט אריין דאס ריסוירטש-טעמע. צו באארבעטן א נאוד, קליקט מען אויף די דריי פינטעלעך און דערנאך אויף "open". דארט זענען שוין אלע פרטים אויסגעפילט. אויב מען האט א ספק, קען מען זיי איבערקוקן און אריינשטעלן די אייגענע פרטים. מען זעט דארט אויך דעם אויספיר (output) פון יענעם נאוד.

פאר יעדן נאוד זענען פאראן עטלעכע אפציעס:

- מען קען לויפן נאר דעם איינעם נאוד.
- מען קען אים אויסמעקן (delete).
- מען קען אים אויסשאלטן (deactivate). דאן לויפט דער גאנצער וואורקפלאו אן דעם עלעמענט, און מען קען אזוי עקספערימענטירן.

די געצייג פון יעדן אגענט (34:54:35)

דער אויספיר פון דער קאנפיגוראציע גייט צום ריסוירטש-אגענט. דארט ווערן געגעבן פיר געצייג: OpenAI, SerpAPI, אן HTTP request, און א parser. עס ווערט ערקלערט אז די OpenAI API שליסלען קאסטן געלט, אבער SerpAPI האט א פרייע ווערסיע. דער קורס זאגט צו צו ווייזן שפעטער ווי אזוי מען באקומט א SerpAPI שליסל, און וואס SerpAPI איז. דאס איז דער זוך-טייל, און דארט ווערט אריינגעשטעלט דער SerpAPI שליסל.

די זוך-רעזולטאטן גייען ווייטער צום פאקט-טשעקינג אגענט. ער באשטעטיגט יעדן פונד דורך זוכן א באשטעטיגונג אין אמווייניגסטנס צוויי נאך אומאפהענגיגע קוואלן. נאר דאן צייכנט ער אן א פאקט אלס "verified". ער קוקט אויך אויף שליסל-ווערטער און דאס גלייכן.

די רעזולטאטן גייען דערנאך צום באריכט-שרייבן אגענט. ער נוצט OpenAI מיט אן output parser און שרייבט א קלארן באריכט פון טויזנט ווערטער, נאר מיט די באשטעטיגטע פאקטן. אזוי איז מען זיכער אז נאר באשטעטיגטע פאקטן קומען אריין.

דערנאך גייט עס צום פארמאטירונג-אגענט, וואס נוצט אויך OpenAI מיט א parser. ער פארוואנדלט דעם באריכט אין א פראפעסיאנעלן HTML מיט ריינעם סטייל, צוגעפאסט פאר אן אימעיל. דער קורצער אימעיל-דראפט גייט צום Gmail אקאונט, וואס שיקט אוועק דעם ענדגילטיגן באריכט.

וועז אלע אינפארמאציע און אלע API שליסלען זענען אריינגעשטעלט, קליקט מען "execute workflow" און די גאנצע אויטאמאציע לויפט. מען דארף נישט קאפירן און אריינקלעבן יעדן פאקט אדער אים איבערקוקן. אלעס ווערט געטון אין עטלעכע מינוט. מען קען אויך באשטימען א צייט-פעריאד וועז צו שיקן די קונים ידיעות וועגן א סעיל אדער א דיעל, אלעס אויטאמאטיש.

HTTP, Gmail און די אלגעמיינע סעטינגס (34:56:37)

וועז מען נוצט די n8n AI ביטא-ווערסיע, בלייבן נאך נאכן בויען דעם וואורקפלאו עטלעכע בקשות וואס מען דארף אננעמען. למשל, דער HTTP request tool מוז מען אליין אויספילן. מען קלייבט א מעטאד, למשל

DELETE, HEAD, OPTIONS, PATCH אדער POST, לויט ווי מען וויל. מען שטעלט אריין דעם URL לינק און די אויטענטיפיקאציע, וואס איז א הויפט-טייל פון דער אויטאמאציע.

כדי צו שיקן דעם Gmail דורך אן אקאונט, דארף מען credentials. מען קליקט "add new credentials" און מען קען זיך איינשרייבן מיט Google. אין דער לעקציע ווערט דאס דערווייל נישט געטון. אונטער "operations" זעט מען אז מען קען אין Gmail אויך אויסמעקן אימעילס, באקומען אימעילס, אנצייכענען אלס געלייענט אדער נישט געלייענט, אראפנעמען א לעיבל, און נאך אסאך. מען דארף נאר פארבינדן דעם אימעיל מיטן אויטאמאציע-געצייג.

יעדער עלעמענט וואס מען אויטאמאטירט האט די זעלבע סעטינגס:

- "Always output data": געווענליך איז דאס אויסגעשאלטן, אבער עס ווערט גערעטן עס אנצושאלטן ביי יעדער אויטאמאציע.
- "Execute once": אויב מען וויל אז דער עלעמענט זאל ארבעטן נאר איין מאל.
- "Retry on fail": אויב עס דורכפאלט, פרובירט עס נאכאמאל, אזוי אז מען דארף עס נישט אליין ווידער אנצינדן.
- "On error": ביי א טעות קען מען אויסקלייבן צי ווייטער צו גיין, צו אפשטעלן דעם וואורקפלאו, אדער ווייטער צו גיין מיטן error אויספיר.
- "Notes": מען קען פארשרייבן נאטיצן וועגן דעם נאוד, און אויך באשטימען צי זיי זאלן ארויסגעוויזן ווערן.

וועו מען שטעלט אויס די צוויי זאכן, דעם HTTP און Gmail, איז די גאנצע אויטאמאציע פארטיג. עס האט געדויערט איין קליק, אן אליין בויען אן אגענט. דאס, ווערט ערקלערט, איז די כח פון דעם AI וואס איז איינגעבויט אין n8n.

א פראמפט פון GPT פאר א מיטינג-וואורקפלאו (34:58:40)

דער קורס ווייזט אן אנדער מעטאד: מען פרעגט GPT גלייך ער זאל געבן א פראמפט פאר n8n. מען בעט: "I need a prompt to generate a workflow that automates the creation of meetings". מען קען אויך נוצן Gemini אדער אן Azure API שליסל. GPT גיט א פראמפט וואס הייבט זיך אן: "Create a step-by-step automated workflow that simplifies the process of scheduling meetings".

מען קאפירט דעם פראמפט, גייט צוריק צו n8n AI, גיט אים אריין, און ווארט עטלעכע סעקונדעס. ווי פריער, שאפט עס א פלוס וואו מען דארף אויספילן עטלעכע פרטים, געבן ערלויבענישן, און אריינשטעלן API שליסלען ווען נויטיג. דערנאך איז דער וואורקפלאו פארטיג. ווען מען זאל דאס אלעס בויען מיט דער האנט, וואלט עס געדויערט אמווייניגסטנס עטלעכע שעה. מיט דער n8n AI ביטא-ווערסיע געשעט עס אין עטלעכע מינוט.

א ריסוירטש-אגענט וואס גיט אויס אן Excel: trigger און set (35:00:10)

דער קומענדיגער פראיעקט איז א ריסוירטש-אגענט, געבויט מיט AI. מען גיט א פראגע, און ער אנאליזירט אלע פרטים און גיט צוריק אין אן Excel בלאט אלע רעזולטאטן, אריינגערעכנט insights און רעקאמענדאציעס.

צו ערשט לייגט מען צו א manual trigger: מען קליקט דעם פלוס-צייכן און קלייבט "manual trigger". מען דארף געדענקען אז אין דעם manual trigger קען מען נישט טוישן קיין פרטים. ווייל מען דורכגעבן דאטא

פונעם trigger, דארף מען א set נאוד. מען זוכט "set", קלייבט "edit fields", און דערנאך א מאָד, למשל "manual mapping". מען קען צולייגן עטליכע פעלדער, מיט נאמען און ווערט, און אויך טוישן דעם array טיפ.

אין דעם ביישפיל איז דער מאָד manual mapping, און דאס פעלד הייסט "question". דארט שרייבט מען דאס ריסוירטש וואס מען וויל, למשל "top trends marketing trends 2025". דאס איז דער אריינגאנג פארן קומענדיגן טייל. דערנאך קליקט מען "execute step".

זוכן אין Google דורך SerpAPI (35:01:46)

די פראגע מוז דורכגיין א Google זוך, און דערפאר נוצט מען SerpAPI. מען זוכט "Serp" און זעט "SerpAPI official". וועלן עס איז נאך נישט אינסטאלירט, קליקט מען און עס אינסטאלירט זיך. אונטער דעם זענען פאראן אסאך נאודס, און מען דארף קלוג אויסקלייבן לויט דער אקציע:

- "search Amazon" פאר לינקס צו פראדוקטן און אינפארמאציע וועגן א פראדוקט.
- Google Flights, Finance, Lens.
- Google Play פאר שפילן און אפליקאציעס.
- זוכן אין YouTube.

דא קלייבט מען דעם פשוט'ן Google search. מען מוז שאפן אן אקאונט מיט אן API שליסל. מען גייט צו SerpAPI און מען שרייבט זיך איין מיט Google, GitHub, אדער אימעיל און פאסווארד. די ערשטע אפציע דארט איז דער API שליסל. דאס איז א פריי צוטריטליכער אקאונט. מען קען באנייען (regenerate) דעם שליסל אויב ער איז שוין געוואָרט, אדער שאפן א נייעם. מען קאפירט אים, גייט צוריק צום וואורקפלאו, שטעלט אים אריין, און היט אים אפ. מען קען אויך טוישן דעם HTTP request, אבער געווענליך איז דאס אלעס. נאכדעם ווערט דער אקאונט געהאלטן פאר יעדע מאל מען נוצט SerpAPI.

אין די סעטינגס איז דער resource "search", און די operation איז "Google search". דער query איז די פראגע פונעם פריערדיגן נאוד, אין JSON פארמאט. מען קען צולייגן א location, למשל New York אדער Delhi. מען קען אויך צולייגן נאך פעלדער, למשל א צייט-פעריאד, אויספיר-פארמאט, אדער Google place ID. דערווייל איז דאס נישט נויטיג, און מען קליקט "execute step".

נאך די זוך-רעזולטאטן ווערט צוגעלייגט נאך א set נאוד. דער טעם: מען זעט דעם אויספיר פונעם פריערדיגן נאוד, און דאס מאכט עס גרינג צו געפינען טעותים (debug). אין דער לעקציע ווערט אזא set נאוד צוגעלייגט נאך יעדן נאוד. פאר אן עקספערט אין אויטאמאציע איז דאס נישט נויטיג, אבער אנדערע וועלן עס וועלן כדי צו קענען דעבאגן.

אנאליז מיט Google Gemini (35:05:29)

די זוך-אינפארמאציע מוז דורכגיין א Gemini מאדעל פאר אנאליז. מען זוכט "Google Gemini" און קלייבט אן אקציע לויט דער ארבעט. פאר א ווידעא נוצט מען "analyze video". מען קען אויך שאפן ווידעא, אודיא, אאז"וו. דא דארף מען נאר "message a model", ווייל מען וויל בלויז אנאליזירן דאס טעמע.

פאר Gemini דארף מען אן API אקאונט:

- מען גייט צו Google AI Studio און מען שרייבט זיך איין מיט א Google אקאונט.

- מען זוכט "API keys" און קליקט "get API key".
- מען שאפט א נייעם שליסל אדער נוצט אן אלטן. מען גיט אים א נאמען און קלייבט א פראיעקט, למשל "chatbot", און מען קליקט "create key".
- מען קאפירט דעם שליסל, גייט צוריק צו n8n, און קליקט "create new credentials". דעם host לאזט מען ווי ער איז, מען שטעלט אריין דעם שליסל, און מען היט אפ.

מען זעט דערנאך "credential successfully created" אינעם פערזענליכן ארט. דער resource איז "text", ווייל מען ארבעט נישט מיט ווידעא, אודיא אדער בילדער. אין דער רשימה פון Gemini מאדעלן ראט דער קורס צו נוצן דעם לעצטן מאדעל, און אין דער לעקציע ווערט געוצט 2.5 Flash.

דער פראמפט איז א JSON וואס באשטימט וועלכן אויספיר מען וויל פון Gemini. וויל מען פיר אויספירן, לייגט מען צו פיר פראמפטס. דער פראמפט באציט זיך צום "edit field" נאוד, דאס הייסט צום אויספיר פונעם פריערדיגן נאוד.

פארקויפן אן n8n אגענט און א נייע לעקציע (35:08:05)

אריינשרייבן די רעזולטאטן אין Google Sheets (35:08:05)

אין דעם טייל פון דער לעקציע ווערט ווייטער געבויט דער AI ריסוירטש-אגענט אין n8n. ביי איין שריט ווייזט זיך א רעזולטאט ווי "undefined". דער קורס זאגט אז מען וועט שפעטער זוכן וואס עס איז די פראבלעם, און דערווייל קען מען אויספירן דעם שריט און אים פארמאכן.

דערנאך ווערט צוגעלייגט נאך א Set נאוד. ער דינט נאר צו זען דעם פריערדיגן רעזולטאט. דארט ווערן ארויסגעגעבן פיר אויסגאנג-פעלדער, ווייל מען דארף פיר סארטן אינפארמאציע, און די דאטא דארף מען אריינשטעלן אין אן Excel טאבעלע.

פאר דער Excel טאבעלע ניצט מען Google Sheets, ווייל עס איז גרינגער. מען זוכט "Google" צווישן די נאודס, מען לאגט זיך איין אין זיין Google חשבון, און מען קלויבט Google Sheets. דארט ווייזט זיך א רשימה פון אקציעס: שאפן א spreadsheet, אויסמעקן א spreadsheet, צולייגן (append), צולייגן ווערטן, אויסליידיגן (clear), אויסמעקן א.א.וו. אין דעם ביישפיל וויל מען צולייגן ווערטן. מען שטעלט אויף דעם Google חשבון, און ביי די Document operations קלויבט מען "append or update row".

פארבינדן דעם דאקומענט און מאפן די קאלומנען (35:09:06)

דער קורס ערקלערט ווי אזוי מען פארבינדט דעם דאקומענט מיט דער טאבעלע:

- מען גייט אריין אין Google Sheets און מען שאפט א נייע טאבעלע. אין דער לעקציע איז שוין דא אזא טאבעלע מיטן נאמען "AI search results".
- אין דער טאבעלע זענען פיר קאלומנען: Question, Summary, Key Insights און Recommendations. "Question" איז די פראגע וואס מען האט אריינגעגעבן ביים אנהייב, און פון דער פראגע וויל מען באקומען א סיכום, הויפט-פונקטן און עצות.
- מען קאפירט דעם URL פון דער טאבעלע. דער טייל פונעם URL נאך "/d" איז דער ID פון דער טאבעלע.
- צוריק אין n8n קלויבט מען "By ID" און מען לייגט אריין דעם ID. מען קען אויך אריינלייגן דעם גאנצן URL. אין דעם ביישפיל פירט עס צו דער זעלבער "AI research" טאבעלע.

• פון דער רשימה ווייזן זיך דערנאך די בלעטער (sheets). אין איין Excel טאבעלע קען מען האבן עטליכע בלעטער, און יעדער בלאט קען האלטן אן אנדער אויטאמאציע.

נאכדעם קומט "Mapping Column Mode". דאס איז דער ארט וואו מען באשטימט וועלכע דאטא גייט אריין אין וועלכן קאלומען. מען קלויבט "Map each column manually". אלע פיר קאלומנען פון דער טאבעלע באווייזן זיך, און ביי יעדן קאלומען שטעלט מען אריין וועלכע דאטא עס זאל באקומען. דערנאך דריקט מען "Execute step" אדער מען היט עס אפ.

א גאנצע אויטאמאציע (35:11:41)

וועז מען לאזט לויפן דעם גאנצן workflow, האט מען אן אויטאמאציע וואס ארבעט אזוי: מען גיט אריין א פראגע, וואס מען קען טוישן אין דעם Edit Field נאוד, און אלע רעזולטאטן וועלן אויטאמאטיש אריינגעשריבן אין דער טאבעלע. דאס זענען דער סיכום, די הויפט-פונקטן וואס Gemini שאפט, און די עצות ווי אזוי צו ניצן די טעמעס.

מען קען אויך באשטימען אין וועלכער שעה פונעם טאג, אדער אין וועלכן טאג, די אויטאמאציע זאל אנהייבן לויפן. מען קען אויך פשוט דריקן אויפן קנעפל און אנהייבן דעם workflow. א גרינער נאוד מיינט אז ער האט נישט קיין פראבלעם און ער לויפט. אין דער לעקציע זעט מען דעם מעלדונג "Workflow executed successfully".

עס ווערט אויך דערמאנט אז מען קען צולייגן tools, למשל workflow tools, Azure , אנדערע ווערסיעס פון ChatGPT א.א.וו., כדי מען זאל עס קענען ריכטיג מאפן.

ווי אזוי צו פארקויפן דעם workflow: JSON און README (35:12:45)

נאכדעם וואס דער workflow איז אויפגעשטעלט, ווייזט דער קורס דעם מערסט געווייגלעכן וועג ווי אזוי עס צו פארקויפן.

ערשטנס דארף מען אראפלאדן דעם גאנצן workflow אין JSON פארמאט. מען דריקט אויף די דריי פינטעלעך, און דארט איז דא אן אפציע "Download". אזוי באקומט מען דעם workflow ווי א JSON טעקע. אין דער לעקציע הייסט די טעקע "workflow.json", און מען קען טוישן דעם נאמען לויטן פראיעקט, למשל "AI agent". די טעקע איז דער n8n מאדעל. וועט עס ימפארטירט זי, באקומט אינגאנצן דעם זעלבן AI אגענט.

מען שאפט אויך א README טעקע, אין דער לעקציע מיטן נאמען "AI research agent". עס איז וויכטיג אז דער וואס קויפט זאל פארשטיין ווי עס ארבעט און וואס מען דארף האבן. די README אנטהאלט פיר הויפט-שריט: n8n import, צולייגן די API שליסלען, פארבינדן די טאבעלע, און די ערשטע לויף. עס ווערט אויך ערקלערט ווי אזוי צו עקספארטירן: מען פאקט צוזאמען אין א ZIP טעקע די JSON טעקע מיט אן PDF, און דאס איז דער פראדוקט.

אין דער README שטייט:

• א באשרייבונג ווי אזוי די אויטאמאציע לויפט פון אנהייב ביזן סוף, און וואס מען באקומט.

• "This package includes": א קורצער איבערבליק פונעם פראדוקט. מען באטאנט אז מען דארף נאר אומזיסטע חשבונות. אזוי קען א קויפער דורכליינען די טעקע און באשליסן צי ער וויל עס קויפן.

- Setup instructions אין פינף שריט: מען עפנט n8n, מען דריקט Own workflows, New צו שאפן א נייעם, די דריי פינטעלעך א.א.וו. אזוי קען דער קויפער אראפלאדן און באקומען די זעלבע אויטאמאציע אויף זיין סיסטעם. מען דערמאנט אויך אז די Google Sheets טאבעלע דארף האבן צוטריט.
- טעסטן דעם אגענט, און Customization: וועלכע אנפאסונגען מען קען אנבאטן פאר די וואס ווילן זיי.
- געדאנקען ווי אזוי צו פארדינען (monetization), און troubleshooting.

Gumroad: ארויפלייגן דעם פראדוקט (35:15:19)

די הויפט-פראגע איז וואו צו פארקויפן. דער קורס רעקאמענדירט שטארק Gumroad. דאס איז אן אומזיסטע אפליקאציע און וועבזייטל צו פארקויפן אויטאמאציע-מכשירים און אנדערע טעכנאלאגישע זאכן, אויך וויידעא-לעקציעס, פון יעדן ארט אויף דער וועלט.

אויפן dashboard איז דא אן איינפירונג צום זייטל ("Getting started", "Make an impression" א.א.וו.). צו צולייגן א פראדוקט טוט מען אזוי:

- מען גייט צו Products און דריקט New product.
- מען גיט א נאמען וואס פאסט צו דער אויטאמאציע, אין ביישפיל "AI research agent".
- מען קלויבט דעם סארט: digital product, course, ebook, membership א.א.וו. דא איז עס א digital product. די אפציע services, וואס איז אויף קאמיסיע, פאסט נישט דא.
- מען שטעלט א פרייז לויט ווי גוט דער פראדוקט קען גיין אויפן מארק, אין א ראם וואס מענטשן וועלן זיך באטראכטן. אין דער לעקציע: $29.
- אויף דער וויטערדיגער זייט באשרייבט מען דעם פראיעקט. מען קען אריינלייגן די README, אדער עטליכע פונקטן דערפון. מען לייגט צו א cover בילד, למשל א סקרינשאט פונעם פראגראם וועט ער לויפט. מען לייגט צו א thumbnail, א call to action, און א סיכום פונעם פראיעקט.
- מען קען טוישן דעם פרייז. אויך זענען דא צוויי אפציעס: אז דער קונה זאל צאלן וויפיל ער ווייל, אדער צאלן אין אויסצאלונגען. מען קען אויך צולייגן פארשידענע ווערסיעס פונעם זעלבן פראיעקט און אזוי מער פארדינען.
- Settings: באגרענצן די צאל פארקויפונגען, לאזן קונים קלויבן א קוואנטיטעט, עפנטלעך וויזן די צאל פארקויפונגען, צייכענען דעם פראדוקט ווי אן e-publication, באשטימען א refund policy, און פארלאנגען shipping אינפארמאציע. דאס אלעס שטעלט מען לויט ווי מען וויל.
- דערנאך דריקט מען Save and continue, מען לייגט צו דעם אינהאלט, און Publish and continue.

באצאלונג, אנאליטיק און פרייזן (35:18:30)

וועט קיין payment method איז נישט צוגעלייגט, קען מען עס צולייגן אויף דער פראדוקט-זייט: PhonePe, Google Pay אדער א באנק-חשבון. דערנאך וועלט דער פראדוקט פארעפנטלעכט, און מען קען אים קויפן.

Gumroad האט נאך מעגלעכקייטן: collaborators, checkouts, emails צו מענטשן וואס האבן אייך געשריבן, און workflows וואס לאזן אויטאמאטיזירן אלעס אין Gumroad , פון באצאלונגען ביז צוטריט. ביי analytics זעט מען וויפיל מען האט פארדינט, ביי payouts די אויסצאלונגען, און עס זענען אויך דא נייעס פון Gumroad. אין settings קען מען צולייגן מיטגלידער פון א טים מיט זייער ראלע און email, כדי זיי זאלן באקומען צוטריט. מען קען אויך אויספילן די באצאלונג-פרטים און דאס לאנד, וואס מאכט גרינגער צו

פארדינען, און עס זענען דא passwords און advanced settings. דער קורס זאגט אז אלעס איז פשוט, און דער באניצער-אינטערפעיס איז זייער נוצלעך פאר איינעם וואס הייבט ערשט אן מיט workflows אדער אנדערע טעכנאלאגישע פראדוקטן. מען לייגט ארויף דעם פראיעקט אין ZIP פארמאט, און נאך דער באצאלונג ווערט עס געשיקט צום קונה.

ווען פרייזן: אין אינדיע קאסטן אויטאמאציע-מכשירים 2,000 ביז 5,000 (רופיס), לויט ווי קאמפליצירט זיי זענען. אין דער וועלט קען מען נעמען $49 ביז $150 א חודש. מיט דער אומזיסטער ווערסיע קען מען גרינג פארקויפן צו דריי ביז פינף קליענטן. ווען מען וויל עס אויסברייטערן צו א געשעפט, קען מען קויפן n8n.

מען קען אויך פארקויפן דורך LinkedIn מיט א DM. דער קורס ווייזט א מוסטער: "היי, איך האב געבויט אן AI ריסוירטש-אגענט וואס קען אייך שיקן וועכנטלעכע באריכטן וועגן אייער פאך. אט איז א מוסטער." מען זאל שטענדיק צושיקן א מוסטער, כדי זיי זאלן פארשטיין ווי דער workflow ארבעט. מען זאל זיי אויך געבן אן אומזיסטן פרואוו-באריכט, כדי זיי זאלן זיין זיכער מיטן קויפן. לויטן קורס, ווען מען שיקט דעם מוסטער צו 20 מענטשן, וועלן כאטש צוויי אדער דריי זאגן יא.

א נייע לעקציע: AI workflow אויטאמאציע (35:20:35)

דא הייבט זיך אן א נייער טייל. די פראגע ווערט געשטעלט: וואס אויב מען קען אויטאמאטיזירן דעם גאנצן workflow, און לאזן AI טון די שווערע ארבעט? מען שפארט צייט, מען פארמינערט טעותים, און מען קען וואקסן דאס געשעפט מיט קלוגע AI מכשירים. די לעקציע וועט ווייזן ווי אזוי צו שאפן AI workflows וואס קענען פארבעסערן די געשעפט-ארבעט, אן שרייבן קיין קאוד.

די לעקציע איז געמיינט פאר א געשעפטסמאן, א מארקעטער, א פראגראמירער, אדער א טייל פון א טים וואס וויל ארבעטן קלוגער. מען וועט לערנען די יסודות פון AI workflow אויטאמאציע, די בעסטע מכשירים אין 2026, און די פראקטישע שריט צו בויען אייגענע workflows אן קאוד. אויפן סוף וועט מען קענען שאפן workflows וואס שפארן צייט, פארגרעסערן די פראדוקטיוויטעט, און לאזן זיך גרינג אויסברייטערן.

די סדר פון דער לעקציע:

- וואס איז AI workflow אויטאמאציע, און פארוואס עס איז וויכטיג.
- די דעפיניציע, די הויפט-מעלות, און פאר וועמען עס פאסט.
- די AI מכשירים וואס מען דארף, ווי Zapier, Make, Notion AI, ChatGPT און נאך.
- בויען א no-code AI workflow פון אנהייב: מען נעמט א פאקטישן פאל, מען שטעלט אויף א trigger, און מען ניצט AI פאר דאטא.

AI-ארבעטס-פלוס אויטאמאציע: יסודות און מכשירים (35:22:08)

וואס די לעקציע וועט באהאנדלען (35:22:08)

דער קורס הייבט אן א נייעם אפטייל וועגן AI-ארבעטס-פלוס אויטאמאציע. עס ווערט צוערשט אויסגערעכנט וואס דא וועט קומען: ווי AI נעמט באשלוסן און ניצט באדינגונגס-לאגיק, און אמת'דיגע ביישפילן פון אויטאמאציע פאר קליינע ביזנעסער, פאר מאכער פון אינהאלט, פאר מארקעטינג-ליט און פאר פארקויפס-טימס.

דערנאך קומען מער אנגעשטרענגטע טעכניקן און פראמפט-אינזשעניריע: אייגענע פראמפטס, web hooks, API'ס און AI אגענטן. דערנאך די אפטע טעותים און ווי מען ווייכט זיי אויס: טריגערס וואס זענען נישט ריכטיג איינגעשטעלט, שוואכע AI פראמפטס, און ווי מען טעסט די ארבעטס-פלוסן. צום סוף קומט א שמועס ווי אזוי מען לאזט וואקסן די אויטאמאציע.

וואס איז AI-ארבעטס-פלוס אויטאמאציע (35:22:39)

עס ווערט ערקלערט אז אין דער היינטיגער שנעלער דיגיטאלער עקאנאמיע איז די "פאַרנומענע ארבעט" (busy work) א שטילער מערדער פון וואוקס. AI-אויטאמאציע ערזעצט נישט סתם מענטשליכע מי. זי פארשטארקט עס, אז מען זאל זיך קענען קאנצענטרירן אויף די אויפגאבעס וואס ברענגען דעם גרעסטן נוצן.

טראדיציאנעלע אויטאמאציע גייט לויט דער לאגיק "אויב דאס, דאן יענץ". די לאגיק איז שטייף, און זי צעברעכט זיך ווען דער אינפוט ענדערט זיך אביסל. AI -אויטאמאציע לייגט אריין קינסטליכע אינטעליגענץ, ווי LLM'ס (גרויסע שפראך-מאדעלן), אין די סדר-שריט. אזוי קען די סיסטעם טראכטן, קאטעגאריזירן, און שאפן אינהאלט אדער באשלוסן. אנשטאט נאר צו פירן דאטא פון פונקט A צו פונקט B, קוקט AI אויף די דאטא, פארשטייט וואס דאס איז, און באשטימט וואס דער בעסטער קומענדיגער שריט איז.

א פשוט'ער משל: טראדיציאנעלע אויטאמאציע איז ווי א טראנספארט-בענדל וואס פירט א קעסטל פון איין זייט צימער צו דער אנדערער. AI-אויטאמאציע איז ווי א קלוגער סארטירער. ער עפנט דאס קעסטל, לייענט דעם צעטל, באשליסט צי דער אינהאלט איז דרינגנד, און שרייבט א קורצן סיכום פארן מענעדזשער איידער ער לייגט עס אויף דער ריכטיגער פאליצע.

די הויפט מעלות (35:24:12)

- ספּאָרן אסאך צייט: AI קען איבערנעמען איבערגע'חזר'טע אויפגאבעס וואס פאדערן טראכטן, ווי אנגרייטן אימעילס, מאכן סיכומים פון מיטינגס, אדער אריינשרייבן דאטא. אזוי קען מען צוריקגעווינען 10 ביז 20 שעה פון דער ארבעטס-וואך.
- ווייניגער מענטשליכע טעותים: AI ווערט נישט מיד פיר אזייגער נאכמיטאג. זי האלט א גלייכע מדרגה פון געוויקייט ביים באארבעטן דאטא, אזוי אז טעותים פון קאפירן-און-אריינלייגן אדער פארפעלטע מעלדונגען ווערן א זאך פון דער פארגאנגענהייט.
- מער סקאלירבארקייט: מען קען באהאנדלען 1,000 "לידס" (מעגליכע קונים) אזוי גרינג ווי AI .10 לאזט קליינע טימס אריסגעבן די מאס ארבעט פון א גרויסן טים, אן אננעמען מער מענטשן.

טראדיציאנעל קעגן AI-באטריבענע פלוסן (35:25:15)

אין דער לעקציע ווערט געוויזן א דיאגראם וואס פארגלייכט די צוויי סארטן לויט פיר פונקטן:

- סארט לאגיק: די טראדיציאנעלע איז "אויב-דאן", א שטרענג באדינגונגס-באזירטער מאדעל. די AI-באטריבענע איז פראבאביליסטיש: זי קאטעגאריזירט און טראכט, זי איז פלעקסיבל, און ס'איז דא א שטיקל לערנען דורכן ניצן LLM.
- סארט אינפוט: די טראדיציאנעלע ניצט סטרוקטורירטע דאטא, למשל ספרעדשיטס. די AI-באטריבענע קען נעמען קולער און בילדער, א גרויסע פארשידנקייט פון אומסטרוקטורירטע דאטא, וואס מאכט אין דער פראקטיק אסאך ארבעט גרינגער.

• באשליסן: ביי דער טראדיציאנעלער איז עס בינאר, יא אדער ניין. די AI-באטריבענע ניצט קאנטעקסט: זי מאכט סיכומים, טוט "סענטימענט-אנאליז" (דערקענט די שטימונג אין א טעקסט) א.א.וו.

• אויפהאלטונג: ביי דער טראדיציאנעלער איז זי הויך, ווייל די סיסטעם צעברעכט זיך וועז דער פארמאט ענדערט זיך. ביי דער AI-באטריבענער איז זי נידעריג, ווייל זי קען זיך צופאסן צו פארשידענע טעקסטן און דאטא.

פאר וועמען איז דאס (35:26:47)

דער קורס באטאנט אז AI-אויטאמאציע איז נישט נאר פאר די טעכנאלאגישע ריזן. זי איז א "כח-פארמערער" פאר יעדן וואס פירט דיגיטאלע אויפגאבעס:

• אנטרעפרענערס: אויטאמאטיזירן דאס געפינען לידס און דאס אריינפירן נייע קונים, כדי זיך צו קאנצענטרירן אויף סטראטעגיע.

• טימס: פארגרינגערן אינערליכע קאמוניקאציע, פראיעקט-פארוואלטונג, דערהיינטיגונגען און טראנסקריפציעס פון מיטינגס.

• מארקעטערס: שאפן פערזענליכע רעקלאמע-טעקסטן, פלאנירן סאשל-מידיע, און אנאליזירן די שטימונג פון קונים אויף גרויסע מאסן.

• פראגראמירער: אויטאמאטיזירן דאקומענטאציע פון קאוד, נאכפאלגן "באגס" (פעלערן), און איבערגעזאגט אז אין 2026 איז די וועלט פון AI-מכשירים אריבער פון פשוט'ע פארבינדערס צו אינטעליגענטע "עקאסיסטעמען". כדי צו בויען א באמת קאנקורענטפעאיגן ארבעטס-פלוס, דארף מען פארשטיין די ארכיטעקטורישע שטארקייט פון יעדן מכשיר.

Zapier: דער אינטואיטיווער דירעקטער (35:27:50)

דער ערשטער מכשיר איז Zapier. מען קען דערמיט שאפן ארבעטס-פלוסן, און מען קען עס זיך פארשטעלן ווי א דירעקטער וואס איז גרינג צו ניצן. Zapier איז דער באקאנטסטער נאמען אין אויטאמאציע, און עס בלייבט דער אומבאשטריטענער מלך פון פארבינדונגען, מיט איבער 8,000 אינטעגראציעס. עס ארבעט ווי א "אוניווערסאלער איבערזעצער" פארן אינטערנעט.

די גרעסטע ענדערונג ביי Zapier איז אז עס האט אריינגענומען AI אגענטן. אנשטאט א גלייכן A-צו-B טריגער, קען מען היינט בויען שטענדיגע AI אגענטן וואס "וואוינען" איבער אלע אפליקאציעס. למשל, מען קען טריינירן אן אגענט אויף די PDF -דאקומענטן פון דער פירמע, און אים זאגן: וועז א קונה פרעגט א טעכנישע שאלה אין Slack, זוך אין אונזערע הענטביכער און גרייט אן ענטפער, אבער שיק עס נאר ארויס אויב די שטימונג פונעם קונה איז נישט פארביטערט.

Zapier: פריי-פרובירן, פלענער און פרייזן (35:29:24)

Zapier גיט א געווייסע מאס פריי-קרעדיט. אויפן באצאל-בלאט זענען דא צוויי וועריסיעס:

• Zapier Pro trial: וועז מען עפנט א נייע חשבון, וועלט מען אויטאמאטיש איינגעשריבן אין א 14-טאגיגער פראבע. מען קען ניצן "פרעמיום" אפליקאציעס, למשל PayPal און Facebook lead ads, וועב-הוקס (וואס וועלן שפעטער ערקלערט ווערן), און AI-פעלדער. ס'זענען אבער דא באגרענעצונגען: נאר 1,000 טאסקס,

30 שריט, און די פונקציעס auto replay און pay per task billing זענען נישט פאראן. די פראבע קען מען פארלענגערן גלייך פון דעם Zapier חשבון.

• Zapier for teams: אינטערוואלן פון איין מינוט צווישן טריגערס, ביז 25 באניצער, ראלעס און ערלויבעניש פאר באניצער, געטיילטע פארבינדונגען צו אפליקאציעס, און א געטיילטער ארבעטס-ראום. די לימיט איז 1,000 טאסקס לויטן היינטיגן פלאן. ארויס- און אריינפירן פון "זעפס" איז נישט פאראן, און SAML single sign-on אויך נישט. כדי צו פארלענגערן א טים-פראבע דארף מען בעטן דורך א לינק און זיך פארבינדן מיטן טים.

דאס אלעס איז אין די פריי-וועלס, אבער מען קען גיין צו באצאלטע. אין דער דאקומענטאציע וועגן אפגרעידן וועלט געבראכט א ביישפיל: אויב דער היינטיגער פלאן קאסט $50 א חודש און מען גייט ארויף צו $80 אינמיטן דעם באצאל-ציקל, און עס בלייבן נאך 15 טעג, באקומט מען $25 קרעדיט און מען באצאלט $55 ביים אפגרעידן. דאס הענגט אפ פון טאסק צו טאסק, ווייל יעדע ארגאניזאציע האט אנדערע אויטאמאציעס. מען וועלט געראטן צו לייענען די דאקומענטאציע וועגן ענדערן דעם באצאל-אינטערוואל, אראפגיין אויף א נידעריגערן פלאן, אדער אנולירן.

make.com: דער וויזועלער ארכיטעקט (35:32:25)

דער צווייטער מכשיר איז make.com, אמאל באקאנט אלס Integromat. אויב Zapier איז פאר גיכקייט, איז Make פאר פינקטלעכקייט און גרויסע מאסן. אין 2026 איז עס דער באליבטער מכשיר ביי אויטאמאציע-אינזשעניערן און פירער פון אפעראציעס.

עס וועלט געוויזן אז Make שטעלט צו לייזונגען פאר מארקעטינג, פארקויפן, אפעראציעס, קונה-ערפארונג, פינאנצן, אינפארמאציע-טעכנאלאגיע, מענטשן און ארבעטספלאַץ, און פראדוקט. מען קען זוכן לויטן געהעריגן געביט. ס'זענען אויך דא "קעיס סטאדיס", למשל אויטאמאציע פאר סאשל-מידיע, מיט בלאגס און אינטעגראציעס ווי ChatGPT ,Claude, Browse AI, OpenAI און Facebook Pages.

מען קען אנהייבן מיט פארטיגע מוסטערן, למשל: נעמען א סיכום פון א וועבזייטל'ס אינהאלט און שאפן א Facebook-פאסט מיט ChatGPT און Browse AI. מען קען אויך בויען אן אייגענעם ארבעטס-פלוס.

make.com: פרייזן און מעגליכקייטן (35:34:00)

מען קען באצאלן יערליך אדער חודש'ליך:

• Free: 1,000 קרעדיטס א חודש, גלייך ווי ביי Zapier. עס האט וויזועלע ארבעטס-פלוסן אן קאוד, 3,000 אינטעגראציעס מיט אפליקאציעס, routers און filters, קונה-שטיצע, און א מינימום פון 15 מינוט צווישן לויפן.

• Core: $9 א חודש, דאס בעסטע פאר פרילענסערס. עס האט אומבאגרענעצטע אקטיוועט "scenarios", מער קאנטראל איבער פלאנירטע scenarios, א העכערע דאטא-טראנספער לימיט, און צוטריט צום Make API.

• מיט "priority execution": $16 א חודש.

• Teams : בערך $29 א חודש. טימס און טים-ראלעס קענען דאס ניצן, און מען קען שאפן און טיילן אייגענע scenarios.

- Custom pricing: פאר ארגאניזאציעס מיט זייער קריטישע ביזנעסער. עס האט שטיצע פאר אייגענע פונקציעס, אינטעגראציעס אויף ענטערפרייז-שטאפל, 24/7 שטיצע, צוטריט צו א "value engineering" טים, און אנגעשטרענגטע זיכערהייט.

צום סוף ווערט דערמאנט אז מען קען מיט make.com זיך אויך פארבינדן מיט MCP סערווערס, ארויסציען אינהאלט, און טון וועב-זוכונגען.

מכשירים פאר אויטאמאציע און א Zapier ביישפיל (35:36:01)

דער סוף פון דעם פריערדיגן מכשיר: מעגליכקייטן, אינטעגראציעס און זיכערהייט (35:36:01)

דער קורס ענדיגט די באשרייבונג פונעם פריערדיגן מכשיר. מען קען דערמיט מאכן אן אגענט, וואס איז גאר אינטערעסאנט. מען קען אויך מאכן א "toolkit", און עס זענען פאראן גרייטע ביישפילן פון ארבעטס-פלוסן (workflow scenarios), למשל צו מאכן א קאוד-אפליקאציע. ווער עס וויל אויפבויען א ארבעטס-פלוס אן קיין קאוד (no-code workflow builder), קען דאס אויך טון. מען קען ניצן סיסטעם-ווערִיאבלס (system variables) און אייגענע ווערִיאבלס (custom variables).

ביי אינטעגראציעס זענען פאראן מער ווי 3,000 אינטעגראציעס. מען קען צולייגן אייגענע אפליקאציעס (custom apps) און מען האט צוטריט צו ענטערפרייז-אפליקאציעס וואס ליגן אויף די פירמע'ס אייגענע סערווערס (on premises).

ביים אויספירן ארבעטס-פלוסן (workflow execution) זענען פאראן געוויסע גרעניצן און סעטינגס:

- די מאקסימום צאל אקטיוויע סצענאריאס.
- אויספירן עטליכע סצענאריאס אויפאמאל (parallel scenario execution).
- די מאקסימום גרייס פון א פייל.
- וויפיל דאטא מען מעג איבערפירן (data transfer).

דערנאך קומט "governance and observability", ד.ה. קאנטראל און איבערזיכט: מען קען האבן אן אומבאגרעניצטע צאל באניצער, מען קען אויפשטעלן א "grid", עס זענען פאראן טימס און טים-ראלעס, און אנאליטיק. לויט דעם קורס גיבן זיי אויך זייער גוטע סופָארט.

ביי זיכערהייט איז פאראן צוויי-שריטיגע אידענטיפיקאציע (two-factor authentication), איינמאליגע אריינלאגירונג פאר אלע סיסטעמען (single sign-on), און "domain claim". דער מכשיר ניצט AWS פארן האסטינג, און די דאטא-צענטערס געפינען זיך אין צפון-אמעריקע און אין אייראפע. דער קורס עצה'ט מען זאל דעם מכשיר אליין אויספרובירן.

דער דריטער מכשיר: ChatGPT אלס דער "מוח" פון אויטאמאציע (35:37:02)

דער דריטער מכשיר איז ChatGPT. לויט דעם קורס קען יעדער דעם מכשיר, און רוב מענטשן ניצן אים שוין טאג-טעגליך. עס איז געווארן איינער פון די וויכטיגסטע מכשירים, און עס האט געמאכט א רעוואלוציע אין אויטאמאציע. דער קורס רופט עס דער "קאגניטיווער מוח" פון 2026.

ChatGPT איז לויט דעם קורס ארויסגעוואקסן פון א פשוט'ן שמועס-פענסטער. היינט איז עס דער "API-first" מוח פון דער וועלט-עקאנאמיע. מיט אנדערע ווערטער, עס איז דער מאטאר וואס ברענגט שכל'דיגע באטראכטונג (reasoning) אריין אין די "zaps" (ביי Zapier) און אין די "scenarios" (ביי Make).

עס האט טיפע אינטעגראציעס און פיל מעגליכקייטן:

- Function calling און באנוץ פון מכשירים (tools): מען קען היינט געבן ChatGPT ערלויבעניש צו ניצן אנדערע מכשירים. א ביישפיל פונעם קורס: מען זאגט ChatGPT "קוק אן די קלאגע פונעם קונה. אויב עס איז א טעכנישער פעלער (bug), נוץ Jira צו עפענען א טיקעט. אויב עס איז א פראבלעם מיט א רעכענונג (billing), נוץ Stripe צו קוקן וואס איז געווען זיין לעצטע צאָלונג."
- Multimodal: ChatGPT קען היינט זיין א וויזועלער שריט אינעם ארבעטס-פלוס. א ביישפיל: א קונה שיקט א בילד פון א צובראכענעם טייל. דער AI קוקט אויפן בילד, דערקענט דעם נומער פונעם טייל לויט זיין טריינירן, און לאזט אויס א באשטעלונג אינעם לאגער-סיסטעם (inventory).

ווען די מאדעלן וואס מען קען ניצן אדער אריינשטעלן: לויט דעם קורס לויפט איצט ווערסיע 5.2. מען האט אויך צוטריט צו עלטערע (legacy) מאדעלן, אבער דער קורס עצה'ט צו ניצן די לעצטע ווערסיע, ווייל זי איז זייער אוואנסירט.

די "reasoning" מאדעלן פון ChatGPT ווערן גענוצט פאר אויטאמאציע וואו עס שטייט אסאך אויפן שפיל (high-stakes). זיי שאצן נישט סתם דאס קומענדיגע ווארט, נאר זיי טראכטן דורך דעם פראבלעם. דערפאר זענען זיי צופאס למשל פאר אן אויטאמאטישער איבערקוק פון א געזעצליכן קאנטראקט, אדער פאר א קאמפליצירטע פינאנציעלע פאָרויסזאגונג.

אין קורצן קען מען ChatGPT ניצן פאר:

- שרייבן טעקסט.
- אנאליזירן געפילן אין טעקסט (sentiment analysis).
- ארויסנעמען דאטא פון אומארגאניזירטע קוואלן, למשל צומישטע אימעילס.
- קאמפליצירטע באשלוסן.

דער פערטער מכשיר: Notion AI – א פארבונדענער ארבעטס-פלאץ (35:39:04)

דער פערטער מכשיר איז Notion AI. לויט דעם קורס איז דאס א "connected workspace", א פארבונדענער ארבעטס-פלאץ. Notion איז געווען א מכשיר פאר דאקומענטן און דאטא-באזעס, און איז היינט פארוואנדלט געווארן אין אן "AI-native" אפערירונג-סיסטעם.

Notion AI האט לעצטנס ארויסגעברענגט אן אגענט. דער אגענט ווייסט אלעס וואס דער באניצער ווייסט. דורך "connectors" האט ער צוטריט צו דער גאנצער Slack היסטאריע און צו Google Drive. מען קען אים פרעגן למשל: "וועלכע אויפגאבן זענען נאך אפן פון דעם פראיעקט וואס מיר האבן נעכטן געבויט אין Slack?" און דער אגענט וועט צונויפשטעלן א בלאט מיט א קורצע איבערזיכט.

עס איז פאראן אויך א "set and forget" מעגליכקייט, ד.ה. מען שטעלט עס איין איינמאל און מען דארף שוין נישט מער טראכטן דערפון. למשל, מען שטעלט אויף א דאטא-באזע פאר נאטיצן פון מיטינגס, און מען לייגט צו אן AI אייגנשאפט וואס הייסט "action item". יעדעס מאל ווען א מיטינג ענדיגט זיך, ליינט Notion AI

אויטאמאטיש איבער דעם טראנסקריפט און פילט אויס די אויפגאבן, אן דעם וואס מען זאל דארפן קליקן אויף א קנעפל.

נאך א מעגליכקייט איז פראגעס און ענטפערס פאר טימס. Notion AI ארבעט ווי א ביבליאטעקאר 24 שעה א טאג. אנשטאט אז אן ארבעטער זאל פרעגן אין Slack "וואו איז די פאליסי פון באצאלטע פריי-טעג (?PTO)", קען מען פרעגן Notion AI, און עס ענטפערט מיטן געוויזן דאקומענט און דעם ריכטיגן אפטייל.

דער מכשיר איז לויט דעם קורס דאס בעסטע פאר:

- פירן אינערליכע וויסן אין א פירמע.
- נאכפאלגן פראיעקטן.
- טימס וואס ווילן אז זייערע דאקומענטן זאלן זיין לעבעדיג און זיך אליין ארגאניזירן.

די פרייזן:

- אומזיסט: $0 א חודש, מיט געוויסע מעגליכקייטן.
- א פלאן פאר $10 א חודש: אומבאגרעניצטע בלאקס פאר צוזאמענארבעט, אומבאגרעניצט ארויפלאדן פיילס, אייגענע פארמס (custom forms), א.א.וו.
- Business: אריינגערעכנט Notion AI, SSO, א מער פיינער קאנטראל איבער דאטא-באזעס, באשטעטיגן בלעטער, און פארמס מיט באדינגונגען (conditional form logic).
- Enterprise: אסאך אינטעגראציעס מיט אנדערע מכשירים, מיט נאך מעגליכקייטן.

דער פינפטער מכשיר: Pabbly – א ביליגע אלטערנאטיוו צו Zapier (35:41:05)

דער קומענדיגער מכשיר איז Pabbly. לויט דעם קורס האט ער זיך אויסגעהאקט א גרויסן פלאץ אין מארק, ווייל ער איז די מערסטע קאסט-עפעקטיווע אלטערנאטיוו צו Zapier. ער צילט ספעציעל אויף דעם "SaaS טאקס", ד.ה. די מאנאטליכע אפצאלן פאר סאפטווער-סערוויסעס וואס הרג'ענען דעם פראפיט פון קליינע געשעפטן.

פארוואס איז ער באזונדער?

- מען באצאלט נישט פאר אינערליכע שריטן: ביי Zapier ווערט א פילטער אדער א "data formatter" געראכנט ווי א "task", און דאס קאסט געלט. ביי Pabbly זענען זיי אומזיסט. מען באצאלט נאר פאר דער לעצטער אקציע, למשל שיקן אן אימעיל.
- איינמאליגע צאלונג אויף אייביג (lifetime deals): Pabbly באט אפטמאל אן אן איינמאליגע צאלונג פאר אויטאמאציע אויף אייביג. פאר א סטארטאפ קען דאס אויסלאזן אן איינשפארונג פון ארום $2,000 א יאר.
- גוט פאר מארקעטינג: Pabbly האט טיפע אינטעגראציעס מיט נישט-געווענליכע מארקעטינג מכשירים, למשל מכשירים צו שיקן אימעילס און SMS gateways, וואס Zapier באגרעניצט אמאל אדער רעכנט שווער געלט דערפאר.

דער מכשיר פאסט פאר אנטערפרענערס וואס היטן אפ דעם בודזשעט, פאר מארקעטינג אגענטורן וואס צייגן אסאך "leads" (מעגליכע קונים), און פאר באניצער וואס פיינט'ן מאנאטליכע אבאנעמענטן.

א פראקטישער ביישפיל: בויען א ארבעטס-פלוס אין Zapier מיט Copilot (35:42:41)

דערנאך ווייזט דער קורס ווי אזוי מען בויט אן אויטאמאציע אין Zapier. מען גייט אריין אין zapier.com. אויף דער היים-בלאט זעט מען די ארבעטס-פלוסן וואס זענען שוין פריער געמאכט געווארן. מען קליקט "create" און מען וועלט א נייעם "zap".

עס זענען פאראן צוויי וועגן צו בויען:

- מאנועל: מען קליקט אויף "trigger" און מען שטעלט אויף טריגערס און אקציעס איינס נאכן אנדערן.
- מען באשרייבט דעם ארבעטס-פלוס פאר דעם Copilot, און ער בויט עס אויף.

אינעם ביישפיל ווערט געשריבן א פראמפט בערך אזוי: "שאף א ארבעטס-פלוס אויטאמאציע מיט Google Sheets. די טאבעלע אנטהאלט: נאמען, אימעיל, און קוואל (source). די קוואלן זענען LinkedIn, Twitter, Instagram און Facebook. Leads פון Instagram, Twitter אדער Facebook ווערן באטראכט ווי לעדס מיט הויכע פריאריטעט, און מען שיקט דעמענטשפרעכנד אן אימעיל." מען לייגט אויך צו דעם אייגענעם אימעיל-אדרעס.

אינעם פראמפט ווערט אויך באשריבן א דריי-שריט־גער פלוס:

- טריגער פון Google Sheets.
- ניצן Zapier AI צו צייכענען יעדן ליד ווי הויך אדער נידעריג אין פריאריטעט.
- שיקן אן אימעיל, יעדע 15 מינוט.

דער Copilot הייבט אן אויטאמאטיש צו בויען דעם ארבעטס-פלוס. מען זעט ווי דער אגענט גייט שריט ביי שריט: ער לייגט צו די שריטן פונעם zap, ער קלייבט אויס די פעלדער, ער דערהיַנטיקט די שריטן, ער פילט אויטאמאטיש אויס די ווערטן, און ער אנאליזירט און גיט צוריק די דאטא. לויט דעם קורס איז דאס דער גרינגסטער וועג.

ביים דריטן שריט (דער אימעיל) שטייט אינעם גוף פונעם אימעיל "New or high priority lead detected", מיט אלע פרטים וואס ווערן אריינגעלייגט. מען קען אריינלייגן נאך טעקסט אדער דאטא, און מען קען אויך אנצינדן "read receipts" (א באשטעטיגונג אז דער אימעיל איז געליינט געווארן). אינעם ביישפיל לאזט מען עס אזוי ווי עס איז.

דערנאך פראבירט מען אויס דעם שריט (test), און ער ווערט געענדיגט. מען קוקט אריין אינעם אימעיל, און מען זעט אז דער אימעיל איז טאקע אנגעקומען. לויט דעם קורס איז דאס א רעוואלוציע וואס קומט אין דער AI וועלט, און עס וועט אויטאמאטיזירן אסאך זאכן.

איבערחזרן די שריטן און אריסגעבן דעם פלוס (35:47:22)

ווען דער ארבעטס-פלוס איז פארטיג, קען מען אים אריסגעבן (publish), און דאן ווערט ער "live", ד.ה. ער הייבט אן צו ארבעטן. אויב מען חזר'ט איבער, איז דער פלוס זייער פשוט:

- ערשטנס ליינט מען די טאבעלע. אינעם ביישפיל הייסט זי "untitled", און א טאבעלע מיט דאטא איז צוגעבונדן.
- צווייטנס אנאליזירט Zapier AI די דאטא און גיט צוריק א רעזולטאט.

- דריטנס שיקט מען ארויס דעם אימעיל, לויט דער פריאריטעט וואס איז געפונען געווארן: צי עס איז א ליד מיט נידעריגע אדער מיט הויכע פריאריטעט.

דער קורס באטאנט אז מיט אגענטן קען מען גרינג שאפן ארבעטס-פלוס אויטאמאציע, אפילו אן אסאך וויסן. די אגענטן וועלן אפטוון די ארבעט. אבער מען מוז שרייבן א זייער באשרייבנדיגן, פרטימ'דיגן פראמפט, ווייל דאס איז דער שליסל צו באקומען די ארבעט ריכטיג געטון.

באשלוסן און באדינגונג-לאגיק מיט ChatGPT: קלאסיפיצירן לעדס (35:48:22)

דער קומענדיגער טייל באהאנדלט ווי אזוי מען מאכט באשלוסן און ניצט באדינגונג-לאגיק (conditional logic) מיט ChatGPT, כדי צו מאכן קלוגע באשלוסן. דער עיקר דא איז: מען נעמט די פראגעס פונעם באניצער אדער די אריינגעגעבענע דאטא, מען פראצעסירט זיי דורך דעם AI מאדעל, און דערנאך באשליסט מען וואספארא קלוגע אקציע צו טון.

דער ביישפיל איז קלאסיפיצירן לעדס. שטעלט אייך פאר א סיסטעם פאר צונויפקלייבן לעדס, וואו מענטשן שיקן אריין זייערע פרטים. לויט זייערע ענטפערס קען ChatGPT זיי איינטיילן אין דריי סארטן: הויכן, מיטלסטן אדער נידעריגן אינטערעס.

למשל, איין ליד שרייבט: "איך דארף א לייזונג וואס זאל מיר העלפן אויטאמאטיזירן מיינע סאשעל מידיא קאמפיינס. וויפיל וועט עס קאסטן פאר א טים פון פינף?" א צווייטער ליד שרייבט: "קענט איר מיר דערציילן מער וועגן די פראדוקט-מעגליכקייטן?" ChatGPT ענטפערט דאן למשל "א דאנק פארן זיך פארבינדן", לויט דעם אינטערעס פונעם ליד, און געפינט אויס די פאסיגע פראדוקט-קאטעגאריע.

דאס איז דער געדאנק וואס מען וויל איינפירן, און מען הייבט אן צו מאכן די זעלבע סימולאציע: מען גיט ChatGPT דעם פראמפט און מען וואלט אויפן ענטפער.

Zapier-אגענטן און דער יסוד פון AI-אגענטן (35:49:56)

פראמפטן פאר א קונה-סערוויס אגענט און באדינגונגען (35:49:56)

אין דעם טייל פון קורס זעט מען ווי דער GPT-טייל אין Zapier גיט צוריק פאסיגע לייזונגען אויף א שאלה. למשל, ביי א פראגע וועגן פירן סאשעל-מעדיע, שלאגט ער פאר מכשירים ווי Hootsuite, Sprout Social און Buffer. דערנאך פרובירט מען א צווייטן פראמפט, למשל: "קענסטו מיר דערציילן מער וועגן די פראדוקט-אייגנשאפטן?" מען דריקט Enter, און אויפן סמך פון די געגעבענע מכשירים הייבט דער מאדעל אן צו געבן די פאסיגע ענטפערס.

דערנאך וועלט געוויזן ווי אזוי מען פאסט צו א פראמפט פאר א ספעציפישער לייזונג. נעמט א משל: מען בויט אן AI-אגענט, און אסאך קונים שיקן אריין קלאגעס אויפן פארטאל אז זיי האבן פראבלעמען מיט באצאלונגען. אנדערע קונים שרייבן ווידער: "איך וויל מער וויסן וועגן אייערע פראדוקט-אייגנשאפטן." די פראגע איז ווי אזוי GPT זאל באשליסן וואס צו ענטפערן. דערפאר שרייבט מען אים א פראמפט מיט קלארע כללים:

- אויב די שאלה אנטהאלט דאס וואלט payment אדער billing, ענטפערט מען: "א דאנק פארן זיך פארבינדן. עס זעט אויס אז איר האט א פראבלעם מיט א חשבון. מיר וועלן אייך גלייך פארבינדן מיט אונזער פינאנץ-אפטיילונג."

- אויב די שאלה אנטהאלט דאס ווארט features, ענטפערט מען: "א דאנק פאר אייער אינטערעס. דא איז מער אינפארמאציע וועגן אונזערע פראדוקט-אייגנשאפטן."
- אין אלע אנדערע פעלער ענטפערט מען למשל: "א דאנק פארן זיך פארבינדן. אונזער שטיצע-מאנשאפט וועט זיך באלד מיט אייך פארבינדן."

דער קורס באמערקט אז דאס איז גארנישט אנדערש ווי די פראגראמירונג-באדינגונגען if און else וואס מען לערנט ביי פראגראמירן. אויף אזא אופן קען מען שאפן א פראמפט מיט if/else באדינגונגען, און דאס זעלבע וואס מען האט פריער געטון אין Zapier קען מען דא אויך אריינשטעלן.

א פאקטישער ביישפיל: כאפן און סארטירן "לידס" (35:51:57)

דערנאך ווערט באהאנדלט א פאל פון דער פאקטישער וועלט: lead generation. דאס איז דער פראצעס פון כאפן מעגליכע קונים און זיי פארוואנדלען אין קונים וואס צאלן באמת. עס איז א זייער וויכטיגע ארבעט פאר פארקויף-מאנשאפטן, בפרט אין פעלדער ווי ריעל-עסטעיט, SaaS און e-commerce. געווענליך קומען לידס פון פארשידענע מקורות, ווי וועבזייטלעך, אויסגעפילטע פארמס און landing pages.

מיט Zapier קען מען אויטאמאטיזירן דאס כאפן פון די לידס-דאטא פון א לידס-פארם. די אינפארמאציע ווערט געהאלטן אין Google Sheets, און Zapier צעטיילט אויטאמאטיש די לידס אין גרופעס און לאזט לויפן די פאסיגע וועיטערדיגע שריט. דער סדר איז אזוי:

- דער טריגער: א נייע פארם ווערט אריינגעשיקט, למשל דורך Google Forms. די פארם קען אנטהאלטן נאמען, אימעיל, פירמע אדער אינדוסטריע.
- קוואליפיצירן דעם לית: איז ער אינטערעסירט, נישט אינטערעסירט, אדער מען דארף זיך שפעטער נאכפרעגן.
- יעדעס מאל וואס א מעגליכער קונה פילט אויס די פארם, באמערקט Zapier דאס אריינשיקן און הייבט אן דעם ארבעט-שטראם.
- די אקציע: די לידס-דאטא ווערט אויטאמאטיש אפגעהיט אין Google Sheet ווי א טאבעלע. יעדעס אריינשיקן שאפט א נייע שורה מיט ארגאניזירטע קאלאנעס: נאמען, אימעיל-אדרעס, פירמע-נאמען און אינדוסטריע. מען קען אויך דערהיינטיגן דעם סטאטוס פונעם לית.

אויף דעם אופן ווערט דער Google Sheet א שיינע דאטא-באזע פאר פירן לידס. די פארקויף-מאנשאפט באקומט א גרינגע איבערזיכט פון אלע מעגליכע קונים, און מען קען דורכפירן וועיטערדיגע שריט, צעטיילן די לידס אין קאטעגאריעס און זיי סארטירן. Zapier האט איינגעבויטע פילטערס און ארבעט-שטראמען וואס מען קען איינשטעלן צו קאטעגאריזירן לידס לויט זייערע ענטפערס. למשל, לידס וואס זענען אנגעצייכנט ווי "אינטערעסירט" גייען אריין אין א ליסטע פון הויכע פריאריטעט, און די איבעריגע גייען אריין אין אן אלגעמיינער נאכפרעג-ליסטע. אזא צעטיילונג זארגט אז די פארקויף-מאנשאפט קען אוועקשטעלן אירע כוחות אויפן מערסט עפעקטיוון אופן.

Webhooks, פילטערס און דער Copilot וואס בויט א ארבעט-שטראם (35:54:02)

דער קורס ווייזט נאך עטליכע פיינע מעגליכקייטן אין Zapier, למשל webhooks, פילטערן, און אימעיל-שטיצע. מען קען זיך וועגן צו דער אפיציעלער דאקומענטאציע פון דעם API. דארט זעט מען למשל ווי מען שיקט דאטא אין form-encoded פארמאט, מיט ערשטן נאמען און לעצטן נאמען, אדער אין JSON פארמאט.

א webhook איז עפעס ווי א URL, און וועמ מען שאפט א webhook ניצט מען דעם URL. אויב מען וויל באניצן Zapier webhooks, גייט מען אריין אין Google Sheets, און אין Apps Script שרייבט מען א JavaScript פונקציע. אין דער פונקציע שאפט מען א פאריאבל url וואס האלט דעם Zapier webhook . אין Zapier קלאפט מען אויף דער פאסיגער event, און מען דארף דורכגיין א ריכטיגע איינשטעלונג: ערשט שאפט מען א טריגער, און אין דער configuration-אפטיילונג שטעלט מען אריין די נייטיגע פרטים.

מען קען אבער אויך בעטן דעם Copilot אין Zapier אז ער זאל אליין בויען דעם גאנצן ארבעט-שטראם. מען שרייבט אים א פראמפט ווי: "שאף א webhook מיט א טריגער-אקציע, מיט פילטער-באדינגונגען פאר לידס פון הויכער ביז נידעריגער פריאריטעט, ווי א דריי-שריטיגער ארבעט-שטראם פאר צעטיילן לידס, און צום סוף שיק אן אימעיל". מען לייגט צו די אייגענע אימעיל-אדרעס. דער אגענט הייבט אן צו ארבעטן און שאפט זיין אייגענעם ארבעט-שטראם. עס קען נעמען א שטיק צייט. ער מעקט ערשט אויס דאס אלטע, שאפט דעם טריגער אלס ערשטן שריט, און דערנאך הייבט ער אן צולייגן די נייע שריט.

מען זעט ווי ער שאפט ערשט דעם webhook. אין דער configure-אפטיילונג קלייבט ער א child key, און מען באקומט די פאסיגע אפציעס. ער שטעלט אויך איין דעם פילטער (Filter by Zapier). דער באניצער דארף גארנישט טון, דער אגענט באארגט אלעס. נאכן ערשטן שריט בויט ער אויס דעם גאנצן ארבעט-שטראם. דער קורס האלט אז דאס ארבעט זייער גוט, און רעקאמענדירט צו באטראכטן די באצאלטע ווערסיע פון Zapier, ווייל לויט דער מיינונג אין קורס קען מען דערמיט גרינג דורכפירן רוב אויטאמאציע-ארבעט.

א וויכטיגע עצה: ווען מען שרייבט א פראמפט, זאל מען געבן א טיפערן באגריף און א דעטאלירטע באשרייבונג פון דעם וואס מען וויל, און דער אגענט וועט ארבעטן לויט דעם. Zapier קען גרינג אריינשטעלן GPT. מאנכמאל דערקענט ער אויטאמאטיש דעם OpenAI חשבון, און מען דארף נאר אנגעבן די אימעיל-אדרעס. מען זעט ווי ער לייגט צו אלע שריט. דערנאך קלאפט מען continue, און ער הייבט אן טעסטן. מען קען אויך טעסטן יעדן שריט באזונדער.

אן איבערבליק פון א נייעם קורס: דאטא-באזע אגענטן (35:58:46)

דא הייבט זיך אן א נייער טייל, און עס ווערט פארגעלייגט זיין פלאן. מען וועט אנהייבן פון פארשטיין דעם באגריף פון דאטא-באזע אגענטן און זייער ראלע אין אויטאמאטיזירן און פארבעסערן ארבעט מיט דאטא-באזעס. דערנאך וועט מען בויען דעם ערשטן AI-אגענט פון גאר אנהייב, און פארשטיין זיינע הויפט-טיילן. מען וועט זען ווי אזוי ארבעטן מיט CSV-דאטא דורך AI-אגענטן, כדי צו אנאליזירן און באארבעטן דאטא. דערנאך וועט מען פארבינדן דעם אגענט צו א SQL דאטא-באזע, כדי צו שטעלן שאלות צום דאטא און עס פירן אין רעאלער צייט. מען וועט אריינשטעלן די function calling מעגליכקייט פון Azure OpenAI, כדי צו פארשטארקן דעם אגענט. צום סוף וועט מען באניצן דעם Assistants API, כדי צו פארבעסערן SQL-שאלות און די ארבעט מיט דער דאטא-באזע, און אזוי פארבעסערן די לייסטונג פונעם אגענט. נאך דעם יסודות-טייל איז צוגעלייגט א קוויז צו פרובירן דאס פארשטענדעניש.

וואס איז אן AI-אגענט? (35:59:50)

די ערשטע לעקציע באהאנדלט די טעאריע הינטער AI-אגענטן, איידער מען גייט אריבער צו פראקטישער ארבעט אין שפעטערדיגע מאדולן. דער יסוד-וויסן העלפט פארשטיין ווי אזוי AI-אגענטן ארבעטן, פארוואס זיי זענען וויכטיג, און ווי אזוי מכשירים ווי Azure, OpenAI און LangChain מאכן גרינגער זיי צו בויען.

אן AI-אגענט איז א ווייכווארג-פראגראם וואס קען דערקענען זיין סביבה, באארבעטן אינפוט, און נעמען פאסיגע שריט לויט דעם אינפוט. די שריט קענען זיין פון ענטפערן א באניצער'ס שאלות ביז איבערזעצן א זאץ, צוזאמענפאסן א דאקומענט, רופן אן API, אדער ארבעטן מיט דאטא-באזעס. מען קען זיך פארשטעלן אן AI-אגענט ווי א ווירטועלער געהילף, נאר אזא וואס מען קען אליין בויען, קאנטראלירן און צופאסן. Siri, Alexa און ChatGPT זענען ביישפילן פון AI-אגענטן אין דער פאקטישער וועלט. זיי נעמען אויף מענטשליכן אינפוט, געווענליך דורך רעדן אדער טעקסט. זיי פארשטייען וואס מען פרעגט, און גיבן צוריק א פאסיגן ענטפער.

וואס מאכט היינטיגע AI-אגענטן אזוי שטארק איז אז זיי זענען געבויט אויף פארגעשריטענע מאשין-לערנען מאדעלן, בפרט גרויסע שפראך-מאדעלן (LLMs). די מאדעלן זענען טריינירט צו פארשטיין מענטשליכע שפראך און קאנטעקסט. זיי לאזן דעם אגענט פארשטיין נישט נאר די ווערטער אליין, נאר אויך די כוונה און די באדייטונג הינטער זיי. דערפאר קענען זיי פירן א באדייטנדיגן שמועס און דורכפירן קאמפליצירטע ארבעט, אן דעם וואס מען זאל זיי מוזן ספעציעל פראגראמירן פאר יעדן מעגליכן פאל.

די בוי-שטיינער פון אן AI-אגענט (36:01:51)

צו בויען אן ארבעטנדיגן אגענט דארף מען פארשטיין פון וואס ער איז געמאכט. געווענליך האט ער די פאלגנדע טיילן:

- אינפוט-באארבעטונג: דאס איז דער ערשטער שריט, וואו דער אגענט באקומט דעם אינפוט. מערסטנס איז עס טעקסט, אבער עס קען אויך זיין קול, בילדער אדער אנדערע פארמאטן, לויטן צוועק. אין דעם קורס פאקוסירט מען אויף טעקסט. דער שריט באשטייט פון כאפן דעם באניצער'ס שאלה און זי צוגרייטן צום פארשטיין. דאס קען אנטהאלטן ריינמאכן דעם טעקסט, אראפנעמען איבעריגע צייכנס, אדער אים איבערפירן אין א סטרוקטורירטן פארמאט.
- דער מאדעל: אין הארץ פונעם אגענט ליגט א שפראך-מאדעל ווי GPT-4. ער איז אחראי צו פארשטיין דעם אינפוט און ארויסגעבן א פאסיגן ענטפער. ער נעמט דעם אינפוט, דאס הייסט דעם פראמפט, און לויט זיין טריינירן אויף ריזיגע דאטא-סעטן פון מענטשליכער שפראך גיט ער א קלארן ענטפער וואס פאסט צום קאנטעקסט. ער טוט דאס דורך מעגליכקייטן (probabilities): אין עיקר זאגט ער פאראויס וואס קומט ווייטער אין דער ריי ווערטער, לויטן אינפוט וואס ער האט באקומען.
- ארויסגעבן דעם אויספיר: נאכן באארבעטן דעם אינפוט גיט דער מאדעל ארויס אן אויספיר. דאס קען זיין א טעקסט-ענטפער, אן איבערגעזעצטער זאץ, א צוזאמענפאסונג, אדער אפילו א באשלוס צו טון א שריט, ווי רופן א פונקציע אדער זוכן אין א דאטא-באזע. די קוואליטעט פון אויספיר הענגט שטארק אפ אין ווי קלאר דער פראמפט איז, און אין די מעגליכקייטן פונעם מאדעל.
- אקציע אדער דורכפירונג: ביי מער פארגעשריטענע אגענטן איז דער ענטפער נישט נאר טעקסט. ער קען אויך אנצינדן אנדערע פונקציעס. למשל, דער אגענט קען איינשטעלן א זיצונג, אויסרעכענען א חשבון, שיקן א מעלדונג, אדער פרעגן א וויסנס-באזע. דאס מאכט דעם AI-אגענט נישט נאר א שמועס-מכשיר, נאר א מכשיר וואס טוט פאקטישע ארבעט.

נאטירליכע שפראך-באארבעטונג (NLP) (36:04:05)

NLP) נאטירליכע שפראך-באארבעטונג) איז דער פעלד אין AI וואס לאזט קאמפיוטערס פארשטיין, טייטשן און ארויסגעבן מענטשליכע שפראך. דער AI-אגענט ניצט NLP צו צעטיילן דעם אינפוט, פארשטיין די באדייטונג הינטער אים, און ארויסגעבן ענטפערס וואס קלינגען ווי פון א מענטש. שפראך-מאדעלן ווי GPT-4

זענען טרייניירט אויף פארשידנארטיגע זאמלונגען טעקסט, אריינגערעכנט ספרים, וועבזייטלעך און אפגעשריבענע שמועסן. דאס לאזט זיי דערקענען קאנטעקסט, טאן, גראמאטיק און באדייטונג.

אגענטן מיט Azure OpenAI, LangChain און CSV דאטא (36:04:37)

ווי NLP מאדעלן פארשטייען צוויי-טייטשיגקייט (36:04:37)

וואס מאכט די היינטיגע NLP מאדעלן אזוי באזונדער איז זייער פעיקייט זיך צו באהאנדלען מיט צוויי-טייטשיגקייט. דער קורס ברענגט א ביישפיל: וועלן א באניצער זאגט "Can you book a table for two?", פארשטייט דער מאדעל אז דאס ווארט "book" מיינט דא מאכן א רעזערוואציע פאר א טיש, און נישט ליינען א בוך. די פעיקייט צו פארשטיין דעם קאנטעקסט איז דאס וואס לאזט אגענטן רעדן מיט מענטשן אויף א נאטירלעכן און עפעקטיוון אופן.

פארוואס Azure OpenAI פאר AI אגענטן (36:04:58)

Azure OpenAI גיט צוטריט צו שטארקע AI מאדעלן, ווי GPT-4, דורך א קלאוד-באזירטן אינטערפעיס. עס ווערן געברענגט דריי סיבות פארוואס דאס פאסט גוט פארן בויען AI אגענטן:

- פארגעשריטענע שפראך-פארשטאנד: GPT-4 קען פארשטיין קאמפליצירטע שפראך-איינגאבן און ארויסגעבן ענטפערס פון הויכער קוואליטעט אין פילע שפראכן. ער כאפט די פיינע נואנסן, דעם טאן און וואס דער באניצער וויל באמת, און דאס איז נויטיג כדי די שמועסן זאלן זיין באדייטנדיג.
- פילזייטיגקייט: דער זעלבער מאדעל קען טון א בריייטע ריי ארבעטן: chatbots, צוזאמפאסן טעקסטן, איבערזעצונג, מאכן באריכטן און נאך. דאס מאכט אים זייער בייגזאם, סיי פאר פראגראמירער און סיי פאר ביזנעסער.
- וואקסונג און זיכערהייט: Azure קימערט זיך מיטן האסטן און וואקסן פונעם מאדעל, אזוי אז מען דארף זיך נישט זארגן וועגן אינפראסטרוקטור. עס גיט אויך זיכערהייט אויפן שטאפל פון גרויסע פירמעס, און דאס איז נויטיג פאר אפליקאציעס וואס ארבעטן מיט סענסיטיווע אינפארמאציע אדער מיט קונים.

אז מען באניצט זיך מיט GPT-4 דורך Azure, באקומט מען צוטריט צו איינעם פון די שטארקסטע AI מאדעלן וואס עקזיסטירן היינט, אן דארפן פירן א גרויסע אינפראסטרוקטור און אן דארפן אליין טריינירן א מאדעל.

וואס איז LangChain (36:06:11)

LangChain איז א שטארקער framework אין Python וואס איז געבויט כדי גרינגער צו מאכן דאס בויען אפליקאציעס וואס לויפן אויף גרויסע שפראך-מאדעלן. GPT-4 אליין קען ארויסגעבן ענטפערס לויט דער איינגאבע, אבער LangChain העלפט פראגראמירער בויען א געארדנטע אינטעראקציע מיט אים, מיט עטליכע טריט. דאס גיט LangChain:

- פירן די פראמפטן: מען קען גרינגער פירן פארשידענע פראמפטן פאר פארשידענע ארבעטן.
- זכרון און קאנטעקסט: LangChain העלפט דעם אגענט געדענקען די פריערדיגע מעלדונגען אין א שמועס, אזוי אז דער שמועס גייט ווייטער פון וואו ער האט זיך אפגעשטעלט.
- געצייג און API פארבינדונג: דער מאדעל קען ארבעטן מיט דרויסנדיגע געצייג, דאטאבאזעס און APIs.

• אייגענע ארבעט-פלוסן: מען קען באשטימען ווי דער אגענט זאל רעאגירן אין פארשידענע מצבים, אריינגערעכנט צווזאמבינדן עטליכע טריט איינס נאכן אנדערן.

בקיצור, LangChain לייגט צו א שיכט פון ארגאניזירן ארום GPT-4, און דאס העלפט בויען פולע אגענטן מיט קאמפליצירטע אויפפירונגען.

די גרונט-טריט פון אויפשטעלן דעם אגענט (36:07:13)

פאר מען הייבט אן די פראקטישע ארבעט, ווייזט דער קורס די גרונט-טריט פון אויפשטעלן א AI אגענט מיט Azure OpenAI און LangChain:

• API ענדפונקט און שליסל: כדי צו באניצן GPT-4 דורך Azure, דארף מען צו ערשט אויפשטעלן דעם API endpoint און באקומען אן API key. דאס לאזט די אפליקאציע זיך אויסווייזן און שיקן בקשות צו די Azure OpenAI סערוויסעס. די דאזיגע פרטים דארף מען האלטן זיכער.

• שאפן דעם פראמפט: א פראמפט איז א טעקסט וואס מען שיקט צום מאדעל כדי ער זאל טון א געוויסע ארבעט. עס קען זיין גאנץ פשוט, למשל: "איבערזעץ דעם זאץ good morning אויף פראנצויזיש און שפאניש". וואס בעסער דער פראמפט, אלס בעסער דער ענטפער. צו לערנען שרייבן גוטע פראמפטן איז א הויפט-קונסט ביים ארבעטן מיט שפראך-מאדעלן.

• רופן דעם מאדעל: וועו דער פראמפט איז גרייט, שיקט מען אים צום מאדעל מיט LangChain'ס invoke מעטאד. די פונקציע פארבינדט זיך מיט די Azure OpenAI סערוויסעס, שיקט דעם פראמפט און באקומט צוריק דעם ענטפער. הינטער די קוליסן פירט invoke די קאמוניקאציע מיטן API פונעם מאדעל, און גיט צוריק דעם ארויסגעגעבענעם ענטפער. דעם ענטפער קען מען ווייזן דעם באניצער, אדער ווייטער באארבעטן.

צוזאמפאסונג פון דער לעקציע (36:08:47)

דער קורס פאסט צוזאם די הויפט-פונקטן:

• א AI אגענט איז א סיסטעם וואס קען פארשטיין איינגאבן, מאכן באשלוסן און ענטפערן אויף א קלוגן אופן.

• אזעלכע אגענטן זענען געבויט אויף שפראך-מאדעלן ווי GPT-4, וואס קענען פארשטיין און שרייבן מענטשליכע שפראך.

• Azure OpenAI גיט א זיכערן און וואקסנדיגן צוטריט צו GPT-4. דאס פאסט אויסגעצייכנט פארן ארויסשטעלן AI לייזונגען פאר גרויסע פירמעס.

• LangChain העלפט ארדענען די אינטעראקציעס, פירן דעם קאנטעקסט, און בויען פארגעשריטענע ארבעט-פלוסן פארן אגענט.

• אויפשטעלן דעם אגענט באשטייט פון אויפשטעלן די APIs, שאפן פראמפטן, און רופן דעם מאדעל דורך קאוד.

פון דא אן גייט מען איבער פון טעאריע צו פראקטיק: ווי אזוי אויפצושטעלן די Azure OpenAI אינסטאנצן, אויפשטעלן LangChain, שרייבן די ערשטע פראמפטן, און ארויסשטעלן דעם אגענט צו טון אמת'דיגע ארבעטן. עס ווערט אויך דערמאנט א quiz וואס איז צוגעלייגט פאר בעסערע פארשטאנד.

אנהייב פון קאוד: אימפארטן און אויפשטעלן דעם מאדעל (36:09:51)

די נייע לעקציע לערנט ווי אזוי צו ארבעטן מיט CSV דאטא מיט א קלוגן AI אגענט: ווי אזוי צו לאדן א דאטא-סעט, אויסריינ׳גן פעלנדיגע דאטא, און באניצן AI כדי צו פרעגן פראגעס און באקומען ענטפערס פון דער דאטא.

דער קאוד הייבט אן מיט אימפארטירן צוויי גרונט-ביבליאטעקן פון Python: os און pandas. דער os מאדול לאזט דעם סקריפט ארבעטן מיט דער אפערירונג-סיסטעם. דאס איז באזונדערס נוצלעך כדי צו נעמען אויף א זיכערן אופן environment variables, ווי API keys אדער endpoint URLs. די pandas ביבליאטעק, וואס ווערט געווענליך אימפארטירט אלס pd, ווערט ברייט באנוצט פאר באארבעטן און אנאליזירן דאטא. כאטש זי ווערט נישט גלייך באנוצט אין דעם שטיקל קאוד, איז זי מסתמא דא צוליב דער דאטא וואס דער מאדעל וועט שפעטער באארבעטן.

נאכדעם ווערן אימפארטירט ווייז-געצייג פון IPython: Markdown, HTML און display . זיי זענען נוצלעך אין Jupyter notebook אדער אין אנדערע אינטעראקטיווע Python סביבות, ווען מען וויל ווייזן א שיינעם, פארמאטירטן ארויסגאנג אנשטאט פשוטן טעקסט. למשל, אויב מען וויל ווייזן א טאבעלע, ארויסווייזן דעם ענטפער פונעם מאדעל מיט גראבע אותיות, אדער ווייזן אינהאלט אלס HTML, העלפן די געצייג אז מען זאל עס גרינגער קענען ליינען.

דערנאך קומט די פארבינדונג מיט LangChain. מען אימפארטירט HumanMessage פון langchain.schema. דאס שטעלט מיט א מעלדונג וואס ווערט געשיקט צום chatbot, און העלפט אויסשפילן שמועסן אויף א ריינעם און געארדנטן אופן, ווען מען בויט שמועס-פלוסן אדער פרובירט אויס פראמפטן. דערנאך אימפארטירט מען AzureChatOpenAI פון langchain_openai. דאס איז א ספעציעלער קלאס וואס ארבעט ווי א פארבינדער צווישן LangChain און דער ווערסיע פון OpenAI וואס ווערט געהאסט אויף Azure. ער לאזט פראגראמירער נוצן מאדעלן ווי GPT-3.5 אדער GPT-4 דורך זייער Azure אינפראסטרוקטור, אויף א זיכערן און וואקסנדיגן אופן.

צום סוף שאפט מען אן אינסטאנץ פון AzureChatOpenAI. מען שטעלט אן די API ווערסיע, 2023-05-15, כדי עס זאל פאסן. מען שטעלט אויך אן דעם deployment name, וואס איז GPT4, דאס הייסט דער GPT-4 מאדעל וואס מען האט ארויסגעשטעלט אינעם Azure פארטאל. דער Azure endpoint ווערט געזומען אויף א זיכערן אופן פון אן environment variable מיט os.getenv . אזוי האלט מען סענסיטיווע אינפארמאציע, ווי URLs , ארויס פונעם קאוד. מיט דעם איז דער מאדעל גרייט צו נעמען מעלדונגען און צוריקגעבן ענטפערס מיטן GPT-4 מאטאר.

פון שמועסן צו דאטא: וואס מען קען בויען (36:13:32)

ווען דער מאדעל איז אויפגעשטעלט, קען מען אים שיקן מעלדונגען מיט HumanMessage, און באקומען קלוגע ענטפערס וואס פארשטייען דעם קאנטעקסט. דאס איז א גוטער יסוד פאר chatbots, פאר אסיסטענטן און פאר AI געצייג וואס ארבעטן מיט דאטא. דער קורס חזר׳ט איבער אז דער קאוד שטעלט אויף אלעס וואס מען דארף כדי צו שמועסן מיט GPT-4 דורך Azure׳ס פלאטפארמע, אנשטאט דורך OpenAI׳ס דירעקטן API. דערמיט קען מען בויען אלערליי קלוגע אפליקאציעס: chatbots, ווירטועלע אסיסטענטן, אויטאמאטישע באריכט-מאכער און נאך.

לאדן און אויסרייניגן דעם CSV דאטא-סעט (36:14:33)

די קומענדיגע שורה קאוד טוט צוויי זאכן מיט אמאל: זי ליינט א דאטא-סעט און רייניגט אים א ביסל אויס.

- ערשטנס, pd.read_csv מיטן פלאץ פונעם טעקע באניצט pandas כדי צו לאדן א CSV טעקע (comma-separated values) אריין אין Python. דער טעקע-וועג, data/all-states-history.csv, ווייזט אויף א טעקע אינעם פאלדער data אינעם יעצטיגן ארבעט-פאלדער. וועז די שורה לויפט, ליינט זי אין די גאנצע טעקע און שפייכלט זי אלס א DataFrame. דאס איז ענליך צו א טאבעלע, נאר אינעווייניג אין Python. די טאבעלע קען אנטהאלטן רייען פון טעגליכע COVID דאטא פאר פארשידענע שטאטן.
- דערנאך קומט .fillna(value=0), וואס איז אן אויסרייניגונג. אמאל פעלן אין א דאטא-סעט וועלטן, דאס הייסט עס זענען דא ליידיגע צעלן. אזעלכע צעלן קענען ברענגען טעותים אדער פאלשע רעזולטאטן ביים אנאליזירן. דער קאוד גייט דורך די גאנצע DataFrame און פארביט יעדן פעלנדיגן וועלט (NaN, "not a number") מיט א נול. דאס איז א שנעלער וועג צו זיכער מאכן אז דער דאטא-סעט איז פולשטענדיג און מען קען זיכער מיט אים ארבעטן.

בכלל ברענגט די שורה אריין די COVID דאטא אין Python, און פילט אויס אלע ליידיגע צעלן מיט נולן. אזוי איז די דאטא ריין, פולשטענדיג און גרייט פאר אנאליז.

בויען א pandas DataFrame אגענט (36:16:39)

דערנאך אימפארטירט מען AgentType פון LangChain. כאטש דער אימפארט ווערט נישט גלייך באנוצט אין די קומענדיגע טריט, ווערט ער אפט צוגעלייגט ווען מען וויל באשטימען אדער צופאסן וועלכער סארט אגענט מען האט: א שמועס-אגענט, אדער איינער וואס ארבעט מיט געצייג ווי pandas. מען קען עס זען ווי א וועג צו באצייכענען אדער איינשטעלן ווי דער אגענט פירט זיך.

נאכדעם אימפארטירט מען א זייער נוצלעכע פונקציע, create_pandas_dataframe_agent , פון langchain_experimental.agents.agent_toolkits, LangChain'ס עקספערימענטאלן מאדול. די פונקציע שאפט אן אגענט וואס קען ליינען, פארשטיין און ארבעטן מיט א pandas DataFrame . אין עסענץ מאכט זי GPT-4 פעיג צו ארבעטן ווי א קלוגער דאטא-אנאליטיקער: ער פארשטייט די טאבעלע מיט דער דאטא, און קען ענטפערן פראגעס דערויף נאר דורכן ליינען פראמפטן אין פשוטן ענגליש.

דאן בויט מען דעם אגענט אליין:

```
create_pandas_dataframe_agent(llm=model, df=df, verbose=True) = agent
```

מען גיט אריין צוויי הויפט-באשטאנדטיילן. llm=model איז דער גרויסער שפראך-מאדעל וואס מען האט פרייער אויפגעשטעלט, GPT-4 אויף Azure. df=df איז די DataFrame מיטן COVID-19 דאטא-סעט. אזוי זאגט מען LangChain צו פארבינדן די כח פון GPT-4 מיט דער סטרוקטור פון דער דאטא. די אפציע verbose=True מיינט אז דער אגענט וועט ווייזן זיין אינעווייניגסטן געדאנקען-גאנג ביים ענטפערן. דאס איז זייער גוט צום לערנען און צום געפינען טעותים.

צום סוף פרובירט מען אויס דעם אגענט מיט)"how many rows are there?"agent.invoke(. מען פרעגט א פראגע אין נאטירלעכער שפראך, פונקט ווי מען רעדט מיט א מיטארבעטער, און דער אגענט פארטייטשט הינטער די קוליסן די פראגע אויף די פאסיגע pandas אפעראציע. אין דעם פאל קוקט ער וויפיל רייען (רעקארדס) עס זענען דא אינעם דאטא-סעט, און גיט צוריק די צאל, אן דעם וואס מען זאל דארפן שרייבן .shape אדער אנדערן טעכנישן קאוד.

פראמפט-פרעפיקס, סופיקס און א פראגע וועגן טעקסאס (36:19:17)

דער קומענדיגער טייל קאוד באשטימט א CSV_PROMPT_PREFIX, וואס דינט ווי אן איינפירונג צו די אנווייזונגען פארן אגענט. ער זאגט דעם AI צו ערשט איינשטעלן די display אפציעס פון pandas, אזוי אז ער זאל קענען זען אלע קאלאמען קלאר, און נאכדעם נעמען די נעמען פון די קאלאמען. דער טריט מאכט זיכער אז דער AI פארשטייט די סטרוקטור פונעם דאטא-סעט פאר ער פרובירט ענטפערן אויף פראגעס. דער משל וואס ווערט געברענגט: עס איז ווי צו זאגן א מענטשליכן אנאליטיקער, "קוק צו ערשט איבער דעם גאנצן spreadsheet און ווער באקאנט מיט די קעפלעך, פאר דו טוקסט זיך איין אין די ציפערן".

דערנאך באשטימט מען CSV_PROMPT_SUFFIX. דאס איז דער וויכטיגסטער טייל, וואס לערנט דעם AI צו טראכטן פארזיכטיג און מיט א סדר:

- פאר ער גיט דעם לעצטן ענטפער, מוז ער פרובירן כאטש צוויי פארשידענע מעטאדן צו אויסרעכענען דעם רעזולטאט.
- אויב די מעטאדן גיבן נישט דעם זעלבן רעזולטאט, זאל ער זיך איבערטראכטן און פרובירן נאכאמאל, ביז ער באקומט גלייכע ענטפערס.
- אויב מען קען דאך נישט דערגרייכן גלייכקייט, זאל ער מודה זיין אז ער איז נישט זיכער.
- אויב די ענטפערס שטימען און ער איז זיכער, זאל ער געבן א גוט-פארמאטירטן ענטפער אין markdown, מיט א פרטימדיגער ערקלערונג.
- דאס וויכטיגסטע: דער אגענט באקומט אן אנווייזונג נישט צו נוצן קיין פריערדיגע וויסן, נאר דאס וואס ער לערנט זיך פון דער דאטא וואס ער אנאליזירט יעצט.

דערנאך קומט די פראגע: וויפיל פאציענטן זענען געווען אין שפיטאל אין יולי 2020 אין טעקסאס, און אין גאנץ לאנד, אלס סך-הכל פון אלע שטאטן? מען זאל באניצן די קאלאמע hospitalizedIncrease. די פראגע פאדערט פונעם אגענט אויסצורעכענען צוויי ווערטן: ערשטנס, די סומע פון שפיטאל-אויפנאמען אין טעקסאס אין יולי 2020; צווייטנס, די סומע פון שפיטאל-אויפנאמען אין אלע שטאטן אין דער זעלבער צייט. מען האט קלאר אנגעזאגט וועלכע קאלאמע צו נוצן, hospitalizedIncrease, כדי דער AI זאל נעמען די דאטא פונעם ריכטיגן פלאץ.

צום סוף רופט מען agent.invoke, וואו מען לייגט צוזאם דעם CSV_PROMPT_PREFIX, די פראגע און דעם סופיקס, און שיקט די פולע אנווייזונג צום אגענט וואס לויפט אויף GPT-4. דער פרעפיקס איז דאס אויפשטעלן, דערנאך קומט די פראגע, און דער סופיקס באשטימט דעם געדאנקען-גאנג.

אגענט וואס רעדט מיט א SQL דאטא-באזע (36:21:52)

סוף פון דער CSV-לעקציע: א פראמפט וואס מאכט פונעם אגענט א פארזיכטיגן אנאליטיקער (36:21:52)

דער קורס פארענדיקט דא די פריערדיגע לעקציע, וואו מען האט צונויפגעשטעלט אלע אנווייזונגען אין איין קלוגן פראמפט. ווען דער אגענט באקומט אזא פראמפט, גייט ער נאך די אנווייזונגען שריט נאך שריט: ער קאנטראלירט די קאלאמען, רעכנט אויס דעם רעזולטאט מיט כאטש צוויי באזונדערע מעטאדן, פארגלייכט די רעזולטאטן, און ערשט דערנאך גיט ער א גוט-סטרוקטורירטן ענטפער.

עס ווערט ערקלערט אז דער ציל איז נישט סתם צו באקומען א שנעלן ענטפער. מען לערנט דעם אגענט זיך צו פירן ווי א פארזיכטיגער דאטא-אנאליטיקער: איינער וואס איבערקוקט זיין ארבעט, ערקלערט ווי ער איז אנגעקומען צום רעזולטאט, און מאכט נישט קיין האנחות. דאס ווערט אנגערופן א שיינע באנוץ פון פראמפט-אינזשעניערינג, כדי צו באקומען AI-ענטפערס וואס זענען פארלעסליך, דורכזיכטיג און באזירט אויף אמת'דיגע דאטא.

וועמ מען לויפט דעם גאנצן קאוד, זעט מען דעם רעזולטאט, און מען זעט אויך די "observations" (באמערקונגען) וואס דער אגענט דרוקט ארויס.

צום סוף פון דער לעקציע ווערט איבערגעחזרט וואס מען האט גערלערנט: ווי אזוי צו שטעלן א AI-אגענט וואס העלפט אנאליזירן דאטא פון א CSV-טעקע. מען האט אריינגעלאדנט די דאטא, אויסגעפילט פעלנדיקע אינפארמאציע, און געשטעלט פראגעס צום אגענט כדי צו באקומען ענטפערס. דער קורס מוטיגט ווייטער צו פראקטיצירן, כדי מען זאל קענען לייכט לייזן אלערליי דאטא-פראבלעמען.

לעקציע דריי: פארוואס פארבינדן AI מיט א SQL דאטא-באזע (36:22:56)

די דריטע לעקציע הייבט זיך אן מיט דער טעמע: ווי אזוי צו פארבינדן א AI מיט א SQL דאטא-באזע. דער פלאן איז צו בויען א סיסטעם וואו מען קען פרעגן א פשוט'ע פראגע אין גלאטן ענגליש, למשל: "ווי פיל פאציענטן זענען געווען אין שפיטאל אין New York אין אקטאבער 2020?", און דער AI וועט אליין אויסשרייבן און לויפן די SQL-פראגעס (queries) און ברענגען דעם ענטפער. דאס שפארט א סך צייט, ווייל מען דארף שוין נישט אליין שרייבן קאמפליצירטע SQL-queries.

די אימפארטן: OS, IPython און LangChain (36:23:27)

דער קורס גייט דורך דעם קאוד שורה נאך שורה:

- import os – דעם מאדול האט מען שוין געזען פריער. ער לאזט Python רעדן מיט דער אפערירונג-סיסטעם. מערסטנס באנוצט מען אים כדי צו לייענען "environment variables", למשל דעם Azure API-שליסל אדער די פארבינדונג-שטריקלעך (connection strings) פון דער דאטא-באזע. אזוי דארף מען נישט אריינשרייבן סודות'דיגע אינפארמאציע גלייך אין קאוד. דאס איז א נארמאלע פראקטיק כדי די אפליקאציע זאל זיין זיכער און בייגעוודיג.
- from IPython.display import Markdown, HTML, display – די אימפארטן זענען בעיקר נוצלעך ווען מען ארבעט אין Jupyter notebooks אדער ענליכע סביבות. זיי לאזן ארויסווייזן דעם רעזולטאט אויף א שיין-פארמאטירטן אופן: אנשטאט צו דרוקן א פשוט'ן טעקסט, קען מען ווייזן Markdown-קעפלעך, פעטע אותיות אדער HTML-טאבעלעס. דאס מאכט די AI-ענטפערס ריין און גרינג צו לייענען.
- from langchain.chat_models import AzureChatOpenAI – דאס ברענגט אריין דעם אינטערפעיס צו באנוצן די GPT-מאדעלן פון OpenAI דורך Azure. LangChain שטיצט סיי דעם אריגינעלן OpenAI API, סיי דעם Azure-ווערסיע; דא קלייבט מען אויס בפירוש דעם מאדעל וואס איז געהאסט אויף Azure. דאס מיינט אז דער מאדעל לויפט דורך דעם Microsoft-וואלקן (cloud), וואס גרויסע פירמעס האבן אפט בעסער ליב צוליב תקנות (compliance) און צוליב איינפלעכטונג מיט זייערע אנדערע סיסטעמען.
- from langchain.agents import create_sql_agent – דא, ווי דער קורס זאגט, "הייבט זיך אן די שמחה". די פונקציע לאזט שאפן א אגענט וואס ארבעט מיט א דאטא-באזע דורך נאטירלעכע שפראך. מיט פשוט'ע

וועדטער: מען שטעלט אויף א AI וואס פארשטייט אן ענגלישע פראגע, למשל "וועלכער פראדוקט האט געהאט די העכסטע פארקויפן אין יוני", און פארוואנדלט זי הינטער די קוליסן אין אמת'דיגע SQL-queries.

• from langchain.agents.agent_toolkits import SQLDatabaseToolkit – דאס איז א ספעציעלער גענעראטאר (toolkit) געמאכט צו פארבינדן LangChain-אגענטן מיט SQL דאטא-באזעס. ער גיט מכשירים און הילפס-פונקציעס וואס בויען א בריק צווישן שפראך און SQL, אזוי אז דער אגענט ווייסט ווי אזוי צו רעדן מיט דער דאטא-באזע, ארויסנעמען די סכעמע-אינפארמאציע (schema) און שאפן ריכטיגע queries.

• from langchain.sql_database import SQLDatabase – די שורה איז וויכטיג ווייל זי דעפינירט די אמת'דיגע פארבינדונג צו דער SQL דאטא-באזע. מען באנוצט עס צו פארבינדן זיך מיט סערווערס ווי PostgreSQL, MySQL, SQLite א.א.וו., כדי צו געבן דעם אגענט צוטריט צו די דאטא-טאבעלעס. ווען מען איז פארבונדן, ווערט דער אביעקט דער "טויער" דורך וועלכן דער אגענט פארשט די דאטא און לויפט queries.

• from langchain_openai import AzureChatOpenAI – דאס זעט אויס ווי א דאפלטער אימפארט פון פריער, אבער ער קומט פון אן אנדער פאד. לויט דער LangChain-ווערסיע וואס מען באנוצט, קען זיין אז מען באנוצט langchain_openai אלס א באזונדער פעקל, וואס האט ספעציעלע שטיצע און פארבעסערונגען פאר Azure. בקיצור, דאס פארזיכערט אז דער אגענט באנוצט דעם יעצטיגן GPT-4 מאדעל וואס איז אויפגעשטעלט אויף Azure.

אין א סך-הכל ערקלערט דער קורס: דער קאוד שטעלט אלעס צוגרייט צו שאפן א AI-אגענט וואס פארבינדט זיך מיט א SQL דאטא-באזע און ענטפערט אויף פראגעס אין גלאטן ענגליש. מיט GPT דורך Azure קען מען שטעלן נאטירלעכע פראגעס, און דער אגענט שרייבט און לויפט די SQL-queries פאר אייך, ברענגט די רעזולטאטן, און ערקלערט וואס ער האט געטון. דאס איז א ריזיגער צייט-שפארער פאר דאטא-גרופעס, אנאליטיקער, און יעדן וואס וויל ארבעטן מיט דאטא-באזעס אן אליין שרייבן queries.

SQLAlchemy, pandas און דאס לאדנען די COVID-דאטא (36:27:37)

• from sqlalchemy import create_engine – דאס ברענגט אריין א שטארקן מכשיר פון דער SQLAlchemy ביבליאטעק. די פונקציע create_engine שאפט א פארבינדונג צווישן דעם Python-קאוד און א SQL דאטא-באזע, צי מען באנוצט MySQL, PostgreSQL, SQLite אדער אן אנדער סיסטעם. דער "engine" איז א בריק וואס לאזט שיקן SQL-באפעלן פון Python און באקומען רעזולטאטן אויף אן עפעקטיוון אופן. עס ווערט באצייכנט אלס איינע פון די פארלעסליכסטע און מערסט-באנוצטע מעטאדן צו ארבעטן מיט דאטא-באזעס אין Python.

• import pandas as pd – דאס ברענגט אריין pandas מיטן געווײנטלעכן קיצור pd. pandas איז נויטיג פאר ארבעט מיט טאבעל-דאטא אין Python: לאדנען, אנאליזירן, רייניגן און באארבעטן דאטא-סעטס, אויף אן אופן וואס איז סיי שטארק סיי באקוועם. עס איז בעיקר נוצלעך ביי גרויסע CSV-טעקעס אדער ווען מען פירט אריבער דאטא צו און פון דער דאטא-באזע.

• df = pd.read_csv(...).fillna(0) – די שורה טוט צוויי זאכן. ערשטנס לייענט זי די CSV-טעקע פונעם data-פאלדער מיט pd.read_csv און לאדנט זי אריין אין א pandas DataFrame וואס הייסט df. די טעקע אנטהאלט, לויט דעם קורס, היסטארישע COVID-19 דאטא פאר אלע אמעריקאנער שטאטן. צווייטנס,

fillna רייניגט די דאטא דורך פארביטן יעדן פעלנדיקן אדער ליידיקן ווערט מיטן נומער 0. דאס פארזיכערט אז ווייטערדיגע חשבונות אדער queries וועלן נישט געשטערט ווערן דורך פעלנדיקע ווערטן.

אין א סך-הכל: דער קאוד גרייט צו א דאטא-פראיעקט וואו מען פארבינדט זיך מיט א דאטא-באזע און ארבעט מיט ווירקלעכע דאטא-סעטס. מען האט די מכשירים צו לייענען סטרוקטורירטע דאטא פון א טעקע, זי רייניגן, און זי אדער אנאליזירן אדער אריינשטופן אין א SQL דאטא-באזע פאר קאמפליצירטערע queries.

שאפן די SQLite דאטא-באזע און אריינשרייבן די טאבעלע (36:29:44)

- database_file_path = "db/test.db" – דאס באשטימט וואו די SQLite דאטא-באזע-טעקע געפינט זיך. SQLite איז א לייכטע, טעקע-באזירטע דאטא-באזע-סיסטעם, דאס הייסט אז אלעס ווערט געהאלטן אין איין איינציגער .db טעקע. דא באשטימט מען אז די דאטא-באזע זאל זיין אין א פאלדער וואס הייסט db, מיטן נאמען test.db. אויב די טעקע עקזיסטירט נאך נישט, שאפט SQLite זי אויטאמאטיש ביים פארבינדן.
- engine = create_engine(f"sqlite:///{database_file_path}") – דאס שאפט א פארבינדונג-engine מיט SQLAlchemy, דורך וועלכן Python רעדט מיט דער SQLite דאטא-באזע. דער פארמאט "sqlite:///" מיט א פאד דערנאך איז דער סטאנדארט connection string פאר SQLite . מען זאגט אזוי צו זאגן: "הערסט, Python, דא וואוינט די דאטא-באזע, באנוץ דעם engine צו שיקן און באקומען דאטא."
- דערנאך לאדנט מען די דאטא מיט א file URL – וועלכע טעקע מען האט ליב – און df = pd.read_csv(file_url).fillna(value=0). דאס איז דאס זעלבע ווי פריער: מען לייענט די CSV-טעקע מיט די היסטארישע COVID-19 דאטא אריין אין א DataFrame וואס הייסט df, און fillna פארביט פעלנדיקע ווערטן מיט נולן, כדי צו פארמיידן פראבלעמען ביי אנאליז אדער ביים אוועקלייגן די דאטא אין א דאטא-באזע.
- צום סוף, df.to_sql – דא וועלט די דאטא אריינגעשריבן אין דער דאטא-באזע:

df.to_sql("all_states_history", con=engine, if_exists="replace", index=False)

"all_states_history" איז דער נאמען פון דער טאבעלע וואס מען שאפט. con=engine זאגט פאר pandas צו באנוצן דעם SQLAlchemy engine וואס מען האט יעצט געשאפן. if_exists="replace" מיינט: אויב א טאבעלע מיטן זעלבן נאמען עקזיסטירט שוין, פארביט זי מיט די נייע דאטא. index=False פארזיכערט אז pandas שרייבט נישט אריין זיין שורה-אינדעקס אלס א באזונדערע קאלאמע אין דער דאטא-באזע.

דער סיסטעם-פראמפט MSSQL_AGENT_PREFIX: כללים פארן אגענט (36:31:49)

דערנאך קומט א פראמפט וואס הייסט MSSQL_AGENT_PREFIX. דאס איז א סיסטעם-פראמפט געשריבן צו אנווייזן א AI-אגענט ווי אזוי צו ארבעטן זיכער און גענוי מיט א SQL דאטא-באזע, ספעציעל Microsoft SQL Server אדער סיי וועלכן אנדערן דיאלעקט וואס ווערט באשטימט דינאמיש. דער קורס גייט דורך אלע כללים אין פראמפט:

- ער הייבט זיך אן מיט באשטימען די ראלע: "You are an agent designed to interact with a SQL database." דאס זאגט פארן מאדעל גענוי וואס ער דארף טון, אנשטאט צו זיין אן אלגעמיינער אסיסטענט; דער מאדעל וועט זיך קאנצענטרירן אויף SQL-queries לויט דעם באניצערס פראגע.
- וען א באניצער פרעגט א פראגע, מוז דער אגענט שאפן א ריכטיגע SQL-query אין דעם באשטימטן דיאלעקט (למשל T-SQL), זי לויפן, קוקן אויפן רעזולטאט און צוריקגעבן א ענטפער.

• סיידן דער באניצער בעט בפירוש א געוויסע צאל רעקארדס, דארף דער אגענט שטענדיג באגרעניצן דעם רעזולטאט צו א פאסיגער צאל, כדי נישט צו פארפלייצן דעם באניצער מיט צו פיל דאטא.

• דער אגענט זאל סארטירן די רעזולטאטן לויט א רעלעוואנטער קאלאמע, כדי דער באניצער זאל זען די וויכטיגסטע וועדטן צום ערשט.

• מען זאל אויסמיידן "SELECT *" (אלע קאלאמען); דער אגענט זאל בעטן נאר די קאלאמען וואס זענען נויטיג פאר דער פראגע. דאס פארבעסערט סיי די שנעלקייט סיי די קלארקייט פונעם רעזולטאט.

• דער פראמפט דערמאנט אז דער מאדעל האט צוטריט צו מכשירים פאר ארבעט מיט דער דאטא-באזע, אבער ער מוז שטענדיג איבערקוקן זיינע queries איידער ער לויפט זיי. אויב א query פאלט דורך, זאל ער פארריכטן דעם טעות און פרובירן נאכאמאל, אנשטאט אויפהערן. דאס מאכט דעם AI מער פעסט און פעאיג צו לערנען פון טעותים בעת דער ארבעט.

• איינער פון די וויכטיגסטע זיכערהייט-כללים: דער אגענט טאר קיינמאל נישט לויפן DML -באפעלן ווי INSERT, UPDATE, DELETE אדער DROP. די באפעלן קענען טוישן אדער אויסמעקן דאטא, און וויבאלד דער אגענט איז געמאכט נאר צו לייענען און אנאליזירן, זענען אזעלכע אקציעס שטרענג פארבאטן.

• דער אגענט טאר קיינמאל נישט טרעפן אדער זיך פארלאזן אויף פריערדיגע וויסן; ער מוז ענטפערן נאר לויט די אמת'דיגע רעזולטאטן פון די queries וואס ער לויפט. דאס היט אפ די גלייבווערדיגקייט, און די ענטפערס זענען באזירט אויף ווירקלעכע דאטא און נישט אויף השערות.

• דער ענטפער זאל צוריקקומען אין Markdown-פארמאט, כדי עס זאל זיין ריין און גרינג צו לייענען. אבער דער מאדעל טאר נישט באנוצן "backticks" ווען ער שיקט SQL-באפעלן צו דער דאטא-באזע; די זענען נאר פארן פארמאטירן דעם לעצטן ענטפער וואס דער באניצער זעט, נישט פארן לויפן די query.

• יעדער ענטפער מוז האבן אן "Explanation"-אפטייל, וואו דער אגענט ערקלערט ווי אזוי ער איז אנגעקומען צום ענטפער, און לייגט צו די אמת'דיגע query וואס ער האט באנוצט.

• אויב די פראגע האט נישט צו טון מיט דער דאטא-באזע, אדער ס'איז נישטא גענוג קאנטעקסט צו ענטפערן, זאל דער אגענט פשוט ענטפערן "I don't know".

• דער אגענט זאל באנוצן נאר טאבעלע-נעמען וואס די מכשירים גיבן אים, און קיינמאל נישט אויסטראכטן אדער אננעמען אז א טאבעלע עקזיסטירט.

דער פראמפט MSSQL_AGENT_FORMAT_INSTRUCTIONS: די סטרוקטור פונעם דענקען (36:35:28)

דער צווייטער פראמפט, MSSQL_AGENT_FORMAT_INSTRUCTIONS, איז געמאכט צו אנווייזן דעם AI-אגענט ווי אזוי צו פארמאטירן זיין דענק-פראצעס און זיינע ענטפערס ביי ארבעט מיט א SQL דאטא-באזע. ער פארזיכערט אז יעדער שריט – פונעם פארשטיין די פראגע, ביזן לויפן די queries און אנקומען צום לעצטן ענטפער – איז דורכזיכטיג, לאגיש און גרינג נאכצופאלגן. דאס איז בעיקר וויכטיג ביי דאטא-באזע-ארבעט, וואו געניקייט און די מעגלעכקייט נאכצושפירן יעדן שריט זענען קריטיש.

די סטרוקטור גייט אזוי:

• Question – די אריגינעלע פראגע פונעם באניצער.

• Thought – דער אגענט'ס איבערטראכטונג, וואס ער טראכט אדער פלאנירט צו טון ווייטער.

- Action – וועלכע מכשיר אדער מעטאד דער אגענט וועט באנוצן.
- Action Input – די אמת'דיגע SQL-query וואס דער אגענט וויל לויפן.
- Observation – דער רעזולטאט וואס די דאטא-באזע האט צוריקגעגעבן נאכן לויפן די query.

די שריטן Thought, Action, Action Input און Observation קענען זיך איבערחזרן עטליכע מאל, וועז דער אגענט פרובירט פארשידענע סטראטעגיעס אדער פארבעסערונגען ביז ער קומט צום ריכטיגן ענטפער. וועז דער אגענט איז זיכער, שליסט ער מיט א Final Thought וואס זאגט אז ער וויסט דעם ענטפער, און דערנאך דער Final Answer אליין. דער ענטפער דארף זיין באזירט בלויז אויף די באמערקונגען (observations) פונעם לויפן די queries, קיינמאל נישט אויף האנחות אדער אויף וויסן פונעם טרייגירן. די סטרוקטור פארזיכערט אז אלעס וואס פירט צום לעצטן ענטפער איז פארשריבן און לאגיש באגרינדעט.

דאס ביישפיל אין פראמפט: COVID-טויטפעלן אין Texas (36:37:01)

דער פראמפט אנטהאלט אויך א ביישפיל-שמועס, כדי דער אגענט זאל געניי פארשטיין ווי אזוי צו ענטפערן. אין ביישפיל ווערט דער אגענט געפרעגט ווי פיל מענטשן זענען געשטארבן פון COVID-19 אין Texas אין 2020. ער בויט א SQL-query וואס קלייבט אויס די death-קאלאמע פון דער COVID-tracking טאבעלע, וואו דער state איז "TX" און די דאטע הייבט זיך אן מיט 2020. די צוריקגעקומענע ווערטן ווערן אויסגערעכנט, און דער אגענט באשליסט די סך-הכל צאל דורך זיי צוזאמענרעכענען. דער לעצטער ענטפער דארף זיין 27437.

דערנאך ווערט אנגעהויבן צו לויפן דעם קאוד, כדי צו זען ווי אזוי עס ארבעט אין פראקטיק.

דאטאבאַזע-אגענטן: SQL, function calling און Assistants API (36:38:02)

דער LLM, די דאטאבאַזע און דער SQL-טולקיט (36:38:02)

דער קורס ערקלערט וויטער דעם בלאק קאוד וואס איניציאליזירט דעם שפראך-מאדעל. דער וואַריאַבל LLM באקומט אן אינסטאַנץ פון AzureChatOpenAI. דאס איז LangChain'ס וועג זיך צו פארבינדן מיט א GPT-מאדעל וואס איז אויפגעשטעלט (deployed) אויף Microsoft Azure.

עטליכע פאראמעטערס ווערן ערקלערט:

- די OpenAI API ווערסיע 2023-05-15 באשטימט מיט וועלכע ווערסיע פון API מען ארבעט.
- דער Azure deployment מיטן נאמען GPT-4-1106 איז דער נאמען וואס מען האט אליין געגעבן צו דעם GPT-4 deployment אין דער אייגענער Azure סביבה.
- דער Azure endpoint ווערט געשטעלט צו os.getenv. דאס ברענגט דעם endpoint URL אויף א זיכערן אופן פון אן environment variable, אזוי אז די credentials און endpoints בלייבן באשיצט.
- temperature און max tokens קאנטראלירן ווי דער מאדעל פירט זיך. temperature=0 מאכט דעם מאדעל מער דעטערמיניסטיש: פאר דעם זעלבן אינפוט וועט ער אלעמאל געבן דעם זעלבן ענטפער. דאס איז אידעאל וועז מען פרעגט סטרוקטורירטע דאטא, למשל א דאטאבאַזע.
- max_tokens=500 באגרעניצט די לענג פונעם ענטפער. אזוי קאנטראלירט מען די פערפארמענס און די גרייס פונעם רעזולטאט.

דערנאך פארבינדט מען זיך צו דער SQLite דאטאבאַזע דורך SQLDatabase.from_uri . דאס זאגט LangChain וואו די דאטאבאַזע געפינט זיך. SQLite איז א לייכטע דאטאבאַזע וואס איז געבויט אויף א טעקע, און דא ווייזט מען פשוט אויף דער DB-טעקע וואס מען האט פריער דעפינירט. דער DB-אביעקט וואס מען באקומט לאזט דעם AI-אגענט אויספארשן די טאבעלעס, פארשטיין די סכעמע און לויפן SQL queries.

צום סוף שאפט מען א טולקיט מיט SQLDatabaseToolkit. דער קורס פארגלייכט דעם טולקיט צו א מעתיק (איבערזעצער) צווישן דעם שפראך-מאדעל און דער SQL דאטאבאַזע. ער בינדט צוזאמען דעם GPT-מאדעל מיט דער פארבונדענער דאטאבאַזע, אז זיי זאלן קענען ארבעטן בשותפות. מיט דעם טולקיט קען דער אגענט באקומען א שאלה אין פשוטן ענגליש, שרייבן דעם פאסיגן SQL query, אים לויפן אויף דער דאטאבאַזע, און צוריקגעבן א גוט-ערקלערטן ענטפער. אלעס געשעט אויטאמאטיש.

די שאלה און דער SQL agent executor (36:40:08)

דערנאך דעפינירט מען די שאלה וואס דער אגענט דארף ענטפערן. די שאלה איז קלאר און ספעציפיש: וויפיל פאציענטן זענען געווען אין שפיטאל אין New York אין אקטאבער 2020, און וויפיל האספיטאליזאַציעס זענען געווען אין אלע סטעיטס צוזאמען אין דער זעלבער צייט. די שאלה זאגט אויך אז דער ענטפער דארף זיך באזירן אויף א קאלום מיטן נאמען hospitalizedIncrease. דער קאלום אנטהאלט ווארשיינליך טעגליכע דאטא איבער דעם צווואוקס אין האספיטאליזאַציעס. אזוי שטעלט מען דעם קאנטעקסט, און דער אגענט ווייסט פונקטליך וועלכע דאטא צו נוצן.

דערנאך שטעלט מען אויף א SQL agent executor. דא קאנפיגורירט מען דעם אגענט צו ארבעטן סיי מיטן שפראך-מאדעל (GPT-4) סיי מיט דער דאטאבאַזע. דער אגענט באקומט דעטאלירטע אנווייזונגען ווי צו טראכטן, ווי צו באארבעטן א שאלה, און ווי צו האנדלען. די אנווייזונגען זארגן אז ער זאל נוצן די ריכטיגע SQL-געצייג: שרייבן א query לויט דער שאלה, ארויסנעמען רעזולטאטן פון דער דאטאבאַזע, און געבן א ענדגילטיגן ענטפער.

מען שטעלט אויך געוויסע פאראמעטערס, למשל צו באגרעניצן די צאל רעזולטאטן צו 30, און צו לאזן דעם אגענט ווייזן זיין טראכט-פראצעס אין verbose מאדע. דאס איז נוציק פאר debugging.

דער טראכט-פראצעס און די אקציעס פונעם אגענט זענען דעפינירט מיט אכטונג, כדי ער זאל ארבעטן מעטאדיש:

- אנאליזירן די שאלה
- אויסטראכטן א פאסיגן SQL query
- לויפן דעם query אויף דער דאטאבאַזע
- קאנטראלירן דעם רעזולטאט

דער אגענט ווערט אויך אנגעזאגט צו פארמאטירן דעם ענדגילטיגן ענטפער קלאר און קורץ, צוזאמען מיט אן ערקלערונג ווי ער איז אנגעקומען צום רעזולטאט. אזוי באקומט דער באניצער נישט נאר א ענטפער, נאר ער פארשטייט אויך ווי דער אגענט איז צו אים אנגעקומען.

לויפן דעם אגענט און דער רעזולטאט (36:41:41)

וועז מען רופט agent_executor_SQL.invoke(question), הייבט דער אגענט אן צו באארבעטן די שאלה אין נאטירלעכער שפראך. ער טייטשט אויס אז די שאלה פרעגט וועגן האספיטאליזאַציעס אין New York און

איבער גאנץ לאנד, און באשטימט וועלכע דאטא ער דארף. דערנאך שרייבט ער א SQL query און לויפט אים אויף דער פארבונדענער דאטאבאַזע. ער נעמט ארויס נאר די נויטיגע רעזולטאטן, לויט די פארמאט-אנווייזונגען. צום סוף שטעלט ער פאר דעם ענדגילטיגן ענטפער אין markdown פארמאט, צוזאמען מיט אן ערקלערונג ווי ער איז אנגעקומען צום רעזולטאט, אריינגערעכנט דעם SQL query.

אין דעם רעזולטאט זעט מען די צאל 53485 פאציענטן וואס זענען געווען אין שפיטאל איבער גאנץ לאנד, אין אלע סטעיטס. דאס איז דער ענדגילטיגער ענטפער וואס דער אגענט האט אויסגעטייטשט.

אין דער לעקציע האט מען פארבונדן דעם AI-אגענט צו א SQL דאטאבאַזע און אים געלערנט צו ענטפערן שאלות וועגן דאטא אין נאטירלעכער שפראך. מען האט אויפגעשטעלט די געצייג, געשריבן א query וועגן האספיטאליזאַציעס, און געלאזט דעם AI באארבעטן די אינפארמאציע און צוריקגעבן א קלארן, פארשטענדלעכן ענטפער. דער קורס מוטיגט ווייטער צו עקספערימענטירן מיט queries.

Function calling אין Azure OpenAI – דאס וועטער-ביישפיל (36:43:46)

די נעקסטע לעקציע עפנט א פונקציע פון Azure OpenAI וואס הייסט function calling. דאס לאזט דעם AI טון מער ווי נאר שמועסן: ער קען ברענגען דאטא אין עכטער צייט, אדער דורכפירן אקציעס, למשל קאנטראלירן דאס וועטער אדער ארויסנעמען געזונט-דאטא. מען גייט דורך דעם קאוד בלאק נאך בלאק:

- דער ערשטער בלאק שטעלט אויף א זיכערע פארבינדונג צו Azure OpenAI מיט environment variables. דער AzureOpenAI קליענט ווערט איניציאליזירט צו רעדן מיט GPT-מאדעלן וואס זענען אויפגעשטעלט אויף Azure.
- א פונקציע געבט צוריק נאכגעמאכטע (mock) וועטער-דאטא פאר געוויסע שטעט. אויב דער ארט איז נישט פאראויס דעפינירט, געבט זי צוריק "unknown" אלס טעמפעראטור.
- א פראבע-רוף טעסט די פונקציע פאר New York. זי דארף צוריקגעבן א פאראויס-באשטימטע (hardcoded) טעמפעראטור אין Fahrenheit.
- א באניצער-מעסעדזש ווערט דעפינירט וואס מען שיקט צום מאדעל. ער פרעגט וועגן דעם וועטער אין דריי שטעט.
- די דעפיניציע פונעם וועטער-געצייג באשרייבט די סכעמע פון דער פונקציע get_current_weather. אזוי פארשטייט דער AI וען און ווי אזוי צו רופן זי.
- די ערשטע chat request שיקט דעם באניצער-מעסעדזש צוזאמען מיט דער געצייג-דעפיניציע צו GPT-4. דער מאדעל באשליסט אליין צי ער וויל נוצן די וועטער-פונקציע.
- באהאנדלען דעם tool call: אויב GPT-4 באשליסט צו נוצן דעם געצייג, נעמט דער קאוד ארויס דעם פונקציע-רוף, לויפט אים, און לייגט צו דעם רעזולטאט צו דער שמועס-היסטאריע.
- א צווייטע completion ווערט געשיקט צום מאדעל מיט די דערהיינטיגטע מעסעדזשעס, וואס אנטהאלטן שוין די רעזולטאטן פונעם געצייג. GPT-4 ענטפערט מיטן פולן קאנטעקסט.

Function calling מיט COVID-דאטא אין SQLite (36:45:49)

דער צווייטער טייל פון דער לעקציע וועגדט די זעלבע טעכניק אן אויף דאטאבאַזע-דאטא:

- לאדן דעם דאטא-סעט: COVID-דאטא ווערט געלאדן פון א CSV אין א pandas DataFrame, און ליידיגע ווערטן ווערן אויסגעפילט מיט נול.

• דער צענטער שריט: שאפן א SQLite דאטאבאַזע. דער קאוד היט אפ דעם DataFrame אלס טאבעלע, און וועט ס'עקזיסטירט שוין א טאבעלע מיטן זעלבן נאמען, ווערט זי פארביטן.

• א פונקציע פרעגט די האספיטאליזאַציע-צוווואוקס: זי ברענגט די צאל נייע האספיטאליזאַציעס פאר א געוויסן אמעריקאנער סטעיט און דאטום פון דער דאטאבאַזע.

• א צווייטע פונקציע ברענגט די צאל נייע positive COVID-19 פעלער פאר א געוויסן סטעיט און דאטום פון דער SQLite דאטאבאַזע.

• א פראבע-רוף פרעגט די האספיטאליזאַציע-צוווואוקס פאר Alaska דעם 5טן מערץ 2021.

• א באניצער-מעסעדזש שטעלט אויף א שאלה וועגן האספיטאליזאַציע-דאטא פאר Alaska אויף א געוויסן דאטום.

• דעפינירן SQL-געצייג: דער בלאק רעגיסטרירט צוויי SQL-געצייג פארן מאדעל, איינס פאר האספיטאליזאַציע-צוווואוקס און איינס פאר positive פעלער אויף א געוויסן דאטום.

• א SQL chat request: דער מאדעל באקומט דעם באניצער-מעסעדזש און די געצייג-דעפיניציעס, און באשליסט צי ער דארף א געצייג כדי צו ענטפערן.

• לויפן דעם SQL tool call: אויב דער מאדעל שלאגט פאר צו רופן א פונקציע, לויפט דער בלאק זי מיט די ריכטיגע ארגומענטן און לייגט צו די רעזולטאטן צו דער שמועס-היסטאריע.

• א לעצטער רוף צו GPT-4: יעצט זענען די פונקציע-רעזולטאטן שוין אין די מעסעדזשעס, און דער מאדעל נוצט זיי צו ענטפערן גענוי אויף דער אריגינעלער שאלה פונעם באניצער.

דער קורס פאסט צוזאמען אז מען האט געלערנט ווי צו מאכן דעם AI קלוגער דורך אים פארבינדן צו דאטא אין עכטער צייט מיט function calling, צי ס'איז וועטער-אינפארמאציע צי שפיטאל-דאטא.

לעקציע 5: Assistants API פאר SQL דאטאבאַזע (36:48:37)

אין לעקציע פינף לערנט מען ווי צו נוצן די Assistants API מיט א SQL דאטאבאַזע:

• אימפארטירן ביבליאטעקן: AzureOpenAI צו פארבינדן זיך צו OpenAI דורך Azure, json צו באהאנדלען סטרוקטורירטע דאטא, און os צו דערגרייכן environment variables ווי API keys אויף א זיכערן אופן.

• אימפארטירן helper פונקציעס: מען לאדט א אייגענעם helper מאדול און אימפארטירט פון אים ספעציפישע פונקציעס. די פונקציעס באארבעטן שאלות וועגן COVID-דאטא, ווי positive פעלער אדער האספיטאליזאַציע-צוווואוקס.

• שאפן דעם AzureOpenAI קליענט מיט זיכערע API keys און endpoints פון environment variables. די ווערסיע זארגט פאר קאמפאטיביליטעט מיט געוויסע API-פונקציעס.

• שאפן דעם assistant און א thread: מען שאפט א assistant וואס איז ספעציאליזירט צו ענטפערן COVID-שאלות מיט GPT-4 און SQL-געצייג. א נייער שמועס-thread ווערט געעפנט, און ער דינט אלס א קאנטיינער פאר דער סעסיע.

• שיקן א באניצער-מעסעדזש: מען לייגט צו א מעסעדזש צום thread וואס שטעלט פאר אן עכטע שאלה, אזוי אז דער assistant זאל קענען באארבעטן און ענטפערן לויט די דאטא-סעטס אדער געצייג וואס ער האט.

• ברענגען אלע מעסעדזשעס: מען ברענגט און דרוקט די גאנצע מעסעדזש-היסטאריע פונעם thread אין א ליינבארן JSON פארמאט. דאס איז נוציק צו נאכפאלגן וואס מ'האט געזאגט און ווי דער assistant האט געענטפערט.

• לויפן דעם assistant: מען הייבט אן א run אויף א געווישן thread. דער assistant באארבעט דעם לעצטן באניצער-מעסעדזש און שאפט א ענטפער מיט זיינע געצייג אדער דאטא-קוואלן.

• נאכקוקן דעם סטאטוס און באהאנדלען געצייג: א שלייף (loop) קאנטראלירט כסדר דעם סטאטוס פונעם assistant און ווארט ביז ער איז פארטיג. אויב דער assistant דארף א געצייג-פונקציע כדי ווייטער צו גיין, לויפט דער קאוד די ריכטיגע פונקציע און שיקט צוריק דעם רעזולטאט, אזוי אז דער assistant קען פארזעצן זיין ארבעט.

• ענדגילטיגע מעסעדזשעס: נאכדעם וואס דער assistant ענדיגט, ברענגט מען ווידער אלע מעסעדזשעס כדי צו ווייזן דעם ענדגילטיגן ענטפער לויט די באארבעטע געצייג און דאטא.

• JSON רעזולטאט: מען דרוקט דעם ענטפער אין א גוט-פארמאטירטער JSON סטרוקטור מיט indentation, וואס מאכט אים לייכטער צו ליינען און צו debuggen.

• ארויפלאדן דעם CSV און נוצן Code Interpreter: מען לאדט ארויף דעם COVID CSV דאטא-סעט און שאפט א נייעם assistant וואס נוצט דעם Code Interpreter געצייג. דער assistant קען יעצט ליינען און באארבעטן די CSV-טעקע גלייך מיט Python קאוד.

• דער לעצטער בלאק ארבעט ווי פריער: ער פאלגט נאך דעם פראגרעס פונעם assistant, דרוקט אויס די אפגעלאפענע צייט און דעם סטאטוס, און ווען ער ענדיגט, דרוקט ער אויס די שמועס-היסטאריע אין ליינבארן JSON, אריינגערעכנט דעם ענטפער לויט דער ארויפגעלאדענער CSV.

סוף פונעם דאטאבאַזע-אגענט קורס (36:52:11)

דער קורס פאסט צוואמען דעם טייל וועגן בויען דאטאבאַזע-אגענטן. מען האט געלערנט צו שאפן אינטעליגענטע אגענטן וואס ארבעטן מיט דאטאבאַזעס, ווי AI-אגענטן קענען אויטאמאטיזירן אויפגאבעס, אנאליזירן דאטא, און מאכן דאטאבאַזע-פארוואלטונג גלאטער און עפעקטיווער. מ'האט זיך נישט באגרעניצט מיט טעאריע: מען האט געזען די אגענטן אין אקציע מיט פראקטישע דעמאס. מען האט פארבונדן אגענטן צו CSV-טעקעס און SQL דאטאבאַזעס, גענוצט function calling פון Azure OpenAI , און פארבעסערט queries מיט דער Assistants API. לויטן קורס קען מען יעצט נוצן AI-אגענטן אלס פערזענלעכע דאטאבאַזע-געהילפן, פון פרעגן דאטא ביז אויטאמאטיזירן פראצעסן.

אנהייב פון א נייעם טייל: Zapier (36:53:13)

דער נעקסטער טייל הייבט אן צו בויען דעם ערשטן AI-אגענט וואס פארבינדט זיך צו סאשעל מעדיע און אויטאמאטיזירט אויפגאבעס. דער ערשטער שריט איז צו שאפן א Zapier אקאונט. ווער ס'האט שוין איינעם קען דעם טייל איבערשפרינגען. פאר נייע באניצער איז דאס אומזיסט, און מען גייט דורך די שריט: מען עפנט דעם געווינטלעכן וועב-בראוזער און גייט צום אפיציעלן וועבזייטל פון Zapier. ווייל מען איז שוין געווען איינגעלאגט, איז די היים-בלאט גלייך ארויסגעקומען.

א Zapier אגענט פאר סאשעל-מידיא (36:54:15)

איינלאגירן אין Zapier און די פלענער (36:54:15)

וועו מען עפנט Zapier, בעט מען זיך איינצולאגירן מיט סיי וועלכן אימעיל-אדרעס. דאס איז זייער גרינג: מען לאגירט זיך איין דורכן אימעיל, און נאכן איינלאגירן באקומט מען 14 טעג אומזיסטע פרובירונג (free trial) פון דעם Zapier חשבון. נאך די 14 טעג דארף מען זיך אויפהייבן (upgrade) צו איינעם פון די פלענער.

דער קורס ווייזט די פלענער: וועו מען קליקט אויף upgrade זעט מען פיר סארטן פלענער: Free, Professional, Team און Enterprise. יעדער פלאן האט אנדערע מעגליכקייטן, און מען קען אויסווייילן דעם פלאן וואס פאסט פאר די אייגענע באדערפענישן, און דאן ווייטער גיין צום באצאלן. דערווייל קען מען אבער הנאה האבן פון דער אומזיסטער פרובירונג די קומענדיגע 14 טעג. דער קורס רעקאמענדירט שטארק אויסצופרובירן אלע מעגליכקייטן פון Zapier אין די 14 טעג, און ערשט דערנאך אויסווייילן א פלאן לויט די באדערפענישן. מען זאל נישט פארלירן צייט, נאר גלייך אנהייבן.

שאפן דעם אגענט און צולייגן א טריגער (36:55:17)

דער ערשטער שריט איז צו שאפן דעם אגענט. מען זעט דעם Create סימבאל און מען קליקט דערויף. דארט געפינט מען אסאך אפציעס: Zaps, Tables, Forms, Canvas, Chatbots און Agents . מען ווילט אויס Agents. מען קען דעם אגענט געבן א נאמען לויט דעם פראיעקט; אין דער לעקציע ווערט ער אנגערופן "Social Media Agent".

מען זעט צוויי קנעפלעך: Configure און Activity. מען קליקט אויף Configure; דאס איז א זייער פשוט'ער וועג אנצוהייבן מיט אגענטן.

דערנאך דארף מען צולייגן א טריגער. עס ווערט ערקלערט אז א טריגער איז דער אנהייב-פונקט, אדער וי א וועקער-גלעקל וואס לאזט אן דעם אגענט. פון דארט הייבט דער אגענט אן צו פונקציאנירן און ארבעטן. מען לייגט עס צו דורכן קנעפל Add Trigger, און דארט געפינט מען פארשידענע אפציעס, למשל א טריגער פון אן אנדער אפליקאציע ווי Google Sheets, Google Calendar, Notion, Gmail, Dropbox און נאך אסאך.

פארבינדן Google Docs און די פראמפטן (36:56:19)

צום ערשט פארבינדט מען Google Docs. מען זוכט "Google Docs", און דארט קען מען אויסזוכן סיי וועלכן דאקומענט. מען דארף פארבינדן דעם Zapier חשבון מיטן Google Docs חשבון: מען קליקט Connect, דערנאך "Yes, continue to Google Docs", מען ווילט אויס דעם חשבון, און מען קליקט Add Trigger.

דער קורס ווייזט ווי דער "Social Media" דאקומענט זעט אויס: דארט זענען אנגעשריבן אלע סארטן פראמפטן וואס מען ווייל, ווי Facebook פראמפטן און Instagram פראמפטן. עס ווייזט ביישפילן ווי דער פראמפט צום AI אגענט זאגט אים ווי אזוי צו טון א געווסע ארבעט; אלע אנווייזונגען זענען קלאר אויסגעשריבן אינעם דאקומענט. עס ווערט געזאגט אז דער דאקומענט וועט צוגעלייגט ווערן אין דער באשרייבונג אונטער דעם וויידעא, כדי מען זאל קענען נוצן די פראמפטן פון דארט.

פארוואס מען טיילט צו די ארבעט אויף באזונדערע אגענטן (36:57:19)

מען וועט טון פיר ארבעטן באזונדער. עס ווערט ערקלערט: אויב מען לאזט דעם AI אגענט טון אלע ארבעטן מיטאמאל, קען ער "hallucinate", ער קען ווערן צעמישט און נישט געבן ריכטיגע רעזולטאטן. דעריבער ווערט שטארק רעקאמענדירט צו טון יעדע ארבעט באזונדער.

די ערשטע ארבעט איז צו הייסן דעם אגענט פאָרשן וועגן דעם געביט (niche). דעריבער שאפט מען צום ערשט דעם פארש-אגענט (research agent).

באשטימען דעם צייט-פלאן (36:57:49)

צום ערשט באשטימט מען א צייט-פלאן (schedule) פארן ערשטן אגענט דורך Zapier. מען קען אויסווייילן סיי וועלכע אָפטקייט: יעדן טאג, יעדע שעה, יעדן חודש, יעדע וואך, לויט ווען מען וויל אז דער סאשעל-מידיא אגענט זאל פאסטן אויף LinkedIn, Instagram אדער Facebook. אין דער לעקציע וויילט מען אויס יעדן טאג, און מען שטעלט די צייט, למשל יעדן טאג 7:00 אינדערפרי, און מען קליקט Save. אזוי ווערט דער טריגער באשטימט דורך Zapier.

פארבינדן Google Sheets און א נייע רייע (36:58:20)

איצט פארבינדט מען א Google Sheet מיטן Zapier חשבון. דער קורס ווייזט ווי די שיט וועט אויסזען: זי וועט האבן קאלאמען פאר דאטום, טיטל, לינק צום ארטיקל, הויפט-שורה (headline), און א קורצע צוזאמענפאסונג (summary) פונעם ארטיקל.

מען וויילט אויס Google Sheets, און מען זעט פארשידענע אפציעס ווי Lookup Spreadsheet, Range Data און אזוי ווייטער. מען וויילט אבער Create Spreadsheet Row: יעדעס מאל וואס דער אגענט ווערט אנגעלאזט, ווערט געשאפן א נייע רייע מיט די לעצטע ארטיקלען. מען קליקט Connect כדי צו פארבינדן מיטן Google חשבון.

נאכן פארבינדן זעט מען די אפציע "Let your agent select a value for this field". עס ווערט געזאגט אז עס איז בעסער עס אזוי איבערצולאזן, דאס הייסט לאזן דעם אגענט אליין אויסווייילן די ווערט.

דער ערשטער פראמפט: דער פארש-אגענט (36:59:21)

איצט גיט מען דעם ערשטן פראמפט צום AI אגענט. די אנווייזונג הייבט זיך אן: "דיין ציל איז צו אידענטיפיצירן און צונויפשטעלן א ליסטע פון AI אגענטן און פירמעס וואס מען שרייבט וועגן זיי אין די נייעס אנליין". דערצו זענען געגעבן שרייב-אנווייזונגען פאר יעדן ארטיקל וואס ער געפינט: וואס צו שרייבן, וועלכע סארט הויפט-שורות און צוזאמענפאסונגען מען דארף געבן.

א וויכטיגער עצה: וואס קלארער די אנווייזונגען, אלס בעסערע רעזולטאטן קען מען ערווארטן. מען זאל פרובירן געבן געניוערע און מער ספעציפישע אנווייזונגען צום AI אגענט. דערנאך לייגט מען צו די Google Sheet וואו מען האט געשאפן די Spreadsheet Row.

אויספרובירן דעם ערשטן אגענט (37:00:22)

מען קליקט אויף Preview און מען פרובירט דעם אגענט. מען זעט ווי דער "Social Media Agent" ווערט געטעסט: דער צייט-פלאן ווערט אנגעלאזט, און דערנאך ווערט די Google זוכונג פארענדיגט. מען קען זען

וואס עס טוט זיך אינעווייניג דורכן קליקן אויף די דעטאלירטע אנווייזונגען; דארט ווערט געוויזן די דעטאלירטע ארבעט אין הינטערגרונט, און ווי דער AI אגענט ברענגט ארויס אלע ארטיקלען.

דעם רעזולטאט זעט מען אין דער Google Sheet : דער אגענט האט געבראכט פיר "טרענדינג" ארטיקלען וועגן AI אגענטן. יעדעס מאל וועלן צוגעלייגט ווערן נייע רייען מיט די לעצטע ארטיקלען. מען באשטעטיגט (approve) די בקשה אז ער זאל דורכפירן די אקציע און דערהיינטיגן די שיט לויט די אנווייזונגען, און אזוי זעט מען ווי די שיט ווערט דערהיינטיגט יעדן טאג.

א סיכום פונעם ערשטן אגענט און פארעפנטלעכן (37:01:57)

דאס איז געווען דער ערשטער אגענט: מען האט פארבונדן די Google Sheet מיטן Zapier חשבון, און דער אגענט האט דורכגעפירט די פארשונג ווען די אקטועלע נייעס-ארטיקלען וואס מען וויל פאסטן אויף סאשעל-מידיא. די ערשטע ארבעט איז פארענדיגט: פארשן, און ארויסשטעלן דעם רעזולטאט אין דער שיט. מען האט צוגעלייגט די Google Sheet דורכן Add Tool באפעל, און מען האט צוגעלייגט דעם Google Docs דאקומענט אלס א וויסן-מקור (knowledge source) פונעם אגענט.

עס ווערט באטאנט: ווען איין ארבעט איז פארטיג, איז וויכטיג צו פארעפנטלעכן (publish) דעם אגענט, אים האלטן באזונדער, און דערנאך אנהייבן א נייעם, כדי די קומענדיגע ארבעט זאל געטון ווערן באזונדער.

דער צווייטער אגענט: פאסטן אויף Facebook (37:02:58)

די קומענדיגע ארבעט איז צו שאפן אן אקציע וואס פאסט די ארטיקלען וואס דער AI האט געשאפן אויף דעם Facebook חשבון. אזוי ווערט געוויזן ווי מען פארבינדט דעם AI אגענט מיט סיי וועלכער סאשעל-מידיא, אנגעהויבן מיט Facebook.

- מען רעדאגירט דעם טריגער און מען פארבייט אים.
- מען פארבינדט ווידער מיט Google Sheets, און מען קליקט New Spreadsheet Row.
- מען ווילט אויס די שיט מיטן נאמען "AI Agent", און דעם בלאט (worksheet) "Sheet1".
- מען קליקט Save.

דער פראמפט זאגט: "שאף און שרייב א Facebook פאסט מיט הויכע באטיילִיגונג (high engagement) וואס וועט כאפן די אויפמערקזאמקייט פון אסאך באניצער". ער באשרייבט ווי דער פאסט זאל אויסזען, למשל: הייב אן מיט א שטארקער דרייסטער דערקלערונג, גיב גלייך ווערט, האלט עס פראקטיש און מיט קלארע שריטן צו טון. ער באשטימט אויך דעם טאן און סטיל: אויף א שמועס-אופן און נאטירלעך, מיט קורצע פארמען פון ווערטער (contractions). מען זאל געבן א פראמפט אזוי ספעציפיש ווי מעגליך; די פראמפטן געפינען זיך אין דעם דאקומענט אין דער באשרייבונג. דארט זענען אויך פארשידענע ביישפילן פון פאסטן, כדי דער AI אגענט זאל זיך קענען אנלענען אויף זיי און שאפן דעם פאסט לויט דעם.

צולייגן Facebook אלס געצייג (37:04:32)

מען לייגט צו א Tool, דעם Facebook חשבון: מען קליקט Facebook Pages, ווייל מען דארף שאפן א פאסט אויף א בלאט. עס ווייזן זיך פארשידענע אפציעס ווי Change Page Profile, Create Page Photo א.א.וו., און מען ווילט אויס Create Page Post. אין דער לעקציע איז Facebook שוין פארבונדן מיט Zapier, אבער עס ווערט ערקלערט אז עס איז זייער גרינג צו פארבינדן: ביים Configure קען מען פארבינדן דעם Facebook חשבון.

אין קורצן, וואס איז געטון געווארן: מען האט צוגעלייגט א טריגער פון Google Sheets, אזוי אז וען א נייער פאסט ווערט געשאפן אדער דערהיינטיגט אין דער שיט, לאזט זיך אן די ארבעט. דאן פארבינדט עס זיך מיט Facebook, און פונקט 7:00 אינדערפרי פאסט עס דעם פאסט אויפן Facebook חשבון.

טעסטן דעם Facebook אגענט (37:05:33)

מען קליקט Test Agent. צום ערשט לאזט זיך אן דער Google Sheets טריגער; עס אנאליזירט דעם דאטא-מקור, און עס ברענגט אלע אינהאלטן פאר סאשעל-מידיא. דער אגענט נעמט ביישפילן פון די סאשעל-מידיא פאסטן אינעם דאקומענט וואס איז געגעבן געווארן אלס וויסן-מקור. דערנאך שאפט ער דעם Facebook פאסט און ווייזט די מעלדונג וואס ער וועט פאסטן. מען קען זען דעם טעקסט פריער, און דערנאך באשטעטיגט מען די בקשה. די אקציע ווערט דורכגעפירט, און אויפן Facebook חשבון זעט מען אז דער פאסט איז אויטאמאטיש געשאפן און ארויפגעלייגט געווארן דורכן "Social Media Agent", ווייל דער טריגער האט זיך אנגעלאזט.

אזוי קען מען אויטאמאטיזירן Facebook: מען דארף נישט שאפן קיין פאסט; דער אגענט שאפט דעם פאסט און פאסט אים אויך אויף Facebook, אן קיין שום ארבעט פון באניצער.

דער דריטער אגענט: LinkedIn (37:07:07)

נאכן פארעפנטלעכן דעם Facebook אגענט, פארבייט מען ווידער דעם געצייג און לייגט צו LinkedIn. עס זענען דא אפציעס ווי Create Company Update און Create Share Update; מען ווילט Create Share Update, וואס שאפט און פאסט א דערהיינטיגונג אויפן LinkedIn חשבון. אין דער לעקציע איז LinkedIn שוין פארבונדן, אבער ביים Configure קען מען גרינג פארבינדן דעם אייגענעם חשבון.

ביי די פעלדער ווילט מען אויס "Let your agent generate a value for this field", ווייל אזוי באקומט מען גענויערע רעזולטאטן. עס זענען דא אנדערע אפציעס, ווי Set a Specific Value, אדער לאזן דעם אגענט אויסוויילן פון גע‌וויסע ברירות, אבער עס ווערט געזאגט אז עס איז בעסער צו לאזן דעם אגענט אליין אויסוויילן; עס ווערט באטאנט אז דאס איז אויסגעטעסט געווארן און עס ארבעט זייער גוט. מען קליקט Save.

דער LinkedIn פראמפט (37:08:11)

דער פראמפט צום "Social Media Agent" זאגט: "דו ביסט א LinkedIn אינהאלט-סטראטעגיסט, און דיין ציל איז צו שרייבן א LinkedIn פאסט וואס ברענגט רעזולטאטן (high converting), וואס כאפט אויפמערקזאמקייט, גיט וועלט און טרייבט צו אקציע". אינעם פראמפט זענען אויך אונטערגעשטראכן די הויפט-עלעמענטן פון א גוט-ארבעטנדיגן LinkedIn פאסט, וואס מען דארף אכטונג געבן אויף:

- א שטארקער רוף צו אקציע (CTA)
- עכטקייט (authenticity)
- טיפקייט
- א געארדנטער, גרינג-צו-ליינען פארמאט

דער ציל איז אז דער אגענט זאל שאפן א זייער ספעציפישן און שטארקן פאסט פארן LinkedIn חשבון.

אויטאמאטישע פאסטן און א זעלבסט-לערנענדיקער אגענט (37:09:12)

דער LinkedIn-אגענט פירט אויס די ארבעט (37:09:12)

אין דער לעקציע ווערט ווייטער געוויזן דער אגענט פאר LinkedIn. אין די אינסטרוקציעס האט מען אריינגעגעבן ביישפילן: מוסטערן פון פאסטן און "הויך-לייסטנדיקע" LinkedIn-ביישפילן. די ביישפילן זענען דא כדי דער אגענט זאל זיך פון זיי קענען אנלערנען און שאפן א פאסט פונקט אין דעם סטיל פון אונזער LinkedIn-אקאונט. צום סוף פון די אינסטרוקציעס שטייט קלאר: וועז דו ביסט פארטיק, לייג עס ארויף אויף מיין LinkedIn-בלאט.

דערנאך פרובירט מען אויס דעם אגענט, און מען זעט ווי ער ארבעט שריט נאך שריט:

- ערשט ענדיקט ער דאס זוכן אויפן אינטערנעט (web browsing).
- דערנאך אנאליזירט ער די דאטא.
- נאכדעם לאדנט ער ארויף דעם אינהאלט פונעם דאקומענט מיט ביישפילן פון סאשעל-מידיע פאסטן.
- צום סוף שאפט דער AI דעם טעקסט פונעם פאסט, וואס זאל ארויפגעלייגט ווערן אויפן LinkedIn-אקאונט.

איידער דער פאסט גייט ארויס, בעט דער אגענט א באשטעטיגונג. מען דריקט אויפן קנעפל "approve", און באלד ווערט געמאלדן אז די אקציע איז פארטיק און דער פאסט איז מיט הצלחה ארויס אויף LinkedIn. פון דארט קען מען גלייך אריבערגיין צום LinkedIn-בלאט, און מען זעט אז דער פאסט איז יעצט ארויפגעלייגט געווארן דורכן AI-אגענט.

עס ווערט אונטערגעשטראכן: דער פאסט איז אויטאמאטיש געשאפן געווארן און אויטאמאטיש ארויפגעלייגט, און דער מענטש אליין האט גארנישט געטון. דער קורס פרעגט: ווי פיל פאסטן קען א מענטש אליין שרייבן און ארויפלייגן יעדן טאג? דאס איז נישט מעגליך פאר א מענטש, און דערפאר דארף מען אויטאמאטירן אזעלכע ארבעטן דורך AI-אגענטן. אויפן זעלבן אופן קען מען שאפן אסאך פאסטן פאר Facebook, Instagram און LinkedIn. אזוי ווי מען האט שוין געמאכט אגענטן פאר Facebook און LinkedIn, ווערט יעצט געוויזן ווי צו מאכן איינעם פאר Instagram.

א נייער אגענט פאר Instagram (37:11:17)

ערשט פארעפנטלעכט (publish) מען דעם אגענט. דערנאך גייט מען צוריק צום טריגער און רעדאגירט אים. מען פארבינדט אים נאכאמאל מיט דעם Google Sheet מיטן נאמען "AI agent", מיטן בלאט "sheet one", און מען היט עס אפ. דערנאך ענדערט מען די אינסטרוקציעס: מען נעמט ארויס LinkedIn און מען פארבינדט אנשטאט דעם Instagram.

די וויסנס-קוועל (knowledge source) פון סאשעל-מידיע ביישפילן איז שוין צוגעלייגט. אבער מען דארף געבן באזונדערע אינסטרוקציעס פאר Instagram. די סיבה איז, אז דער פראמפט פאר LinkedIn, Instagram און Facebook איז גאנץ אנדערש: Instagram און Facebook זענען מער אומפארמעלע פלאצן, און LinkedIn איז מער פארמעל. דערפאר דארף דער פראמפט זיך צופאסן צו דער נאטור פון יעדער סאשעל-מידיע. מען קאפירט דעם פראמפט פון א דאקומענט (docs), וואו מען האט שוין פריער צוגעגרייט פראמפטן פאר אלע סאשעל-מידיעס.

נאכדעם לייגט מען צו א געצייג (tool). מען קלייבט "Instagram for business", מיט דער אקציע צו פארעפנטלעכן בילדער און ווידעאס אין א פאסט. מען קאנפיגורירט עס, דריקט אויף "new account" כדי צו פארבינדן דעם אקאונט, און דערנאך היט מען עס אפ.

דער אינהאלט פונעם Instagram-פראמפט (37:13:28)

דער פראמפט הייבט זיך אן אזוי: "דו ביסט א געניטער Instagram אינהאלט-סטראטעגיסט, און דיין ציל איז צו שרייבן אן אינטערעסאנטע Instagram באשרייבונג (caption) וואס כאפט אן די אויפמערקזאמקייט". נאכדעם ווערן אויסגערעכנט די הויפט-עלעמענטן פאר הצלחה, און די סטילן פון באשרייבונגען. ביי די ביישפילן שטייט: קורץ און שארף (punchy), פרייליך, און מער אינטערעסאנט. דער קורס באמערקט, אז אזוי פאסט מען אויף Instagram: נישט אויף א פארמעלן אופן, נאר מער פריי. די ביישפילן זענען צוגעלייגט כדי דער אגענט זאל שאפן א פונקטליכן פאסט.

דערנאך קען מען מאכן אן "agent preview " און אויספרובירן דעם אגענט. ער שאפט דעם פאסט פאר Instagram, און אויפן Instagram-אקאונט זעט מען דעם פאסט, פונקט ווי עס איז געווען ביי LinkedIn.

איבערבליק און עצות (37:14:32)

אויף אזא אופן קען מען אויטאמאטירן אלע סאשעל-מידיעס: LinkedIn, Instagram און אפילו Facebook. עס ווערט באטאנט ווי גרינג דאס איז: מיט בלויז עטליכע שריט ווערט די גאנצע ארבעט אויטאמאטיש. דער קורס מוטיקט מען זאל נישט אפשטעלן דא. עס ווערט דערמאנט, אז מען האט נאך 14 טעג, און אין דער צייט קען מען בויען נאך אגענטן, פארבינדן אנדערע סאשעל-מידיעס, נוצן די פראמפטן אויף זיי, און זען ווי עס ארבעט פאר דעם אייגענעם אינהאלט.

מען קען אויך צופאסן דעם פראמפט. ווען דער AI שאפט א טעקסט און מען וויל ענדערונגען, דריקט מען אויפן קנעפל "reply" און מען שרייבט ווי מען וויל די באשרייבונג, אדער ווי דער פאסט זאל אויסזען. דער אגענט דערהיינטיקט עס דאן לויט דעם. דערצו האט מען דירעקט פארבונדן Google Sheets און Google Docs. דער פראמפט ווערט פון דארט גענומען, און דערפאר ווערט יעדע ענדערונג וואס מען מאכט אין דעם דאקומענט גלייך אויך דערהיינטיקט אינעם אגענט.

די יסודות פון א זעלבסט-לערנענדיקן AI-אגענט (37:15:04)

א נייער טייל הייבט זיך אן: ווי צו בויען א זעלבסט-לערנענדיקן (AI (self-learning-אגענט. עס זענען דא אסאך וועגן דאס צו טון. דא נוצט מען די פאלגנדע מיטלען:

- די פלאטפארמע n8n.
- Airtable: דאס איז די וואלקן-שפייכלונג, וואו דער אגענט געדענקט זאכן וועגן דעם באניצער. ער שפייכלט דארט יעדן פרט וואס מען גיט אים איבער.
- Telegram: א מיטל פון קאמוניקאציע. מען קען רעדן צום אגענט, אדער שרייבן פארלאנגען און פרעפערענצן, ווי למשל וואס מען האט ליב און וואס נישט, וואו מען וואוינט, און אלע פרטים וואס מען וויל אז דער אגענט זאל אויסלערנען.
- א LLM-פלאטפארמע, וואס העלפט דעם אגענט צו געדענקען און צו ענטפערן אויף די פארלאנגען. מען קען נוצן יעדן LLM; דא נוצט מען ChatGPT פון OpenAI.

דער Telegram-טריגער און א נייער באט (37:16:37)

אויף דער n8n Cloud פלאטפארמע שאפט מען א נייעם workflow, כדי צו האבן א ליידיקע קאנווע וואו מען לייגט צו אלעס שריט נאך שריט. אלס ערשטן שריט לייגט מען צו א טריגער: Telegram, מיט דער אפציע "on message". דאס מיינט, אז יעדע מעלדונג וואס מען שיקט פון Telegram קומט אן צו דער קאנווע. ווייל מען האט נאך נישט קיין אויטענטיפיקאציע (credential), שאפט מען א נייע. דערצו דארף מען האבן אן access token.

מען פרובירט ערשט די וועבזייטל-ווערסיע פון Telegram, אבער דארט איז נישט דא קיין אפציע צו שאפן א קאנאל. דערפאר לאדנט מען אראפ און מען אינסטאלירט די ווערסיע פאר Windows. אין דער אפליקאציע גייט מען צום פראפיל און מען שאפט א נייעם קאנאל. מען קען אים געבן א נאמען לויט'ן געשמאק, אפילו א שיינעם נאמען ווי "Jarvis". דא קלייבט מען א פשוטן נאמען: "self-learning AI agent". די באשרייבונג איז אפציאנאל און מען לאזט זי ליידיג. מען קלייבט "private", מען לייגט נישט צו קיין מיטגלידער, און אלעס אנדערש בלייבט ווי עס איז.

דערנאך שרייבט מען "@BotFather", וואס גיט א ברייטע אויסוואל פון אפציעס צו בויען דעם באט. מען דריקט אויפן לינק, דריקט "start", און מען שרייבט /newbot כדי צו שאפן א נייעם באט. ערשט גיט מען דעם באט א נאמען. דער נאמען מוז זיין יחידותדיק, ווייל אסאך מענטשן האבן שוין אויסגעטראכט שיינע נאמען, און מען טאר זיי נישט איבערחזרן. דא נעמט מען דעם נאמען "Simplilearn self-learning".

נאכדעם דארף מען קלייבן א באניצער-נאמען (username), וואס מוז זיך ענדיקן מיט "bot". מען פרעגט צי BotFather קען געבן פארשלאגן, אבער עס קען נישט; עס נעמט נאר אן וואס מען גיט אים. די ערשטע פרוווען זענען שוין פארנומען, למשל א נאמען מיט "simply" און א נאמען מיט דעם פולן נאמען מיט "1". צום סוף ארבעט "Simplilearn training". BotFather מעלדט "מזל טוב" אויף דעם נייעם באט, און גיט א token צו נוצן דעם) HTTP APIעס הייבט זיך אן מיט 81290).

מען דריקט אויפן token, און ער ווערט קאפירט. דערנאך לייגט מען אים אריין אין n8n. עס ווערט געווארנט: דעם token טאר מען נישט טיילן מיט קיינעם. ער דארף זיין פארדעקט און פארזיכערט. מען דריקט "save", n8n פרובירט דעם קשר און עס ווייזט "success". אזוי האט מען געשאפן דעם ערשטן Telegram-טריגער.

צולייגן Airtable און א personal access token (37:21:20)

דער קומענדיקער שריט איז די דאטא-באזע וואס שפייכלט אלעס. מען קלייבט Airtable, מיט דער אקציע "search records". דא דארף מען ווידער אן אויטענטיפיקאציע. מען האט שוין אן Airtable-אקאונט מיט א טאבעלע וואס איז פריער געשאפן געווארן. אויף דער הויפט-זייט דריקט מען אויפן פראפיל-בילדל אויף דער רעכטער זייט, און דארט קלייבט מען "Builder hub". אין דער אפטיילונג פאר פראגראמירער (developer) קלייבט מען "personal access tokens", און דערנאך "create new token".

מען גיט דעם token א נאמען ("AI agent"), און דערנאך דארף מען צולייגן scopes, דאס הייסט די רעכטן וואס דער token באקומט. פאר אזא יסודותדיקן פראיעקט איז גענוג די ערשטע אפציע, "records read". אבער כדי צו זיין זיכער, און ווייל מען קען זיי אפשר דארפן אין דער צוקונפט, קלייבט מען אלע scopes. וועז אלע זענען צוגעלייגט, ווערט די אויסוואל-ליסטע (drop-down) ליידיג. מען קען אויך צולייגן א base: מען קלייבט אלע יעצטיגע און צוקונפטיקע bases פאר דעם workspace. צום סוף היט מען אפ די ענדערונגען. אויב מען האט שוין א token, קען מען שאפן א נייעם דורך דער אפציע וואס דארט געפינט זיך.

בויען דעם AI-אגענט אין n8n מיט Airtable (37:23:23)

פארענדיקן די Airtable-פארבינדונג (37:23:23)

אין דעם טייל פון קורס ווערט ווייטער געבויט דער workflow אין n8n . מען האט שוין געשאפן א נייעם access token אויפן Airtable פלאטפארמע (דארט ווייזט זיך דער קנעפל "create new token" אין בלויער פארב). מען גייט צוריק צום builder hub, וואו מען לייגט צו א נייע credential, און אין דעם פעלד "access token" קאפירט מען אריין דעם token וואס מען האט ערשט געשאפן.

דער קורס ווארנט: מען טאר נישט ווייזן די סודות'דיגע tokens און API שליסלען פאר קיינעם. מען זאל זיי האלטן פארדעקט (masked).

דערנאך דריקט מען save און מען טעסט די פארבינדונג. עס קומט א מעלדונג אז די פארבינדונג איז געטעסט מיט הצלחה (Connection tested successfully). מען פארמאכט דעם פענצטער און מען קערט זיך אום צום canvas.

אויסשטעלן דעם "search records" שריט (37:23:54)

אויפן canvas זעט מען אז עס פעלט נאך עפעס. מען האט נאך נישט ארײנגעלייגט אלע פאראמעטערס פאר דעם "search records" שריט. מען קליקט אויף אים. דער resource און די operation זענען שוין אויסגעשטעלט, אויך דער base. וואס עס פעלט איז דער table.

מען דארף אויסקלייבן דעם table וואס מען האט פריער געשאפן אויפן Airtable פלאטפארמע, מיטן נאמען "AI agent learning". מען קען אים זען אויף דער רעכטער זייט ווען מען סקראלט אראפ. וועט עס וויל, קען אויך שאפן א נייעם table. אבער ווייל דער table איז שוין דא, נעמט מען דעם וואס עקזיסטירט שוין.

אינעם table זענען שוין דא עטליכע קאלומנעס, למשל "material title". ווייל מען ארבעט מיט א פרובירונג-ווערסיע (trial), ווערט ערקלערט אז מען דארף נישט קיין איבעריגע קאלומנעס און איינטראגן. דעריבער מעקט מען אויס אלע אלטע קאלומנעס און איינטראגן וואס מען האט פריער צוגעלייגט.

עס בלייבט איין קאלומנע מיט איין איינטראג, און דעם איינטראג קען מען רעדאגירן. דער געדאנק איז אזוי: ווען מען וועט שרייבן פראמפטס פארן AI-אגענט, זאל דער ערשטער איינטראג זיך ארויסווייזן אלס רעזולטאט ביים פרובירן. דעריבער שרייבט מען אריין: "Simplilearn tutorial on self-learning and understanding AI agents". מען מאכט א refresh, אז אלעס זאל זיך אפהיטן, און מען גייט צוריק צו n8n.

אויסקלייבן די קאלומנע און דעם limit (37:25:25)

אין n8n האט מען שוין אויסגעקליבן דעם table פון דער רשימה (from list). יעצט קלייבט מען אויס די קאלומנע: "learning material". דעם נאמען קען מען טוישן. ביים ערשטן פרובירן האט מען גענוצט דעם נאמען "learning material", און מען נוצט אים דא ווידער. אבער מען קען אים אויך אויסשטעלן ווי מען וויל.

דערנאך דארף מען צולייגן עטליכע expressions:

- מען שאלט אויס דעם "return all".
- ווען "return all" איז אויסגעשאלטן, איז דער limit לויט דער גרונט-איינשטעלונג 100. מען וויל נישט אזא גרויסן ענטפער, דעריבער שטעלט מען 50, ווייל ס'איז נאר א פרובירונג-ווערסיע.

מיט דעם איז דער "search records" טייל אינגאנצן פונקציאנירנדיג.

צולייגן aggregate און merge (37:26:25)

מען פארקלענערט דעם zoom און מען סדר'ט אויס די triggers און אלע טיילן פונעם workflow. יעצט לייגט מען צו דעם aggregate שריט, אז ער זאל צוזאמנעמען די פארשידענע עלעמענטן.

אין דער פאראמעטערס-אפטיילונג פון aggregate:

- מען קלייבט "individual fields".
- בייַ "fields to aggregate" קאפירט מען דעם קאלומנע-נאמען פונעם Airtable table, דהיינו "material title". מען לאזט אים אזוי ווי ער איז, מען גייט צוריק, מען קלעבט אים איין און מען היט אפ.

דערנאך פארלענגערט מען דעם aggregate דורך קליקן אויפן פלוס. מען זוכט די merge פונקציע און מען קליקט דערויף. די merge פונקציע נעמט צוויי אריינגאבעס (inputs). מען קען צופירן סיי דעם trigger אינפוט און סיי דעם table אינפוט צו דער merge אפטיילונג.

דער mode ווערט "combine", מיט דער אפציע "all possible combinations". מען קערט זיך אום צום canvas.

יעצט פירט מען א ליניע גלייך פונעם Telegram trigger . דער געדאנק: לויט די אריינגאבעס פונעם Telegram trigger קען דער workflow גיין דורך search records און aggregate , אדער גלייך פונעם Telegram trigger אליין. איין אינפוט קומט פונעם aggregate, און "input 1" קומט גלייך פונעם trigger.

צולייגן דעם AI-אגענט (37:27:59)

יעצט קומט דער הויפט-טייל: צולייגן א AI-אגענט. מען ארגאניזירט ערשט איבער דעם canvas אז ער זאל אויסזען ציכטיג. מען קליקט אויפן פלוס און מען קלייבט "AI". מען האט פרובירט זוכן "advanced AI", אבער אוא אפציע איז נישטא, דעריבער נעמט מען פשוט "AI", די ערשטע אפציע.

עס עפנען זיך פארשידענע אפציעס. ערשט לייגט מען צו דעם "AI agent", און שפעטער שטעלט מען אים אויס לויט די באדערפענישן. מען קלייבט א AI-אגענט פאר פרובירן.

מען קוקט איבער די פאראמעטערס: ער איז פארבונדן מיטן chat trigger node, מיט "prompt user message", און אלע נויטיגע expressions זענען גרייט. מען לאזט אלעס ביי די גרונט-איינשטעלונגען איידער מען הייבט אן מיט שמועסן אדער איינשטעלונגען, און מען קערט זיך אום צום canvas.

פרובירן דעם Telegram באט (37:29:01)

פריער מעקט מען אויס די פארבינדונג צום אגענט, כדי צו טעסטן ערשט די triggers. מען שאלט אים אויס (deactivate), און ער איז אינגאנצן אומאקטיוו.

יעצט פרובירט מען דעם workflow וואס מען האט ביז אהער געבויט, דורך שיקן א מעלדונג אדער צוויי. כדי צו אקטיווירן דעם באט נוצט מען דעם סימבאל @ מיטן נאמען פונעם באט. דא האט דער באט א נאמען ענלעך צו "simply training bot". מען שרייבט עס און מען דריקט enter.

איידער מען לייענט די אפציעס, סקראלט מען אביסל אראפ. דארט זעט מען דעם נאמען פונעם באט, אנגעצייכנט אין בלויער פארב, און מען קליקט דערויף. דער קורס גיט א משל: ווי אויף סאציאלע נעצן, וואו מען צייכנט אן (tag) א חבר מיט @. למשל, אז דער חבר הייסט XYZ, שרייבט מען "@XYZ". אזוי אויך דא שרייבט מען @ מיטן נאמען פונעם באט, און מען קומט צו אים.

עס עפנט זיך א פענצטער פאר שמועסן מיטן באט, וואס הייסט "Simplilearn self-learning bot". מען קליקט אויפן start קנעפל אונטן און מען שיקט מעלדונגען. די ערשטע מעלדונג איז: "This is a trial". מען דריקט execute.

דער workflow איז נאך נישט געווען אקטיווירט, פונדעסטוועגן האט ער אויפגענומען די צוויי מעלדונגען, און עס קומט אז דער workflow איז דורכגעפירט מיט הצלחה. ביידע טעקסטן זענען אנגעקומען: די ערשטע איז "start", און די צווייטע איז "hi this is a trial". מען זעט אז דער workflow ארבעט. יעצט קען מען צופארבינדן דעם AI-אגענט און אים אקטיווירן.

אויסשטעלן דעם פראמפט און דעם system message (37:30:35)

יעצט קומט דער וויכטיגער טייל: אויסשטעלן דעם AI-אגענט, און א צווייטן Airtable פאר די ענטפערס. מען טאפלט-קליקט אויפן AI agent. מען זעט דעם Telegram trigger און דעם "material title", וואס איז די ערשטע קאלומנע, מיט דער מעלדונג "Simplilearn tutorial on self-learning and understanding AI agents".

ביי "source of the prompt" קלייבט מען "define below", און דארט שרייבט מען דעם פראמפט. מען דארף אויסגעפינען די קאלומנע און זי איינשלעפן אין "prompt user". דערצו נוצט מען JSON: מען שלעפט עס אהין און מען לאזט אפ, און דער פראמפט ווערט אויטאמאטיש געמאכט.

דערנאך לייגט מען צו אן אפציע: דעם system message. די גרונט-איינשטעלונג פונעם system message איז ענלעך צו "you are a helpful assistant", אבער מען קען שרייבן אן אייגענעם. דער קורס באמערקט אז דאס איז נישט קיין expression, נאר א פעסטער (fixed) טעקסט. דעריבער קען מען אים טוישן אויב מ'דארף.

די קליינע ענדערונג וואס מען מאכט: דער system message זאל אנהייבן מיט דעם איצטיגן דאטום און צייט, אז מען זאל וויסן אז ס'גייט פאר אין עכטער צייט. דערצו דארף מען טוישן דעם system message פון fixed אויף expression. דאן באקומט מען פארשלאגן און אויטאמאטישע קאוד-אויספילונג. מען שרייבט $now און מען דריקט tab, כדי אויסצוקלייבן דאטום און צייט, און דאס ווערט דער רעזולטאט: current date and time.

דער אינהאלט פונעם פראמפט (37:32:39)

מען קען צולייגן א מעלדונג וואס ערקלערט דעם AI-אגענט ווי ער זאל זיך פירן. דער קורס ווייזט דעם טעקסט שנעל. דער workbook ווערט פארלינקט אין דער באשרייבונג, און מען קען אים פון דארט אראפלאדן. מען קען אויך נעמען הילף פון Claude: מען שרייבט עטליכע פונקטן, און Claude גיט א שיינעם פראמפט פארן אגענט.

דער אינהאלט פונעם פראמפט, אין קורצן:

- "דו ביסט א הילפסאמער און שטארקער AI-געהילף", מיט דער פעאיגקייט צוצולייגן און צו לייענען די אריינגאבע, און צו לערנען מער ווען דעם באניצער פון דעם וואס ער האט שוין געטיילט.
- אויב ער ווייסט עפעס נישט, זאל ער פרעגן כדי צו לערנען מער ווען דעם באניצער.
- דער שמועס זאל זיין ווי א שמועס צווישן מענטשן, נישט ווי א באט וואס רעדט.
- אויבן קען ער ווייזן די לעצטע מעלדונג אדער זכרון וואס ער האט באקומען: "your most recent memories". דארט נוצט מען א JSON קאוד-שטיקל, כדי ארויסצוציען דעם לעצטן איינטראג.

אויף דער רעכטער זייט זעט מען אז עס ציט ארויס די מעלדונג פון Airtable: "Simplilearn tutorial on self-learning and understanding AI agents", דער ערשטער איינטראג. אזוי דארף עס ארבעטן, און דאס איז דער "recent memory". שפעטער דארף מען נאך אויסשטעלן דעם אגענט מיט API שליסלען און דאס גלייכן.

ריכטלינעס פארן ענטפער (37:34:08)

אין פראמפט שטייט אויך:

- מען זאל תמיד געבן דעם לעצטן ענטפער אין א שמועסדיגן אופן, נישט ווי א רשימה פון blog posts.
- מען זאל עס האלטן פשוט, אן צו פיל היראַרכיע.

דער קורס ערקלערט פארוואס: אויב דער ענטפער איז צו לאנג, קען דער באניצער זיך פילן צומישט און עפעס פארפעלן. וועז עס איז קורץ, איז עס גרינג און נוצליך.

מען שרייבט אויך: נאכן אויסטראכטן דעם ענטפער, זאל מען באטראכטן א TLDR וועדרסיע, וואס מיינט "too long, didn't read". מען זאל תמיד געבן א שמועסדיגן, שארפן און לעבעדיגן ענטפער, און פונדעסטוועגן בלייבן באשטימט (assertive).

אזא פראמפט קען מען שרייבן אליין, נעמען הילף פון Claude, אדער נעמען די פארלינקטע טעקע. דערנאך פארמאכט מען דעם פראמפט און מען קערט זיך אום צום canvas.

אויסקלייבן דעם chat model (37:35:08)

אויפן canvas זעט מען אז מען דארף נאך פארריכטן עטליכע זאכן. דאס ערשטע איז א chat model. מען קען קלייבן יעדן מאדעל וואס איז פאראן, למשל DeepSeek. ווי עס ווערט דערמאנט, האט מען געדעדט וועגן AI מאדעלן אינעם אנהייב פונעם קורס, און מען קען נעמען וועלכן LLM מען וויל: א לאקאלן, OpenAI אדער DeepSeek.

דא נעמט מען OpenAI. דאס ערשטע וואס מען דארף איז credentials. דער קורס ערקלערט אז credentials זענען גארנישט מער ווי די API שליסלען. מען גייט צום OpenAI וועבזייטל, וואו מען שאפט די API שליסלען, און אין דער לעקציע ווערט געזוצט א פרייע וועדרסיע.

פארענדיקן דעם n8n אגענט און יסודות פון AI אגענטן (37:36:08)

שאפן דעם OpenAI API שליסל און פארבינדן דעם מאדעל (37:36:08)

דער קורס האלט אינמיטן בויען א "זעלבסט-לערנענדיקן אגענט" אין n8n. דער ערשטער שריט אין דעם חלק איז צו שאפן א נייעם API שליסל ביי OpenAI. מען גייט אריין אין דעם אפטייל "API keys", מען זעט דארט די אפציע "Create new secret key", און מען גיט דעם שליסל א נאמען: "Self-learning agent". דער "project type" בלייבט אויף "default", און די "permissions" ווערן געשטעלט אויף "all". דערנאך קליקט מען אויף "Create secret key".

עס ווערט געגעבן א ווארנונג: מען טאר קיינמאל נישט טיילן די געהיימע שליסלען מיט קיינעם. די דעמאנסטראציע ניצט א פרייען אקאונט, אבער ווען מען ניצט א באצאלטן אקאונט מיט געקויפטע "tokens", קען א צווייטער זיי מיסברויכן, און מען קען דארפן באצאלן גאר הויכע סומעס.

מען קאפירט דעם שליסל, מען מאכט צו דאס פענסטער, און מען לייגט אריין דעם שליסל אין דעם n8n פעלד. וועט עס האט שאלות אדער ספיקות, קען זיך באניצן מיט דעם "Ask Assistant", וואס איז אנגעצייכנט מיט עטליכע פארבן, און עס וועט העלפן. דער טעסט פון דער פארבינדונג ענדיגט זיך מיט הצלחה. מען גייט צוריק, מען מאכט צו דאס פענסטער אן אפשפייכלען עפעס נאכאמאל, און מען זעט א רשימה פון מאדעלן. מען קען אויסקלייבן וועלכן מאדעל מען וויל. אין דער דעמאנסטראציע בלייבט מען ביים דיפאלט מאדעל.

צולייגן דעם זכרון: Simple Memory (37:37:09)

צוריק אויפן "canvas", דארף מען איצט באשטימען דעם זכרון, וואס זאל געדענקען וואס עס וועט פארשריבן. דאס איז די "memory", א דאטא-באזע וואס היט אפ אלע איינטראגונגען און רעזולטאטן. עס זענען דא עטליכע אפציעס פאר זכרון: "Simple Memory", דער דיפאלט פון n8n, אדער אנדערע ווי PostgreSQL און MongoDB. מען קלייבט אויס "Simple Memory", ווייל עס פאדערט נישט קיין "credentials", כדי עס זאל בלייבן פשוט.

די "context window length" שטייט אויף 5, און מען לאזט עס אזוי, כדי נישט צו קאמפליצירן די זאך. ביי "connected chat" קלייבט מען אויס "define below". דארט דארף מען באשטימען דעם "key". דער שליסל איז נישט מער ווי דער ID פון דער רייַ (דער איינטראגונג) אין זכרון וואס איז די לעצטע. מען כאפט דעם ID פונעם רעכטן זייט און מען שלעפט אים ("drag and drop") אריין אין דעם "key" פעלד, וואו מען זעט דעם אנגעצייכנטן קורסאר. אזוי וועט דאס פעלד אויטאמאטיש איבערגעביטן פון "fixed" אויף "expression", און דער קאוד וועט אויטאמאטיש אויסגעפילט. דער נוצן דערפון: יעדעס מאל וואס עס קומט צו א נייע איינטראגונג, רעקארדירט דער סיסטעם דעם אדרעס פון דער איינטראגונג, און גיט צוריק די רעזולטאטן פונעם לעצטן זכרון.

צולייגן א געצייג: Airtable אלס "Memory Tool" (37:38:43)

דער לעצטער טייל פון דעם אגענט איז א "tool " (געצייג). עס זענען דא א ברייטע אויסוואל פון געצייג, למשל MCP און Pinecone, אבער מען קלייבט אויס Airtable. אין דעם אנהייב פונעם פראיעקט האט מען שוין געשאפן אלע נויטיגע "tokens", דעריבער זעט מען אויפן "parameters" בלאט אז דער "Airtable Personal Access Token" איז שוין אויסגעקליבן.

מען קען געבן דעם געצייג א נאמען. דער דיפאלט איז "Get a record", און מען ביט עס אויף "Memory Tool". ביי "tool description" לאזט מען עס זיך אויטאמאטיש באשטימען. די "resource" דארף זיין "record", און די "operation" "create or update" (אדער פשוט "create"). ביי "base" קלייבט מען אויס פון דער רשימה די Airtable דאטא-באזע וואס מען האט פריער געשאפן, די "AI agent learning" טאבעלע. ביי "table" קלייבט מען אויס, אויך פון דער רשימה, די קאלאנע מיטן נאמען "learning materials".

פאר די קאלאנעס קען מען אויסקלייבן "manual", און דאן דארף מען אליין אריינשרייבן דעם "material title". מען קען אבער אויך נעמען הילף פונעם AI אגענט אין n8n. ווי פריער, קען מען שלעפן און אריינלייגן, אדער ניצן דעם "autocomplete". וועז דאס פעלד שטייט אויף "expression" מאדע, וועט עס אויטאמאטיש אויסגעפילט. דער באניצער פרובירט קודם צו שרייבן אליין מיט געווייגלטע קלאמערן, נאר דערנאך באשליסט מען צו ניצן דעם "autocomplete" כדי נישט צו ריזיקירן. דער אינסטרוקציע-טעקסט וואס מען שרייבט איז בערך: "Add memory, write a memory for future reference about the user", אין בערך 100 ביז 150 אותיות אדער ווייניגער. עס וועט דערמאנט אז מען האט באשלאסן צו פארמיידן לאנגע שמועסן, און דאס איז דער גאנצער ציל דא.

צום סוף קליקט מען אויף "Let the model decide this parameter", און דער מאדעל שרייבט אויטאמאטיש די באשרייבונג, אזוי אז מען זאל נישט קענען אדורכפאלן. צוריק אויפן "canvas" זעט מען אז אלעס איז אויסגעשטעלט: דער "OpenAI Chat Model", די "Simple Memory", און דער "Memory Tool".

דער ענטפער-נאוד אין Telegram (37:41:49)

די לעצטע זאך איז נאכאמאל דער Telegram טייל. פון די "send" אפציעס קלייבט מען אויס "Send a text message". די פאראמעטערס זענען שוין אויסגעשטעלט. עס ווערט ערקלערט פארוואס מען דארף צוויי Telegram נאודס: אין אנהייב איז דא דער "trigger" וואס הייבט אן דעם פראצעס, אבער מען דארף אויך א נאוד אינעם סוף, וואס דינט נאר צום ענטפערן, כדי דער "bot" זאל קענען צוריקגעבן רעזולטאטן.

ביי "chat ID" נעמט מען דעם ID פונעם "Telegram Trigger". ווי מען האט פריער באשטימט דעם ID פון דער לעצטער איינטראגונג, אזוי ציט דער סיסטעם ארויס די דאטא לויטן ID, און גיט צוריק די רעזולטאטן. די רעזולטאטן זענען נישט מער ווי די איינטראגונגען וואס מען האט געגעבן דעם AI אגענט. די "resource" שטייט אויף "message", און די "operation" אויף "send message".

א טעות און דער תיקון (37:42:50)

ווען מען טעסט, זעט מען אז די איינטראגונגען ווערן פראצעסירט, אבער דער סיסטעם קען נישט צוריקגעבן קיין ענטפער, און עס ווייזט זיך א טעות. דער קורס מיינט אז דער גרונט איז ווארשיינליך אז מען האט נישט קאנפיגורירט דעם "output", כאטש עס קען זיין אויך אן אנדער סיבה. א מעגליכע סיבה איז אויך א "premium" אבאנעמענט: ביי וועלכן LLM מען ניצט נאר (למשל DeepSeek, OpenAI אד"ג), דארף מען האבן א פארריעט פון "tokens" ווערט בערך 10 ביז 20 דאלאר.

דער תיקון: מען ביט דעם "text" פעלד אויף דעם "expression":

json.output

"Reply markup" בלייבט אויף "none" ווי דער דיפאלט. צוריק אויפן "canvas", זעט מען אז די טעות איז פארריכט, און די "triggers" פון Telegram ארבעטן ווי געריכט. דאן אקטיווירט מען דעם "workflow" און מען שפייכלט אים אפ.

שמועסן מיטן Bot אין Telegram (37:44:20)

מען גייט צוריק צו Telegram און מען הייבט אן צו שמועסן מיטן "chatbot". די ערשטע מעלדונג: "Hi, would you like to talk or would you like to know about me?". דערנאך: "Do you remember our previous conversations or shall we begin from scratch?". ערשט ענטפערט דער סיסטעם, אבער דערנאך ווערט דער ענטפער אויסגעמעקט, און עס איז קלאר אז עפעס איז נישט אין ארדענונג. דער קורס מיינט אז דאס איז צוליב די "tokens": מען האט גענוצט א פרייע וועריע פון ChatGPT, און מען דארף א "premium" וועריע מיט צוגעלייגטע "tokens". דער קורס זאגט אז אזוי ארבעט עס: מען לאדנט אראפ דעם "workflow", מען לייגט צו "tokens", און עס זאל ארבעטן.

א נייע לעקציע: דער סדר היום (37:44:51)

דא הייבט זיך אן א נייע לעקציע מיט א סדר: ערשטנס די יסודות פון AI אגענטן, דערנאך וויאזוי AI אגענטן זענען אנדערש פון LLMs, וויאזוי AI אגענטן ארבעטן, די צוקונפטיגע ריזיקעס פון AI אגענטן, געדאנקען וועגן AI אגענטן, און צום סוף, דער וויכטיגסטער פונקט: וויאזוי צו בויען אן AI קול-אגענט (voice agent).

וואס איז אן AI אגענט? (37:45:23)

אין זיין עיקר, איז אן AI אגענט א ווייכווארג-פראגראם וואס איז געבויט צו טון ארבעטן אליין (אויטאנאמיש). נישט ווי טראדיציאנעלע ווייכווארג, וואס פאלגט שטרענג פאראויס-באשטימטע כללים, קענען AI אגענטן מאכן באשלוסן לויט פארשטאנד, אינטעראקציע און דאטא אין דער רעאלער צייט. זיי ניצן מאדערנע מאדעלן ווי ChatGPT, Claude, Gemini אד"ג.

מען קען עס פארשטעלן ווי א דיגיטאלער געהילף וואס פאלגט נישט נאר שטייפע אינסטרוקציעס. אנשטאט צו זאגן "check if Tony is available on Friday and book a meeting", קען מען פשוט זאגן "schedule a meeting with Tony at the earliest convenient time in the next month". אויף אזא בקשה טוט דער אגענט אזוי: ערשט קוקט ער אין אייער קאלענדאר אויף פרייַ צייטן, דערנאך גייט ער צו Tony'ס קאלענדאר און געפינט א צייט וואס פאסט מיט אייערע, ער באשטימט די בעסטע צייט, און שיקט ארויס איינלאדונגען. אלעס דאס געשעט אליין.

דער אונטערשייד צווישן AI אגענטן און LLMs (37:46:24)

כאטש AI אגענטן ניצן גרויסע שפראך-מאדעלן ווי GPT-4, גייען זיי ווייטער פון בלויז שאפן טעקסט. דער הויפט-אונטערשייד: LLMs שאפן ענטפערס לויט דער דאטא אויף וועלכער זיי זענען טריינירט געווארן, אבער זיי האבן נישט קיין אקטיוו שייכות מיט דער וועלט. AI אגענטן, אויף דער אנדערער זייט, קענען צוקומען צו דרויסנדיקער דאטא, פלאנירן ארבעטן, און אויספירן אקציעס, אלעס אין דער רעאלער צייט.

א משל: וועו מען פרעגט אן LLM וועגן א ספארט-שפיל וואס איז פארגעקומען נעכטן, קען ער האבן אדער נישט האבן דערהייַנטיקטע ידיעות דערווען. אבער AI אגענטן קענען אראפנעמען לעבעדיקע דאטא פון וועב, פון דאטא-באזעס אדער פון APIs, און געבן אן אקוראטן ענטפער אויפן ארט.

"Hallucinations" (37:47:27)

עס ווערט ערקלערט דער באגריף "hallucination". מען קען טראכטן אז דאס איז נאר ביי מענטשן, אבער אויך אן LLM קען "האלוצינירן". א משל: עס זענען דא פארשידענע סארטן עפל, ווי Fuji, McIntosh, Granny אד"ג. אבער וועו מען שרייבט "tell me about a watermelon apple" אדער "orange mango", וואס זענען אינגאנצן נישט רעאל, אדער גאר אנדערע פרוכטן, קען אן LLM געבן פאלשע און אוממעגליכע אינפארמאציע וואס הערט זיך אן גלייבווערדיג. דאס הייסט "hallucination". עס ווערט צוגעלייגט אז די נייערע LLMs האבן א מאסשטאב צו מעסטן זייערע האלוצינאציעס.

וויאזוי ארבעטן AI אגענטן: פיר פעאיגקייטן (37:47:57)

AI אגענטן ארבעטן ווי פארגעשריטענע פראבלעם-לייזער מיט פיר הויפט-פעאיגקייטן:

- פלאנירן
- אינטעראקציע מיט געצייג
- זכרון און צוטריט צו וויסן
- אויספירן אקציעס

פלאנירן: AI אגענטן הייבן אן מיט א ציל, צי עס איז פארשן מארק-טרענדס אדער פארפאסן אן אימעיל. זיי צוטיילן דעם ציל אויף שריט וואס מען קען אויספירן, לויט א "chain of thought" מעטאד, כדי צו פארבעסערן די אויספירונג. דאס נעמט אוועק דאס נויט פון פאראויס-באשטימטע מענטשלעכע "triggers".

אינטעראקציע מיט געצייג: נישט ווי פשוטע AI מאדעלן, קענען AI אגענטן ארבעטן מיט געצייג: זוכן אויפן אינטערנעט, פרעגן דאטא-באזעס, אדער ניצן APIs. דאס פארברייטערט זייערע פעאיגקייטן ווייטער פון א פעסטן, אומבאוועגלעכן וויסן.

זכרון און צוטריט צו וויסן: AI אגענטן קענען ארויסנעמען און אפהיטן אינפארמאציע מיט טעכניקן ווי RAG, דאס הייסט "retrieval augmented generation". דאס דערלויבט זיי אריינצונעמען דאטא וואס איז ספעציפיש פאר א פירמע, מארק-פארשונג, אדער שאלות פון קונה-סערוויס, כדי צו שאפן דערהיינטיקטע ענטפערס.

אויספירן אקציעס: די לעצטע שטופע גייט ווייטער פון נאר פראצעסירן אינפארמאציע. AI אגענטן נעמען אקציעס, למשל שרייבן באריכטן און פירן ארבעטן.

בויען אן AI קול-אגענט מיט Vapi (37:49:30)

די צוקונפט און די סכנות פון AI אגענטן (37:49:30)

דער קורס ענדיקט פריער דעם טייל ווען AI אגענטן קענען טון: זיי קענען ארבעטן מיט פארשידענע אפליקאציעס און אפילו רעדן מיט אנדערע AI אגענטן. דאס מאכט זיי פאר שטארקע אויטאמאציע-געצייג, וואס קענען אליין דורכפירן קאמפליצירטע ארבעטס-פראצעסן.

דערנאך גייט מען איבער צום נעקסטן טייל: די צוקונפט און די ריזיקעס פון AI אגענטן. עס ווערט ערקלערט אז AI אגענטן ברענגען א נייע תקופה פון אויטאמאציע און עפעקטיווקייט. אבער ווייל זיי ארבעטן אליין, ברענגט עס אויך סכנות. אן גוטע זיכערהייטס-מיטלען פון פריער, קען אן AI אגענט פאלש פארשטיין דעם ציל. דער קורס ברענגט א ביישפיל: אויב מען זאגט פאר אן AI אן קאנטראל אז ער זאל ברענגען שלום אויף דער וועלט, קען ער קומען צו עקסטרעמע לייזונגען. דערפאר ווערט באטאנט אז מען דארף האבן מענטשליכע אויפזיכט, עטישע פראגראמירונג און א פאראנטווארטליכע אריינפירונג, כדי מען זאל קענען באניצן AI אגענטן זיכער און עפעקטיוו.

לעצטע געדאנקען וועגן AI אגענטן (37:50:15)

אלס סיכום זאגט דער קורס אז AI אגענטן טוישן ווי מיר ארבעטן און ווי מיר האבן צו טון מיט טעכנאלאגיע. זיי קענען פלאנירן, ארבעטן מיט געצייג, אפהיטן וויסן און דורכפירן אויפגאבעס. דערפאר זענען זיי אסאך מער אנטוויקלט ווי די געווענליכע LLM מאדעלן.

אנהייבן מיט Vapi און א פרובירונג-שמועס (37:50:30)

יעצט קומט מען צום וויכטיגסטן טייל פון דער לעקציע: ווי אזוי מען בויט אן AI קול-אגענט. צו בויען אן AI קול-אסיסטענט באניצט מען זיך מיט א געצייג וואס הייסט Vapi. אויפן וועבזייטל זעט מען פארשידענע ביישפילן פון באנוץ, וויכטיגע אייגנשאפטן פון Vapi, און אויך בלאגס, קאריערע-זייטן, אינפארמאציע פאר סטארטאפס און דאקומענטאציע.

אויפן דעשבארד איז שוין דא א פארטיגער Vapi אסיסטענט, מיט וועמען מען קען גלייך רעדן. מען גיט די נויטיגע ערלויבענישן (צוטריט צום מיקראפאן), און דער אסיסטענט הייבט אן: "ברוך הבא צו Vapi, איך בין אן אסיסטענט מיט וועמען מען קען רעדן ווי מיט א מענטש. ווי גייט דיר דער טאג?"

אין דעם שמועס פרעגט דער אסיסטענט ווי אזוי מען האט זיך דערוואוסט פון Vapi. די ענטפער אין דער לעקציע: מען האט געזוכט אויף וועבזייטלעך ווי צו בויען אן אגענט וואס ענטפערט אויף אימעילס, אויף טעלעפאן-רופן און מאכט באשטימונגען (אפוינטמענטס), און אזוי איז מען אנגעקומען צו Vapi. דער אסיסטענט לויבט דעם פלאן און פרעגט וועלכע ספעציפישע פונקציעס מען וויל.

דא ווערט דערציילט דער פאקטישער ביישפיל פון וואס מען וויל בויען, א מעדיצינישער אגענט וואס ארבעט פאר א דאקטאר:

- ער באקומט טעלעפאן-רופן פון פאציענטן.
- ער טשעקט ווען דער דאקטאר איז פריי.
- ער שאפט די פריסטע מעגליכע באשטימונג.
- אויב דער פאציענט איז שוין אן עקזיסטירנדער פאציענט פון דעם דאקטאר, דארף מען זיך נישט זארגן וועגן פאפירן.
- אויב ער איז נייער ביים דאקטאר, דארף ער געבן געוויסע פרטים, און דער אגענט זאמלט איין אלע פרטים.
- ער קוקט אין דעם דאקטארס קאלענדאר, שטעלט איין די באשטימונג, און שיקט ארויס א באשטעטיגונג צום פאציענט.

דער אסיסטענט ענטפערט אז דאס איז א פרעכטיגער באנוץ: אן אגענט וואס פירט פאציענטן-רופן, טשעקט ווען דער דאקטאר איז פריי און באהאנדלט פאפירן, קען שפארן א סך צייט סיי פאר פאציענטן סיי פאר דאקטוירים. ווען מען זאגט אז מען וויל טיפער אויספארשן די אפציעס, איינשטעלונגען און מעגליכקייטן, און פרובירן בויען איינעם אין דער פראקטיק, שלאגט דער אסיסטענט פאר צו קוקן אין דער דאקומענטאציע, אדער אז ער קען העלפן פירן דורך דער איינשטעלונג. צום סוף זאגט מען אז מען וויל צוריקגיין צום דעשבארד און דארט אנהייבן, און דער אסיסטענט ווינטשט הצלחה און זאגט אז מען קען שפעטער צוריקקומען אויב מען דארף הילף.

נאכן שמועס ווערט באמערקט ווי ריאליסטיש עס האט געקלונגען. עס האט אויסגעזען ווי מען רעדט מיט א פאקטישן מענטש, ווי למשל א קאסטומער-סערוויס ארבעטער.

איינשטעלן דעם מאדעל: פראווייידער, פראמפט, טעמפעראטור און טאקענס (37:54:38)

יעצט הייבט מען אן בויען אן אגענט פאר זיך. עס זענען דא עטלעכע אפטיילונגען: Model, Transcriber, Voice, Tools, Analysis און Advanced. מען גייט דורך זיי שריט ביי שריט.

אין דער Model אפטיילונג קען מען אויסקלייבן יעדן AI פראווייידער: Google, א Custom LLM אויב מען האט אן אייגענעם, Groq, DeepSeek, Mistral, און נאך אסאך אנדערע. די ערשטע ברירה איז OpenAI מיט דעם מאדעל) GPT-4oאין לעקציע ווערט ער אנגערופן "GPT-4 cluster"). מען קען אויך צולייגן טעקעס, למשל אויב מען האט אליין געשריבן א פראמפט אדער א ספעציעל צוגעשטעלטן פראמפט, קען מען אים דא אריינלייגן.

דערנאך קומט די אפציע Temperature. לויט דער ערקלערונג אויפן זייטל, קאנטראלירט די טעמפעראטור ווי צופעליג דער אויסגאנג איז: א העכערע צאל גיט מער צופעליגע ענטפערס, א נידעריגערע צאל, נאענט צו נול, גיט מער באשטימטע ענטפערס. אין דער לעקציע ווערט געזאגט אז בעצם רעדט מען דא פון "האלוצינאציע". די ערקלערונג אין קורס:

- א מינימאלע טעמפעראטור מאכט דעם שמועס מערסטנס מענטשליך, אזוי אז מען פילט ווי מען רעדט מיט א פאקטישן מענטש.
- טעמפעראטור נול קלינגט ווי א ראבאט, אן געפילן און אן אויף-און-אראפ אין טאן.
- א צו הויכע טעמפעראטור גיט אן איבערגעטריבענעם, איבער-באגייסטערטן טאן, וואס קלינגט אויסגעטראכט און האלוצינירנדיק, נישט באקוועם און נישט זיכער.

דערפאר שטעלט מען איין 0.5, וואס מאכט אז עס זאל קלינגען מער מענטשליך און פארשטענדליך. מען דארף געבן אן אפטימאלע טעמפעראטור.

דערנאך קומט Max Tokens: דאס איז די מאקסימאלע צאל טאקענס וואס דער אסיסטענט טאר שאפן ביי יעדן ריי אין שמועס. וועו מען הייבט אן א שמועס מיטן אגענט, באניצט ער ביז 250 טאקענס צו פארשטיין דעם אריינגאב אדער באפעל און צו געבן א ענטפער.

דער טראנסקריבער: שפראך, Nova 3, הינטערגרונט-קלאנג און שליסל-וועדטער (37:57:11)

אין דער Transcriber אפטיילונג שטעלט מען איין ווי דער טראנסקריבער פירט זיך. מען קלייבט אויס Deepgram אלס פראווויידער, די שפראך איז ענגליש, דער מאדעל איז Nova 3, און עס איז דא אן אפציע פאר הינטערגרונט-קלאנג.

דער קורס ערקלערט די אפציע מיט א ביישפיל: לאמיר זאגן מען בויט אן AI אגענט פאר א שפיטאל, און מען וויל אז ער זאל זיין אזוי ריאליסטיש אז וועו א קליענט רעדט מיט אים, זאל ער נישט באמערקן אפילו קיין סימן אז ער רעדט נישט מיט א מענטש נאר מיט אן AI אגענט. דעמאלט קען מען אנצינדן די הינטערגרונט-קלאנג אפציע. וועו זי איז אנגעצונדן, גיט זי א הינטערגרונט-קלאנג וואס פאסט צום פעלד. ביי א שפיטאל, למשל, קען מען הערן קולות פון אן אמבולאנס אדער פון מענטשן וואס גייען ארום, ענליך צום פאקטישן ארט, אבער מיט זהירות.

מען קען אויך צולייגן שליסל-וועדטער, למשל מעדיצינישע טערמינען ווי ICU, נוירא-כירורג, אפציעס פון אפעראציע, אקסידזשען. אויב מען וויל צולייגן טערמינען וואס געהערן צום מעדיצינישן פעלד, לייגט מען זיי דארט אריין.

דאס קול, געצייג און אנאליז (37:58:40)

אין דער Voice אפטיילונג זענען דא אסאך קולות אויסצוקלייבן. למשל, אויב מען וויל א פראפעסיאנעלן אינדישן קול, איז דא אזא אפציע. עס זענען דא נעמען ווי Hannah אדער Paige, וואס ווערט באשריבן ווי א טיפער טאן, באַרואיגנדיק און פראפעסיאנעל, און אויך Harry און Lily. די ערשטע ברירה איז Elliot, און לויט דער לעקציע איז Elliot דער זעלבער קול וואס האט געשמועסט אין אנהייב. מען קלייבט אויס Lily, און מען קען זי פרובירן הערן.

אין דער Tools אפטיילונג קען מען אויסקלייבן ספעציפישע געצייג און שוין-פארטיגע פונקציעס.

אין דער Analysis אפטיילונג קען מען זאמלען דאטא וועגן דעם וואס פאסירט אין שמועס. מען קען היטן וועלכע אינפארמאציע דער קליענט האט איבערגעגעבן, וואס פאר אן אויסטויש עס איז פארגעקומען מיטן AI אסיסטענט, און ווי ער האט געענטפערט. מען קען אויך באשטימען א צייט-גרענץ פאר אפשאצן צי דער שמועס איז געלונגען. עס נעמט בערך 30 סעקונדעס אפצושאצן צי ס'איז געווען א געלונגענער שמועס.

אדוואנסירטע איינשטעלונגען און טעקסט-מעסעדזשעס (37:59:59)

אין די Advanced איינשטעלונגען זענען דא זאכן ווי פריוואטקייט, דערקענען א ווָאיסמעיל, און איינשטעלונגען פאר א טיימאוט פונעם רוף. מען קען באשטימען וויפיל צייט צו ווארטן אויף א ענטפער, און אויב עס קומט נישט קיין ענטפער פון דער צווייטער זייט, קען דער אגענט אליין אפשניידן דעם רוף. דאס דארף מען האלטן אין זינען.

מען קען אויך געבן עטלעכע אפציעס דורכן דריקן אויף נומערן: למשל "1" פאר איין שפראך, "2" פאר אן אנדערע שפראך. ביי א מעדיצינישן אגענט קען מען מאכן "1" פאר א נוירא-כירורג און "2" פאר א הארץ-כירורג.

דערצו קען מען אנצינדן טעקסט-מעסעדזשעס. עס איז שוין דא א פארגעשריבענער מעסעדזש, און מען קען אים טוישן. ווייל מען האט אויסגעקליבן דעם קול Lily, שרייבט מען אין די אדוואנסירטע איינשטעלונגען א מעסעדזש ווי: "דאס איז Lily פון ABC illness partners, דאס איז אייער באשטימונג, האט א גוטן טאג." עס זענען אויך דא איינשטעלונגען פאר מעסעדזשעס ווען דער שמועס שטייט: די מאקסימאלע צאל אזעלכע מעסעדזשעס און די מאקסימאלע צייט פון שטילקייט.

לאטענסי, טעלעפאן-נומערן און איבערפירן רופן (38:01:41)

אזוי איז געגאנגען דער גאנגער פלאן פאר דעם מאדעל. דערנאך ווערט דערמאנט די לאטענסי (די פארשפעטיקונג אין ענטפערן): זי הענגט אפ פון ווען מאדעל און וועלכן פראווייידער מען קלייבט אויס. אין דער לעקציע איז די מיינונג אז OpenAI מיט GPT-4o איז די בעסטע ברירה, ווייל ער איז די ערשטע ברירה, דערפאר טוישט מען דארט גארנישט.

אויף דער לינקער זייט זעט מען אויך אסיסטענט-workflows, טעלעפאן-נומערן און אנדערע זאכן. מען קען למשל באשטימען: אויב דער אגענט הערט א געווייסן שליסל-ווארט, זאל ער איבערפירן דעם רוף צו דיר. אדער ביי געווייסע רופער, אדער אויב א מענטש זאגט "ס'איז א נויט-פאל, איך דארף זיך בארעטן מיט עמעצן", קען מען באשטימען א גרופע שליסל-ווערטער, און דער רוף ווערט איבערגעפירט פונעם אגענט צו דיר. דאס טוט מען דורכן געבן די אייגענע טעלעפאן-נומער. עס איז אויך דא אן אפטיילונג פאר געצייג און טעקעס, וואו מען קען ארויפלייגן אלעס וואס מען דארף.

צום סוף גייט מען צוריק צום הויפט-זייטל און צו די workflows. דארט קען מען זען ווי דער מאדעל ארבעט, און ווייטער פארזעצן מיטן מאדעל וואס מען האט ביז יעצט אויפגעשטעלט.

א שטים-אגענט אין פראקטיק, און די יסודות פון AI-אגענטן (38:02:49)

א פרואוו-געשפרעך מיט דעם שטים-אגענט Riley (38:02:49)

אין דער לעקציע ווערט ווייטער געוויזן ווי מען בויט א שטים-אגענט אויף דער פלאטפארמע) Vapiאין די קעפשנס הייסט עס "WPY" אדער "Vappy"). מען קען דארט שאפן עטליכע מאדעלן. ביי דיפאלט איז שוין דא אן אסיסטענט מיטן נאמען Riley, און מען האט פרובירט צו טוישן דעם נאמען פון Riley אויף Lily, כדי צו זען צי די ענדערונג ווערט אנגענומען.

דער קורס ערקלערט אז וועו מען וויל א נייעם AI-אסיסטענט, קען מען צולייגן נאך אגענטן אונטער Riley. מען קען דעם אגענט טעסטן און מיט אים רעדן. ערשט וועו מען איז צופרידן מיט ווי דער אגענט ארבעט, גייט מען ווייטער און מען פארעפנטלעכט אים ("publish").

דערנאך ווערט אריינגעשטעלט א פראמפט וואס איז שוין געווען גרייט. מען קאפירט אים אריין אין דעם פלאץ פאר דעם פראמפט, און מען הייבט אן רעדן מיט דעם אסיסטענט. נאך א קורצער רגע ווערט דער מייקראפאן אקטיוויזירט, און דער אגענט רעדט.

מען הערט דעם גאנצן שמועס:

• דער אגענט: "א דאנק פארן רופן Wellness Partners. דא רעדט Riley, אייער אסיסטענט פאר באשטעלונגען. ווי קען איך אייך היינט העלפן?"

• דער באניצער זאגט "שלום" און פרעגט ווי עס גייט. דער אגענט ענטפערט אז עס גייט גוט, און פרעגט ווי ער קען העלפן באשטעלן א טערמין.

• דער באניצער זאגט אז ער וויל זען א נעוראכירורג (neurosurgeon). ער וויל מאכן סקענס, וויסן ווי עס שטייט מיטן מוח, און באקומען מעדיצין קעגן זיינע מיגרענען.

• דער אגענט ענטפערט אז ער וועט גערן העלפן, און פרעגט צי דער באניצער איז שוין אמאל געווען אין דער קליניק, אדער צי דאס איז זיין ערשטער באזוך.

• דער באניצער זאגט אז דאס איז זיין ערשטער באזוך.

• דער אגענט זאגט אז ווייל עס איז דער ערשטער באזוך, דארף ער נעמען גרונט-אינפארמאציע: פולער נאמען, געבורט-טאג און א טעלעפאן-נומער.

• דער באניצער גיט דעם נאמען "Tony Stark" און א טעלעפאן-נומער וואס הייבט זיך אן מיט 9986 און ענדיגט זיך מיט 1234.

• דער אגענט דאנקט און פרעגט נאכאמאל דעם געבורט-טאג. דער באניצער ענטפערט: 2טן אפריל 1984.

• דער אגענט זאגט אז ער קוקט נאך וואס איז פריי ביים נעוראכירורג און פאר די סקענס, און ער מאכט א קורצע פויזע. דערנאך שלאגט ער פאר צוויי טערמינען: מיטוואך, 15טן נאוועמבער, 10:00 אינדערפרי, אדער פרייטאג, 17טן נאוועמבער, 2:00 נאכמיטאג.

• דער באניצער זאגט אז ביידע פאסן, אבער ער קלייבט דעם צווייטן.

• דער אגענט באשטעטיגט דעם טערמין פרייטאג, 17טן נאוועמבער, 2:00 נאכמיטאג. ער בעט קומען 20 מינוט פריער כדי אויסצופילן די פארמען פאר נייע פאציענטן. מען דארף מיטברענגען דעם אינשורענס-קארטל, א פאטא-אידענטיפיקאציע, און א ליסטע פון אלע מעדיצינען וואס מען נעמט יעצט. דער טערמין וועט דויערן בערך 45 ביז 60 מינוט.

• דער אגענט פרעגט צי דער באניצער וויל באקומען א רוף אדער א טעקסט פאר דעם טערמין, אלס דערמאנונג. דער באניצער זאגט יא.

• דער אגענט שטעלט אן די דערמאנונג, דאנקט, און פרעגט צי ער קען נאך מיט עפעס העלפן. דער באניצער זאגט אז דאס איז אלעס, און דער אגענט ענדיגט: "מיר קוקן ארויס אייך צו זען דעם 17טן נאוועמבער."

וואס מען קען טון מיט אזעלכע אגענטן, און וויפיל עס קאסט (38:05:52)

דער קורס באמערקט אז דער נאמען איז נישט געטוישט געווארן פון Riley אויף Lily, אבער דאס איז נישט קיין גרויסע פראבלעם. דער שמועס אליין איז געווען זייער גוט.

אויף דעם אופן, ווערט ערקלערט, קען מען שאפן אייגענע AI-שטים-אגענטן לויט דעם געביט אין וועלכן מען ארבעט:

- אן AI-אסיסטענט פאר ביזנעס-צוועקן;
- אן אסיסטענט פאר "קאלד-קאלינג": רופן מענטשן, ערקלערן זיי דעם ביזנעס, און באשטעלן א מיטינג;
- אגענטן וואס מען בויט פאר קליענטן וואס ווילן האבן אן AI-אסיסטענט, און אזוי פארדינען געלט.

מען זעט אויפן עקראן אז Vapi גיט 1,000 פרואוו-מינוטן אומזיסט. נאך דעם דארף מען באצאלן פאר זייערע סערוויסעס, און מעגליך אויך פאר באצאלטע מאדעלן. דער קורס זאגט אז דאס איז נישט קיין גרויסער סכום. מען קען קויפן א גוטע מאס טאקענס, מאדעלן און מינוטן, שאפן ערשטקלאסיגע אסיסטענטן, און דאס קען ברענגען "פאסיווע הכנסה".

וואס איז אן AI-אגענט? (38:07:25)

נאכדעם הייבט זיך אן אן איינפירונג פון יסוד. אן AI-אגענט איז א ווייכווארג-פראגראם וואס פארשטייט וואס עס טוט זיך ארום אים, נעמט באשלוסן, און טוט אקציעס אליין. מען קען עס פארגלייכן מיט א קלוגן אסיסטענט: ער באמערקט זיין סביבה, פארארבעט די אינפארמאציע, און האנדלט אויף אן אופן וואס העלפט דערגרייכן א באשטימטן ציל.

דער הויפט-אונטערשייד איז אז ער דארף נישט קיין שטענדיגע מענטשלעכע אנווייזונג. ווען ער איז איינמאל אויפגעשטעלט, ארבעט ער זעלבסטשטענדיג, און נעמט באשלוסן לויט דער דאטא וואס ער באקומט. מיט פשוטע ווערטער: אן AI-אגענט איז א סיסטעם וואס קען באמערקן, טראכטן און האנדלן, אלעס כדי צו פארענדיגן א טאסק אויף אן עפעקטיוון אופן.

פארוואס זיי וואקסן אזוי שנעל אין 2026 (38:08:25)

כדי דאס צו פארשטיין, דארף מען קוקן אויף דער היסטאריע פון AI:

- 1950ער יארן: AI איז נאר געווען א געדאנק, מיט זייער פשוטע סיסטעמען און באגרעניצטע פעאיגקייטן.
- 2010ער יארן: דיפ-לערנען האט אלעס געטוישט. מאשינען האבן אנגעהויבן לערנען מוסטערן פון ריזיגע מאסן דאטא, ספעציעל דורך נויראל-נעצן.
- 2022: דער אמת'ער דורכברוך איז געקומען מיט שמועס-AI ווי ChatGPT. AI האט פלוצעם געקענט פארשטיין און שאפן שפראך ווי א מענטש, און עס איז געווארן צוגענגליך פאר יעדן, נישט נאר פאר פארשער.
- 2026: די תקופה פון "מולטימאדאלן" AI. עס פארארבעט טעקסט, בילדער, אודיא און אפילו ווידעא, אלעס צוזאמען.

צוליב דעם ענטפערן AI-אגענטן שוין נישט נאר אויף פראגעס, נאר זיי טוען טאקע ארבעט. דארט קומט די צען-פאכיגע העכערונג אין פראדוקטיוויטעט: זיי אויטאמאטיזירן ארבעט-פראצעסן, נעמען באשלוסן, און פירן

אויס אקציעס איבער עטליכע געצייג. דאס איז נישט נאר א פארבעסערונג, נאר אן איבערגאנג: פון AI וואס העלפט, צו AI וואס האנדלט.

די השפעה אין דער וועלט און אויפן ביזנעס (38:09:26)

דער קורס ברענגט פאלגנדע ציפערן:

- פיראליטעט: שאפער און ביזנעסער דערגרייכן מיט AI -אגענטן קאנסיסטענט איבער 500,000 קוקער אויף YouTube.
- צייט: טאסקס וואס האבן פריער גענומען 40 שעה א וואך, ווערן היינט שטארק פארקלענערט, ווייל דער אגענט נעמט איבער די איבערגעחזרטע ארבעט וואס נעמט אוועק צייט.
- עפעקטיווקייט: מער ווי דריי מאל העכער. דאס מיינט שנעלערע אויספירונג, וויניגער ארבעט מיט די הענט, און בעסערע רעזולטאטן.
- הכנסה: אין געוויסע פעלער ברענגען אגענטן $5,000 ביז $50,000 א חודש. אזוי ווערן זיי פון א געצייג א מקור פון הכנסה.
- אנגענומען ווערן: א וואוקס פון 300% יעדעס יאר. פירמעס עקספערימענטירן שוין נישט נאר, נאר זיי אינוועסטירן אקטיוו.
- דער מארק: מען ערווארט אז ער וועט דערגרייכן איבער $2 ביליאן. מען שטייט נאך אינעם אנהייב, מיט ריזיגע וואוקס-מעגליכקייטן.

די דריי יסוד-טעכנאלאגיעס (38:10:57)

- NLP)פאראַרבעטן נאטירלעכע שפראך): ערמעגליכט AI צו פארשטיין מענטשלעכע שפראך און דערויף ענטפערן, סיי אין טעקסט סיי אין שטים. עס כאפט דעם טייטש, דעם קאנטעקסט און די כוונה, ווי מענטשן רעדן צווישן זיך.
- אויטאמאציע: דא ליגט דער אמת'ער כח. דער אגענט טוט איבערגעחזרטע טאסקס אן מענטשלעכן איינמישן, ווי שיקן אימעילס, דערהיינטיגן דאטא, אדער פירן ארבעט-פראצעסן. ער טוט דאס עפעקטיוו און אלעמאל אויפן זעלבן אופן.
- API: לאזט דעם אגענט זיך פארבינדן מיט דרויסנדיגע געצייג און סערוויסעס. למשל, ער קען ציען דאטא פון א דאטא-באזע, שיקן מעסעדזשעס דורך WhatsApp, אדער אנצינדן אקציעס אין אנדערע ווייכווארג.

ווען מען פאראייניגט אלע דריי (NLP פארן פארשטיין, אויטאמאציע פארן אויספירן, און API פארן פארבינדן) באקומט מען א פולע סיסטעם וואס טראכט, האנדלט, און ארבעט צוזאמען מיט דער דיגיטאלער וועלט.

די וויכטיגסטע געצייג און פריימווערקס (38:11:59)

- LangChain: איינס פון די מערסט פאפולערע פריימווערקס פאר אפליקאציעס וואס לויפן אויף LLM. עס פארבינדט עטליכע שריט אין איין ארבעט-פראצעס, למשל פראמפטס און געצייג.
- AutoGen)אין די קעפשנס "autog"): גייט א שריט ווייטער, מיט זעלבסטשטענדיגע אגענטן וואס פלאנירן טאסקס, פירן זיי אויס, און חזרן איבער, כמעט ווי אן אומאפהענגיגער ארבעטער.
- Google Cloud AI Studio: צו אנטוויקלען און ארויסלאזן מאדעלן אויף גרויסן פארנעם, ספעציעל פאר פראדוקציע-אפליקאציעס מיט א שטארקער אינפראסטרוקטור.

- OpenAI API: גיט צוטריט צו שטארקע שפראך-מאדעלן וואס פארשטייען טעקסט, שאפן טעקסט, און טראכטן דורך. עס לויפט הינטער אסאך מאדערנע אפליקאציעס.
- n8n: א געצייג פאר אויטאמאציע פון ארבעט-פראצעסן. מען פארבינדט אפליקאציעס וויזועל, אן שרייבן אסאך קאוד.
- Hugging Face: א פלאטפארמע מיט אדוואנסירטע NLP- און מאשין-לערנען-מאדעלן, צוואמען מיט געצייג צו טריינירן, צופאסן ("fine-tuning") און ארויסלאזן.

צוואמען גיבן די אלע געצייג אלעס וואס מען דארף: פון בויען און טריינירן מאדעלן, ביז ארויסלאזן פולשטענדיגע אגענטן.

אויסקלייבן דעם ריכטיגן פראיעקט (38:13:30)

- אנהייבער-שטאפל: פשוטע פראיעקטן, ווי טשעט-באטס אדער אויטאמאציע פון אימעילס. מען לערנט ווי AI ענטפערט, ווי מען בויט ארבעט-פראצעסן, און ווי געצייג פארבינדן זיך. דאס בויט א פעסטן יסוד.
- מיטל-שטאפל: מער קאמפליצירטע סיסטעמען, ווי אגענטן מיט עטליכע געצייג, אדער אויטאמאטישע באשטעלונגען. מען לערנט צו פאראייניגן פארשידענע API'ס.
- אדוואנסירטער שטאפל: פולשטענדיג זעלבסטשטענדיגע סיסטעמען, ווי ביזנעס-באטס אדער אגענטן מיט מאשין-לערנען, וואס נעמען באשלוסן מיט מינימאלן מענטשלעכן איינמישן.

די עצה: נישט שפרינגען גלייך צו אדוואנסירטע פראיעקטן. מען זאל אנהייבן פשוט, בויען זיכערקייט, און ארויפגיין ביסלעכווייז. עס גייט נישט וועגן קאמפליצירטקייט, נאר וועגן קאנסיסטענץ און וואוקס.

א ביישפיל: אגענט פאר אימעיל און קאלענדער (38:15:03)

דער אגענט סארטירט אימעילס, באשטעלט מיטינגען, און פירט דעם קאלענדער. אזוי בויט מען אים:

- Gmail API לאזט דעם אגענט לייענען די אריינקומענדיגע אימעילס.
- n8n שאפט דעם ארבעט-פראצעס, וואס באשטימט וואס זאל געשען מיט יעדן נייעם אימעיל.
- GPT-4 איז דער "מוח". ער פארשטייט דעם אינהאלט, באשטימט וואס איז דרינגענד, און באשליסט וועלכע אקציעס צו טון.
- צום סוף שיקט דער אגענט קלוגע ענטפערס אויטאמאטיש, אדער באשטעלט מיטינגען אין דעם קאלענדער.

דער רעזולטאט: מען שפארט 12 ביז 15 שעה א וואך. און מען קען זיך פארשטעלן ווי גרויס דער געווינס איז, וועו מען ניצט אזא אויטאמאציע אויף נאך געביטן.

נאך א ביישפיל: אגענט פאר שאפן און שעדזשולן אינהאלט (38:16:05)

דער לעצטער ביישפיל אין דעם שטיק איז אן אגענט פאר סאשעל-מידיע מענעדזשערס, שאפער און ביזנעסער וואס ווילן בלייבן קאנסיסטענט אנליין. ער שאפט אידייען פאר אינהאלט, און פארעפנטלעכט זיי אויטאמאטיש אויף זעקס אדער מער פלאטפארמעס.

צען AI-אגענט פראיעקטן און דער ערשטער בוי-שריט (38:16:35)

אן אגענט פאר סאשעל-מעדיע אינהאלט (38:16:35)

דער קורס ענדיגט צו ערקלערן אן AI-אגענט וואס שאפט און פובליקירט אינהאלט פאר סאשעל-מעדיע אויטאמאטיש, אן דארפן יעדן טאג ארבעטן מיט די הענט. עס ווערט ערקלערט ווי ער ארבעט, שריט נאך שריט:

- עס הייבט זיך אן מיט אריינגעגעבענע טעמעס: טרענדס, שליסל-ווערטער, אדער באשטימטע טעמעס אויף וועלכע מען וויל זיך קאנצענטרירן.
- דערנאך ניצט מען OpenAI צו שאפן אינהאלט פון הויכער קוואליטעט: קעפלעך (captions), פאסטן, אידייעס, און אפילו פארשידענע ווערסיעס צוגעפאסט פאר יעדער פלאטפארמע באזונדער.
- דערנאך קומט n8n, א געצייג וואס מען ניצט אסאך פאר אויטאמאציע-ארבעטס-פלוסן. עס פארבינדט אלעס צוזאמען, פון שאפן דעם אינהאלט ביז פלאנירן ווען צו פובליקירן און ביזן פובליקירן גופא.
- צום סוף פארשיקט דער אגענט אליין די פאסטן, און זארגט אז דער אינהאלט קומט ארויס אין דער ריכטיגער צייט אויף די ריכטיגע פלאטפארמעס.

דער רעזולטאט: מען קען אויטאמאטיזירן ביז זיבן פאסטן א טאג, קאנסיסטענט, אן ווערן אויסגעמוטשעט. דאס שפארט צייט, און עס האלט אויך א שטארקע און אקטיוועט פרעזענץ אויף דער אינטערנעט. אנשטאט פארברענגען שעהן מיט שאפן און פאסטן מיט די הענט, קען מען זיך קאנצענטרירן אויף סטראטעגיע און וואוקס, און דער אגענט נעמט איבער די אויספירונג.

אן אגענט פאר הוצאות און בודזשעט (38:17:37)

דער ווייטערדיגער ביישפיל איז זייער פראקטיש: אן AI-אגענט וואס פירט חשבון פון הוצאות און העלפט מאכן א בודזשעט. דער קורס רופט אים "דיין פערזענליכער פינאנץ-גורו", וואס קוקט שטענדיג נאך ווי מען גיט אויס געלט און העלפט נעמען קליגערע געלט-באשלוסן. ער פירט אויטאמאטיש חשבון פון די הוצאות, טיילט זיי איין אין קאטעגאריעס, און גיט אפילו עצות ווי מען קען שפארן.

ווי ער ארבעט:

- עס הייבט זיך אן מיט א באנק-API, וואס ברענגט זיכער אראפ די דאטא פון די טראנזאקציעס: וואו מען האט אויסגעגעבן, וויפיל, און ווען.
- n8n אויטאמאטיזירט דעם ארבעטס-פלוס: עס ארגאניזירט די דאטא און פירט זי ווייטער צו באארבעטן.
- GPT-4 דינט אלס דער שכל-שיכט. עס אנאליזירט די מוסטערן פון אויסגעבן, געפינט איבעריגע הוצאות, און שאפט נוצליכע איינזיכטן.
- צום סוף גיט דער אגענט בודזשעט-איינזיכטן און ווארנונגען, למשל ער ווארנט ווען מען גיט אויס צופיל, אדער ער שלאגט פאר וועגן ווי צו שניידן די קאסטן.

דער רעזולטאט: אין דורכשניט קען א באניצער שפארן בערך $450 א חודש, נאר דורכדעם וואס ער ווייסט בעסער וואס טוט זיך מיט זיין געלט און באשליסט קליגער. אנשטאט נאכשרייבן יעדע הוצאה מיט די הענט, ארבעט דער אגענט ווי א קלוגער פינאנץ-געהילף, און מען פירט זיך מיטן געלט מער עפעקטיוו און לייכטער.

א פארשונג- און צוזאמענפאסונג-געהילף (38:18:43)

אן אנדערע שטארקע אנוועגדונג איז א co-pilot פאר פארשונג און צוזאמענפאסונג. דער קורס רופט אים א "וויסן-נינדזשע", וואס קען דורכגיין ריזיגע מאסן אינפארמאציע און ארויסגעבן פונקט וואס מען דארף, אין א שטיקל פון דער צייט. ער זוכט אין פארשידענע מקורות, קלייבט צוזאם די שייכותדיגע אינפארמאציע, און שאפט קלארע און קורצע צוזאמענפאסונגען פון דער פארשונג.

- עס הייבט זיך אן מיט web scraping: דער אגענט קלייבט דאטא פון פארשידענע וועבזייטלעך, ארטיקלען און אנדערע מקורות.
- דערנאך קומט די אנאליז-שטאפל: מען פילטערט דעם אינהאלט און מען קלייבט אויס נאר די מערסט שייכותדיגע אינפארמאציע.
- דערנאך ניצט מען OpenAI פארן צוזאמענפאסן: דא פארוואנדלט דער אגענט גרויסע מאסן אינהאלט אין קורצע צוזאמענפאסונגען וואס זענען גרינג צו פארשטיין.
- צום סוף גיט דער אגענט ארויס א געארדנטן באריכט, וואס מען קען ניצן צו נעמען באשלוסן, פאר פרעזענטאציעס, אדער פאר וויטערע פארשונג.

די ווירקונג: אן ארבעט וואס נעמט געווענליך 12 שעה ווערט יעצט פארטיג אין בלויז 20 מינוט. אנשטאט פארברענגען שעהן מיט ליינען און צוזאמשטעלן אינפארמאציע, טוט דער אגענט די שווערע ארבעט, און מען קען זיך קאנצענטרירן אויף די איינזיכטן און אויף טון.

א שרייבער פאר רעזומעס און באגלייט-בריוו (38:20:17)

דער קורס רופט דעם אגענט "דאס געהיימע וואפן פון א ארבעט-זוכער". ער פאסט צו די רעזומע לויט דער באשרייבונג פון דער ארבעט, און ער שאפט פערזענליכע באגלייט-בריוו (cover letters) וואס זענען צוגעפאסט צו יעדער אפליקאציע באזונדער.

- ער הייבט אן מיט אנאליזירן די באשרייבונג פון דער ארבעט, און ער געפינט די וויכטיגע קענטענישן, פאדערונגען און שליסל-ווערטער נאך וועלכע די רעקרוטירער זוכן.
- דערנאך ניצט מען GPT-4 צו צופאסן די רעזומע און צו מאכן ATS-אפטימיזאציע. דער שריט זארגט אז די רעזומע פאסט צו דער ארבעט, און אז זי איז געאייגנט פאר applicant tracking systems, סיסטעמען וואס אסאך פירמעס ניצן צו פילטערן קאנדידאטן.
- צום סוף שאפט דער אגענט א צוגעפאסטן באגלייט-בריוו. ער איז נישט קיין אלגעמיינער בריוו, נאר ספעציפיש פאר דער ארבעט און פאר דער פירמע.

די ווירקונג: א קאנדידאט קען זען מער ווי 300% מער רופן צו אינטערוויוס, נאר ווייל זיינע אפליקאציעס זענען מער שייכותדיג און בעסער אפטימיזירט. אנשטאט שיקן די זעלבע רעזומע אומעטום, העלפט דער אגענט ארויסשטעכן, און יעדע אפליקאציע ווערט קליגער, מער געצילט און מער עפעקטיוו.

אויטאמאטישע פראדוקט-ליסטינגס און פרייז-נאכפאלגער (38:21:19)

דאס איז א ביזנעס-אנוועגדונג מיט א גרויסע ווירקונג, א "סופער-אגענט פאר e-commerce" וואס העלפט געשעפטן בלייבן קאנקורענצפעאיג אין עכטער צייט. ער סינכראניזירט אויטאמאטיש די פראדוקט-ליסטינגס, און ער פאסט צו די פרייזן אויף פלאטפארמעס ווי Amazon, Shopify און eBay, אלעס אן דארפן אריינמישן די הענט.

• צו ערשט פארבינדט זיך דער אגענט מיט די API's פון די פלאטפארמעס, און אזוי קומט ער צו צו די ליסטינגס און די פרייז-דאטא אויף די פארשידענע מארקעטפלעצער.

• דערנאך קוקט ער שטענדיג נאך די פרייזן פון די קאנקורענטן. דאס איז קריטיש אין e-commerce, וואו אפילו א קליינער פרייז-אונטערשייד קען ווירקן אויף די פארקויפן.

• מיט n8n ווערט דער גאנצער ארבעטס-פלוס אויטאמאטיזירט, און די דערהיינטיגונגען פאסירן גלייך און קאנסיסטענט.

• צום סוף ניצט דער אגענט דינאמישע פרייז-סטראטעגיעס: ער פאסט צו די פרייזן לויט דער קאנקורענץ, לויט דער פראגע (דימאנד), אדער לויט פריער-באשטימטע כללים.

דער רעזולטאט: געשעפטן קענען זען ביז 45% מער איינקונפט, נאר דורכדעם וואס זיי בלייבן קאנקורענצפעאיג און רעאגירן שנעל אויפן מארק. אנשטאט נאכקוקן די קאנקורענטן און טוישן פרייזן מיט די הענט, האלט דער אגענט דאס געשעפט שטענדיג אפטימיזירט אין עכטער צייט.

א באט פאר שפיל-סטראטעגיע (38:22:19)

דאס איז אן אייגנארטיגע אנוועندונג: א "שפיל-גאון", אן AI-אגענט וואס העלפט שפילער נעמען קליגערע באשלוסן אין עכטער צייט. ער אנאליזירט וואס עס טוט זיך יעצט אין שפיל, און ער שלאגט פאר די בעסטע סטראטעגיעס בשעת מען שפילט.

• דער אגענט פארבינדט זיך מיט דער API פון דער שפיל, וואס גיט דאטא אין עכטער צייט וועגן דער סביבה פון דער שפיל: שפילער, פאזיציעס, פונקטן אדער געשעענישן אין שפיל.

• דערנאך קומט computer vision: דער אגענט קען זען און פארשטיין וואס עס טוט זיך אויפן עקראן. דאס איז באזונדערס נוצליך ביי שפילן וואו נישט אלע דאטא זענען גלייך צוגענגליך דורך API's.

• דערנאך מאכט GPT-4 די סטראטעגיע-אנאליז: עס באארבעט די גאנצע אינפארמאציע, עס אפשאצט די מעגליכע צוגן, און עס זאגט פאראויס וואס איז דער בעסטער וועג.

• צום סוף גיט דער אגענט עצות וואס ווערן ארויפגעלייגט אויפן עקראן (overlay). דאס הייסט, ער ווייזט אנווייזונגען גלייך אויפן עקראן אין עכטער צייט, און דער שפילער קען גלייך נעמען בעסערע באשלוסן.

דער רעזולטאט: שפילער קענען דערגרייכן ביז 78% געווינען, דורכדעם וואס זיי ניצן קליגערע סטראטעגיעס וואס זענען געבויט אויף דאטא. אנשטאט זיך פארלאזן אויפן אינסטינקט, ארבעט דער אגענט ווי א טרענער אין עכטער צייט.

א הילף-דעסק אין פילע שפראכן (38:23:52)

דער אגענט איז א "פראדוקטיוויטעט-קראפט", באזונדערס פאר פירמעס וואס באדינען קונים אין פארשידענע לענדער און שפראכן. ער באהאנדלט קונה-שטיצע טיקעטס אין מער ווי 85 שפראכן, און ער לייזט בערך 89% פון די פראבלעמען.

• עס הייבט זיך אן מיט א מעסעדזש פון א קונה: פון א טשעט, פון אן אימעיל, אדער פון סיי וועלכע פלאטפארמע.

• די סיסטעם דערקענט אויטאמאטיש די שפראך פון דער מעסעדזש, גלייך, אן דארפן אריינגעבן עפעס מיט די הענט.

• GPT-4 שאפט אן ענטפער: עס פארשטייט די פראגע פונעם קונה, עס כאפט וואס ער ווייל, און עס גיט א קלארן, געוויסן ענטפער וואס קלינגט ווי פון א מענטש.

• צום סוף ווערט דער ענטפער אריינגעלייגט אין די טיקעטינג-סיסטעמען, אזוי אז מען קען נאכקוקן דעם ענין, נאכגיין דערויף, און פירן ביז דער לייזונג.

דער גרעסטער מעלה דא איז גיכקייט און די מעגליכקייט צו וואקסן. דער אגענט ענטפערט אין ווייניגער ווי 30 סעקונדעס. דאס פארבעסערט שטארק די דערפארונג פונעם קונה, און עס שניידט די ארבעטס-קאסטן. אנשטאט זיך פארלאזן אינגאנצן אויף גרויסע מאנשאפטן וואס רעדן פילע שפראכן, קענען געשעפטן ניצן אן AI-אגענט צו געבן שנעלע, קאנסיסטענטע קונה-שטיצע איבער דער גאנצער וועלט.

א פיטנעס- און מאלצייט-טרענער (38:25:26)

דער אגענט ארבעט ווי א פערזענליכער געזונט-געהילף, וואס העלפט מענטשן בלייבן קאנסיסטענט מיט זייערע צילן אין פיטנעס און ערנערונג. ער שאפט פערזענליכע טורנונג-פראגראמען און מאלצייטן לויט די צילן, די באליבטע זאכן, און דעם לעבנס-שטייגער פון יעדן איינעם.

• עס הייבט זיך אן מיט דעם וואס דער באניצער גיט אריין: צילן ווי אראפגיין אין וואג אדער באקומען מוסקלען, וואס ער עסט יא און וואס נישט, אדער וויפיל צייט ער האט.

• GPT שאפט א צוגעפאסטן פלאן, טורנונגען און פארשלאגן פאר מאלצייטן, ספעציפיש לויט די באדערפענישן פונעם באניצער.

• די דאטא ווערט אפגעהיט אין דאטא-באזעס, אזוי אז די סיסטעם קען נאכקוקן דעם פארשריט, דערהיינטיגן די פלענער, און האלטן קאנסיסטענץ מיט דער צייט.

• צום סוף שיקט דער אגענט טעגליכע דערהיינטיגונגען דורך SMS אדער אן אפליקאציע, און אזוי בלייבט דער באניצער פארנומען, מאטיווירט, און אויפן ריכטיגן וועג.

דער רעזולטאט: באניצער קענען זען באדייטנדע פארבעסערונגען, למשל ביז 15 פונט וויניגער אין בלויז 90 טעג. אנשטאט נאכגיין אלגעמיינע פלענער, גיט דער אגענט א ספעציעלן צוגאנג צו געזונט, וואס פאסט זיך צו און בלייבט קאנסיסטענט.

א קאנטראלירער פאר סמארט-מכשירים (38:26:29)

דאס איז א "פראדוקטיוויטעט-קראפט" פאר סמארט-הייזער, געמאכט כדי דאס טעגליכע לעבן זאל זיין באקוועמער און עפעקטיווער. דער אגענט קען קאנטראלירן מער ווי 50 סמארט-מכשירים אין שטוב, דורך קול-באפעלן און דורך אויטאמאטיזירטע רוטינען: פון לעמפ און פאכערס ביז זיכערהייט-סיסטעמען און אנדערע מכשירים.

• עס הייבט זיך אן מיטן קול: דער באניצער גיט א באפעל, ווי אנצינדן די לעמפ אדער טוישן די טעמפעראטור.

• דאס ווערט באארבעט דורך Alexa אדער דורך Google's API, וואס פארוואנדלען דאס קול אין באפעלן וואס מען קען אויספירן.

• n8n באהאנדלט די אויטאמאציע פונעם ארבעטס-פלוס, און באשליסט וועלכע אקציע מען דארף אויסלעזן.

• דערנאך קומען IFTTT כללים, וואס פארבינדן די פארשידענע מכשירים און באשטימען אויטאמאטישע לאגיק, למשל אויסלעשן די לעמפ ווען מען גייט ארויס פון שטוב.

- צום סוף ווערן די באפעלן אויסגעפירט דורך די פראטאקאלן פון די סמארט-מכשירים, וואס לאזן גלאט קאמוניקירן מיט אלע פארבונדענע מכשירים.

דער רעזולטאט: ביז 40% שפארונג אין ענערגיע, צוזאמען מיט מער באקוועמליכקייט און א קליגער לעבן. אנשטאט קאנטראלירן יעדן מכשיר באזונדער מיט די הענט, שאפט דער אגענט א קלוגע סביבה, וואו די שטוב רעאגירט אויטאמאטיש.

וועלכן פראיעקט צו בויען ערשט: "Home Automation Maestro" (38:28:02)

נאך די צען פראיעקט-אידייעס ווערט באמערקט אז רוב מענטשן קוקן אזא ליסטע, זאגן "שיין", און גייען ווייטער. דערפאר רעקאמענדירט דער קורס איין פראיעקט צו בויען ערשט, דער "home automation maestro". די סיבה: עס איז א זייער נוצליכע אויטאמאציע וואס מען קען מאכן אין איין טאג, עס איז נישט צו שווער צו בויען, און ווען עס ארבעט פילט זיך עס ווי כישוף.

דער פראבלעם מיט היינטיגע סמארט-הייזער: מען האט דא לעמפ, דארט אן AC, א שלאס, א TV, א רעדנער. זיי זענען אלע "סמארט", אבער זיי רעדן נישט איינער מיטן אנדערן. דעריבער מוז מען עפענען די Philips Hue אפליקאציע, דערנאך די Google Home אפליקאציע, דערנאך די אפליקאציע פונעם סמארט-שלאס: פינף פארשידענע אפליקאציעס נאר צו זאגן "איך גיי צו דער ארבעט".

דער home automation maestro פארעכט דאס. ער ווערט דער מוח וואס פארבינדט אלעס. מען שיקט אים איין מעסעדזש אויף WhatsApp, למשל "איך גיי ארויס", און ער פארשטייט וואס דאס מיינט: ער שליסט צו אלע טירן, לעשט אויס די לעמפ, שטעלט דעם AC אויף eco mode, און פארמאכט די פירהאנגען, אלעס מיט איין מאל. מען דארף מער נישט שפרינגען פון איין אפליקאציע צו דער אנדערער. דער קורס רעקאמענדירט צו בויען דעם פראיעקט אין איין סוף-וואך. עס נעמט בערך 3 ביז 4 שעה, און צום סוף האט מען עפעס וואס פארבעסערט טאקע דאס לעבן יעדן טאג.

צוגרייטונג: Python און Node.js (38:29:03)

דער ערשטער שריט איז Python, די פראגראמיר-שפראך מיט וועלכער מען וועט בויען דעם backend. עס ווערט ערקלערט מיט א משל: Python איז די שפראך וואס זאגט דעם קאמפיוטער וואס צו טון, ווי מען גיט אנווייזונגען צו א ראבאט, און Python מאכט די אנווייזונגען פארשטענדליך פארן ראבאט. מען האט אויסגעקליבן Python ווייל עס איז גרינג צו לערנען, אסאך פירמעס און פראגראמירער איבער דער וועלט פארשטייען עס, און עס איז פיל-זייטיג: מען קען דערמיט בויען אלעס, פון וועבזייטלעך ביז AI-פראגראמען.

- מען גייט אויף python.org און מען לאדנט אראפ די לעצטע ווערסיע. דארט קען מען זען דעם נומער פון דער ווערסיע, און מען דריקט אויפן download קנעפל.
- וען Python איז שוין אינסטאלירט, עפנט מען דעם) terminalמען זוכט "terminal" אין סיסטעם, ער איז דארט פון זיך אליין), און מען שרייבט דעם באפעל:

version-- python

- אויב Python איז אינסטאלירט, ווייזט עס די ווערסיע. אין דער לעקציע זעט מען Python 3.13.7.

דער ווייטערדיגער שריט, וואס איז אפציאנאל, איז אינסטאלירן Node.js, א געהילף צו לויפן א React frontend. מען דארף עס אויב מען וויל ארבעטן מיט א React frontend, וואס מען ווען ניצן אין דעם קורס

צו ווייזן פראדוקט-דאטא. React איז א JavaScript framework צו שאפן אינטעראקטיוע באניצער-אינטערפייסעס, דער טייל פון דער אפליקאציע וואס דער באניצער זעט און ארבעט מיט אים.

אינסטאלירן די געצייג און ערשטע FastAPI אפליקאציע (38:31:07)

NodeJS: פארוואס מען דארף עס און ווי אזוי מען אינסטאלירט (38:31:07)

דער קורס ערקלערט אז NodeJS איז א געצייג וואס לאזט לויפן JavaScript קאוד אינדרויסן פון א בראוזער. דאס איז זייער וויכטיג אויב מען וויל לויפן דעם React פראנט-ענד לאקאל אויף די אייגענע סיסטעם. מען דארף נישט קענען JavaScript פאר דעם קורס, אבער וועז NodeJS איז אינסטאלירט, איז גרינגער אויפצושטעלן דעם פראנט-ענד טייל פונעם פראיעקט.

ווי אזוי מען אינסטאלירט עס:

- מען גייט צו node.js.org.
- דארט זעט מען דריי אפציעס: Windows installer (MSI), און standalone binary . מען זאל דריקן אויפן Windows installer קנעפל און אינסטאלירן פון דארט.
- אין דער לעקציע ווערט Windows אויסגעקליבן, ווייל די סיסטעם איז Windows. ווער עס האט Mac OS אדער Linux, קען אויסקלייבן זיין אייגענע אפעריירונג-סיסטעם און אראפלאדן.

נאכן אינסטאלירן גייט מען צוריק צום טערמינאל און מען שרייבט דעם באפעל node --version כדי צו זען וועלכע ווערסיע פון NodeJS איז אינסטאלירט. וועז עס איז ריכטיג אינסטאלירט, ווייזט עס עפעס ווי v22.19.0, און דאס איז די ווערסיע-נומער פון Node אויף דער סיסטעם.

FastAPI און Uvicorn: וואס זיי זענען און ווי אזוי מען אינסטאלירט זיי (38:32:08)

נאכדעם וואס Python און NodeJS זענען אפגעהאנדלט, גייט דער קורס איבער צו די הויפט-טיילן פונעם פראיעקט, אנגעהויבן מיט FastAPI. עס ווערט נאכאמאל ערקלערט אז FastAPI איז דער framework מיט וועלכן מען בויט דעם בעק-ענד פון דער אפליקאציע. מען קען עס זיך פארשטעלן ווי א געצייג וואס מאכט גרינג צו באהאנדלען בקשות פון באניצער. למשל: וועז איינער דריקט אויף א קנעפל צו צולייגן א פראדוקט, שיקט עס צוריק די ריכטיגע דאטא, למשל א ליסטע פון די פראדוקטן. FastAPI איז מאדערן, גרינג צו ניצן און טאקע שנעל, און דערפאר איז עס א גוטע ברירה פאר בויען וועב-אפליקאציעס.

דערנאך קומט Uvicorn. דער קורס באמערקט אז עס הייסט "Uvicorn" מיט א U, נישט "unicorn". דאס איז דער סערווער וואס לויפט די FastAPI אפליקאציע. עס איז ווי א מאטאר וואס ברענגט די אפליקאציע צום לעבן, און עס באהאנדלט די בקשות וואס קומען פון דער דרויסנדיקער וועלט, למשל א באניצער וואס ארבעט מיט דער אפליקאציע. די צוויי ארבעטן צוואמען: FastAPI בויט דעם בעק-ענד, און Uvicorn לויפט אים.

די אינסטאלאציע:

- אינעם טערמינאל שרייבט מען pip install fastapi. דאס זאגט פאר Python צו אינסטאלירן FastAPI אויף דער סיסטעם. מען זעט אויפן עקראן אז FastAPI איז אינסטאלירט, און מען זעט אויך די פעקלעך (packages) וואס עס האט מיטאינסטאלירט.
- ווער עס וויל דערהייבן pip צו א נייערע ווערסיע, קען ניצן python.exe -m pip install--upgrade pip.

• דערנאך שרייבט מען pip install uvicorn, וואס אינסטאלירט דעם Uvicorn סערווער כדי מען זאל קענען לויפן די FastAPI אפליקאציע. מען זעט אז Uvicorn איז אויך אינסטאלירט.

Pydantic: קאנטראלירן די דאטא (38:33:40)

דערנאך ווערט ערקלערט א וויכטיגער באגריף: Pydantic. דאס איז א געצייג וואס FastAPI ניצט צו קאנטראלירן (validate) די דאטא וואס מען שיקט צו און פון דער אפליקאציע. דער משל: ווען מען פילט אויס א פארם מיט נאמען און עלטער, פאסט Pydantic אויף אז דער נאמען זאל זיין א) stringטעקסט) און דער עלטער א נומער. אזוי ווערט פארזיכערט אז די דאטא איז גילטיג און וועט נישט פאראורזאכן טעותים אין דער אפליקאציע.

אויסער דעם פארמאטירט Pydantic די דאטא אויף אן אופן וואס מאכט גרינג דערמיט צו ארבעטן. ווען מען שיקט פראדוקט-אינפארמאציע, ווי נאמען, באשרייבונג און פרייז, פאסט Pydantic אויף אז אלעס זאל זיין ריכטיג פארמאטירט. אין דער אפליקאציע פונעם קורס וועט מען ניצן Pydantic צו קאנטראלירן די פראדוקט-דאטא ווען מען לייגט צו, דערהיַנטיקט אדער מעקט אויס פראדוקטן.

VS Code, די Python פארלענגערונג און עפענען דעם פראיעקט (38:34:40)

דער ערשטער שריט איז VS Code. VS Code איז שוין אינסטאלירט אויף דער סיסטעם, און מען זאל זיכער מאכן אז מען האט די לעצטע ווערסיע. איידער מען עפנט א טעקע, איז דא א וויכטיגע זאך: מען דארף אינסטאלירן די Python extension. מען גייט אין דעם extensions חלק, זוכט "Python", און מען קלייבט אויס די אפציע וואס הייסט פשוט Python, נישט די debugger, און מען דריקט דערויף. דער קורס ווארנט אז ווער עס האט עס נאך נישט אינסטאלירט, זאל עס אינסטאלירן כדי דער קאוד זאל ארבעטן אין גאנצן.

דערנאך גייט מען צו File. מען קען עפענען א פאלדער אדער א טעקסט-טעקע. אין דער לעקציע ווערט געעפנט א פאלדער וואס איז שוין צוגעגרייט אין Downloads, מיטן נאמען fast API demo products. אויף דער לינקער זייט, אינעם Explorer, זעט מען אלע טעקעס גרייט, און אלעס וועט ווערן געוויזן שריט ביי שריט. אויף דער רעכטער זייט איז דא GitHub Copilot אין agent mode, און עס ווערט צוגעזאגט צו ווייזן ווי אזוי מען ניצט עס.

שאפן און אקטיווירן א virtual environment (38:36:12)

נאכדעם וואס Python איז אינסטאלירט, שאפט מען א) virtual environmentא באזונדערע סביבה) צו פארוואלטן די dependencies, דאס הייסט די פעקלעך וואס דער פראיעקט דארף. אינעם טערמינאל שרייבט מען:

;gt&נאמען;lt& m venv- python

מען קען געבן וועלכן נאמען מען ווייל. דער ערשטער פרואוו מיטן נאמען venv האט געגעבן א טעות, אז עס קען נישט קאפירן א טעקע אין דעם Downloads/fast API פאלדער. דערפאר האט מען עס ווידער געמאכט מיטן נאמען fastapi, און דאס מאל איז די סביבה געשאפן געווארן. אינעם טעקע-חלק עפנט זיך א נייער פאלדער מיטן נאמען, און אינעווייניג זעט מען די Lib פעקלעך, די Scripts און נאך.

דערנאך אקטיווירט מען די סביבה:

- אויף Windows שרייבט מען דעם נאמען פון דער סביבה און דערנאך activate\Scripts , דאס הייסט activate\Scripts\fastapi. דער ערשטער פרואוו האט נישט געארבעט צוליב א פאלשן נאמען. נאכן פארריכטן דעם נאמען ווייזט עס אז די סביבה איז אקטיווירט.
- אויף Mac אדער Linux שרייבט מען bin אנשטאט Scripts, דאס הייסט fastapi/bin/activate.

דערנאך קאנטראלירט מען נאכאמאל אינעם טערמינאל צי FastAPI און Uvicorn זענען אינסטאלירט, מיטן באפעל pip install fastapi uvicorn, וואס באהאנדלט ביידע מיט איין מאל. עס ווייזט אז אלע פעקלעך זענען שוין אינסטאלירט, ווייל מען האט דאס שוין פריער געטון. דער באפעל ווערט דא נאר געלאפן כדי צו קאנטראלירן.

א פשוטער Python ביישפיל: א פונקציע (38:38:18)

מען שליסט צו דעם טערמינאל, גייט צו File און שאפט א נייע טעקע מיטן נאמען hello.py. דער ציל איז צו פארשטיין די יסודות פון Python, באזונדערס ווי אזוי פונקציעס ארבעטן. דער קאוד זעט אויס אזוי:

:def greet(name)

"!{name} ,Hello"return f

"John" = name

און צום סוף: ((print(greet(name.

בשעת'ן שרייבן זעט מען אז VS Code ווייזט אויטאמאטיש א פארשלאג, און דער קורס באמערקט אז דאס העלפט שטארק ביים קאודירן.

די ערקלערונג, שורה נאך שורה:

- def greet(name) דעפינירט א פונקציע מיטן נאמען greet, וואס נעמט איין ארגומענט: name.
- "!{name} ,Hello"return f גיט צוריק א באגריסונג. דאס ניצט Python'ס f-string , א פארמאטירטע string וואס לייגט אריין דעם ווערט פון name אינעם טעקסט.
- "John" = name גיט דעם טעקסט John צום וואריאבל name.
- ((print(greet(name רופט די פונקציע greet מיטן נאמען John און דרוקט אויס די באגריסונג.

מען לויפט דעם קאוד דורכן דריקן "Run Python File", און דער רעזולטאט ערשיינט: !Hello, John אזוי שרייבט מען א פשוטן Python קאוד.

(דא קומט אריין א קורצע רעקלאמע פאר א באצאלטן קורס אין generative AI און מאשין-לערנען, וואס ווערט איבערגעהיפט.)

די ערשטע FastAPI אפליקאציע (38:40:53)

יעצט שאפט מען א פשוטע FastAPI אפליקאציע, און דער קורס זאגט אז עס איז גאר פאסיג פאר אנהייבער. מען שאפט א נייע טעקע, fastapi.py, און מען שרייבט:

from fastapi import FastAPI

דער קורס ווארנט אז דאס איז case sensitive, דאס הייסט עס מאכט א חילוק צווישן קליינע און גרויסע אותיות. ביי fastapi שרייבט מען מיט קליינע אותיות, און דער קלאס FastAPI ווערט געשריבן מיט "API" אין גרויסע אותיות. מען דארף שטארק אכטונג געבן אויף די אותיות. אזוי אימפארטירט מען דעם FastAPI קלאס, וואס איז דער ארץ פון דער אפליקאציע.

דערנאך לייגט מען א באמערקונג (comment): "step: create an instance of FastAPI", און מען שרייבט app = FastAPI(). GitHub Copilot שלאגט דעם קאוד שוין פאר. וועז מען שטעלט דעם מויז דערויף, ערשיינט די אפציע "accept: Tab", און וועז מען דריקט Tab, איז דער קאוד גרייט. עס ווערט באמערקט אז די אייגנשאפטן מאכן דאס קאודירן פיל גרינגער.

דערנאך שרייבט מען א comment: "define a simple route", און דער קאוד איז:

@app.get("/")

def read_root():

return {"message": "Hello, FastAPI"}

די ערקלערונג:

- from fastapi import FastAPI אימפארטירט דעם הויפט-קלאס פון דער אפליקאציע.
- app = FastAPI() שאפט אן instance פונעם FastAPI קלאס. דער instance וועט באהאנדלען אלע HTTP בקשות וואס קומען אריין.
- @app.get("/") איז א decorator וואס זאגט פאר FastAPI זיך צוצוהערן צו GET בקשות אויפן הויפט-אדרעס (root URL), דאס הייסט "/".
- def read_root() איז די פונקציע וואס ווערט געפופן וען א GET בקשה ווערט געמאכט צום root URL.
- צום סוף גיט די פונקציע צוריק א JSON ענטפער מיט דער מעלדונג "Hello, FastAPI".

לויפן די אפליקאציע מיט Uvicorn (38:43:28)

כדי צו לויפן די אפליקאציע, איז נישט גענוג צו דריקן "Run". מען דארף געבן א באפעל אינעם טערמינאל:

uvicorn main:app --reload

אין דער לעקציע הייבט מען אן צו שרייבן דעם באפעל מיט uvicorn און reload.

FastAPI: Swagger, CRUD און Pydantic (38:43:58)

דער לאקאלער סערווער און זיין אדרעס (38:43:58)

נאכן אנלויפן דעם באפעל ווייזט זיך אן אויספיר אין טערמינאל: uvicorn קוקט נאך ענדערונגען אין די דירעקטאריעס, און דער סערווער לויפט אויף http://127.0.0.1:8000. דער קורס באטאנט אז דאס איז זייער וויכטיג, און מען קען גלייך דרוקן אויף דעם לינק כדי צו עפענען די וועבזייטל. ביים פרובירן ווייזט זיך א פעלער אז דער בלאט קען דערווייל נישט געליינט ווערן, אבער פריער ווערט ערקלערט וואס דער אדרעס מיינט.

דער אדרעס איז דער לאקאלער סערווער-אדרעס פאר א FastAPI אפליקאציע וואס לויפט אויף אייער אייגענעם קאמפיוטער. וועז מען לאזט לויפן א FastAPI אפליקאציע מיט uvicorn, הייבט זיך אן דער סערווער

אויפן אייגענעם קאמפיוטער. 127.0.0.1, אויך באקאנט אלס localhost, איז דער IP-אדרעס וואס ווייזט אויפן אייגענעם קאמפיוטער. 8000 איז דער פארט-נומער ווו די FastAPI אפליקאציע לויפט, און עס איז דער סטאנדארט-פארט פון uvicorn. מען קען אים טוישן אויב נויטיג, למשל דורך צולייגן צום באפעל uvicorn main:app --reload א --port מיט אן אנדער נומער.

Swagger UI מיט דער הילף פון GitHub Copilot (38:45:01)

ווען די FastAPI אפליקאציע לויפט, קען מען זי אויספרובירן אינטעראקטיוו דורך Swagger UI, וואס ווערט אויטאמאטיש געשאפן. FastAPI גיט אזא אינטעראקטיוועט דאקומענטאציע וואס ווייזט אלע API-וועגן (routes), און מען קען זיי פרובירן גלייך פונעם בלעטערער. אזוי איז זייער גרינג נאכצוקוקן צי אלעס ארבעט ווי געדארפט. כדי צו זען Swagger UI מוז די אפליקאציע לויפן.

אין דער לעקציע ווערט געוויזן ווי מען נעמט צו הילף GitHub Copilot. מען שרייבט צום Copilot-אגענט א פראמפט ער זאל לאזן לויפן Swagger מיט דעם פארט פאר דער FastAPI אפליקאציע. מען זעט ווי דער אגענט ארבעט שריט ביי שריט:

- ער זוכט אינעם workspace נאך FastAPI אפליקאציע-דעפיניציעס און נאך דעם פונקט פון וואו דער סערוואר הייבט זיך אן.
- ער עפנט דעם פייל fastapi.py, דעם קאוד וואס איז שוין פריער געשריבן געווארן, ליינט אים און שטעלט צונויף א פלאן פון אויפגאבן.
- ער שרייבט אז ער וועט אנהייבן דעם uvicorn סערוואר פונעם virtual environment פונעם פראיעקט, און אז Swagger וועט זיך ווייזן ביים אדרעס /docs.
- ער שאפט א קורצע to-do רשימה און לייגט פאר א באפעל מיט --reload און --host.

דעם באפעל קען מען אליין לאזן לויפן אינעם טערמינאל, אדער לאזן דעם agent mode עס טון. מען וועלט Allow, און דער אגענט לאזט לויפן דעם uvicorn סערוואר. אלעס וואס דער agent mode טוט קען מען נאכקוקן אינעם טערמינאל-אפטייל פון VS Code. דערנאך מעלדט דער אגענט אז דער סערוואר איז געלאפן, דערהיינטיקט די to-do רשימה און ערקלערט ווי צו עפענען Swagger. ער שרייבט אז די FastAPI אפליקאציע איז אנגעהויבן געווארן פונעם virtual env, גיט דעם לינק צו Swagger UI, און גיט אויך קורצע הערות און ניצליכע באפעלן: ווי דער סערוואר הייבט זיך אן, די דירעקטאריע פונעם פייל, אז מען קען אפשטעלן דעם סערוואר מיט Ctrl+C אינעם טערמינאל ווו ער לויפט, און אן אנדער אופן אנצוהייבן פונעם הויפט-פאלדער פונעם פראיעקט, דורך python -m uvicorn fastapi:app --reload מיטן פארט.

פראבלעמען מיטן פארבינדן, און וואס דער אגענט טוט (38:47:35)

ביים עפענען דעם לינק וועבסייט דער בלעטערער "the site can't be reached". מען גייט צוריק צו Copilot אין VS Code און שרייבט אז מען קען נישט עפענען דעם לינק, עס שטייט "page not found". דער אגענט ענטפערט אז ער וועט אנהייבן דעם סערוואר פון דאס ניי כדי צו כאפן די לאגס פונעם אנהייב. מען דרוקט Allow, און אינעם טערמינאל זעט מען ווי ער ברענגט אריין עטליכע ביבליאטעקן, הייבט אן א reloader פראצעס, און uvicorn לויפט אויף דעם פארט. דערנאך שלאגט ער פאר צו שאפן א נייעם פייל app.py כדי אויסצומיידן א קאנפליקט פון נעמען, און דערנאך uvicorn נאכאמאל אנצוהייבן. מען דרוקט נאכאמאל Allow, דער באפעל לויפט, אבער דער בלאט קען נאך אלץ נישט דערגרייכט ווערן. דער קורס מיינט אז ס'קען זיין א פראבלעם מיט דער פארבינדונג.

אויספרובירן די וועגן (routes) (38:49:08)

דער קומענדיגער שריט איז אויספרובירן די וועגן. אין Swagger UI זעט מען אלע API -וועגן אויסגערעכנט, ווי GET, POST, PUT, DELETE, און יעדן איינעם קען מען פרובירן דורך אריינשרייבן דאטא, למשל א נייע פראדוקט, און דרוקן Execute. מען בעט דעם agent mode עס צו טון. דער אגענט שרייבט אז ער וועט אנהייבן דעם סערווער אין הינטערגרונט, אינעם package פאלדער, און שיקן עטליכע HTTP בקשות צו פרובירן די וועגן פון רשימה און פון שאפן א פראדוקט, דאס הייסט GET און POST פרובעס. ער וואלט אז וועב-אינהאלט קען זיין שעדליך, און מען דרוקט Allow. ער קוקט קודם נאך די פארבינדונג צום סערווער, לאזט אים לויפן, און פירט אריבער stdout און stderr צו פיילן, וואלט א ביסל און ליינט זיי.

דאס מאל וואלפט עס א פעלער. דער אגענט ערקלערט: אויב מען זעט "page not found" ביים /docs, וואס קען פאסירן אויך ביי אייך, דארף מען זיך פארזיכערן אז uvicorn לויפט אויפן פארט 8000. אויב נישט, הייבט מען אים אן מיט די באפעלן פון פריער. מען דארף זיך אויך פארזיכערן אז קיין firewall אדער אנדערער פראצעס בלאקירט נישט דעם פארט. אויב עס איז דא א קאנפליקט פון נעמען ביים אריינברענגען, צוליב א פייל-נאמען אינעם הויפט-פאלדער, שרייבט ער אז ער האט שוין פריער צוגעלייגט א זיכערן app.py, אבער דער package האט אין זיך דעם געמיינטן API.

אלס קומענדיגע שריט לייגט ער פאר אפצושטעלן דעם Python REPL אינעם טערמינאל און דאן אנהייבן אויטאמאטישע טעסטס און מעלדן די רעזולטאטן מיט ביישפיל-ענטפערס. מען בעט אים אפצושטעלן די דאפלטע Python פראצעסן אינעם טערמינאל, אנהייבן פון דאס ניי מיט די אויטאמאטישע טעסטס און מעלדן די רעזולטאטן, אז דער פעלער זאל זיך מער נישט ווייזן. דאס קען נעמען א שטיק צייט.

זען דעם ענטפער (38:51:10)

נאכן אויספירן א בקשה ווייזט Swagger UI גלייך דעם ענטפער, אזוי אז מען זעט צי דער API גיט צוריק דעם ערוואַרטעטן רעזולטאט. למשל, מען קען פרובירן דעם POST /products ענדפוינט דורך אויספילן די נויטיגע פרטים פון דער פראדוקט: name, description, price, quantity. וועז מען דרוקט Execute, ווייזט זיך דער רעזולטאט וואס באשטעטיגט אז די פראדוקט איז מצליח צוגעלייגט געוואָרן.

דריטער שריט: CRUD אפעראציעס פאר פראדוקטן (38:51:41)

דער דריטער שריט איז שאפן CRUD אפעראציעס, דאס הייסט Create, Read, Update, Delete, צו פארוואלטן פראדוקטן אין א FastAPI אפליקאציע. די אפעראציעס ווערן געטון דורך די HTTP מעטאדן POST, GET, PUT און DELETE.

מען עפנט א נייעם פייל מיטן נאמען crud.py און שרייבט:

- from fastapi import FastAPI, און from typing import List, Optional.
- app = FastAPI().
- א הערה "in memory storage for the products", און products = [], א ליידיגע רשימה. דער קורס ערקלערט אז דאס איז א שפייכלער אינעם זכרון, און שפעטער וועט עס פארביטן ווערן מיט א דאטאבאזע.

שאפן א פראדוקט: מען שרייבט @app.post("products/") און א פונקציע create_product מיט די פאראמעטערס name (string), description (string), price (float) און quantity (integer). די פונקציע שטעלט צונויף די פראדוקט, לייגט זי צו מיט products.append(product), און גיט צוריק די מעלדונג

"Product added successfully" צוזאמען מיט דער פראדוקט. עס ווערט ערקלערט: די POST מעטאד דערלויבט צולייגן א נייע פראדוקט. די פונקציע נעמט אן די פרטים דורך HTTP POST און שפייכלט די פראדוקט אין דער רשימה אינעם זכרון. דער app.post@ decorator באשטימט דעם וועג וואס ווערט זיך צו צו POST בקשות ביי /products, און די פונקציע נעמט אן name, description, price און quantity אלס איינגאנג כדי צו באשטימען א פראדוקט.

דערהיינטיקן א פראדוקט: דער ענדפוינט דערלויבט צו דערהיינטיקן א פראדוקט וואס איז שוין פאראן. מען גיט אן דעם product ID און די פעלדער (name, price, description) וואס מען וויל טוישן, און עס קומט צוריק א מעלדונג.

אויסמעקן א פראדוקט: מען לייגט א הערה "Deleting" און שרייבט:

@app.delete("/products/{product_id}")

def delete_product(product_id: int):

אינעווייניג: אויב product_id < len(products), טוט מען products.pop(product_id) און גיט צוריק "Product deleted successfully". אויב נישט, גיט מען צוריק "Product not found". דער decorator @app.delete הערט זיך צו צו אלע DELETE בקשות ביי /products/{product_id}. אויב דער ID איז פאראן אין דער רשימה, ווערט די פראדוקט אראפגענומען מיט pop, און דער API ענטפערט מיט א הצלחה-מעלדונג. אויב די פראדוקט איז נישט געפונען, קומט א פעלער-מעלדונג.

פעלערן און וואלידאציע מיט Pydantic (38:55:26)

נאכדעם וואס מען האט געזען ווי די CRUD פונקציעס שאפן, דערהיינטיקן און מעקן אויס א פראדוקט, ווערט געוויזן ווי מען באהאנדלט פעלערן און וואלידאציע מיט Pydantic. Pydantic העלפט באשטימען דאטא-מאדעלן און נאכקוקן די דאטא וואס ווערט געשיקט צום API. FastAPI קוקט אויטאמאטיש נאך צי די דאטא שטימט מיטן מאדעל, און ווארפט א פעלער אויב נישט.

מען שאפט נאך א פייל, pydantic.py, און שרייבט:

- from pydantic import BaseModel, from fastapi import FastAPI, from typing import List.
- app = FastAPI().
- products = [], און "in memory storage for products" א הערה.
- name (str), description "Pydantic model for the product", און א מאדעל מיט די פעלדער (str), price (float) און quantity (int). מען נעמט אן דעם פארשלאג פונעם עדיטאר מיט Tab.

שאפן א פראדוקט מיט Pydantic: מען לייגט א הערה פאר creating a product, שרייבט א POST וועג ביי / products און א פונקציע create_product וואס נעמט אן א product. זי טוט products.append(product) און גיט צוריק די מעלדונג "Product added successfully" מיט דער פראדוקט. דערנאך לייגט מען צו א נייע הערה פארן קומענדיגן טייל: באקומען אלע פראדוקטן.

Pydantic מאדעל פאר פראדוקטן און וואלידאציע (38:57:32)

דער קורס פירט ווייטער מיטן FastAPI פראיעקט פון א אינוואנטאר-אפליקאציע. מען לייגט צו א רוט וואס ברענגט אלע פראדוקטן: א דעקאראטאר פון סארט app.get מיטן פאד products , וואו מען שטעלט response_model אלס א ליסטע (List) פון Product. דערנאך שרייבט מען א פונקציע get_products וואס גיט צוריק די ליסטע פון פראדוקטן.

דערנאך ווערט ערקלערט דער קאוד. עס איז דא א Pydantic מאדעל מיטן נאמען Product , וואס ירשנט פון BaseModel. די קלאס באשטימט די סכעמע (schema) פון א פראדוקט: זי פארזיכערט אז די פעלדער name, description, price און quantity ווערן געשיקט, און אז זיי האבן די ריכטיגע סארט דאטא: string פאר נאמען און באשרייבונג, float פאר פרייז, און integer פאר קוואנטיטעט.

מען זעט ווי דער מאדעל ווערט גענוצט אינעם API דורכן POST רוט אויף products. די דאטא קומט אריין אלס א Product אביעקט, און FastAPI קאנטראלירט אויטאמאטיש די דאטא קעגן דעם מאדעל. אויב די דאטא איז נישט גילטיג, למשל מען שיקט א string אנשטאט א נומער פאר price, גיט FastAPI צוריק א 400 Bad Request טעות, מיט אלע פרטים פון וואס איז דורכגעפאלן ביי דער וואלידאציע. אזוי ארבעט די טעות-באהאנדלונג.

פארבינדן מיטן פראנט-ענד דורך CORS (38:58:34)

נאכן אויפשטעלן דעם FastAPI באק-ענד, איז די צייט עס צו פארבינדן מיט א פראנט-ענד ווי React אדער Vue.js. דערצו דארף מען זיך פארזיכערן אז CORS ווערט ריכטיג באהאנדלט, כדי דער פראנט-ענד און דער באק-ענד זאלן קענען רעדן איינער מיטן אנדערן אן שטערונגען. וואס CORS מיינט איז שוין פריער ערקלערט געווארן אינעם קורס, און מען קען צוריקגיין און עס איבערהערן.

FastAPI ערלויבט גרינג אריינלייגן CORS דורך CORSMiddleware , וואו מען שטעלט איין און מען ערלויבט cross-origin בקשות צווישן פראנט-ענד און באק-ענד. דאס פארזיכערט א גלאטע קאמוניקאציע צווישן די פארשידענע טיילן פון דער אפליקאציע.

מען שאפט א נייע טעקע cors.py. די שריט:

- מען אימפארטירט FastAPI פון fastapi, און CORSMiddleware פון fastapi.middleware.cors.
- מען שאפט app = FastAPI().
- מען שטעלט א ליסטע origins, וואו מען גיט אריין דעם localhost מיטן פארט 8000.
- מען רופט app.add_middleware מיט CORSMiddleware, וואו allow_origins איז די origins ליסטע, allow_credentials איז True, און מען ערלויבט אויך די methods און די headers.

דערנאך ווערט געוויזן ווי אזוי מען רופט דעם FastAPI באק-ענד פון א React פראנט-ענד, מיט א ביישפיל פון JavaScript: מען רופט fetch אויפן localhost פארט מיטן פאד products, דערנאך .then וואו מען פארוואנדלט דעם ענטפער מיט response.json(), און נאך א .then וואו מען דרוקט אויס די דאטא מיט console.log. די פשוטע fetch בקשה רופט דעם get_products רוט אויפן באק-ענד און שרייבט אויס דעם ענטפער אין דער browser console.

פארבינדן צו א PostgreSQL דאטאבאזע (39:00:39)

ווייטער ווערט געוויזן ווי אזוי מען פארבינדט זיך צו א דאטאבאזע. דער ביישפיל איז PostgreSQL, אבער מען קען נוצן יעדע אנדערע דאטאבאזע. מען שאפט נאך א טעקע database.py. ווייל מען האט שוין די CRUD אפעראציעס, איז די צייט זיך צו פארבינדן מיט PostgreSQL פאר א שטענדיגע (persistent) אפהיטונג פון דאטא.

אינעם טערמינאל גיט מען אריין דעם באפעל:

pip install psycopg2

דאס אינסטאלירט דעם psycopg2 פעקל פאר PostgreSQL. מען וואלט עטליכע סעקונדעס ביז עס ווערט אינסטאלירט. עס קומט ארויף א מעלדונג אז א נייע ווערסיע פון pip איז פאראן, אבער מען דארף איצט נישט דערהיינטיגן; די פעקלעך זענען שוין אינסטאלירט.

אין database.py שרייבט מען import psycopg2, און דערנאך, מיט א קאמענטאר "connect to PostgreSQL database", שאפט מען א פארבינדונג conn דורך psycopg2.connect מיט די פאלגנדע ווערטן:

- dbname איז inventory_db
- user איז user
- password איז password
- host איז localhost
- port איז 5432

דערנאך שאפט מען cursor = conn.cursor().

שאפן א טאבעלע און פארמאכן די פארבינדונג (39:02:40)

צו שאפן א טאבעלע גיט מען אריין א SQL באפעל CREATE TABLE IF NOT EXISTS products, מיט די קאלומען: id אלס SERIAL PRIMARY KEY, name אלס VARCHAR, description אלס TEXT, price אלס FLOAT, און quantity אלס integer. דערנאך רופט מען conn.commit(), cursor.close() און conn.close().

דער קורס ערקלערט יעדן טייל:

- די פונקציע psycopg2.connect שטעלט אויף די פארבינדונג צו דער PostgreSQL דאטאבאזע מיט די געגעבענע פרטים: נאמען פון דער דאטאבאזע, באניצער, פאסווארט, host און פארט.
- די פונקציע cursor.execute פירט אויס דעם SQL באפעל וואס שאפט די products טאבעלע אויב זי עקזיסטירט נאך נישט.
- די פונקציע conn.commit היט אפ אלע ענדערונגען וואס מען האט געמאכט אין דער דאטאבאזע, ווי למשל שאפן די טאבעלע.
- cursor.close און conn.close פארמאכן דעם cursor און די פארבינדונג צו דער דאטאבאזע.

אויספרובירן דעם API און ניצן GitHub Copilot (39:03:48)

דער קומענדיגער שריט איז אויספרובירן דעם API. דערצו קען מען נוצן Swagger UI, וואס FastAPI שטעלט צו אויטאמאטיש. מען לאוט לויפן די אפליקאציע מיט uvicorn. אינעם טערמינאל גיט מען אריין:

reload-- uvicorn main:app

דאס הייבט אן דעם FastAPI סערווער. אין דער לעקציע קומט ארויף א טעות: "error loading ASGI", ער קען נישט אימפארטירן דעם מאדול main. וועז עס לויפט, זעט מען די רוטס POST, GET, PUT און DELETE פאר פארוואלטן די פראדוקטן, און מען קען אויך צוקומען צו א צווייטן אויסזען פון דער דאקומענטאציע.

מען קען אויך נוצן GitHub Copilot דערפאר. מען בעט Copilot צו שאפן א רוט וואס ברענגט א פראדוקט לויט זיין ID. Copilot ענטפערט אז עס וועט צולייגן א GET רוט אין דער FastAPI טעקע פונעם פראיעקט, וואס גיט צוריק א פראדוקט לויט ID דורך א געווישן דאטאבאזע-העלפער, און דערנאך דערהיינטיגן די todo ליסטע. נאכדעם זאגט עס אז עס איז פארטיג, און עס גיט אנווייזונגען ווי אזוי שנעל אויסצופרובירן: מען קען אנהייבן די אינוואנטאר אפליקאציע מיט א באפעל אויב זי לויפט נאך נישט, און עס פאַרשלאגט אליין אנצוהייבן דעם סערווער. מען בעט עס טאקע אנהייבן דעם סערווער.

מען קען זיך אנקוקן די דירעקטאריעס וואס מען האט געשאפן, די virtual environments, די scripts און די FastAPI פונקציעס. פריער איז געווען א טעות, אבער איצט זעט מען אז אלע דאטאבאזעס זענען אריינגענומען און פארבונדן מיט FastAPI, און מען זעט ווי עס איז פארבונדן מיט PostgreSQL. ווייל Copilot נעמט לאנג צו ענדיגן, ווערט געזאגט אז מען קען עס אליין שרייבן און זען ווי עס ארבעט.

סך-הכל פונעם FastAPI טייל (39:05:54)

אין דעם טייל האט מען:

- געשאפן א FastAPI אפליקאציע מיט CRUD אפעראציעס;
- אריינגעלייגט Pydantic פאר וואלידאציע פון דאטא;
- זיך פארבונדן מיט א PostgreSQL דאטאבאזע פאר שטענדיגע אפהיטונג;
- גענוצט GitHub Copilot כדי צו פארשנעלערן די אנטוויקלונג און פארבעסערן די קוואליטעט פונעם קאוד;
- אויספרובירט דעם API דורך FastAPI'ס אינטעראקטיווע דאקומענטאציע.

דער קאוד פאר דעם באק-ענד ווערט צוגעלייגט אלס א GitHub repository אין דער באשרייבונג.

פארוואס Streamlit איז א "game changer" (39:06:25)

דער קורס גייט איבער צו בויען אינטעראקטיווע אפליקאציעס מיט Streamlit, און ערקלערט פארוואס עס איז אזא גרויסע ענדערונג:

- שנעלע אנטוויקלונג: אמאל, צו שאפן א וועב-אפליקאציע געבויט אויף דאטא, האט מען געדארפט א סך צוגרייטונג און לערנען פארשידענע געציג. מיט Streamlit קען מען פארוואנדלען א Python סקריפט אין אן אינטעראקטיווע וועב-אפליקאציע אין עטליכע מינוט. מען דארף נישט קיין פראנט-ענד קענטעניש און נישט לערנען קאמפליצירטע וועב געציג. צי מען שאפט א דעשבארד פאר באווייזן דאטא, העלפט Streamlit עס טון שנעל, און עס איז זייער גרינג אויפצושטעלן.

• פשוטע סינטאקס: דאס איז דער בעסטער טייל. ביי געווענליכע אנטוויקלונג דארף מען זשאנגלירן עטליכע שפראכן ווי HTML, CSS און JavaScript; מיט Streamlit דארף מען נאר Python, וואס מען קען שוין. מען קען צולייגן קנעפלעך, sliders, טשארטס און נאך, מיט נאר עטליכע שורות Python קאוד, אזוי אז מען קען זיך קאנצענטרירן אויף דער דאטא און נישט אויפן קאוד.

• פערפעקט פאר דאטא-וויסנשאפט: Streamlit איז געבויט געווארן מיט דאטא-וויסנשאפט אין זינען. עס פאסט זיך צוזאמען מיט pandas, Matplotlib, Plotly און אנדערע ביבליאטעקן וואס דאטא-וויסנשאפטלער נוצן שוין. מיט איין שורה קאוד קען מען ווייזן די דאטא אין טשארטס אדער טאבעלעס, וואס מאכט גרינג צו טיילן אויספירן און שאפן שיינע אינטעראקטיוו דעשבארדס.

בקיצור, Streamlit לאזט מען זיך קאנצענטרירן אויף דעם וואס איז באמת וויכטיג, די דאטא און די אויספירן, אן פארכאפט ווערן אין די קאמפליקאציעס פון וועב-אנטוויקלונג. עס איז פשוט, אינטואיטיוו, און געמאכט פאר דאטא-וויסנשאפטלער און Python פראגראמירער.

פארגלייך מיט Flask און Dash (39:07:56)

• Flask איז א "micro" וועב-פריימווארק, גוט צו בויען וועב-אפליקאציעס פון אנפאנג. עס איז בייגעוודיג, אבער אפילו פאר א פשוטע אפליקאציע דארף מען אליין באהאנדלען routing, templates און פארוואלטן פארשידענע טיילן. פאר אינטעראקטיוויטעט דארף מען שרייבן א סך אייגענעם קאוד.

• Dash איז א פריימווארק פאר אפליקאציעס געבויט אויף דאטא, גוט פאר אינטעראקטיוו וויזואליזאציעס. אבער Dash איז שווערער צו לערנען ווי Streamlit; מען דארף שרייבן מער boilerplate קאוד צו אויפשטעלן קנעפלעך, sliders און טשארטס.

פארוואס Streamlit שטייט ארויס:

• מען בויט אפליקאציעס מיט עטליכע שורות Python; עס באזארגט אליין דעם UI און דעם סערווער, און מען קאנצענטרירט זיך אויף דאטא-לאגיק און וויזואליזאציעס.

• ווייניגער אפהענגיגקייטן: ביי Flask און Dash דארף מען אפט אינסטאלירן און איינשטעלן JavaScript און פריימווארקס; ביי Streamlit איז אלעס שוין איינגעשטעלט און ארבעט גלייך מיט מינימאלע צוגרייטונג.

• ענגע פארבינדונג מיט pandas, Matplotlib און Plotly, וואס מאכט עס די באליבטע ברירה פאר דאטא-וויסנשאפטלער וואס דארפן שנעל ווייזן זייער דאטא און אויספירן.

ווירקליכע באנוצן פון Streamlit (39:09:28)

• דאטא-דעשבארדס: דאטא-וויסנשאפטלער און אנאליטיקער שאפן דעשבארדס צו זען וויכטיגע מעסטונגען. מיט Streamlit איז עס אזוי פשוט ווי שרייבן Python: מען לאדנט ארויף דאטא, ווייזט עס אין טאבעלעס, און לייגט צו אינטעראקטיוו טשארטס, אלעס אין איין אפליקאציע. צי עס איז פארקויף-דאטא, פינאנציעלע רעזולטאטן אדער מארקעטינג מעסטונגען, העלפט Streamlit פארוואנדלען רויע דאטא אין באדייטפולע בילדער.

• מאשין-לערנען אפליקאציעס: אויב מען האט געבויט א מאדעל אין scikit-learn אדער TensorFlow, קען מען מיט Streamlit שאפן אן אינטעראקטיוו אפליקאציע ווו באניצער גיבן אריין דאטא און זען פאָרויסזאגונגען. למשל, אן אפליקאציע ווו מען לאדנט ארויף בילדער, און Streamlit ווייזט די פאָרויסזאגונגען פונעם מאדעל אין רעאלע צייט.

- פראטאטייפינג און פארשונג: פארשער וואס ווילן שנעל פרובירן אן אידעע אדער טיילן רעזולטאטן קענען שאפן אינטעראקטיווע וויזואליזאציעס און באקומען גלייך רעאקציעס פון אנדערע, צי עס איז ווייזן אנאליז-רעזולטאטן אדער עקספערימענטירן מיט מאשין-לערנען מאדעלן; מען בויט אן אפליקאציע אין קורצער צייט און טיילט עס מיט דער וועלט.
- דאטא-אויספארשונג און אנאליז: מיט Streamlit קען מען שנעל לאדענען דאטא-סעטס, פילטערן די דאטא, און אויספרובירן פארשידענע וויזואליזאציעס, וואס איז זייער נוצלעך פאר דאטא-ווישנשאפטלער.

Streamlit: פון אינסטאלירן ביז דאטא-אויספארשונג (39:10:58)

פארוואס Streamlit, און אינסטאלירן Python און Streamlit (39:10:58)

דער קורס ענדיגט די הקדמה וועגן ווי Streamlit ווערט געניצט אין פאקטישע פראיעקטן: מען קען דורכאויס אינטעראקטיוו אויספארשן גרויסע דאטא-סעטן און טיילן די רעזולטאטן מיט די מיטארבעטער אין דער מאנשאפט אדער מיט די אינטערעסירטע פארטייען. Streamlit איז געבויט כדי מען זאל זיך קענען קאנצענטרירן אויף דער עיקר ארבעט און שנעלער ארויסגעבן די רעזולטאטן, צי עס איז בויען דעשבארדס, ארויסשטעלן מאשין-לערנען מאדעלן, אדער אויספארשן דאטא-סעטן. פון דא און ווייטער ווייזט דער קורס ווי אזוי צו בויען אן אייגענע Streamlit אפליקאציע.

דער ערשטער שריט איז אינסטאלירן די נויטיגע זאכן כדי מען זאל קענען לויפן Streamlit אויפן קאמפיוטער:

- Python – רוב מענטשן האבן עס שוין. וועט עס האט עס נישט, גייט אריין אויף python.org און לאדנט עס אראפ.
- Streamlit – מען עפנט דעם טערמינאל אדער Command Prompt און מען שרייבט: pip install streamlit. דאס אינסטאלירט Streamlit, און מען איז גרייט צו אנהייבן.

נאכן אינסטאלירן ווערט רעקאמענדירט צו שאפן א "ווירטועלע סביבה" (virtual environment). דאס העלפט האלטן די אפהענגיגקייטן (dependencies) פונעם פראיעקט באזונדער און ארגאניזירט. אין טערמינאל שרייבט מען:

m venv myenv- python

דערנאך אקטיווירט מען די ווירטועלע סביבה; אויף Windows נוצט מען דעם צוגעפאסטן באפעל (activate\Scripts\myenv). מען ווייסט אז זי איז אקטיוו ווייל דער טערמינאל ווייזט דעם נאמען פון דער סביבה פארנט אינעם שורה. אינערהאלב דער סביבה אינסטאלירט מען Streamlit מיטן זעלבן באפעל ווי פריער.

דער ערשטער סקריפט: Hello World (39:12:16)

יעצט לאזט מען לויפן די ערשטע Streamlit אפליקאציע. קודם דארף מען גיין צום פאלדער וואו דער טעקע ליגט. אין דעם ביישפיל ליגט א טעקע מיטן נאמען app.py אויפן) Desktopזי איז שוין פריער געשאפן געווארן). די שריט:

- אין טערמינאל שרייבט מען cd desktop און דריקט) Enter אין דער לעקציע איז מען שוין געווען אינעם Desktop פאלדער).

• כדי צו באשטעטיגן אז די טעקע איז דארט, שרייבט מען dir. אויב מען זעט app.py אין דער ליסטע, מיינט עס אז מען געפינט זיך אינעם ריכטיגן פאלדער.

אינעווייניג אין app.py שרייבט מען בלויז צוויי שורות קאוד:

```
import streamlit as st

st.write("Hello World")
```

די ערשטע שורה ברענגט אריין די Streamlit ביבליאטעק; די צווייטע זאגט פאר Streamlit צו ווייזן דעם טעקסט אויפן עקראן. דאס איז אלעס.

כדי צו לויפן די אפליקאציע גייט מען צוריק צום טערמינאל און שרייבט streamlit run app.py (app.py איז דער נאמען פון דער טעקע), און דריקט Enter. Streamlit הייבט אן א לאקאלן אנטוויקלונגס-סערווער. ווען מען לויפט Streamlit דאס ערשטע מאל, קען עס פרעגן פאר אן אימעיל; מען קען פשוט דריקן Enter צו איבערשפרינגען. נאך עטלעכע סעקונדעס באווייזט זיך א "Local URL". מען קליקט דערויף אדער מען קאפירט און לייגט עס אריין אינעם בראוזער – און דארט זעט מען "Hello World" אויפן עקראן. דאס מיינט אז די אפליקאציע לויפט מיט הצלחה.

א וויכטיגער טיפ: מען דארף נישט אפשטעלן דעם סערווער יעדעס מאל מען מאכט ענדערונגען. Streamlit לאדנט אויטאמאטיש איבער די אפליקאציע ווען מען היט אפ (app.py (save – מען רעדאגירט פשוט די טעקע.

ווייזן טעקסט: st.write און markdown (39:14:08)

נאכדעם וואס Streamlit איז אויפגעשטעלט, גייט מען איבער צו די יסודות פון בויען אן אינטעראקטיווע אפליקאציע. Streamlit מאכט עס גרינג צו ווייזן אינהאלט ווי טעקסט און בילדער, און צולייגן אינטעראקטיווע עלעמענטן אן פיל קאוד.

מען הייבט אן מיט ווייזן טעקסט, וואס איז איינס פון די ערשטע זאכן וואס מען דארף אין אן אפליקאציע. די פשוטסטע וועג צו ווייזן סיי וואס אויפן בלאט איז די פונקציע st.write . אין דעם ביישפיל שרייבט מען: st.write("This is a simple text using write").

מען קען אויך פארמאטירן דעם טעקסט מיט markdown, וואס גיט מער קאנטראל איבער ווי דער אינהאלט קוקט אויס. למשל, ווען מען וויל צולייגן קעפלעך (headers) און אונטער-קעפלעך כדי צו געבן סטרוקטור, נוצט מען די פונקציע st.markdown. אין דעם ביישפיל ווערט געשריבן איין שורה "This is a large heading" און א צווייטע "This is a smaller heading". ווען מען לויפט דעם קאוד, ווייזט די ערשטע שורה א גרויסן קעפל און די צווייטע א קלענערן אונטער-קעפל. מיט דער טעכניק קען מען צעטיילן די אפליקאציע אויף אפטיילונגען און ארדנטלעך ארגאניזירן דעם אינהאלט.

אינטעראקטיווע עלעמענטן: slider און checkbox (39:15:13)

Streamlit מאכט עס זייער גרינג צולייגן "ווידזשעטס" ווי סליידערס, קנעפלעך און checkboxes.

א slider לאזט דעם באניצער אויסקלייבן א נומער פון א באשטימטן ראנזש. דער ביישפיל לאזט אויסקלייבן א נומער צווישן 1 און 100: מען שאפט א פאריאבל number מיט דער פונקציע st.slider, מיטן טעקסט "Pick a number", די מינימום-ווערט 1 און די מאקסימום-ווערט 100. צום סוף ווערט געדרוקט "You picked" און דערנאך דער נומער. ווען מען לויפט עס באווייזט זיך דער סליידער, און ווען דער באניצער רירט אים, ווערט

דער אויסגעקליבענער נומער געוויזן אונטן. אין דער דעמא ווערט אויסגעקליבן 24, און עס שטייט "You picked 24".

אויף אן ענלעכן אופן קען מען נוצן א checkbox צו באקומען אינפוט פונעם באניצער. דער ביישפיל איז א קעסטל פאר מסכים זיין צו די באדינגונגען (terms and conditions): מען נוצט די פונקציע st.checkbox, און אויב דאס קעסטל איז אנגעצייכנט, שרייבט מען "You agreed". דאס קעסטל באווייזט זיך, און ווען דער באניצער צייכנט עס אן, ווערט געוויזן די מעלדונג "You agreed".

מעדיע: בילדער און ווידעאס (39:16:28)

Streamlit לאזט אויך צולייגן מעדיע ווי בילדער, ווידעאס און אודיא. כדי צו ווייזן א בילד נוצט מען די פונקציע st.image און מען גיט איר דעם וועג (path) צום בילד. אין דער לעקציע איז דער path צו א בילד וואס איז פריער אראפגעלאדנט געווארן אויפן קאמפיוטער שוין אריינגעלייגט, און דאס בילד ווערט געוויזן אויפן בלאט. מען קען אויך קאנטראלירן די בריים און הייך פונעם בילד דורך געבן די אפציאנעלע ארגומענטן width און height.

פאר א ווידעא נוצט מען די פונקציע st.video מיטן (pathאדער URL) צום ווידעא. וועז מען לויפט עס, ווערט דער ווידעא פונעם געגעבענעם URL אריינגעבויט גלייך אין דער אפליקאציע. מען קען אויך נוצן א לאקאלע ווידעא-טעקע, וועז מען וויל.

דורך צוזאמשטעלן די אלע מעגלעכקייטן קען מען לייכט בויען אינטעראקטיווע באניצער-אינטערפייסעס מיט Streamlit – צי מען ווייזט דאטא, שאפט פארמען, אדער בויט אריין מולטימעדיע – אלעס מיט נאר עטלעכע שורות קאוד.

דאטא-אויספארשונג: אריפלאדן CSV און Excel טעקעס (39:17:12)

Streamlit איז זייער שטארק ביי באהאנדלען און ווייזן דאטא. אין דעם טייל ווערט געוויזן ווי צו אריפלאדן און ווייזן טעקעס ווי CSV אדער Excel, ווי צו ווייזן די דאטא אין א טאבעלע מיט אינטעראקטיווע פילטערס, און ווי צו צולייגן סטאטיסטישע סיכומים און מעטריקס כדי די דאטא זאל זיין מער פארשטענדלעך.

דער ערשטער שריט איז דערלויבן דעם באניצער אריפצולאדן א CSV אדער Excel טעקע. Streamlit האט א איינגעבויטע פונקציע st.file_uploader, וואס לאזט דעם באניצער אויסקלייבן א טעקע פון זיין קאמפיוטער. וועז די טעקע איז אריפגעלאדן, לייענט מען זי מיט pandas כדי צו ווייזן דעם אינהאלט.

דער קאוד ארבעט אזוי:

- מען ברענגט אריין (import) די ביבליאטעקן.
- st.file_uploader גיט די מעגלעכקייט אויסצוקלייבן א טעקע, אדער CSV אדער Excel.
- א if באדינגונג: אויב די אריפגעלאדענע טעקע איז נישט None, ווייזט מען וועלכע טעקע איז אריפגעלאדן געווארן.
- לויט דעם טיפ טעקע נוצט מען pd.read_csv פאר CSV טעקעס און pd.read_excel פאר Excel טעקעס.
- צום סוף ווערט דער data frame געוויזן; די פונקציע st.write ווייזט דעם אינהאלט פון דער טעקע ווי א טאבעלע.

וועזן טאבעלעס מיט pandas (39:18:50)

וועזן די טעקע איז ארויפגעלאדן, איז אפט נוצלעך צו ווייזן די דאטא אין א ריינעם, לייענבארן פארמאט. Streamlit טוט דאס אויטאמאטיש מיט st.write, ווי מען האט פריער געזען. אבער מען קען אויך מער צופאסן ווי די טאבעלע ווערט געווייזן. אין דעם ביישפיל ווערט גענוצט די פונקציע st.dataframe. אין דער דעמא קליקט מען אויף "Browse files" און מען לאדנט ארויף א CSV טעקע. דאס ווייזט דעם data frame ווי אן אינטעראקטיוו טאבעלע, וואו דער באניצער קען גלייך סארטירן און פילטרירן די קאלאנעס – דאס איז דערפאר ווייל מען האט גענוצט די dataframe פונקציע.

אינטעראקטיוו פילטרירן און זוכן (39:19:39)

מען קען מאכן די דאטא נאך מער אינטעראקטיוו דורך לאזן דעם באניצער פילטרירן דעם דאטא-סעט לויט א באדינגונג – למשל ווייזן נאר די שורות וואו א געוויסע קאלאנע שטימט מיט א געגעבענער ווערט. דאס טוט מען מיטן select box ווידזשעט (st.selectbox). די שריט אינעם קאוד:

- מען שאפט א select box.
- מען פילטרירט די דאטא לויט דעם נאמען אדער סיי וועלכע ווערט וואס איז אויסגעקליבן.
- צום סוף ווייזט מען בלויז די געפילטערטע דאטא.

וועזן מען לויפט עס, קען מען אויסקלייבן א ווערט צו פילטרירן; אין דער דעמא קלייבט מען אויס לויט קאליר (color). דער באניצער קלייבט א ווערט פון דער אראפפאלנדיקער ליסטע, און עס ווערן געווייזן נאר די שורות וואו די קאלאנע שטימט מיט דער אויסגעקליבענער ווערט. דאס איז א גוטע וועג צו מאכן די דאטא מער אינטעראקטיוו און באקוועם פארן באניצער.

מעטריקס און סטאטיסטישע סיכומים (39:20:30)

צום לעצט איז אפט נוצלעך צו ווייזן סיכום-סטאטיסטיקס אדער מעטריקס וואס געבן דעם באניצער א שנעלן איבערבליק פון דער דאטא. Streamlit האט איינגעבויטע שטיצע דערפאר: מיט st.metric קען מען ווייזן זאכן ווי די גאנצע צאל שורות, דורכשניטן אדער סומעס. דערצו קען מען ווייזן א באזיס סטאטיסטישן סיכום מיט describe.

דער קאוד האט צוויי טיילן: ערשט ווייזט מען די באזיס סטאטיסטיקס, און דערנאך ווייזט מען ספעציפישע מעטריקס ווי די צאל שורות. די describe מעטאד גיט א סיכום פון סטאטיסטיקס ווי דער דורכשניט (mean), די standard deviation און די פראצענטילן פאר יעדע נומעריש קאלאנע. מיט st.metric קען מען ארויסהייבן וויכטיגע ווערטן, ווי די גאנצע צאל שורות אינעם דאטא-סעט אדער אנדערע שליסל-מעטריקס. אין דער דעמא ווערט געווייזן אז די צאל שורות איז 16.

דורך צוזאמשטעלן די אלע מעגלעכקייטן מאכט Streamlit עס זייער גרינג אויסצופארשן דאטא: ארויפלאדן טעקעס, ווייזן טאבעלעס, צולייגן פילטערס און ווייזן סטאטיסטישע סיכומים – אלעס מיט עטלעכע שורות קאוד. אזוי קען מען בויען אינטעראקטיוו אפליקאציעס, געטריבן פון דאטא, וואס לאזן באניצער אויספארשן און אנאליזירן דאטא-סעטן אן מי.

דאטא-וויזואליזאציע: Matplotlib און Plotly (39:21:42)

איינס פון די בעסטע זאכן ביי Streamlit איז די מעגלעכקייט צו וויזואליזירן דאטא. דער קורס הייבט אן מיט Matplotlib צו צייכענען א פשוטן טשארט; וועזן מען לויפט דעם קאוד, זעט מען ווי די דאטא ווערט געווייזן ווי א

טשארט. פאר אינטעראקטיוו טשארטס ארבעט Streamlit אויסגעצייכנט מיט Plotly, און די לעקציע הייבט אן שרייבן דעם קאוד צו שאפן א Plotly טשארט.

א קליינע Streamlit דעשבאָרד און ארויסשטעלן אויף דער קלאוד (39:22:13)

אינטעראקטיוו טשארטס און מאפעס (39:22:13)

דער קורס ענדיגט דעם טייל וועגן ווייזן טשארטס אין Streamlit. מען זעט אן אינטעראקטיוו טשארט, געמאכט מיט plotly express. דער באניצער קען מיט איר ארבעטן, נישט נאר קוקן אויף א פארפרוירענעם בילד.

א צווייטע שיינע מעגליכקייט איז מאפע-ווייזונגען. אין דער לעקציע ווערט געוויזן א ביישפיל מיט der ביבליאטעק pydeck. מען אימפארטירט זי אונטער דעם קורצן נאמען pdk. מען לאזט לויפן דעם קאוד, און עס ערשיינט א מאפע. די מאפע שטעלט פאר די דאטא פון דער CSV טעקע וואס מען האט פריער ארויפגעלייגט.

עס ווערט ערקלערט אז אזעלכע טשארטס לאזן מאכן א שיינע ווייזונג, געבויט אויף דאטא, אין עטליכע סעקונדעס.

א פראיעקט: COVID-19 דעשבאָרד (39:22:33)

דערנאך גייט מען בויען צוזאמען א קליינעם פראיעקט. דאס איז א מיני-דעשבאָרד וואס ווייזט סטאטיסטיק פון COVID-19. דער דעשבאָרד האט דריי הויפט-טיילן:

- א טיטל.
- א טשארט וואס ווייזט די צאל פאלן לויט דער צייט.
- א טאבעלע מיט די רויע דאטא.

דערצו לייגט מען צו אינטעראקטיוויטעט. די באניצער זאלן קענען פילטערן און זוכן אין די דאטא.

דער קורס גיט א פארגלייך: אזוי ווי מען בויט דאס פאר COVID דאטא, קען מען בויען עפעס ענליכס פאר שטאק-פרייזן. דארט וואלט מען געצייכנט טרענדס לויט דער צייט, און געוויזן הויפט-צאלן ווי די גאנצע צאל פאלן אדער די באווענונגען אין פרייז.

א פאָרויסבליק אויף דעם וואס מען וועט בויען:

- א ליניע-טשארט וואס ווייזט דעם טרענד פון COVID-19 פאלן לויט דער צייט.
- אונטער דער טשארט, א טאבעלע וואס ווייזט די רויע דאטא אין א באקווועמער פארעם.
- אינטעראקטיוויטעט, וואס לאזט דעם באניצער פילטערן לויט זיין ווילן.

שריט ביי שריט: ביבליאטעקן, טיטל און טשארט (39:23:30)

אלס ערשטס אימפארטירט מען די נויטיגע ביבליאטעקן:

- pandas, צו באהאנדלען די דאטא.
- plotly.express, צו מאכן די ווייזונגען.
- streamlit, צו בויען די אפליקאציע.

דערנאך לייענט מען די דאטא מיט דער פונקציע pd.read_csv . מען גיט איר דעם וועג צו דער טעקע complete.csv, וואס מען האט פריער אראפגעלאדן. דער קורס נעמט אן אז אין דער טעקע איז דא א קאלאמע פאר דאטומס, און נאך א קאלאמע פאר דער צאל פאלן. די צוויי קאלאמעס וועט מען צייכענען.

דער ערשטער טייל פון דעם דעשבאָרד איז א טיטל. ער לאזט די באניצער וויסן וועגן וואס די אפליקאציע האנדלט. מען באניצט די פונקציע title, און אויפן עקראן ערשיינט "COVID-19 Dashboard".

דערנאך מאכט מען א פשוטע ליניע-טשארט. זי וויזט ווי די צאל פאלן ענדערט זיך מיט דער צייט, און מען מאכט זי מיט plotly.express. די קאלאמע פון דאטום גייט אויפן x -אקס, און די קאלאמע פון פאלן גייט אויפן y-אקס.

מען וויזט צוויי וועריאבלס פון דעם קאוד:

- איינע פאר דער גאנצער צאל פאלן (total cases).
- איינע פאר נייע טעגליכע פאלן (new daily cases).

כדי צו וויזן די טשארט אויפן עקראן, באניצט מען די פונקציע plotly_chart. עס ווערט ערקלערט אז די איין שורה קאוד שאפט א דינאמישע טשארט. זי פאסט זיך אליין צו צו די דאטא אין דער CSV טעקע. אזוי זעט דער באניצער גרינג ווי די לאגע פון COVID-19 האט זיך אנטוויקלט מיט דער צייט.

די דאטא-טאבעלע (39:25:10)

אונטער דער טשארט לייגט מען צו א טאבעלע מיט די רויע דאטא. אזוי קען דער באניצער זען די פונקטליכע צאלן הינטער דער טשארט. מען באניצט די פונקציע st.write, וואס וויזט דעם data frame אלס טאבעלע.

אויפן עקראן זעט מען דעם גאנצן דאטא-סעט אין טאבעלע-פארעם, מיט די דאטומס און די פאלן פאר יעדן רעקארד.

דער קורס גיט א טיפ: אויב מען וויל וויזן נאר א טייל פון דער טאבעלע, אדער ארויסהייבן געוויסע שורות, קען מען פריער פילטערן דעם data frame.

אינטעראקטיוויטעט מיט א selectbox (39:25:47)

עס ווערט געזאגט אז דאס איז דער בעסטער טייל פון Streamlit: מען קען מאכן די אפליקאציע אינטעראקטיוו מיט נאר עטליכע שורות קאוד.

דער ביישפיל איז א דראפ-דאון ליסטע. דער באניצער קלייבט אויס וועלכן לאנד'ס דאטא ער וויל זען. מען מאכט דאס מיט דעם widget וואס הייסט selectbox. אין דעם ביישפיל באניצט מען שטאטן (states) אנשטאט לענדער.

וועז דער באניצער קלייבט אויס פון דער ליסטע, דערהיינטיגן זיך סיי די טאבעלע סיי די טשארט. זיי וויזן דאן נאר די דאטא פון דעם אויסגעקלייבענעם ארט.

מען קען אויך צולייגן אנדערע כלים:

- sliders, למשל צו פילטערן לויט דאטום.
- checkboxes, צו זען געוויסע מאָסן ווי די גאנצע צאל פאלן אדער טויטפאלן.

אין דער דעמא זעט מען ווי מען פילטערט לויט שטאט, און די דאטא ווערט געוויזן לויט דעם אויסגעקליבענעם שטאט.

דער קורס רעקאמענדירט: ביים בויען און לויפן די אפליקאציע, זאל מען ערקלערן דעם קאוד שריט ביי שריט. למשל, ביים צולייגן די ליניע-טשארט זאל מען ערקלערן:

- וואס די פונקציע px.line טוט.
- ווי זי פארבינדט די קאלאמעס מיט דעם x- און y-אקס.
- ווי מען לייגט צו דעם טיטל.

ביים לויפן דעם קאוד זעט מען די רעזולטאטן גלייך. ערשט ערשיינט די טשארט מיט די דאטא, דערנאך די טאבעלע אונטן. וועז דער באניצער ארבעט מיט דעם selectbox, דערהיינטיגן זיך די דאטא און די טשארט אויפן ארט.

ארויסשטעלן די אפליקאציע אויף Streamlit Cloud (39:26:48)

יעצט וואס די אפליקאציע איז געבויט, דארף מען זי מאכן צוגענגליך פאר דער וועלט. דער קורס גייט דורך די שטאפלען:

- מאכן א requirements.txt טעקע.
- ארויפשטופן דעם קאוד אויף GitHub.
- פארבינדן מיט Streamlit Cloud.
- שאפן א פובליק URL וואס מען קען טיילן.

צום סוף רעדט מען וועגן געווינלעכע פעלערן ביים ארויסשטעלן, און ווי אזוי זיי צו פארריכטן. עס ווערט באטאנט אז Streamlit Cloud מאכט עס זייער גרינג.

ערשטער שריט: עפענען א חשבון אויף Streamlit Cloud . אויב מען האט נאך נישט, איינשרייבן זיך דורך דעם GitHub חשבון. אין דער דעמא איז מען שוין אריין אין Streamlit Cloud.

צווייטער שריט: דער קאוד מוז זיין אין א GitHub repository. אויב נישט, מאכט מען א נייעם repository און מען שטופט אריין דעם פראיעקט. אין דער דעמא האט מען געשאפן א repository מיטן נאמען "new app", און אריינגעלייגט דעם פראיעקט.

די git באפעלן, אין דער סדר:

- git init, צו אנהייבן א repository.
- git add, צו צולייגן די טעקעס.
- git commit -m "initial commit".
- git remote add origin, מיט דעם באניצער-נאמען און דעם repository-נאמען.
- git push origin master.

פארבינדן GitHub מיט Streamlit Cloud (39:28:50)

וועלן דער קאוד ליגט שוין אויף GitHub, גייט מען צום Streamlit Cloud דעשבאָרד. מען קליקט אויף "New app" און מען קלייבט אויס דעם repository וואס מען האט יעצט געשאפן. Streamlit Cloud דערקענט אליין דעם repository. אין דער דעמא קלייבט מען אויס די אפליקאציע, און דעם branch "main". נאכדעם דריקט מען "Deploy", און Streamlit טוט די איבעריגע ארבעט.

די requirements.txt טעקע (39:29:12)

כדי די אפליקאציע זאל לויפן גלאט, דארף מען אנגעבן אלע dependencies, דאס הייסט אלע ביבליאטעקן וואס זי דארף. דאס טוט מען אין א טעקע מיטן נאמען requirements.txt.

אין דעם פראיעקט-טעקע-אָרט מאכט מען די טעקע, און מען שרייבט אריין די נויטיגע ביבליאטעקן. אויב מען באניצט נאך ביבליאטעקן, מוז מען זיי אויך צולייגן.

מען קען אויך שאפן די טעקע אויטאמאטיש מיט דעם באפעל:

requirements.txt ;gt& pip freeze

דאס זארגט אז אלע פעקלעך פון דעם virtual environment ווערן אריינגענומען.

דערנאך שטופט מען די טעקע ארויף אויף GitHub. אזוי ווייסט Streamlit Cloud פונקטליך וועלכע פעקלעך צו אינסטאלירן ביים ארויסשטעלן.

דער פובליק URL (39:30:10)

נאכן ארויסשטעלן שאפט Streamlit Cloud א פובליק URL פאר דער אפליקאציע. מען קען אים טיילן מיט יעדן איינעם, און מען זעט אים אינעם דעשבאָרד. אין דער לעקציע ווערט געוויזן ווי אזא URL זעט אויס. עס ווערט אויך דערמאנט אז מען קען אריינלייגן (embed) די אפליקאציע גלייך אין א וועבזייטל.

געווייַנלעכע פעלערן און ווי זיי צו פארריכטן (39:30:30)

דער קורס זאגט אז ארויסשטעלן איז געווייַנלעך גלאט, אבער עס זענען דא עטליכע פעלערן וואס מען קען טרעפן:

- פעלנדיגע dependencies: אויב די אפליקאציע לאדט זיך נישט ריכטיג, זאל מען זיכער מאכן אז אלע פעקלעך שטייען אין requirements.txt. אויב מען האט עפעס פארגעסן, גיט די קלאוד א פעלער-מעלדונג. מען לייגט צו דאס פעלנדיגע פעקל און מען שטעלט ארויס נאכאמאל.
- אומגילטיגע טעקע-וועגן: אויב מען באניצט לאקאלע טעקעס, דארפן די וועגן זיין ריכטיג. מען דארף געדענקען אז די אפליקאציע לויפט אויף דער קלאוד, און רעלאטיוע וועגן דארף מען אפשר דערהיינטיגן. מען קען האלטן די דאטא אויף GitHub און באניצן אן אויסערליכן לינק.
- די אפליקאציע דערהיינטיגט זיך נישט: אויב ענדערונגען ערשיינען נישט, זאל מען זיכער מאכן אז מען האט זיי אריפגעשטופט אויף GitHub. אמאל נעמט עס עטליכע סעקונדעס ביז Streamlit Cloud דערקענט די ענדערונגען און שטעלט ארויס נאכאמאל.
- די אפליקאציע פאלט אוועק אדער גיט פעלערן: מען קוקט אין די logs אינעם Streamlit Cloud דעשבאָרד. Streamlit גיט א דעטאלירטע פעלער-מעלדונג, וואס העלפט געפינען און פארריכטן דעם פראבלעם. מען זאל אויך פרובירן דעם קאוד לאקאל, כדי צו כאפן באגס פארן ארויסשטעלן.

עצות און ווייטערדיגע שריט (39:31:25)

Streamlit איז געבויט צו זיין שנעל. אבער וועו די אפליקאציע וואקסט, קען די שנעלקייט ווערן א פראבלעם, בפרט מיט גרויסע דאטא-סעטס אדער קאמפליצירטע חשבונות.

א וועג צו פארבעסערן די שנעלקייט איז caching. caching האלט אן די רעזולטאטן פון טייערע פונקציעס, אזוי אז מען דארף זיי נישט איבערשאצן יעדעס מאל וועו די אפליקאציע לויפט. מען ווייזט קאוד פאר caching ביים לאדן די דאטא. עס ווערט ערקלערט אז דאס מאכט די אפליקאציע שנעלער, בפרט ביי גרויסע דאטא-סעטס.

ווייטערע מעגליכקייטן:

- אייגענע themes און branding: מען קען ענדערן דעם אויסזען און דעם געפיל פון דער אפליקאציע.
- extensions און components: למשל file uploaders, כדי באניצער זאלן קענען ארויפלייגן טעקעס. אדער פארגעשריטענע אינטעראקטיוו טשארטס פון ביבליאטעקן ווי Plotly אדער Altair.
- custom components, וואס מען געפינט אין דער Streamlit Components Gallery.

מען רעקאמענדירט צו שטודירן די אפיציעלע Streamlit דאקומענטאציע. דארט זענען דא נאך טיפס, לעקציעס, השראה און קאוד-ביישפילן פאר פארגעשריטענע פיטשערס.

סיכום (39:32:57)

דער קורס פאסט צוזאם וואס מען האט געלערנט: איינשטעלן Streamlit, בויען די ערשטע אפליקאציע, און זי ארויסשטעלן אויף דער אינטערנעט. דערצו האט מען געלערנט ארבעטן מיט דאטא, מאכן שיינע וויזונגען, און צולייגן אינטעראקטיוויטעט. יעצט קען מען אליין אנהייבן בויען אפליקאציעס מיט Streamlit.

גלייך דערנאך הייבט זיך אן א נייער טייל פון דעם קורס. עס ווערט פאָרגעשטעלט וואס דארט וועט קומען:

- וואס איז DevOps, און פארוואס עס איז וויכטיג.
- וואס איז Docker, און זיינע מעלות.
- ווי מען אינסטאלירט Docker.
- די ארכיטעקטור פון Docker, און זיינע קאמפאנענטן.

Docker: קאנטיינערס קעגן ווירטועלע מאשינען (39:33:30)

וואס איז DevOps, און פארוואס דארף מען Docker (39:33:30)

דער קורס הייבט אן דעם נייעם טייל מיט א פראגע: וואס מיינט DevOps? עס ווערט ערקלערט אז DevOps איז א צוזאמענארבעט צווישן צוויי טימס, דער Development -טים (די וואס שרייבן דעם קאוד) און דער Operations-טים (די וואס פירן די סיסטעמען). אזוי קען מען שטענדיג און אן אויפהער צושטעלן אפליקאציעס און סערוויסעס צום באניצער. די צוזאמענארבעט מאכט מעגליך עטליכע פראצעסן: continuous integration, continuous deployment און continuous delivery.

דערנאך ערקלערט דער קורס מיט א משל פארוואס Docker איז וויכטיג. ס'איז דא א פראגראמירער וואס האט א גרויסן סערוואר און וויל אויפשטעלן א סך אפליקאציעס אויף איין איינציגע ווירטועלע מאשין (VM). אבער

וועו מען שטעלט אויף אזויפיל באוונדערע אפליקאציעס אויף איין VM, קען זי נישט טראגן די גאנצע לאסט, און די ווירטועלע מאשין קראכט.

וואס קען ער טון כדי דאס צו פארמיידן? דער ענטפער איז Docker. וועו עס וויל אויפשטעלן א סך אפליקאציעס אויף איין ווירטועלע מאשין, פאר דעם איז Docker די ריכטיגע לעזונג. מיט Docker ווערן נישט די טעותים פון יעדע אפליקאציע אריבערגעטראגן, די ווירטועלע מאשין קראכט נישט, און זי ווערט נישט באווירקט.

Docker קעגן ווירטועלע מאשין: פיר הויפט-פונקטן (39:35:00)

דער קורס שטעלט אוועק א פארגלייך צווישן Docker און א ווירטועלע מאשין. מען דערמאנט אז א ווירטועלע מאשין איז א פולשטענדיגער באשטאנדטייל וואס קען טראגן יעדע סארט לאסט. פונדעסטוועגן זענען דא עטליכע סיבות פארוואס מען וועט אין געוויסע פעלער בעסער אויסקלייבן Docker:

- זכרון-באניץ (Memory): Docker נוצט זייער ווייניג זכרון, ווייל ס'איז נישטא דער באגריף פון א "גאסט אפערירונג-סיסטעם" (guest operating system). א ווירטועלע מאשין נעמט א ריזיגע מאס RAM כדי צו האלטן דעם אפערירונג-סיסטעם לעבעדיג און ארבעטנדיג.
- פערפארמענס: וועו מען נוצט ווייניגער רעסורסן, איז די פערפארמענס בעסער. וועו מען נעמט מער רעסורסן, קען מען נישט ארויסגעבן די זעלבע פערפארמענס.
- פארטאביליטעט (פארטראגבארקייט): א Docker קאנטיינער איז פיל לייכטער צו פארטראגן פון איין ארט צום אנדערן ווי א ווירטועלע מאשין.
- בוט-צייט (Boot time): א ווירטועלע מאשין הייבט זיך אן אין מינוטן, א Docker קאנטיינער אין סעקונדעס.

דער סך-הכל: אין זכרון-באניץ, בוט-צייט און פערפארמענס איז Docker בעסער ווי ווירטועלע מאשינען.

א משל מיט 9 GB זכרון (39:36:32)

דער קורס ברענגט א קלאסישן משל וועגן זכרון. מען זעט א סערווער מיט 9 GB זכרון און דריי אפליקאציעס: app 1, app 2 און app 3. אויף דער בילד ווייזט גרין דעם פארנומענעם זכרון און רויט דעם פארווארפענעם זכרון. ביי איינער פון די מאשינען זענען 4 GB פארנומען און 3 GB פארווארפן. app 2 נוצט בערך 2 GB. ביי אן אנדערער זענען 3 GB פארנומען און 1 GB פארווארפן. ביי ווירטועלע מאשינען זענען אווי אלע 9 GB RAM צוגעטיילט, און די איבעריגע 6 GB וואס ווערן נישט געוונצט קען מען נישט ווידער באנוצן.

דער טעם: אז מען טיילט צו, למשל, 4 GB RAM צו א ווירטועלע מאשין, בלייבט דער זכרון דארט פארשלאסן. עס איז נישט אזוי אז מען נוצט נאר 4 GB פון די 9 GB, און דער רעשט בלייבט פריי פאר אנדערע זאכן. דער רעשט ווערט פשוט פארווארפן, און מען קען אים נישט נוצן פאר קיין אנדערן צוועק.

דאס איז דער חסרון פון א ווירטועלע מאשין: וועו מען טיילט צו זכרון צו א געוויסער ווירטועלע מאשין און מען נוצט אים נישט אינגאנצן, ווערט דער איבעריגער זכרון פארווארפן. ער איז "hardcoded" צוגעטיילט צו יענער מאשין, און מען קען אים נישט אויסנוצן אנדערש.

ביי Docker איז דאס אנדערש. אויב מען נוצט, למשל, נאר 3 GB פון די 9 GB RAM, קען מען דעם אומבאנוצטן זכרון ווידער באנוצן פאר נייע קאנטיינערס. דער זכרון איז נישט באשטימט פאר איין זאך. צי מען נוצט 1, 2, 3 אדער 4 GB, דער פרייער רעשט שטייט צו דער רשות פון אנדערע קאנטיינערס. אזוי האט מען שטענדיג אן

אפטימאלן באניץ פון די רעסורסן, אן אומזיסטע פארשוועדונג. דערפאר זענען Docker קאנטיינערס אזוי עפעקטיוו אין זכרון אין פארגלייך מיט ווירטועלע מאשינען.

פערפארמענס: צווויי אפעריירונג-סיסטעמען קעגן איינס (39:39:05)

דער צווייטער פונקט איז פערפארמענס. עס ווערט ערקלערט ווי א ווירטועלע מאשין איז געבויט, שיכט אויף שיכט:

- אונטן איז די פיזישע אינפראסטרוקטור (די האַרדווער).
- אויף איר ליגט א באזע-אפעריירונג-סיסטעם.
- אויף דעם שטייט דער hypervisor.
- דער hypervisor שאפט די ווירטועלע מאשינען, און יעדע פון זיי האט אינעווייניג א guest operating system.

די פראבלעם איז אז מען האט דא צווויי אפעריירונג-סיסטעמען. דער גאסט-סיסטעם נעמט אויך רעסורסן: זכרון, פלאץ און פראצעסאר. אין יעדן סארט רעסורס נעמט ער זיין חלק. דערפאר, ווען מען שטעלט אויף א זייער קליינע אפליקאציע, נוצט מען ענדליך פיל מער, ווייל דער גאסט-סיסטעם נעמט אוועק דאס רוב רעסורסן, בשעת די אפליקאציע אליין דארף ווייניג.

א Docker קאנטיינער האט נישט קיין גאסט-אפעריירונג-סיסטעם. דער Docker engine איז גענוג צו פירן די קאנטיינערס, און אין די קאנטיינערס איז די אפליקאציע אויפגעשטעלט. דערפאר איז די פערפארמענס פיל בעסער ווי ביי א ווירטועלע מאשין: ס'ווערן נישט פארשוועדעט קיין רעסורסן. ביי ווירטועלע מאשינען לויפט א פולשטענדיגער גאסט-סיסטעם וואס פאדערט רעסורסן, ביי קאנטיינערס דארף מען דאס נישט.

פארטאביליטעט און בוט-צייט (39:40:35)

דער דריטער פונקט איז פארטאביליטעט. אז אן אפליקאציע ארבעט אויף סיסטעם A, מיינט נאך נישט אז זי וועט ארבעטן אויף סיסטעם B. דער קורס דערמאנט א באקאנטע פראבלעם: די אפליקאציע ארבעט פיין אין א "נידעריגער" סביבה (lower environment, ווי dev), און ווען זי קומט אריין אין דער production סביבה, פעלט זי אמאל און קומט נישט אויף ווי געהעריג. דאס איז די פארטאביליטעט-פראבלעם ביי ווירטועלע מאשינען.

ביי קאנטיינערס איז דאס אנדערש. ווען א קאנטיינער ארבעט שוין אין דער dev סביבה און איז גרייט, דארף מען אים נישט ענדערן לויט דער סביבה. דער זעלבער קאנטיינער קען גרינג ארויפגעשטעלט ווערן אין production. וואס מען האט צוגעגרייט אין dev, ארבעט אן פראבלעמען אין production, ווייל דער קאנטיינער אנטהאלט אינעווייניג אלע) dependencies די זאכן וואס די אפליקאציע דארף) און אלע) executablesדי לויפבארע פיילס). מען דארף נישט מאכן קיין ענדערונגען אין קאנטיינער.

דער פערטער פונקט איז בוט-צייט. ביי ווירטועלע מאשינען איז זי אין מינוטן, ביי Docker קאנטיינערס אין סעקונדעס, אפשר אפילו אין מילי-סעקונדעס. דערפאר באטראכט מען א ווירטועלע מאשין ווי א סביבה וואס פרעסט רעסורסן, בשעת Docker איז פיל לייכטער און שנעלער.

וואס איז Docker און וואס איז א Docker קאנטיינער (39:42:10)

דער קורס ווייזט די ארכיטעקטור פון Docker. אונטן איז די פיזישע אינפראסטרוקטור, און אויף איר דער host operating system. ס'איז נישטא קיין hypervisor און קיין גאסט-אפעריירונג-סיסטעם. אנשטאט דעם שטייט

דער Docker engine, וואס פירט די Docker קאנטיינערס. אין יעדן קאנטיינער געפינען זיך די אפליקאציעס צוזאמען מיט אלע נויטיגע binaries און libraries. דער Docker engine איז דער פראצעס וואס באהאנדלט אלע בקשות. וועז דער Docker engine לויפט, זענען אלע קאנטיינערס צוגענגליך.

די דעפיניציע: Docker איז א ווייכווארג-פלאטפארמע פאר virtualization אויפן OS-שטאפל. זי לאזט פראגראמירער און IT אדמיניסטראטארן שאפן, ארויפשטעלן און לויפן אפליקאציעס אין א Docker קאנטיינער, מיט אלע זייערע dependencies. היינט איז Docker א ווייכטיגע געצייג פאר אויפשטעלן אפליקאציעס, צוליב דעם יתרון פון פארטאביליטעט: מען דארף כמעט נישט ענדערן לויט דער פלאטפארמע אדער סביבה. וואס די פראגראמירער בויען, דאס זעלבע גייט ביז production.

א Docker קאנטיינער איז א לייכטער ווייכווארג-פעקל וואס אנטהאלט אלע dependencies: א באזע-אפעריירונג-סיסטעם, frameworks און libraries. ס'איז א פולשטענדיגער פעקל, אזוי אז מען קען די אפליקאציע גלייך אויפשטעלן. מען לייגט אריין סיי דעם מקור-קאוד (source code) סיי די dependencies. אין Docker קאנטיינערס קען מען האלטן כמה אפליקאציעס, כמה פראגראמירונג-שפראכן און כמה frameworks. מען קען דארט אפילו אויפשטעלן דאטא-באזעס.

די מעלות פון Docker (39:44:15)

דער קורס רעכנט אויס די מעלות פון Docker:

- שנעלע ארויפשטעלונג (rapid deployment)
- שנעלע און גרינגע קאנפיגוראציע
- בעסערע עפעקטיווקייט
- פארטאביליטעט
- סקעילאביליטעט (scalability)
- זיכערהייט (security)

דערנאך נעמט מען זיי דורך איינס נאך איינס, אנגעהויבן פון rapid deployment. דער משל איז א וועב-אפליקאציע וואס מען וויל ארויפשטעלן. ס'זענען דא געוויסע שריט וואס מען מוז דורכגיין אין יעדער סביבה. אויב מען האט, למשל, פינף סביבות, מוז מען טון די זעלבע שריט אין אלע פינף, יעדעס מאל אין אן אנדערער צייט.

מיט Docker קאנטיינערס דארף מען נישט לויפן די שריט כמה מאל. אז די שריט זענען אמאל דורכגעפירט געווארן אין איין Docker קאנטיינער, גייט דער זעלבער קאנטיינער ביז production. מען דארף אים נישט ווידער בויען (rebuild). מען האט שוין דעם קאנטיינער, און מען לויפט בלויז איין אדער צוויי באפעלן כדי אים אויפצושטעלן, אן ווידער טון די ארבעט פון אנהייב אן.

Docker: מעלות, אינסטאלאציע און ארכיטעקטור (39:45:47)

די מעלות פון Docker (39:45:47)

דער קורס פירט ווייטער מיט דער רשימה פון די מעלות פון Docker. די ערשטע מעלה וואס ווערט דא דערמאנט איז אז Docker מאכט גרינגער די ארבעט פון ארויסשיקן (deployment) די אפליקאציע אין די production סביבה, אדער אין די העכערע סביבות.

• גרינגע און שנעלע קאנפיגוראציעס: עס ווערט ערקלערט אז די קאנפיגוראציע פון Docker קאנטעינערס איז גאנץ גרינג. מען קען בויען אייגענע, באזונדער-צוגעפאסטע Docker images פון Dockerfiles. די פראצעסן קען מען אויך אריינשטעלן אין Jenkins, אזוי אז CI/CD איז גאנץ מעגליך מיט דער הילף פון Docker, ווען מען וויל דאס טאקע דורכפירן.

• בעסערע עפעקטיווקייט: Docker קאנטעינערס האבן א בעסערע עפעקטיווקייט און א בעסערע פארוואלטונג פון רעסורסן ווי virtual machines.

• טראגבארקייט (portability): עס ווערט איבערגעזע'חזר'ט אז מען קען גרינג אריבערפירן פון איין סביבה צו א צווייטער און פון איין סיסטעם צו א צווייטן. אויך פאר פראגראמירער איז דאס גרינג: מען קען די קאנטעינערס גאר לייכט אריבערגעבן פון איין מאשין צו א צווייטער און פון איין פראגראמירער צום אנדערן.

• סקעילאביליטי (scalability): פונקט ווי מען קען שאפן עטליכע virtual machines, קען מען שאפן עטליכע קאנטעינערס און צעטיילן דעם טראפיק צווישן זיי. דער קורס באמערקט אבער אז די סקעילאביליטי ווערט דורכגעפירט מיט orchestration מכשירים.

• זיכערהייט (security): מען קען אריינשטעלן די בעסטע זיכערהייט-שיטות, ווייל Docker איז א סארט virtualization ענליך צו א virtual machine. דעריבער שטייען אלע די פארשידענע מעגליכקייטן צו דער ניצן, ווען מען וויל מאכן א Docker קאנטעינער זיכער.

אינסטאלירן Docker אויף פארשידענע פלאטפארמעס (39:47:00)

דערנאך גייט מען איבער צו דער אינסטאלאציע. עס ווערט ערקלערט אז אינסטאלירן Docker איז א געוועלטיכע זאך, און Docker איז פארן אויף יעדע פלאטפארמע וואס מען וויל: Ubuntu, Windows און Mac.

• Ubuntu: מען לויפט די באפעלן אויף דער Ubuntu אפעריינדער סיסטעם, און דורך זיי ווערט Docker אראפגעלאדן.

• Windows: דא איז פארן א באזונדערע .exe טעקע, און מען פירט דורך די אינסטאלאציע פון דער .exe טעקע.

• Mac: פאר דער Mac אפעריינדער סיסטעם זענען פארן DMG טעקעס. ווען מען עפנט די DMG טעקע, אינסטאלירט זי דעם קאמפאנענט. אויף Mac קען מען אויך ניצן Homebrew כדי צו אינסטאלירן Docker.

די ארכיטעקטור פון Docker: דער Client און דער Daemon (39:47:48)

דער ווייטערדיקער טעמע איז די ארכיטעקטור פון Docker. דער קורס זאגט אז זי איז גאנץ פשוט, און עס זענען פארן א פאר הויפט-קאמפאנענטן.

דער ערשטער, וואס שטייט פאר'ן באניצער, איז דער Docker Client, אדער די Docker CLI . דאס איז א command line interface וואס נעמט אויף די בקשות פונעם באניצער. למשל, ווען מען לויפט א באפעל, נעמט די CLI אויף די בקשה און שיקט זי ווייטער צום Docker daemon , וואס לויפט אויפ'ן סערוואר, דורך א REST API.

דער Docker daemon לויפט אן אויפהער אויפ'ן סערוואר ווי א פראצעס אין הינטערגרונט, און ער ווערט אינסטאלירט ווי א service. עס ווערט געווארנט: אויב דער daemon לויפט נישט, וועט מען נישט באקומען

קיין ריכטיגן ענטפער פון דער Docker command line . כדי די באפעלן זאלן ווערן דורכגעפירט, מוז דער Docker סערווער, דאס הייסט דער daemon, זיין אויפגעשטעלט און לויפן.

מען קען אויך קאנפיגורירן די Docker CLI זיך צו פארבינדן מיט אנדערע סערווערס, מיט א Docker daemon אויף אן אנדערן סערווער. די פארבינדונג גייט דורך דער REST API : אלע בקשות ווערן געשיקט צום daemon, און דער ענטפער וואס קומט צוריק ווערט איבערגעגעבן צום באניצער.

די דריי קאמפאנענטן אין פרטים (39:49:21)

דער קורס חזר'ט איבער און בריייטערט אויס די קאמפאנענטן:

- Docker Client: באשטייט פון דעם CLI קאמפאנענט וואס נעמט אויף אלע בקשות פונעם באניצער, און ניצט די REST API צו זיך פארבינדן מיטן daemon.
- Docker daemon: לויפט שטענדיג. ווען מען לויפט באפעלן ווי docker pull, docker ps און ענליכע, שיקט מען אין דער אמת'ן א בקשה צום daemon, און דער daemon טוט די גאנצע ארבעט. דער Client אליין טוט נישט די ארבעט. ער איז בלויז א מין interface וואס נעמט אויף די בקשה און שיקט זי ווייטער. דער daemon איז א פראצעס אין הינטערגרונט אויף דער אפערירנדער סיסטעם, און עס איז דער זעלבער פראצעס אויף אלע פלאטפארמעס: Mac, Windows אדער Linux. אומעטום איז דאס די זעלבע יסוד-פונקציע. ער הערט שטענדיג צו דער REST API, און יעדע נייע בקשה באארבעט ער גלייך. צי מען שטעלט אפ א קאנטיינער אדער מען לאזט אים לויפן, ווערט די פאסיגע אקציע דורכגעפירט, און דער ענטפער ווערט צוריקגעשיקט דורך דער REST API צום Docker Client.

דער קורס ערקלערט אויך אז dockerd איז דער פראצעס, אדער די command line , וואס הייבט אן דעם daemon. ווען מען לויפט אויפ'ן סערווער דעם באפעל service docker status, קען מען זען וועלכער באפעל ווערט דורכגעפירט אין הינטערגרונט. דארט זעט מען אז דאס איז דער פראצעס וואס הייבט אן דעם daemon, און ער באקומט געוויסע קאנפיגוראציעס ווי פאראמעטערס. אויך מיט'ן באפעל ps -ef קען מען זען וועלכער Docker פראצעס לויפט.

- Docker Registry: דער דריטער קאמפאנענט. עס איז פארן א פארטיגע cloud סערוויס וואס הייסט Docker Hub, ווו מען קען אפהיטן Docker images. דארט זענען פארן א סך פארטיגע images, אבער מען קען אויך ארויפלייגן אייגענע images. ווער עס ווייל, קען אויך האבן א פריוואטע registry פון זיך אליין. למשל אין JFrog קען מען האבן פריוואטע, אייגענע registries. אלע די פארשידענע repositories קען מען ניצן מיט Docker קאנטיינערס. Docker האט די בייגזאמקייט אז מען קען זיך פארבינדן מיט וועלכער registry אדער repository מען ווייל.

וואס איז א Docker Image (39:51:54)

דערנאך ווערט ערקלערט אין פרטים וואס א Docker image איז. א Docker image איז נישט מער ווי א מוסטער (template) פון אנווייזונגען, וואס מען ניצט כדי צו שאפן דעם ענדגילטיגן Docker קאנטיינער.

דער קורס ברענגט א משל: מען וויל א Tomcat Apache image. וועלכע שריט דארף מען טון?

- צום ערשט קלייבט מען אויס א באזיס אפערירנדע סיסטעם.
- אויף דעם אינסטאלירט מען JDK אדער אנדערע פעקלעך וואס מען דארף פריער האבן.
- אויף דעם קאנפיגורירט מען Tomcat Apache.

דאס זענען די שריט, א סדר פון אנווייזונגען איינס נאך'ן אנדערן, אזוי אז צום סוף האט מען א פארטיגן קאנטעינער וואס מען קען ארויסשיקן כדי צו האלטן Tomcat Apache.

כדי צו בויען א Docker image דארף מען א טעקע וואס הייסט Dockerfile, וואס אנטהאלט אלע די אנווייזונגען. וועו מען לויפט די Dockerfile, ארבעט זי ווי א build מכשיר: יעדע אנווייזונג ווערט באארבעט, און מען באקומט דעם ענדגילטיגן רעזולטאט.

די שיכטן (layers) פון אן Image (39:53:00)

עס ווערט ערקלערט אז Docker images באשטייען פון עטליכע שיכטן, שריט נאך שריט. צום ערשט איז פאראן א באזיס-שיכט, פון וועלכער דאס image הייבט זיך אן. אויף דעם קומט נאך א שיכט, למשל וועו מען האט אינסטאלירט JDK. דערנאך, וועו מען האט קאנפיגורירט Tomcat Apache, קומט נאך א שיכט. די שיכטן גייען אין א סדר פון צולייגן: די עלטסטע הייסט די באזיס-שיכט, און די נייסטע ענדערונגען ווערן געמאכט אין די נייסטע שיכטן.

דער קורס ווייזט א ביישפיל פון א Dockerfile מיט פיר אנווייזונגען, וואס מען רופט אויך "א פיר-שיכטיגע Docker image":

- FROM ubuntu:18.04: מען ציט אראפ Ubuntu 18.04.
- COPY: מען ברענגט אריין א טעקע.
- RUN make: מען לויפט דעם make כלי.
- CMD: מען לויפט די python טעקע.

די באזיס-שיכט איז Ubuntu 18.04. נאכדעם וואס די לעצטע CMD אנווייזונג ווערט באארבעט און דורכגעפירט, באקומט מען א פולשטענדיגע, פארטיגע Docker image, און וועו דער גאנצער build פראצעס ענדיגט זיך, האט מען א קאמפלעטע image אדער קאנטעינער.

דער קורס באמערקט אז די באזיס-שיכט, Ubuntu 18.04, בויט מען נישט אליין. זי ווערט אויטאמאטיש אראפגעלאדן ווי א גאנצע באזיס-שיכט. נאר נאך איר קומען צו layer 1, layer 2 און layer 3, ווייל דארט טוט מען די ענדערונגען. FROM מיינט נישט אז מען מאכט ענדערונגען אין א שיכט. מען ברענגט בלויז אריין א באזיס אפעריינדע סיסטעם, אדער א באזיס image. דאס צולייגן פון שיכטן גייט נאר פאר זיך נאך דער באזיס-שיכט.

קאנטעינער-שיכטן און Copy-on-Write (39:55:32)

וועו א באניצער שאפט א קאנטעינער, ווערט א נייע שיכט ארויפגעלייגט אויפ'ן image . די שיכטן הייסן container layers. די ערשטע אנווייזונג, Ubuntu, איז נישט קיין container layer. מען קען זי רופן א base image layer.

יעדער קאנטעינער האט א באזונדערע container layer וואס מען קען אין איר שרייבן און לייענען, און יעדע ענדערונג אין א קאנטעינער שפיגלט זיך אפ נאר אין די container layers. layer 1 קען האבן געוויסע אנווייזונגען, layer 2 אנדערע, אזוי ווייטער שריט נאך שריט. וועו מען מעקט אויס א קאנטעינער, ווערט די אויבערשטע שיכט אויך אויסגעמעקט. למשל, אויב א קאנטעינער האט געהאט דריי שיכטן און מען מעקט אים אויס, פארשווינדט די אויבערשטע שיכט, ווייל דער קאנטעינער אליין איז אויסגעמעקט.

צום סוף ערקלערט דער קורס אז Docker ניצט א copy-on-write שיטה, סיי ביי images און סיי ביי קאנטעינערס. דאס איז א שיטה צו טיילן און קאפירן טעקעס פאר בעסערער עפעקטיווקייט, וואס ווערט געוונצט צו פארבעסערן די גאנצע פערפארמענס פון די קאנטעינערס. זי מאכט Docker עפעקטיוו דורך פארקלענערן דעם באנוץ פון disk שטח, ווייל דאס קאפירן ווערט דורכגעפירט נאר אין דעם מאמענט ווען מען דארף עס טאקע. דאס איז זייער וויכטיג: ווען עס לויפן א סך קאנטעינערס, מוז מען האלטן דעם באנוץ פון disk און רעסורסן אויפ'ן מינימום, און זיך פארזיכערן אז ער גייט נישט אריבער א געוויסע שיעור. ווען דער באנוץ איז קלענער, ווערט אויך די אלגעמיינע פערפארמענס בעסער.

דאקער: רעדזשיסטרי, קאנטעינערס, Compose און Swarm (39:57:36)

די לעיערס פון א דאקער-אימעדזש און דער Union File System (39:57:36)

דער קורס הייבט אן מיט עטליכע פאקטן וועגן די שיכטן (layers) פון א דאקער-אימעדזש. די באזע-שיכטן זענען אינגאנצן readonly, מען קען זיי נישט מאדיפיצירן.

דער Union File System שפארט זכרון, ווייל ער פארמיידט אז די זעלבע פיילס זאלן זיך דובלירן. אין א Union File System קען מען צוזאמשטעלן אלע שיכטן צו איין אימעדזש. ווען אלע שיכטן ווערן צוזאמגעשטעלט, באקומט מען דאס ענדגילטיגע פראדוקט, וואס מען רופט א דאקער-אימעדזש אדער דאקער-קאנטעינער.

דער סיסטעם לאזט אויך אז דער פייל-סיסטעם זאל אויסזען ווי ער קען באשריבן ווערן. מען רופט דאס copy-on-write. מען ניצט די copy-on-write סטראטעגיע כדי צו מאכן ענדערונגען אין דעם Union File System בשעת מען שאפט די פארשידענע שיכטן.

דער דאקער-רעדזשיסטרי (39:58:20)

דערנאך ערקלערט דער קורס דעם Docker Registry, א זייער וויכטיגער טייל פון דאקער. אפילו ווען מען גרייט צו א דאקער-אימעדזש אויפן אייגענעם קאמפיוטער, איז נישט גרינג אריבערצוטראגן די אימעדזשעס פון איין סיסטעם צום צווייטן, אויב מען האט נישט קיין געצייג וואו מען קען זיי טיילן. דעריבער איז דער רעדזשיסטרי זייער וויכטיג פאר דער פארטאביליטעט פון דאקער.

א דאקער-רעדזשיסטרי איז א סערוויס וואס האלט און פארשפרייט דאקער-אימעדזשעס צווישן באניצער. איין באניצער קען בויען אן אימעדזש און אסאך באניצער קענען אים ניצן. דער ביישפיל: א Jenkins-אימעדזש ווערט צוגעגרייט נאר איין מאל, אבער פארשידענע מענטשן ניצן אים.

א repository איז א זאמלונג פון דאקער-אימעדזשעס, און אין איין repository געפינט מען פארשידענע אימעדזשעס. אינעם רעדזשיסטרי אונטערשיידט דער באניצער צווישן די אימעדזשעס לויט זייערע tag-נעמען.

דער קורס גיט א ביישפיל: עס איז דא אן אימעדזש מיטן נאמען Jenkins. דער נאמען בלייבט דער זעלבער, אבער די tags זענען פארשידן: איינער JDK8, א צווייטער JDK 9 און א דריטער JDK 10. דער חילוק צווישן די דריי: דער JDK8-אימעדזש איז צוגעגרייט און לויפט אויף JDK 8. ווער עס וויל JDK 9 , דארף אראפציען דעם Jenkins-אימעדזש וואו JDK 9 איז אינסטאלירט. מען ניצט אלזא באזונדערע tags לויט די פרעימווארקס אדער runtimes. אזוי קען מען אונטערשיידן צווישן די אימעדזשעס און אויסקלייבן יענעם וואס פאסט פאר די באדערפענישן.

Docker Hub, פריוואטע רעדזשיסטריס, און pull/push (40:00:10)

דאקער האט פון זיך א רעדזשיסטרי אין קלאוד, Docker Hub, וואס ווערט אויטאמאטיש פארבונדן גלייך ביים אינסטאלירן דאקער. ווען מען ארבעט אין א פריוואטער אינפראסטרוקטור, קען מען ניצן דאקער-repositories אויף JFrog Artifactory, אדער מען קען האסטן פריוואטע Docker Hub repositories. האט סיי עפנטליכע סיי פריוואטע repositories, און מען קען שאפן ביידע סארטן.

מען רעדט מיט די אימעדזשעס דורך pull און push:

- ווער עס וויל אראפלאדן א דאקער-אימעדזש, לויפט pull.
- ווער עס וויל ארויפשטעלן זיין אימעדזש אויפן רעדזשיסטרי, לויפט push.

אין ביידע פעלער רעדט מען מיט Docker Hub. ווען מען לויפט א build-פראצעס אדער קאנטעינערס, קען זיין אז געוויסע אימעדזשעס ווערן אראפגעצויגן פון Docker Hub. אויב דער אימעדזש מיט דעם tag געפינט זיך דארט, ווערט pull געלינגען.

דער קורס ווארנט: מיט push קען מען ארויפשטעלן נאר אייגענע (custom) אימעדזשעס. מען קען נישט pushן די פארטיגע אימעדזשעס וואס זענען שוין דא. מען שטעלט ארויף אויף Docker Hub נאר אן אייגענעם אימעדזש, און דערנאך קען מען טון pull און push צו יענע repositories. אויף פארטיגע אימעדזשעס פון דריטע פירמעס קען מען נאר טון pull, נישט push. push קומט נאר פאר ווען מען ארבעט מיט אן אייגענעם אימעדזש.

דער דאקער-קאנטעינער (40:01:42)

דער ווייטערדיגער טייל איז דער Docker container. א דאקער-קאנטעינער איז א פעקל וואס מען קען לויפן, און עס האלט די אפליקאציע צוזאמען מיט אירע dependencies. ווען מען גרייט צו א קאנטעינער, לייגט מען אריין ביידע, די dependencies און דעם source-קאוד פון דער אפליקאציע. דערפאר דארף דער קאנטעינער נישט זוכן קיין dependencies אויפן host-מאשין אדער סערווער.

דאס איז א מעלה פאר פארטאביליטעט: אלעס וואס מען בויט אויף איין סיסטעם געפינט זיך אינעם קאנטעינער, און מען דארף נישט טון קיין ווייטערע ארבעט אויף אנדערע מאשינען. וואס ארבעט אויף איין סיסטעם, ארבעט גרינג אויך אויף א צווייטן.

א קאנטעינער איז אויך גאר לייכט. מען קען אים גרינג אויפשטעלן און לויפן אויף סערווערס, ווייל ער נעמט נישט אזויפיל רעסורסן ווי א virtual machine. ווער עס וויל האסטן זיין אפליקאציע אויפן אייגענעם קאמפיוטער, קען דאס גרינג טון אין א דאקער-קאנטעינער, וואס ניצט אסאך ווייניגער ווי א virtual machine. מען דארף זיך נישט זארגן וועגן דעם host-אפעריירן-סיסטעם און זיינע באדינגונגען.

קאנטעינערס לויפן אינגאנצן אפגעזונדערט, פונקט ווי א virtual machine, אן קיין איינמישונג. זיי טיילן זיך מיט דעם kernel פונעם OS, אבער זיי ווירקן נישט איינער אויפן אנדערן. צוויי קאנטעינערס רעדן נישט מיטאנאנד, סיידן מען שטעלט זיי אויף צו רעדן.

דא זענען אויך data volumes, וואס מען קען טיילן און ניצן צווישן עטליכע קאנטעינערס. ווען עס בלייבט אומגענוצטער פלאץ, קען מען אים טיילן צווישן קאנטעינערס.

קאנטעינערס ווערן געבויט פון דאקער-אימעדזשעס, מיטן באפעל docker run. למשל, מען האט צוגעגרייט אן אייגענעם אימעדזש מיט א Dockerfile, און מען וויל דערפון מאכן א קאנטעינער. דאן לויפט מען docker run. דער באפעל שאפט אן instance פון א דאקער-אימעדזש.

(40:04:00) Docker Compose

דער נעקסטער טייל איז Docker Compose. נארמאל, וועט עס וויל לויפן פינף-זעקס קאנטעינערס, דארף לויפן docker run פינף-זעקס מאל. וועט עס וויל פארבינדן די קאנטעינערס צווישן זיך, דארף דאס פירן דורך דאקער-באפעלן. ס'איז נישט שטענדיג גרינג צו לויפן די באפעלן מיט די הענט אדער זיך זיי געדענקען.

Docker Compose איז א געצייג וואס לאזט לויפן עטליכע קאנטעינערס מיט איין סערוויס. מען האט איין פייל, וואס האלט אין זיך אלע באשרייבונגען פון די קאנטעינערס און זייערע פארבינדונגען. מען לויפט אים, און אלע קאנטעינערס הייבן אן. וועט עס וויל לויפן צוויי קאנטעינערס, לויפט איין פייל אנשטאט צוויי באפעלן. מען קען אויך האבן פינף, זעקס, זיבן אדער וויפיל מען וויל, און זיי ווערן אויפגעשטעלט אויפן סערווער אן ארבעט מיט די הענט.

די קאנטעינערס לויפן אפגעזונדערט, אבער מען קען באשטימען ווי זיי זאלן רעדן מיטאנאנד, און דאס שרייבט מען אריין אינעם Docker Compose פייל. דער פייל איז געשריבן אין YAML-פארמאט. מען באשרייבט אין אים: מען דארף צוויי קאנטעינערס, ווער פון זיי ניצט וועלכן אימעדזש, מיט וועלכע פאראמעטערס, און אויך די פארבינדונג צווישן זיי.

דער קורס גיט א ביישפיל: אן אפליקאציע דארף אן Apache web server און א MySQL דאטאבאזע. מען קען מאכן איין Docker Compose פייל וואס לויפט ביידע קאנטעינערס, אן דארפן אנהייבן אדער איבערשטארטן יעדן באזונדער. דער פייל האלט ביידע, דעם Apache-קאנטעינער און דעם MySQL-קאנטעינער. די מעלה איז אז מען דארף זיי נישט לויפן באזונדער, נאר מען פירט אלעס אויס מיט איין פייל.

Docker Swarm: מאנעדזשער און ווארקער נאודס (40:06:17)

דערנאך ערקלערט דער קורס Docker Swarm. דאס איז דער orchestration-געצייג וואס קומט אריינגעבויט פון דאקער אליין. עס איז א סערוויס פאר קאנטעינערס וואס לאזט IT-אדמיניסטראטארן און פראגראמירער שאפן און פירן א cluster פון swarm nodes.

ווען מען אינסטאלירט נאר דעם יסודות-דאקער, פעלן געוויסע מעגלעכקייטן: high availability , עטליכע instances און scalability. דערפאר דארף מען אן orchestration-געצייג ווי Docker Swarm . מען האט א swarm פון פינף-זעקס node-מאשינען, און אויף זיי קען מען אויפשטעלן אן אפליקאציע-אינפראסטרוקטור וואס איז שטענדיג פאראן. דער swarm ווערט געניצט ווען מען וויל לויפן די אפליקאציע אין פינף, זעקס אדער זיבן קאנטעינערס.

יעדער node אין Docker Swarm האט א Docker daemon פאר די קאמוניקאציע. עס איז דא א master node וואס רעדט מיט אלע node -מאשינען. ער מעלדט זיי למשל: "איך וויל אויפשטעלן א קאנטעינער אויף node 3", און דאן וועט דער קאנטעינער אויפגעשטעלט און לויפט אויף node 3.

א swarm האט צוויי סארטן nodes:

- manager node, דער master node, וואס פירט די רעסורסן.
- worker node, וואס טוט די מערסטע ארבעט.

פון manager node ערווארטעט מען נישט ער זאל האסטן עפעס חוץ די הויפט-קאמפאנענטן. מען קען אבער אויך באשטימען ביים אויפשטעלן, אז די קאנטעינערס זאלן ווערן אויפגעשטעלט נאר אויף ווארקער nodes. דער manager פארבינדט זיך כסדר מיט די ווארקער nodes, און ווען עס קומט אן א נייע בקשה, שטעלט ער אויף דעם קאנטעינער אויף א ווארקער node.

די ווארקער nodes באקומען די בקשות פונעם manager און פירן זיי אויס. אויב א ווארקער node איז נישט אקטיוו אדער נישט online, קען דער master node נישט אויפשטעלן קיין קאנטעינערס אויף יענעם סערווער.

הקדמה צום דעמא: אן אייגענער אימעדזש אויף Ubuntu 20 (40:08:49)

צום סוף פון דעם אפשניט לייגט דער קורס פאר א דעמא: שאפן א דאקער-אימעדזש מיט א Dockerfile אויף Ubuntu 20. מען ניצט Ubuntu 20 אלס באזע-אפעריר-סיסטעם, און דערויף בויט מען אן אייגענעם דאקער-אימעדזש.

דעמא: Docker אויף אן Ubuntu סערווער (40:09:21)

צוגרייטן דעם סערווער און אינסטאלירן Docker (40:09:21)

אין דעם טייל פונעם קורס ווערט געוויזן א פראקטישע דעמא: מען פארבינדט זיך צו א ווירטועלער מאשין און מען גרייט צו א Docker אימאזש. דער ציל איז צו בויען אן אייגענעם (custom) Docker אימאזש. דער באזע-אימאזש (base image) איז דער אפעריר-סיסטעם Ubuntu.

עס ווערט גענומען Ubuntu ווערסיע 20. אין דער לעקציע ווערט דאס באצייכנט אלס די לעצטע ווערסיע פון Ubuntu. מען האט שוין א סערווער צוגעגרייט, און מען גייט אריבער דעם סדר פון טריט:

- ערשטנס אינסטאלירט מען דעם פעקל docker.io, וואס שטעלט אריין Docker אויפן סערווער.
- דערנאך מאכט מען enable דעם סערוויס, און מען פארזיכערט זיך אז דער Docker סערוויס לויפט. אויב דער סערוויס איז אראפ, קען מען גארנישט טון.
- אלס פראבע לויפט מען docker run hello-world. דער באפעל ציט אראפ א Docker אימאזש פון Docker Hub און פירט אויס א קליינע ארבעט. ווען דאס גייט דורך, ווייסט מען אז דער קאנטעינער לויפט און אלעס ארבעט.
- ווער עס ווייל, קען אויך פרובירן דעם Ubuntu Docker אימאזש.

די הויפט-דעמא איז אבער צו בויען אן אייגענעם Docker אימאזש. אויף א באזע פון Ubuntu אינסטאלירט מען PHP, Nginx און נאך קאמפאנענטן. דאס רעזולטאט איז א וועבזייטל געבויט אויף PHP, וואס ווערט געהאסט אויף א וועב-סערווער (עס ווערט דערמאנט Apache, אבער דער סערווער איז Nginx). צום סוף קען מען צוקומען צו דער אפליקאציע.

דער ערשטער טריט פאר דעם אייגענעם אימאזש: מען שאפט א פאלדער, למשל מיטן נאמען nginx-image. אין אים שאפט מען א טעקע מיטן נאמען Dockerfile, און דעם אינהאלט נעמט מען פון א צוגעגרייטן מוסטער.

אינסטאלירן Docker אויפן סערוועד, טריט נאך טריט (40:10:54)

מען גייט צום סערוועד און מען הייבט אן מיט אינסטאלירן Docker. ערשט נאך דעם קען מען ווייטער לערנען וואס די פלאטפארמע איז און ווי מען גרייט צו פארשידענע סארטן קאנטעינערס. דער סערוועד איז אן Ubuntu סערוועד.

- מען לויפט ערשט דעם באפעל apt update. ער דערהיינטיקט לאקאל די ליסטע פון פעקלעך מיט אלע לינקס. דער צוועק: מען זאל זיך קענען פארבינדן צו די Ubuntu אדרעסן און אלעמאל אראפנעמען די לעצטע פעקלעך.
- דערנאך לויפט מען apt install docker.io. דער סיסטעם הייבט אן אראפנעמען פעקלעך און פרעגט אויף באשטעטיגונג. מען ענטפערט "yes". דאן ווערן די פעקלעך אראפגענומען איינס נאכן אנדערן, און די אינסטאלאציע גייט פאר זיך.
- ווען די אינסטאלאציע ענדיגט זיך, קאנטראלירט מען צי Docker איז אינסטאלירט. מען קען לויפן docker version, אדער מען באניצט זיך מיטן service באפעל צו זען צי דער Docker סערוויס לויפט.
- אויב דער סערוויס לויפט נישט, מוז מען אים מאכן enable און אים אנהייבן (start).

דער קורס באטאנט: ערשט נאך דער אינסטאלאציע פון Docker קען מען גיין ווייטער צו שאפן קאנטעינערס און אימאזשעס. אויב Docker איז נישט אינסטאלירט, וועט מען אנשטויסן אין פראבלעמען. דערפאר דארף מען זיך פארזיכערן, נאך פאר סיי וועלכע אנדערע ארבעט, אז Docker איז אינגאנצן אויפגעשטעלט און לויפט. דאס אראפנעמען פון די פעקלעך נעמט א שטיק צייט, און מען ווארט ביז עס איז פארטיג.

קאנטראלירן דעם סערוויס און פרובירן hello-world (40:13:28)

נאך דער אינסטאלאציע לויפט מען דעם באפעל:

```
service docker status
```

דער באפעל ווייזט צי דער Docker סערוויס איז פאראן. אויב מען באקומט נישט קיין ריכטיגן ענטפער, הייבט מען אן דעם סערוויס. אין דער דעמא זעט מען אז פריער איז דער סערוויס געווען inactive , און יעצט איז ער active. מען זעט דעם פראצעס dockerd לויפן, וואס הייבט אן דעם Docker daemon. דער Docker daemon לויפט, און מען קען גיין ווייטער.

דערנאך לויפט מען:

```
docker run hello-world
```

דאס איז א פראבע צו זען צי דער Docker אינסטאנץ ארבעט ווי געהעריג. ווען עס איז דא א פראבלעם מיטן Docker daemon, אדער ער לויפט נישט ריכטיג, באקומט מען א פעלער-מעלדונג. דא איז נישט ארויסגעקומען קיין פעלער. דאס ווייזט אז דער Docker אימאזש לויפט און גיט צוריק דעם ריכטיגן ענטפער. מען זעט אז דער אימאזש איז אראפגעצויגן געווארן אויפן סיסטעם, אויסגעפירט געווארן, און האט געגעבן אן ענטפער.

לויפן אן Ubuntu קאנטעינער מיט bash (40:14:28)

דער קומענדיגער טריט איז צו לויפן אן Ubuntu קאנטעינער, מיט א באפעל ווי:

```
it ubuntu bash- docker run
```

דער באפעל ציט אראפ דעם Ubuntu Docker אימאזש און לויפט דערויף דעם פראגראם bash. דער צוועק: וועט מען וויל טון א פראצעס אדער אן ארבעט, זאל מען זיך קענען פארבינדן אינעווייניג אין קאנטעינער.

ווי עס ארבעט:

- אויב דער אימאזש איז שוין פאראן לאקאל, שאפט Docker גלייך דעם קאנטעינער.
- אויב נישט, ציט Docker פריער אראפ דעם ריכטיגן אימאזש. אין דער דעמא זעט מען ווי דאס געשעט.
- נאכדעם באקומט מען א טערמינאל. מען זעט "root@" מיט אן אידענטיפיקאציע-נומער (ID). דער ID איז דער ID פונעם קאנטעינער, דאס הייסט מען געפינט זיך אינעווייניג אין Docker קאנטעינער.
- מיטן באפעל exit גייט מען ארויס פונעם קאנטעינער.
- מיטן באפעל a- docker ps זעט מען אלע קאנטעינערס וואס מען האט ביז יעצט געלאפן.

אין דער ליסטע זעט מען ביידע קאנטעינערס, hello-world און Ubuntu, און ביידע זענען אפגעשטעלט. אין דער סטאטוס-שפאלט שטייט (0) Exited, ווייל ביידע זענען געלאפן אין אן אינטעראקטיוון מאדוס, און וועט זיי האבן געענדיגט האבן זיי זיך אפגעשטעלט.

וועט עס וויל אז א קאנטעינער זאל ווייטער לויפן, דארף אים לויפן אין detached מאדוס (מיט "-d"), אזוי אז ער זאל אלעמאל לויפן. דערפאר דארף מען אבער שאפן אן אייגענעם Docker אימאזש און אים געהעריג אויפשטעלן.

דער אינהאלט פונעם Dockerfile (40:16:29)

מען שאפט א פאלדער מיטן נאמען) nginx-imageמען קען געבן סיי וועלכן נאמען). אין אים לייגט מען א טעקע מיטן נאמען Dockerfile. וואס עס שטייט אין דער טעקע, לויט דעם וועלט דער Docker אימאזש צוגעגרייט. דער קורס גייט איבער דעם אינהאלט חלק נאך חלק:

- באזע-אימאזש: ערשט באשטימט מען דעם באזע-אימאזש, Ubuntu 20.04.
- LABEL: מען שרייבט לעיבלס מיט אינפארמאציע, למשל די ווערסיע, וועט עס איז דער maintainer, און א באשרייבונג. מען קען זיי צופאסן. דער צוועק: מען זאל וויסן וועלכן פראגראמירער מען דארף זיך וענדן, וועט עס איז דא א פראבלעם מיטן Dockerfile אדער מיטן build סקריפט.
- DEBIAN_FRONTEND=noninteractive: מען שטעלט אויס דעם פאראמעטער. א build פראצעס אין א Dockerfile קען נישט לויפן אין אינטעראקטיוון סעסיע, ער מוז לויפן non-interactive.
- RUN: מיט RUN לויפט מען באפעלן ווי apt update, און מען אינסטאלירט PHP, Nginx און Supervisor. די באפעלן ווערן אויסגעפירט בעת דעם build פראצעס.
- ENV: מען שטעלט אויס סביבה-ווריאבלען (environment variables): nginx_vhost, php_conf, nginx_conf און supervisor_conf.
- COPY: מען טראנספערירט טעקעס פונעם Linux סערווער אריין אין קאנטעינער. למשל די Nginx קאנפיגוראציע מיט דער virtual host טעקע, און אויך די Supervisor קאנפיגוראציע-טעקע. די ווריאבלען ווי supervisor_conf זענען שוין פריער באשטימט, און זיי זאגן וועלכע קאנפיגוראציע מען וויל.
- פאלדערס: מען שאפט די פאלדער-סטרוקטור /var/www/html, כדי מען זאל דארט קענען ארויפלייגן אינהאלט.

- VOLUME: מען שטעלט אויס volumes, כדי די געוויסע דירעקטאריעס זאלן ווערן mounted אויף דער host מאשין (דעם host סערווער). אזוי קען מען זיי אויך באניצן לאקאל.
- start.sh: דער סקריפט start.sh גייט אריין אין דעם CMD באפעל, וואס ווערט אויסגעפירט ווען דער קאנטעינער לויפט. ווען מען וויל לויפן דעם Nginx פראצעס אדער דעם Docker קאנטעינער, מוז אינעווייניג לויפן א פראצעס. נאר ווען א פראצעס לויפט אן אויפהער, בלייבט דער קאנטעינער לעבעדיג.

צונויפשטעלן דעם גאנצן Dockerfile (40:20:02)

מען קאפירט דעם גאנצן אינהאלט. ער שטעלט מיט זיך פאר דעם קאמפלעטן Dockerfile, ביז צום EXPOSE חלק. ביי EXPOSE עפנט מען די פארטן 80 און 443.

נאך דעם Dockerfile דארף מען שאפן נאך טעקעס: די virtual host טעקע, די Supervisor (supervisord) טעקע, און די start.sh טעקע. מען גייט צוריק צום סערווער צו ארבעטן ווייטער.

א אייגענער Docker אימעדזש פאר Nginx מיט PHP (40:20:34)

די הויפט-טעקע (Dockerfile) און די צוגעהעריגע טעקעס (40:20:34)

אין דער לעקציע ווערט ווייטער געוויזן ווי אזוי מען גרייט צו א אייגענעם Docker אימעדזש פאר א וועב-סערווער מיט Nginx און PHP. דער קורס באמערקט אז מען קען שרייבן די טעקעס מיט וועלכן עדיטאר מען וויל, עס איז נישטא קיין באגרענעצונג.

מען זעט ווי די הויפט-טעקע, דער Dockerfile, זעט אויס. אין איר זענען דא עטליכע RUN באפעלן וואס לויפן ביים בויען דעם אימעדזש. נאכדעם קומען environment variables, און נאך דעם נאך variables וואס דינען בלויז צו באשטימען די פאדן (paths) צו די טעקעס און פאלדערס.

עס ווערט ערקלערט אז אינעם Dockerfile קאפירט מען דריי טעקעס פונעם host, דאס הייסט פונעם סערווער, אריין אינעם Docker קאנטעינער:

- די טעקע default
- די טעקע supervisord.conf
- די טעקע start.sh

די דריי טעקעס דארף מען צוגרייטן אין דעם זעלבן איצטיגן דירעקטאריע ווו דער Dockerfile געפינט זיך.

די טעקע default: די virtual host קאנפיגוראציע (40:21:04)

ערשט שאפט מען אינעם איצטיגן דירעקטאריע א טעקע מיטן נאמען default. אין איר שטייט די default virtual host קאנפיגוראציע פאר Nginx. דארט באשטימט מען דעם server אטריבוט, וואס זאגט דעם סערווער אז ער זאל זוכן טעקעס ווי index.html אדער index.php אינעם דירעקטאריע /var/www/html.

אין דער לעקציע ווערט דער אינהאלט אריינקאפירט אין דער טעקע און די טעקע ווערט געשפייכלט.

די טעקע supervisord.conf און די נעמען (40:21:34)

דערנאך קאפירט מען דעם אינהאלט פאר דער צווייטער טעקע און מען שאפט א טעקע מיטן נאמען supervisord.conf.

דער קורס וואלט אז מען דארף נאכקוקן אז די נעמען פון די טעקעס שטימען, ווייל דער Dockerfile רופט זיך אן אויף די דאזיקע נעמען. אויב א נאמען איז אנדערש, וועט דער Dockerfile די טעקע נישט געפינען. מען זעט ווי מען זוכט אינעם Dockerfile מיטן grep באפעל און מען באשטעטיגט אז דער נאמען supervisord שטימט. אזוי שטייט שוין גרייט default און supervisord.conf.

די טעקע start.sh און די execute ערלויבעניש (40:22:44)

די דריטע טעקע איז start.sh. אין איר באשטימט מען ווי אזוי דער פראצעס פון Nginx, דעם וועב-סערווער, זאל זיך אנהייבן. עס ווערט ערקלערט אז דא לאזט מען לויפן דעם supervisord פראצעס צוואמען מיט זיין קאנפיגוראציע-טעקע. מען קאפירט אריין דעם אינהאלט און אזוי איז די supervisor קאנפיגוראציע אויך גרייט.

דערנאך דארף מען געבן start.sh א execute ערלויבעניש. דער קורס ערקלערט פארוואס: אינעם Dockerfile לאזט מען לויפן די טעקע מיט /.start.sh , און אויף אזא אופן קען מען א טעקע נאר לויפן ווען זי האט א execute ערלויבעניש. דערפאר מוז מען טוישן אירע ערלויבענישן מיטן chmod באפעל:

x start.sh+ chmod

אצינד זענען אלע טיילן גרייט: די באפעלן, דער Dockerfile, supervisord.conf, default און start.sh. עס בלייבט נאר דער build פראצעס.

דער build פראצעס (40:23:47)

צו בויען דעם אימעדזש ניצט מען דעם באפעל docker build. דער נאמען פונעם אימעדזש אין דער לעקציע איז nginx-image. דער קורס באטאנט אז מען קען געבן וועלכן נאמען מען וויל:

. t nginx-image- docker build

נאכן בויען קען מען אויסרעכענען די אימעדזשעס און מען זעט דעם נייעם אייגענעם Docker אימעדזש. מען זעט אויך דעם Ubuntu אימעדזש, וואס איז דער base image אויף וועלכן מען בויט, און דערנעבן דעם צוגעגרייטן nginx-image. נאכדעם שאפט מען א webroot פאלדער און מען לאזט לויפן דעם docker run באפעל צו זען אויב מען קען זיך פאקטיש פארבינדן.

ווען מען לאזט לויפן דעם באפעל, נעמט Docker דעם Dockerfile פונעם איצטיגן דירעקטאריע און הייבט אן דורכפירן די ארבעטן: די פעקלעך (packages) ווערן אינסטאלירט און די קאנפיגוראציעס ווערן געמאכט.

פארוואס אלעס דארף גיין אן באשטעטיגונגען (40:24:48)

דער קורס באטאנט א וויכטיגן פרט: אלע אינסטאלאציעס גייען אין א non-interactive מאדע. דער באניצער ווערט קיינמאל נישט געפרעגט פאר א באשטעטיגונג, ווייל אינעם build קען מען נישט אריינגעבן באשטעטיגונגען. דערפאר מוזן די באפעלן געשריבן ווערן אזוי אז זיי זאלן גארנישט פארלאנגען א באשטעטיגונג פונעם באניצער. דער טעם: מארגן קען מען דאס אריינשטעלן אין Jenkins, און דארט וועט קיינער נישט זיצן צו אריינשרייבן באפעלן אדער באשטעטיגונגען בעת דער פראצעס לויפט.

מען זעט ווי די PHP פעקלעך ווערן אינסטאלירט, דערונטער די FPM פעקלעך.

דער אימעדזש איז גרייט (40:25:18)

אין סוף קומט די מעלדונג "Successfully tagged". עס ווערט ערקלערט אז דאס באווייזט אז דער Docker אימעדזש איז אין גאנצן אן פראבלעמען.

צו זען דעם אימעדזש קען מען לאזן לויפן איינעם פון די צוויי באפעלן: docker images אדער docker image ls. די גרייס פונעם אימעדזש איז 233.7 MB, און דאס ווייזט אז דער Docker אימעדזש איז גרייט צו לויפן.

א volume פאר די וועב-בלעטער (40:25:48)

דער נעקסטער שריט איז צו פארבינדן א volume מיטן דירעקטאריע /var/www/html/ בשעת דער קאנטיינער לויפט, ווייל מען וויל אריינשטעלן וועב-בלעטער.

דער קורס ערקלערט די עצה דערפון: מען קען אויך אריינשטעלן די וועב-בלעטער אלס טייל פונעם build, למשל דורך א git clone פון א repository אינעם Dockerfile. אבער דא וויל מען דאס נישט טון בשעתן צוגרייטן דעם אימעדזש. מען האט איצט בלויז א סביבה ווו Nginx איז ריכטיג אויפגעשטעלט און לויפט, און דערנאך באשליסט מען אליין ווי אזוי צו שטעלן דעם אינהאלט.

דערפאר שאפט מען א פאלדער אויפן סערווער און מען טיילט אים מיטן Docker קאנטיינער. ווען מען לייגט אריין HTML אדער PHP טעקעס אין דעם לאקאלן פאלדער, זאל מען קענען זען א וועב-בלאט וואס לויפט.

אין דער לעקציע שאפט מען דעם פאלדער /webroot . די טעקעס וואס מען לייגט אריין אין אים ווערן מיט א Docker volume איבערגעפירט אינעווייניג אינעם קאנטיינער.

דער docker run באפעל (40:26:49)

דער קורס ערקלערט די טיילן פונעם docker run באפעל:

- דער פארט: אין דער לעקציע ניצט מען פארט 80. מען קען אויך נעמען 8080 אדער וועלכן פארט מען וויל.
- דער נאמען פונעם קאנטיינער איז test-container. דעם נאמען קען מען אויך טוישן.
- דער אימעדזש איז nginx-image. וועט עס ניצט אן אנדער אימעדזש, קען אים דא אריינשטעלן.
- /webroot ווערט פארבונדן מיט /var/www/html. אזוי ווערט יעדע טעקע וואס מען לייגט אריין אין webroot באלד געזען אין /var/www/html.

דער באפעל זעט בערך אויס אזוי:

webroot:/var/www/html nginx-image/ v- name test-container-- p 80:80- d- docker run

מען קאפירט דעם באפעל און מען לאזט אים לויפן. דער קורס ווארנט: אויב דער פארט ווערט שוין געניצט, מוז מען זארגן אז דאס וואס ניצט אים זאל אפגעשטעלט ווערן. מען זאל נאכקוקן אז דער פארט וואס מען ניצט איז פריי.

נאכקוקן אז דער קאנטיינער לויפט (40:27:50)

מיטן באפעל docker ps זעט מען די קאנטיינערס וואס לויפן. מיט docker ps -a זעט מען סיי די וואס לויפן און סיי די וואס זענען אפגעשטעלט. מען ניצט docker ps, און מען זעט אז דער קאנטיינער test-container לויפט אויף פארט 80.

צו פרובירן קען מען לאזן לויפן curl localhost:80. מען קען אויך נעמען דעם public IP פונעם סערווער, אים אריינשטעלן אינעם בראוזער און זען דעם אינהאלט דארט. די ענטפער איז אצינד א 403 Forbidden פעלער. דער קורס ערקלערט אז דאס איז אין ארדענונג, ווייל עס זענען נאך נישטא קיין וועב-בלעטער אין /var/www/html, דאס הייסט אין /webroot.

אריינלייגן אינהאלט: HTML (40:28:51)

צו לייגן אינהאלט אין די טעקעס ניצט מען echo באפעלן. מען קען אויך שרייבן די טעקעס מיט vi. מען לאזט לויפן דעם echo באפעל, און אין /webroot ווערט באשאפן א טעקע index.html. וועט עס ווייל, קען שאפן אן אנדער בלאט.

וועט מען לאזט ווידער לויפן curl, זעט מען דעם אינהאלט: דאס וואס מען שטעלט אריין אין index.html קומט ארויס וועט מען לאזט לויפן curl.

אריינלייגן אינהאלט: PHP (40:29:53)

דערנאך שטעלט מען אריין PHP אינהאלט אנשטאט נאר סטאטישן HTML. מען שאפט א טעקע info.php, און אזוי האט /webroot אצינד צוויי טעקעס.

דער קורס דערמאנט אז index.html קען האבן וועלכן אינהאלט מען וויל. מען קען אויך נעמען א גאנצע וועבזייטל: מען קען טון א git clone אין /webroot, און די גאנצע אפליקאציע וועט אנהייבן לויפן.

צו זען info.php מוז מען טוישן דעם URL און אריינשטעלן דעם נאמען פון דער טעקע. דער טעם: index.html איז דער default בלאט וואס ווערט געוויזן אן א נאמען. וועט עס וויל אן אנדער default נאמען, קען טוישן די קאנפיגוראציע פונעם Nginx סערווער, און לויט דעם וועט עס ארבעטן.

וועט מען לאזט לויפן curl מיטן נייעם URL, קומט ארויס דער PHP רעזולטאט: א פולשטענדיגע אינפארמאציע. עס ווערט ערקלערט אז אינעם טערמינאל זעט דאס אויס ווי א גרויסער שוועכער טעקסט, אבער אין א וועב-בראוזער וועט עס זיין א פארשטענדליכע אינפארמאציע, למשל אין טאבעלעס. מען שקראלט ארויף צום אנהייב, און מען זעט אז דער רעזולטאט באווייזט אז דער PHP אינהאלט ארבעט.

סיכום פון דער דעמאנסטראציע (40:30:54)

אזוי האט מען א קאנטעינער וואס לויפט. דער קורס באטאנט אז מען קען דאס טוישן ווי מען וויל. מען קען צוגרייטן א Docker אימעדזש מיטן קאוד שוין אריינגעבויט, למשל מיט א git clone. מען קען אויך האלטן די קאנפיגוראציע און דעם source קאוד אין א git repository און צוגרייטן פון דעם א פולשטענדיגן פעקל.

דאס איז גענוען די דעמאנסטראציע ווי אזוי צו שאפן א אייגענעם Docker אימעדזש מיט א Dockerfile, און ווי אזוי צו לאזן לויפן א קאנטעינער פון אים.

אין סוף פון דעם אפשניט הייבט זיך אן א נייע טעמע: איידער מען לערנט זיך LLMs, ערקלערט דער קורס ערשט דעם פראבלעם וואס זיי לייזן. וועט ארגאניזאציעס בויען אפליקאציעס מיט גרויסע שפראך-מאדעלן, טרעפן זיי אן אויף שווערקייטן, און דער פארזעץ פון דער טעמע קומט אינעם קומענדיגן אפשניט.

LLMOps: פאַרוואלטן LLM-אפליקאציעס פון אנהייב ביז סוף (40:31:54)

וואס איז LLMOps, און ווי עס פארגלייכט זיך מיט DevOps און MLOps (40:31:54)

דער קורס ערקלערט אז ארבעטן מיט א גרויסן שפראך-מאדעל (LLM) מיינט נישט פשוט ארויפשטעלן א מאדעל און דערמיט ענדיגן. עס זענען דא אסאך באוועגלעכע טיילן: דאטא, פראמפטס, APIs, די אינטעראקציע מיט'ן באניצער, אפשאצונג (evaluation), סיסטעם-זיכערהייט, קאנטראל און אן אנגייענדיגע פארבעסערונג. LLMOps איז די דיסציפלין וואס פארוואלט אלע די טיילן איבער'ן גאנצן לעבנס-ציקל פון אן אפליקאציע וואס ארבעט מיט אן LLM. זי קאנצענטרירט זיך אויף אנטוויקלען, טעסטן, ארויסשטעלן, אויפפאסן און פארבעסערן אפליקאציעס וואס באנוצן זיך מיט גרויסע שפראך-מאדעלן.

עס ווערט געמאכט א פארגלייך מיט צוויי אנדערע וויכטיגע באגריפן:

- DevOps – פארבעסערט דעם פראצעס פון אנטוויקלען און ארויסגעבן ווייכווארג. עס העלפט פראגראמירער בויען אפליקאציעס שנעלער, טעסטן און ארויסגעבן ווייכווארג אויף אן עפעקטיווע אופן.
- MLOps – פארברייטערט די זעלבע געדאנקען אויף מאשין-לערנען: פארוואלטן מאדעלן, טריינירונג-פייפליינס, דאטא-סעטס און ארויסשטעלונג.
- LLMOps – גייט נאך א טריט ווייטער און באהאנדלט די ספעציעלע שוועריגקייטן פון גרויסע שפראך-מאדעלן. עס פארוואלט נישט נאר דעם מאדעל אליין, נאר אויך די פראמפטס, דעם קאנטעקסט, די סיסטעמען וואס ברענגען דאטא, באניצער-אפרוף, די קוואליטעט פון די ענטפערס און AI-זיכערהייט.

א פשוטע וועג עס צו פארשטיין: DevOps פארוואלט ווייכווארג, MLOps פארוואלט מאשין-לערנען, און LLMOps פארוואלט אפליקאציעס וואס זענען געטריבן דורך גרויסע שפראך-מאדעלן.

פארוואס מען דארף LLMOps (40:32:56)

פון ערשטן בליק קען אויסזען אז בויען אן אפליקאציע מיט אן LLM איז גאר פשוט: א פראגראמירער קען פארבינדן אן API, שיקן א פראמפט, באקומען אן ענטפער, און אין גאנץ קורצע צייט האבן אן ארבעטנדיגן פראטאטיפ. אבער אריבערגיין פון א פראטאטיפ צו אן אפליקאציע אויף פראדוקציע-שטאפל ברענגט מיט זיך אסאך שוועריגקייטן.

טראדיציאנעלע מאשין-לערנען סיסטעמען האבן שוין אליין שוועריגקייטן, ווי ארויסשטעלן דעם מאדעל, פארוואלטן דאטא, אויפפאסן אויף דער פערפארמענס, און באהאנדלען ענדערונגען אין די מוסטערן פון דער דאטא. LLM-אפליקאציעס ברענגען נאך מער קאמפליצירטקייט:

- קוואליטעט פון די ענטפערס – זי הענגט שטארק אפ פונעם פראמפט. אפילו א קליינע ענדערונג אין די אנווייזונגען קען באדייטנד טוישן דעם ענטפער. דעריבער דארפן פירמעס א סיסטעם צו פארוואלטן, טעסטן און פארגלייכן פארשידענע ווערסיעס פון פראמפטס.
- "האלוצינאציע" – LLMs קענען אמאל געבן ענטפערס וואס הערן זיך ריכטיג אן, אבער זענען אין פאקט פאלש. אין ביזנעס-אפליקאציעס קען פאלשע אינפארמאציע שאפן ערנסטע פראבלעמען. LLMOps העלפט פירמעס אפשאצן און אויפפאסן אויף דער קוואליטעט פון די ענטפערס.
- קאסטן – יעדע אינטעראקציע מיט אן LLM פארברויכט קאמפיוטער-רעסורסן און טאקענס. ווען אן אפליקאציע וואקסט צו טויזנטער באניצער, ווערט פארוואלטן די קאסטן גאר קריטיש.

• זיכערהייט און פריוואטקייט – פירמעס מוזן זיכער מאכן אז סענסיטיווע אינפארמאציע איז באשיצט, און אז באניצער קענען נישט מאניפולירן דעם מאדעל דורך טעכניקן ווי prompt injection און אנדערע אטאקעס.

צוליב די אלע שוועריגקייטן דארפן פירמעס א סטרוקטורירטן צוגאנג צו פארוואלטן LLM-אפליקאציעס.

ווי אזוי ארבעט אן LLM-אפליקאציע: די שיכטן (40:34:23)

כדי בעסער צו פארשטיין LLMOps, ערקלערט דער קורס קודם ווי א טיפישע LLM-אפליקאציע ארבעט. וועג א באניצער רעדט מיט אן AI-אפליקאציע, גייט דער פראצעס געווענליך דורך עטליכע שיכטן:

• די באניצער-שניטשטעל – דארט שרייבט דער באניצער אריין א פראגע אדער א בקשה. דאס קען זיין א chatbot, א מאביל-אפליקאציע אדער א סיסטעם פאר גרויסע פירמעס.

• די אפליקאציע-שיכט – פארוואלט די ביזנעס-לאגיק און באשליסט ווי אזוי די בקשה זאל באהאנדלט ווערן. לויט'ן באדערפענישׂ קען די אפליקאציע זיך באנוצן מיט טעכנאלאגיעס ווי prompt engineering, RAG אדער AI-אגענטן.

• דער LLM אליין – דער מאדעל באקומט די אנווייזונגען צוזאמען מיט'ן שייכותדיגן קאנטעקסט און שאפט אן ענטפער. דאס קען זיין א מאדעל פאר אלגעמיינע צוועקן ווי GPT, Claude, Gemini, אדער אן אפן-קאוד מאדעל ווי Llama.

אבער אין עכטע אפליקאציעס שיקן פירמעס געווענליך נישט יעדע פראגע גלייך צום מאדעל; זיי פארבינדן דעם מאדעל מיט נאך דאטא. למשל, אן אנגעשטעלטער וואס פרעגט וועגן די פירמע'ס כללים, דארף מען אפשר אינפארמאציע פון אינערליכע דאקומענטן. דער סיסטעם זוכט קודם די שייכותדיגע אינפארמאציע, און דערנאך גיט ער דעם קאנטעקסט צום LLM.

צום סוף, נאכדעם וואס דער ענטפער איז געשאפן, דארף מען אויפפאסן (monitoring): מען פאלגט נאך וויכטיגע פאקטארן ווי די קוואליטעט פונעם ענטפער, ווי לאנג עס נעמט (latency), באניצער-אפרוף און קאסטן. LLMOps פארוואלט יעדן שטאפל פון דעם פראצעס, און מאכט זיכער אז דער גאנצער סיסטעם ארבעט פארלעסלעך – פון דער באניצער'ס פראגע ביזן לעצטן ענטפער.

די קאמפאנענטן פון אן LLM-אפליקאציע (40:35:54)

אן אפליקאציע איז נישט בלויז א גרויסער שפראך-מאדעל; עס איז אן "עקאסיסטעם" פון טיילן וואס ארבעטן צוזאמען:

• דער יסוד-מאדעל (foundational model) – גרויסע פאר-טריינירטע מאדעלן געשאפן דורך פירמעס ווי OpenAI, Anthropic, Google, Meta און אנדערע AI-פארשונג פירמעס. זיי געבן די הויפט-"שכל" וואס מען דארף צו שאפן טעקסט, פארשטיין אינפארמאציע, שרייבן קאוד, ענטפערן פראגעס און דורכפירן אויפגאבעס פון לאגישן טראכטן.

• די פראמפט-שיכט – פראמפטס זענען אנווייזונגען וואס פירן דאס באנעמען פונעם מאדעל. א גוט-אויסגעטראכטער פראמפט קען פארבעסערן די געניקייט, די קאנסיסטענץ און די נוצלעכקייט. אין א פראדוקציע-סיסטעם מוזן פירמעס פארוואלטן פראמפטס פארזיכטיג, נאכפאלגן די ענדערונגען, און אפשאצן וועלכע פראמפטס געבן בעסערע רעזולטאטן.

• די דאטא-שיכט – אסאך פירמע-אפליקאציעס דארפן צוטריט צו פריוואטע אדער ספעציפישע אינפארמאציע פון זייער פאך. די דאטא קען אנטהאלטן דאקומענטן, דאטא-באזעס, שמועסן מיט קונים

אדער אינערליכע וויסן-קוועלער. וועקטאר-דאטא-באזעס און embedding-סיסטעמען העלפן אפהיטן און ארויסנעמען די אינפארמאציע אויף אן עפעקטיווע אופן.

• די אפליקאציע-שיכט – דארט רעדט דער באניצער מיט'ן AI-סיסטעם. ביישפילן: קונה-סערוויס, chatbots, AI-געהילפן, קאוד-געצייג און זוך-אפליקאציעס פאר פירמעס.

• די אויפפאסונג-שיכט (monitoring) – וויבאלד LLM-ענטפערס זענען נישט אלעמאל פאראויסצוזען, דארפן פירמעס א סיסטעם צו מעסטן די פערפארמענס, אנטדעקן פראבלעמען, אויפפאסן אויף קאסטן און כסדר פארבעסערן די אפליקאציע.

צוזאמען שאפן די קאמפאנענטן א פולשטענדיגע LLM-אפליקאציע, און LLMOps גיט די ראם צו פארוואלטן זיי אלע עפעקטיוו.

דער LLMOps לעבנס-ציקל (40:37:26)

דער קורס באטראכט דערנאך דעם גאנצן לעבנס-ציקל פון בויען און פארוואלטן אזעלכע סיסטעמען, וואס הייסט דער LLMOps לעבנס-ציקל:

• דאטא-צוגרייטונג – דאטא איז איינע פון די וויכטיגסטע טיילן פון יעדער AI-אפליקאציע, ווייל די קוואליטעט פון דער אינפארמאציע ווירקט דירעקט אויף דער קוואליטעט פונעם רעזולטאט. פירמעס זאמלען דאטא פון פארשידענע קוועלער – דאקומענטן, דאטא-באזעס, אינטעראקציעס מיט קונים, וועבזייטלעך און אינערליכע וויסן-סיסטעמען. די דאטא דארף מען רייניגן, סטרוקטורירן און צוגרייטן אריידער די LLM-אפליקאציע קען זיך דערמיט באנוצן.

• אויסוואל פונעם מאדעל – מען דארף אויסקלייבן דעם ריכטיגן מאדעל לויט די באדערפענישן. פארשידענע מאדעלן האבן אנדערע מעגלעכקייטן, קאסטן, שנעלקייט פון ענטפער און שטאפל פון געניקייט. למשל, א פשוטער chatbot קען גוט ארבעטן מיט א קלענערן און שנעלן מאדעל, בשעת א קאמפליצירטע אפליקאציע פון לאגישן טראכטן קען פאדערן א שטארקערן מאדעל.

• prompt engineering – וויבאלד LLMs ענטפערן לויט אנווייזונגען, ווערט אויסטראכטן עפעקטיווע פראמפטס א וויכטיגער טייל פון דער אנטוויקלונג. גרופעס עקספערימענטירן מיט פארשידענע פראמפטס, טעסטן די רעזולטאטן און פארבעסערן די אנווייזונגען.

• טעסטן און אפשאצן – אנדערש ווי ביי טראדיציאנעלע וויכווארג, וואו מען קען לייכט באשטימען וואס דער ריכטיגער רעזולטאט דארף זיין, איז אפשאצן LLM-אפליקאציעס שוועריר, ווייל די ענטפערס קענען זיך טוישן. גרופעס באנוצן זיך מיט אויטאמאטישע אפשאצונג-מעטאדן, מענטשלעכן אפרוף און קוואליטעט-מעסטונגען.

• ארויסשטעלונג (deployment) – וועו די אפליקאציע גייט דורך די טעסטן, ווערט זי צוגענגלעך פאר באניצער דורך APIs, וועבזייטלעך, מאביל-אפליקאציעס אדער פלאטפארמעס פאר פירמעס.

• אנגייענדיגע אויפפאסונג – נאך דער ארויסשטעלונג פאלגט מען נאך די קוואליטעט פון די ענטפערס, latency, קאסטן, באניצער-צופרידנקייט און זיכערהייט-פראבלעמען, און לויט דעם אפרוף פארבעסערט מען כסדר די אפליקאציע.

דער וויכטיגער פונקט צו געדענקען: LLMOps איז נישט קיין איינמאליגער פראצעס; עס איז אן אנגייענדיגער ציקל וואו אפליקאציעס ווערן כסדר אפגעשאצט, פארבעסערט און אפטימירט.

דאטא-פארוואלטונג אין LLMOps (40:39:30)

דאטא-פארוואלטונג שפילט א קריטישע ראלע, ווייל גרויסע שפראך-מאדעלן זענען נאר אזוי נוצלעך ווי די אינפארמאציע צו וועלכער זיי האבן צוטריט. אלגעמיינע LLMs זענען טריינירט אויף ריזיגע מאסן עפנטלעך-צוגענגלעכע דאטא, אבער אסאך פירמעס דארפן AI-סיסטעמען וואס פארשטייען זייער אייגענע פריוואטע און ספעציאליזירטע אינפארמאציע. ביישפילן: א באנק וויל אן AI-געהילף וואס פארשטייט אירע אינערליכע כללים; א געזונטהייט-ארגאניזאציע דארף א געהילף וואס קען אנאליזירן מעדיצינישע דאקומענטן; און א פירמע וויל אז אירע אנגעשטעלטע זאלן קענען זוכן אינערליכע וויסן.

- דער ערשטער שריט איז זאמלען דאטא – פון דאקומענטן, וועבזייטלעך, דאטא-באזעס, קונה-סערוויס שמועסן און פירמע-סיסטעמען.
- די גענאמלטע דאטא קען מען געווענליך נישט גלייך געבן פאר'ן LLM; זי דארף דורכגיין א צוגרייטונג: רייניגן איבריגע אינפארמאציע, אראפנעמען דאפלטן אינהאלט, פארריכטן פארמאט-פראבלעמען, און ארגאניזירן די דאטא אין באדייטפולע אפטיילונגען.
- document chunking – א וויכטיגע טעכניק: גרויסע דאקומענטן וועלן צעטיילט אין קלענערע שטיקער, ווייל LLMs האבן א באגרענעצונג וויפיל אינפארמאציע זיי קענען פראצעסירן מיט אמאל. די קלענערע שטיקער זענען לייכטער צו זוכן און צו געבן אלס קאנטעקסט פאר'ן מאדעל.
- embeddings – זיי פארוואנדלען טעקסט אין נומעריש פארשטעלונגען וואס כאפן דעם באדייט. דאס דערלויבט דעם סיסטעם צו זוכן לויט'ן באגריף און נישט נאר לויט געוויע שליסל-ווערטער. די embeddings ווערן אפגעהיט אין וועקטאר-דאטא-באזעס ווי Pinecone אדער ChromaDB. וועו א באניצער פרעגט א פראגע, זוכט דער סיסטעם אין דער וועקטאר-דאטא-באזע די מערסט שייכותדיגע אינפארמאציע און גיט זי צום LLM.

עפעקטיוע דאטא-פארוואלטונג העלפט פארבעסערן די געוויקייט, פארמינערן האלוצינאציעס, און מאכן LLM-אפליקאציעס מער נוצלעך פאר עכטע ביזנעס.

prompt engineering און פארוואלטונג פון פראמפטס (40:41:03)

איינע פון די אייגנארטיגע זאכן ביי גרויסע שפראך-מאדעלן איז אז די קוואליטעט פונעם רעזולטאט הענגט שטארק אפ אין דער קוואליטעט פון די אנווייזונגען. די אנווייזונגען הייסן פראמפטס, און אויסטראכטן עפעקטיוע פראמפטס הייסט prompt engineering. א פראמפט איז נישט בלויז א פשוטע פראגע; ער קען אנטהאלטן קאנטעקסט, אנווייזונגען, ביישפילן, כללים און דעם געוואונטשענעם פארמאט פונעם ענטפער.

דער קורס ברענגט א ביישפיל: אנשטאט צו בעטן פונעם מאדעל "summarize this document", איז א בעסערער פראמפט: "You are a business analyst, analyze this document and provide a summary containing key insights, important numbers, risk, and recommendation." דער צווייטער פראמפט גיט מער קאנטעקסט און אנפירונג, און פירט געווענליך צו א בעסערן ענטפער.

אין א פראדוקציע-סביבה ווערט prompt engineering א סטרוקטורירטער פראצעס. גרופעס דארפן פארוואלטן פארשידענע ווערסיעס פון פראמפטס, טעסטן ווי זיי ארבעטן, און פארשטיין וועלכע ווערסיע גיט דעם בעסטן רעזולטאט – ענלעך צו פארוואלטן ווערסיעס פון וווייכווארג-קאוד. א קליינע ענדערונג אין א פראמפט קען פארבעסערן די פערפארמענס, אדער אמאל שאפן אומגעריכטע פראבלעמען. דעריבער פירט LLMOps איין קאנטראל פון פראמפט-ווערסיעס און טעסט-פראקטיקן.

פירמעס שאצן אויך אפ פראמפטס דורך מעטאדן ווי A/B testing: פארשידענע ווערסיעס פון א פראמפט ווערן געטעסט מיט עכטע ביישפילן, און די ווערסיע וואס ארבעט בעסער ווערט אויסגעקליבן. פראמפט-פארוואלטונג איז באזונדערס וויכטיג ווען מען בויט AI-אפליקאציעס וואס שטייען פנים-אל-פנים מיט קונים; א שלעכט-אויסגעטראכטער פראמפט קען ברענגען פאלשע ענטפערס, נישט-קאנסיסטענטע ענטפערס אדער נישט-פאסיגע רעזולטאטן. דעריבער איז prompt engineering נישט נאר א קונץ פאר עקספערימענטירן מיט AI-מאדעלן; אין LLMOps ווערט עס א וויכטיגע אינזשעניר-פראקטיק וואס מען דארף פאר בויען פארלעסלעכע AI-סיסטעמען.

וואס איז RAG (40:43:06)

איינע פון די גרעסטע שוועריגקייטן ביי גרויסע שפראך-מאדעלן איז אז זיי האבן נישט אלעמאל צוטריט צו די לעצטע אדער פריוואטע אינפארמאציע. א מאדעל קען האבן שטארקע שפראך-פארשטאנד, אבער ער ווייסט אפשר נישט ספעציפישע פירמע-דאטא, לעצטע נייעס אדער קאנפידענציעלע אינפארמאציע. Retrieval Augmented Generation, באקאנט אלס RAG, לייזט דעם פראבלעם דורך פאראייניגן ארויסנעמען אינפארמאציע מיט שאפן שפראך.

דער יסוד-געדאנק איז פשוט: אנשטאט צו בעטן דעם LLM ער זאל ענטפערן נאר מיט זיין עקזיסטירנדיגן וויסן, נעמט מען קודם ארויס שייכותדיגע אינפארמאציע פון אן אינדרויסנדיגן וויסן-קוואל און מען גיט זי צום מאדעל. דער פראצעס:

- דער באניצער שיקט א פראגע.
- דער סיסטעם פארוואנדלט די פראגע אין א פארמאט וואס מען קען באנוצן פאר זוכן.
- ער זוכט אין א וויסן-באזע, געווענליך דורך א וועקטאר-דאטא-באזע, און געפינט די מערסט שייכותדיגע דאקומענטן אדער אינפארמאציע.
- די ארויסגענומענע אינפארמאציע ווערט צוגעלייגט צום פראמפט און געשיקט צום LLM.
- דער מאדעל באנוצט זיך מיט דעם עקסטערן קאנטעקסט צו שאפן א געניערן און מער שייכותדיגן ענטפער.

ביישפיל: אן אנגעשטעלטער פרעגט אן AI-געהילף "What is our company's leave policy?" (וואס איז אונזער פירמע'ס כללים פון פרייע טעג?). אן אלגעמיינער LLM וויסט אפשר נישט די אינערליכע כללים, אבער מיט RAG נעמט דער סיסטעם ארויס דעם אפיציעלן דאקומענט פון די כללים פון דער פירמע'ס וויסן-באזע, גיט אים צום מאדעל, און דער מאדעל שאפט אן ענטפער באזירט אויף דער אינפארמאציע.

RAG ברענגט עטליכע מעלות: עס פארמינערט האלוצינאציעס, דערלויבט פירמעס צו באנוצן פריוואטע דאטא, האלט די אינפארמאציע דערהיינטיגט אן איבער-טריינירן דעם מאדעל, און מאכט LLM-אפליקאציעס מער פאסיג פאר פירמע-באדערפענישן. היינט איז RAG געווארן איינע פון די מערסט-באנוצטע ארכיטעקטורן פאר בויען פארלעסלעכע AI-אפליקאציעס פאר פירמעס.

סטראטעגיעס פון ארויסשטעלונג: API קעגן זעלבסט-געהאסט (40:44:39)

ווען מען שטעלט ארויס א גרויסן שפראך-מאדעל זענען דא צוויי הויפט-צוגאנגען: ארויסשטעלונג דורך אן API, און זעלבסט-געהאסטע ארויסשטעלונג. די ריכטיגע ברירה הענגט אפ אין פאקטארן ווי קאסטן, פריוואטקייט, קאנטראל און די מעגלעכקייט צו וואקסן (scalability).

- דורך אן API – מען באקומט צוטריט צום מאדעל דורך א פראווייידער. דאס איז געווענליך דער שנעלסטער וועג ארויסצושטעלן אן LLM, ווייל מען דארף נישט פארוואלטן די אונטערליגנדע אינפראסטרוקטור. עס איז לייכט אינצושטעלן, פשוט צו פארגרעסערן, און ארבעט גוט ווען מען וויל שנעל ארויסגעבן אן אפליקאציע. אבער עס האט באגרענעצונגען: די דאטא ווערט געשיקט צו אן אינדרויסנדיגע סערוויס, וואס קען שאפן זארגן פון פריוואטקייט אדער פון אנהאלטן תקנות; די קאסטן קענען וואקסן ווען די צאל באניצער, בקשות אדער טאקענס וואקסט; און מען האט ווייניגער קאנטראל איבער דער אינפראסטרוקטור פונעם מאדעל.
- זעלבסט-געהאסט – פירמעס לויפן מאדעלן ווי Llama אדער זייערע אייגענע צוגעפאסטע מאדעלן אויף פריוואטער אינפראסטרוקטור. די גרעסטע מעלה איז מער קאנטראל און פריוואטקייט: די פירמע באשליסט וואו דער מאדעל לויפט, ווי אזוי די דאטא ווערט אפגעהיט און ווי אזוי די אינפראסטרוקטור איז איינגעשטעלט. עס קען אויך געבן מער פאראויסצוזעענדיגע קאסטן ביי א גרעסערן היקף. אבער זעלבסט-האסטן פאדערט מער טעכנישע ארבעט.

LLMOps מעטריקס און די יסודות פון DevOps און CI/CD (40:46:14)

אינפראסטרוקטור און דעפלוימענט-סטראטעגיעס אין LLMOps (40:46:14)

דער קורס ענדיגט דעם טעמע פון LLM דעפלוימענט מיט דעם פונקט אז זעלבסט-האסטירן א מאדעל פארלאנגט עקספערטיז. די טימס דארפן פירן סערווערס, GPUs, נייע ווערסיעס פון מאדעל, מאניטארינג, און א דעפלוימענט-פייפליין.

עס ווערט ערקלערט אז ווען LLM אפליקאציעס וואקסן, ווערט אויך די אינפראסטרוקטור וואס שטיצט זיי וויכטיג. עטליכע ביישפילן ווערן דערמאנט:

- קלאוד-פלאטפארמעס, למשל Google Cloud;
- מכשירים צו אנפירן קאנטעינערס, למשל Kubernetes;
- טעכנאלאגיע צו סערווירן מאדעלן, למשל) vLLMאין די אויטאמאטישע אונטערשריפטן שטייט "VLM").

די מכשירים קענען העלפן פארגרעסערן LLM-ארבעט אויף אן עפעקטיווע אופן.

דער הויפט-פונקט: אין LLMOps איז נישטא איין דעפלוימענט-סטראטעגיע וואס פאסט פאר יעדן. דער בעסטער וועג הענגט אפ אין וויפיל די אפליקאציע פארלאנגט פון די פאלגנדע זאכן: שנעלקייט, פריוואטקייט, קאנטראל, קאסטן, עפעקטיווקייט און די מעגליכקייט צו וואקסן.

וואס מען דארף נאכקוקן נאכן דעפלויען (40:46:45)

דער קורס באטאנט אז די ארבעט הערט נישט אויף נאכדעם וואס א LLM אפליקאציע איז ארויסגעשטעלט. מען דארף ווייטער נאכפאלגן עטליכע זאכן:

- לעיטענסי און שנעלקייט: ווי שנעל דער מאדעל ענטפערט אויף א באניצער'ס בקשה. וועז די צייט פון ענטפער ווערט צו לאנג, שאדט עס די איבערלעבונג פונעם באניצער.
- באנוץ פון טאָקענס און קאסטן: אסאך LLM סערוויסעס רעכענען געלט לויט די צאל טאָקענס וואס ווערן פראצעסירט. דערפאר העלפט נאכקוקן ווי אסאך טאָקענס מען פארברויכט, כדי צו קאנטראלירן די הוצאות און צו פארבעסערן די פראמפטס.

- פידבעק פון באניצער: רייטינגס, קאמענטארן און דאס פירונג פונעם באניצער. דאס ווייזט צי די ענטפערס פונעם מאדעל זענען טאקע נוצליך אדער נישט.
- זיכערהייט, בפרט פראמפט-אינדזשעקשאן אטאקעס: ווען מען האלט אן אויג אויף פארדעכטיגע אינפוט, באשיצט מען די אפליקאציע פון מיסברויך.

מעטריקס פאר אפשאצן די קוואליטעט (40:47:15)

פאר דעם אפשאצן ווערן געבראכט די פאלגנדע מעטריקס:

- אקוראטקייט און רעלעוואנץ: מען קוקט צי דער מאדעל גיט ריכטיגע און נוצליכע ענטפערס.
- קוואליטעט פון קאנטעקסט: אין סיסטעמען וואס זענען געבויט אויף RAG מעסט מען אויך צי די אינפארמאציע וואס ווערט ארויסגעזוכט איז רעלעוואנט און העלפט.
- הזיה-ראטע (hallucination rate): ווי אפט דער מאדעל שאפט אינפארמאציע וואס איז פאלש אדער האט נישט קיין יסוד.
- א "LLM as a judge" צאל: א צווייטער גרויסער שפראך-מאדעל שאצט אפ די קוואליטעט פון די ענטפערס וואס זענען געשאפן געווארן.

אריינפיר: א נייער טייל וועגן DevOps און CI/CD (40:47:47)

דא הייבט זיך אן א נייער טייל פונעם קורס. עס ווערט אנגעזאגט דער פלאן:

- ערשט ערקלערן וואס איז דער DevOps פראצעס;
- דערנאך די וויכטיגקייט פון א CI/CD פייפליין;
- דערנאך א מוסטער-פייפליין, דאס הייסט בויען א CI/CD פייפליין מיט Jenkins;
- צום סוף א קליינע דעמא וויאזוי מען שטעלט דאס אויף, כדי א פייפליין זאל ווערן פראצעסירט און דורכגעפירט ווי איין גאנצקייט.

וואס איז DevOps? (40:48:17)

דער קורס ערקלערט אז DevOps איז א צוזאמשטעל פון צווי פראקטיקן: דעוועלאפמענט (אנטוויקלונג) און אפעראציעס.

די אנטוויקלונג-טים האט איר אייגענע ארבעט, צו אנטוויקלען און צוגרייטן דעם סאורס-קאוד. די אפעראציעס-טים איז פאראנטוואורטליך צו דעפלויען דעם קאוד אין א געוויסע סביבה, סיי א פראדאקשן סביבה און סיי אן אנדערע. זי טוט די אלע ארבעטן פון אפעראציעס: שאפן ווירטועלע מאשינען, זיי פירן, מאכן פעטשעס (patching) און נאך אסאך.

די אנטוויקלונג-טים ארבעט שטענדיג אויפן סאורס-קאוד און איז פאראנטוואורטליך אז דער פראדוקט זאל ארבעטן. זי באהאנדלט די פערפארמענס, שרייבט דעם קאוד, און ארבעט צוזאמען מיטן טעסטן כדי צו באשטעטיגן דעם קאוד. זי טוט א גרויסע צאל ארבעטן און ניצט פילע מכשירים: סקריפטינג-מכשירים, מכשירים צו שרייבן קאוד, און אנדערע אנטוויקלונג-מכשירים. דאס איז נויטיג ווייל זי טוט פארשידענע סארטן פראגראמירונג, און עס קען זיין אז אין איין פראיעקט ניצט מען מער ווי איין פראגראמיר-שפראך. דערפאר איז דער פעלד פון DevOps גאר א ברייטער.

די ראלע פון דער אפעראציעס-טים (40:49:48)

פון דער זייט פון אפעראציעס איז דאס א טים וואס פירט די ארבעטס-קראפט. איר ציל איז אז אלע טעגליכע ארבעטן און אפעראציעס זאלן ווערן געפירט אויף אן עפעקטיווע און פראדוקטיווע אופן. בעיקר איז די אפעראציעס-טים פאראנטוואַרטליך צו האלטן די סביבה אויף און ארבעטנדיג, און צו דורכפירן די ארבעט און די אויפהאלטונג וואס מען וויל טון אויף איר.

וואס DevOps ברענגט (40:50:18)

דער קורס רעכנט אויס די וויכטיגסטע מיילשטיינס וואס DevOps העלפט דערגרייכן:

- אפטערע רעליזן: מען האט אויך פאר DevOps ארויסגעגעבן נייע ווערסיעס, אבער נישט אזוי אפט. מען האט עס געטון בערך יעדן קוואַרטאל, יעדע דריי אדער פיר חדשים. מיטן קומען פון DevOps איז די אפטקייט שטארק געשטיגן. געוויסע פירמעס פרובירן ארויסצוגעבן יעדן חודש, אדער אפילו צווייַ מאל א חודש.
- צוזאמענארבעט פון די טימס: די זאך האט זיך שטארק פארבעסערט. אמאל האבן די אפעראציעס- און אנטוויקלונג-טימס נישט צוזאמענגעארבעט. יעדע איז געווען פארנומען מיט איר אייגענע ארבעט. מיט DevOps זענען זיי געקומען צוזאמען, און דאס האט פארגרעסערט די פראדוקטיוויטעט און די פערפארמענס פונעם פראדוקט.
- בעסערע פארוואלטונג: ווייל מען האט איבערגעשטעלט די פראצעסן און איינגעפירט אנטוויקלונג-מכשירים און אויטאמאציע, פארבעסערט זיך די פארוואלטונג פון אלע ארבעטן וואס זענען נישט געווען געפלאנט. דאס פלאנירן ווערט אויך שטארק בעסער.
- שנעלערע לייזונג פון פראבלעמען: ווייל מען ברענגט דעם קאוד צו דער פראדאקשן סביבה אין א קירצערע צייט, ווערן מער באגס פאַרריכט. דערצו איז דער צאל באגס וואס דערגרייכן די פראדאקשן שטארק געפאלן. ווען עס זענען וויניגער פראבלעמען, איז גרינגער זיי שנעל צו פאַרריכטן און דעם תיקון אריינצושטעלן אין דער פראדאקשן.

DevOps אין די גרויסע פירמעס (40:52:52)

דער קורס זאגט אז היינט פירן איין DevOps רוב גרויסע ארגאניזאציעס, סיי פינאנציעלע און סיי סערוויס-פירמעס. יעדע ארגאניזאציע פרובירט עס איינצופירן, ווייל עס שטעלט גאנץ איבער דעם אנטוויקלונג-פראצעס און מאכט אים אויטאמאטיש. די האנט-ארבעט וואס מען האט פריער אריינגעלייגט ווערט אויטאמאטיזירט דורך די מכשירים.

דאס האלט זיך בעיקר אויף א וויכטיגע אייגנשאפט: דער CI/CD פייפליין. ער איז פאראנטוואַרטליך צו ברענגען דעם קאוד צו דער פראדאקשן סביבה אין א קירצערע צייט, און ער איז דער שליסל וואס לאזט מען זאל מער ליפערן צו דער פראדאקשן.

וואס איז א CI/CD פייפליין? (40:53:54)

CI/CD איז דער באגריף פון "continuous integration" און "continuous delivery". מען באטראכט עס ווי דעם רוקנביין פון DevOps. ווען מען פירט איין DevOps אין א פראיעקט, איז דער ערשטער און מינימאלער שריט פון אויטאמאציע געווענליך דער CI/CD פייפליין.

וואס מיינט דאס ווארט "פייפליין"? א פייפליין איז א סעריע פון געשעענישן וואס זענען פארבונדן איינס מיטן צווייטן, א סדר פון שריט. דער קורס ברענגט א ביישפיל פון א געווענליכן דעפלוימענט:

- מען קאמפילירט דעם סאורס-קאוד;
- מען שאפט די ארטיפעקטן;
- מען טעסט;
- מען דעפלויט צו א געוויסע סביבה.

די אלע שריט, וואס מען פלעגט טון מיט די הענט, קען מען אריינשטעלן אין א פייפליין. א פייפליין איז נישט מער ווי א סדר פון די שריט, פארבונדן צווישן זיך, וואס ווערן דורכגעפירט איינס נאכן צווייטן. ער טוט פארשידענע ארבעטן: בויען דעם קאוד, לויפן די טעסט-קעיסעס, און מען קען אויך צולייגן דעם דעפלוימענט.

עס ווערט באטאנט אז דער סדר איז זייער וויכטיג. מען דארף אריינשטעלן אין פייפליין פונקט דעם זעלבן סדר וואס מען ניצט אין דער געווענליכער אנטוויקלונג-ארבעט.

(40:55:27) Continuous Integration (CI)

Continuous Integration הייסט אויך CI. אסאך מכשירים טראגן אין זייער נאמען "CI", און זיי מיינען דערמיט פשוט continuous integration.

דאס איז א פראקטיק וואס פאראייניגט דעם סאורס-קאוד אין א געמיינזאמער ריפאזיטארי, און זי מאכט אויטאמאטיש די באשטעטיגונג פונעם קאוד. זי נעמט אריין אן אויטאמאטישן בילד און אויטאמאטישע טעסט-קעיסעס. זי העלפט אנטדעקן פראבלעמען און באגס גרינג און שנעל, ווי א מעכאניזם וואס כאפט די פראבלעמען פרי.

דער קורס ווארנט: CI ווישט נישט אויס די באגס. זי העלפט נאר זיי לייכטער צו געפינען, ווייל עס רעדט זיך פון אן אויטאמאטישן פראצעס מיט אויטאמאטישע טעסטס. דערנאך דארף די אנטוויקלונג-טים באהאנדלען די באגס איינס נאכן צווייטן. א באג דארף מען איבערשרייבן און פארריכטן לויט די כללים פון אנטוויקלונג. דאס איז נישט קיין אויטאמאט וואס נעמט אוועק די באגס, אבער ער העלפט זיי שנעל צו געפינען און צו באזייטיגן.

(40:56:58) Continuous Delivery (CD) און דער ציל פונעם פייפליין

Continuous Delivery הייסט אויך CD. דאס איז א פאזע אין וועלכער מען מאכט ענדערונגען אין קאוד פאר דעם דעפלוימענט. דא באשטעטיגט מען וואס פונקט מען וויל ליפערן צום קונה, דאס הייסט וואס מען שיקט אריבער צו די קונים.

דער לעצטער ציל פונעם פייפליין איז דער דעפלוימענט. פראגראמירן איז נישט די איינציגע זאך. נאכן שרייבן די פראגראמען און טון די אנטוויקלונג, קומט דער דעפלוימענט: וויאזוי מען וועט אים דורכפירן. דער קורס זאגט אז דאס איז די שוינקייט פון דעם פראצעס, ווייל מען קען באשטימען וויאזוי דער דעפלוימענט זאל ווערן געטון און דורכגעפירט.

ווען מען שטעלט די צוויי פראקטיקן, CI און CD, צוזאמען לויטן סדר, ווערן אלע שריט איין גאנצער אויטאמאטישער פראצעס. דער פראצעס הייסט CI/CD. ווייל עס רעדט זיך פון CI/CD אויטאמאציע, איז דער רעזולטאט אויטאמאציע פון סיי דעם בילד און סיי דעם דעפלוימענט.

די מעלות פון CI/CD אויטאמאציע (40:58:30)

דער קורס ערקלערט אז א CI/CD סיסטעם נעמט זיך אן סיי מיט'ן בויען (build), סיי מיט'ן דורכפירן די טעסט-קעיסעס, און סיי מיט די דעפלוימענטס. וועו מען שטעלט איין CI/CD, קען די טיים מאכן builds און deploys גאר שנעל און עפעקטיוו, ווייל אלעס גייט אויטאמאטיש.

דער ערשטער רווח איז אז עס דארף נישט קיין האנט-ארבעט, און דערמיט פאלט אוועק די געפאר פון מענטשליכע טעותים. מען זעט אפט אז ביים מאכן א דעפלוימענט פארגעסט מען א טייל binaries, אדער עס פעלט עפעס אנדערש. מיט CI/CD פאלט די גאנצע פראבלעם אוועק.

דער צווייטער רווח איז אז דער פראצעס מאכט די טיימס מער אגיל (agile), מער פראדוקטיוו און מער זיכער אין זיך. די אויטאמאציע גיט א חיזוק צום בטחון אז די זאכן וועלן ארבעטן ווי געהעריג און אז עס זענען נישטא קיין פראבלעמען.

וואס איז Jenkins און פארוואס עס איז פאפולער (40:59:01)

עס ווערט געפרעגט: מען הערט אפט אז Jenkins איז א CI-געצייג און א CD-געצייג, אבער וואס איז עס טאקע? דער קורס ערקלערט אז Jenkins ווערט אויך אנגערופן אן "orchestration"-געצייג. עס איז אן אויטאמאטישער געצייג, און די גרעסטע מעלה איז אז עס איז אינגאנצן open source.

עס זענען יא פאראן געצאלטע ענטערפרייז-ווערסיעס, למשל CloudBees, אבער לויט'ן קורס איז נישטא קיין עכטער חילוק אין וואס CloudBees און Jenkins גיבן. דערפאר נוצן אסאך פירמעס Jenkins אזוי ווי עס איז. מען זעט אין גרויסע פירמעס אז זיי גייען נישט צו קיין ענטערפרייז-געצייג ווי CloudBees, נאר זיי נוצן דעם גרונט-Jenkins.

Jenkins מאכט עס גרינג פאר פראגראמירער צו איינשליסן (integrate) זייערע ענדערונגען אינעם פראיעקט. דאס איז זייער וויכטיג, ווייל עס העלפט די טיימס פלאנירן ווי אזוי די ארבעט ווערט געטון. לויט'ן קורס איז די גרינגקייט פאר פראגראמירער די גרעסטע מעלה פון דעם געצייג. דערפאר איז Jenkins א וויכטיגער געצייג וועו מען רעדט פון אזעלכע אויטאמאציעס.

Plugins און די בייגזאמקייט פון Jenkins (41:00:32)

Jenkins דערגרייכט קאנטינועלע אינטעגראציע (continuous integration) מיט'ן הילף פון plugins, און דאס איז נאך א מעלה. עס זענען פאראן אסאך plugins. דער קורס ברענגט א ביישפיל: אויב מען וויל אן אינטעגראציע מיט Kubernetes אדער Docker, זענען די plugins אפשר נישט איינגעשטעלט פון אנהייב. אבער מען האט די מעגלעכקייט זיי צו אינסטאלירן, און דאן וועלן די פונקציעס איינגעבויט און פאראייניגט אין Jenkins.

אין קורץ ערקלערט דער קורס אז Jenkins פאסט צום בעסטן צו בויען א CI/CD פייפליין צוליב עטלעכע סיבות:

- די בייגזאמקייט (flexibility).
- דער open source כאראקטער.
- די שטיצע פאר plugins.

• עס איז גרינג צו באנוצן, מיט א פשוט'ן און קלארן GUI. מען כאפט שנעל וויאזוי Jenkins ארבעט.

דער סוף-רעזולטאט איז א שטארקער געצייג מיט וועלכן מען קען איינפירן CI/CD פאר כמעט יעדן סארט קאוד אדער פראגראמירונג-שפראך: Android, .NET, Java, NodeJS. אלע שפראכן וועלן געשטיצט אין Jenkins.

א CI/CD פייפליין מיט Jenkins – די באדערפענישן (41:01:33)

דער קורס גייט איבער צום CI/CD פייפליין מיט Jenkins. כדי צו אויטאמאטיזירן דעם גאנצן אנטוויקלונג-פראצעס איז א CI/CD פייפליין די לעצטע לעזונג, און כדי צו בויען אזא פייפליין איז Jenkins די בעסטע לעזונג. עס וועלן געברענגט זעקס טריט פאר אן אלגעמיינעם פייפליין. מען קען צולייגן נאך טריט, למשל וועלן מען אינסטאלירט נאך plugins, אבער דאס זענען די גרונט-טריט פאר א מינימאלן פייפליין.

דער ערשטער באדארף איז אז עס מוז זיין א Java JDK אויפ'ן סיסטעם. די מערסטע אפעריירונג-סיסטעמען האבן שוין א Java JRE, אבער דער JRE איז נאר צום לויפן: מען קען לויפן jar ,artifacts-פיילס, די אפליקאציע און דעם קאוד, אבער עס מאכט נישט קיין קאמפיילירונג. פאר קאמפיילירן דארף מען javac, וואס קומט מיט'ן Java JDK. דערפאר דארף מען דעם JDK.

דער קורס דערמאנט אויך אז מען דארף פארשטיין ווי אזוי צו לויפן געוויסע Linux באפעלן, ווייל עס וועלן לויפן אינסטאלאציע-טריט און פראצעסן.

אראפנעמען און לויפן Jenkins (41:03:05)

דער ערשטער טריט איז אראפנעמען און אינסטאלירן דעם JDK. דערנאך נעמט מען אראפ Jenkins פונעם אפיציעלן וועבזייטל jenkins.io/download. דארט איז פארן שטיצע פאר פארשידענע אפעריירונג-סיסטעמען און פלאטפארמעס: א Java פעקל (war-פייל), Docker, Ubuntu, Debian, CentOS, Fedora, Red Hat, Windows, openSUSE, FreeBSD, Gentoo און Mac. מען קען דארט אראפנעמען יעדן סארט artifact פאר יעדער סביבה.

מען נעמט אראפ דעם אלגעמיינעם Java פעקל (דעם war-פייל) אין א באשטימטן פאלדער. דער קורס ברענגט א ביישפיל: מען מאכט א פאלדער מיטן נאמען Jenkins, מען גייט אריין אין אים מיט'ן cd באפעל, און מען לויפט:

java -jar jenkins.war

עס וועלן ערקלערט אז jar און war פיילס זענען artifacts וואס מען קען גלייך לויפן מיט'ן java באפעל. מען דארף נישט קיין web container אדער application container. וועלן דער באפעל לויפט, הייבט די אפליקאציע אן צו ארבעטן.

דערנאך עפנט מען א וועב-בלעטערער און גייט צו localhost מיט'ן פארט. דער קורס זאגט אז Jenkins נוצט דעם 8080 פארט, ענליך צו Apache Tomcat. וועלן די אינסטאלאציע איז פארטיג, עפנט מען localhost:8080. מען קען אויך אריינגיין דורך דעם public IP address מיט'ן פארט, און אזוי צוקומען צו Jenkins אין בלעטערער.

שאפן א פייפליין-דזשאב (41:04:37)

אין Jenkins איז פאראן אן אפציע "create new jobs" (צוויי נעמען ווערן געוויזן פאר דער זעלבער זאך: "new item" אדער "new job"). מען דריקט דערויף, מען קלייבט אויס "pipeline job", און מען גיט א נאמען פארן פייפליין. דערנאך דריקט מען OK.

מען סקראלט אראפ ביז דער "pipeline" אפטיילונג און מען קלייבט אויס "pipeline script". דארט זענען פאראן פארשידענע אפציעס פאר ווי אזוי צו פירן דעם פייפליין:

- מען קען גלייך אריינשרייבן דעם פייפליין-סקריפט דארט.
- מען קען ברענגען דעם Jenkinsfile פון א source code management געצייג, ווי Git אדער Subversion.

אנשטעלן, לויפן און נאכקוקן דעם דזשאב (41:06:38)

דער ווייטערדיגער טריט איז אנשטעלן און לויפן דעם פייפליין-דזשאב. אויב מען האט שוין א GitHub לינק וואו דער Jenkinsfile געפינט זיך, לייגט מען אריין דעם לינק. מען שרייבט אן אז דער פייפליין-סקריפט זאל קומען פונעם Jenkinsfile אין דעם GitHub repository. דערנאך מאכט מען save, און Jenkins נעמט דעם פייפליין פון דארט.

דערנאך דריקט מען "build now", און מען זעט ווי דער build פראצעס גייט פאר. אויב מען דריקט אויף "console output", באקומט מען אלע logs פון וואס עס טוט זיך אינעווייניג: יעדן טריט פונעם פייפליין וואס לויפט.

דער זעקסטער טריט: ווען מען דריקט "build now", זעט מען ווי דער קאוד ווערט ארויסגענומען (checked out) און אראפגענומען פארן build.

שפעטער, אויב מען וויל טוישן דעם GitHub URL, קען מען אנשטעלן דעם עקזיסטירנדן דזשאב פון ניי און טוישן דעם לינק ווען עס פעלט אויס. מען קען אויך מאכן א קאפיע (clone) פונעם דזשאב, און דער קורס רופט דאס איינע פון די בעסטע מעלות.

עס זענען אויך פאראן "advanced settings". דארט לייגט מען אריין דעם GitHub repository URL. דאן ווערט דער Jenkinsfile אראפגענומען, און ווען מען דריקט "build now" לויפן אסאך טריט און אנשטעלונגען. צווישן זיי איז די דעקלאראציע checkout scm, וואס נעמט ארויס דעם קאוד פונעם repository.

Jenkins-פייפליין, MLOps און אגענט-אינטערוויו-פראגעס (41:09:10)

א דעמא: א פייפליין אין Jenkins (41:09:10)

דער קורס ענדיגט די ערקלערונג פון CI/CD-פייפליינס מיט א פראקטישער דעמא. מען זעט דעם Jenkins-פארטאל, וואו עס איז דא אן אפציע "create a job". מען קען קליקן אויף "New Item" אדער גלייך אויף "create a job".

- מען גיט א נאמען, למשל "pipeline", און מען קלייבט אויס דעם סארט job.
- עס זענען פאראן עטליכע סארטן: Freestyle, Pipeline, GitHub Organization און Multibranch Pipeline. אין דער לעקציע קלייבט מען אויס "Pipeline" און מען דריקט OK.

- עס עפנט זיך א קאנפיגוראציע-בלאט פאר דעם פייפליין. ס'איז דא די זעלבע אפטיילונגען ווי ביי Freestyle, למשל General און Build Triggers. אבער די Build Step און Post-build Step זענען אינגאנצן אראפגענומען, ווייל דער פייפליין אליין נעמט זייער פלאץ.

מען קען אריינשטעלן דעם גאנצן פייפליין-סקריפט גלייך דא. Jenkins גיט אויך מוסטערן, למשל א GitHub-Maven-ביישפיל, מיט פארטיגע סטעפס. אז מען לאזט עס לויפן, ארבעט עס גלאט און עס ציט אראפ (check out) קאוד. אבער דער ריכטיגער וועג ביים בויען א CI/CD-פייפליין איז צו האלטן דעם Jenkinsfile אין א ווערסיע-קאנטראל-סיסטעם, און דאס ווייזט מען ווייטער.

דער Jenkinsfile פון SCM, Build Now און די לאגס (41:10:43)

- מען קלייבט אויס "Pipeline script from SCM" און דערנאך Git.
- דער פייל-נאמען פונעם פייפליין-סקריפט איז "Jenkinsfile".
- מען לייגט אריין די אדרעס פון א Git-repository. אין דעם repository איז דא א Maven-build-פייפליין מיט סטעפס פאר CI, פאר דעם build און פאר דיפלוימענט.
- ביי א פריוואטן repository דארף מען צולייגן credentials דורכן קנעפל "Add". דער repository אין דער דעמא איז אבער פובליק, דעריבער דארף מען נישט קיין credentials.

נאכן אפהיטן (save) די קאנפיגוראציע באקומט מען א בלאט מיט אפציעס: "Build Now", אויסמעקן דעם פייפליין, אדער אים איבער-קאנפיגורירן. מען דריקט "Build Now". גלייך ווערט דער פייפליין אראפגעצויגן און עס הייבט אן לויפן. ביז עס לויפט נאך, זעט מען נישט די גאנצע Stage View. מען זעט אבער שוין אז דער "Checkout Code" שטאפל איז פארטיג און מ'גייט ווייטער צום build. נאכן build גייט עס ווייטער צו די קומענדיגע סטעפס.

עס ווערט ערקלערט אז מען קען קוקן דעם "Console Output" כדי צו זען דעם גאנצן לאג. מען קען אויך זען לאגס פאר יעדן שטאפל באזונדער. דאס איז וויכטיג, ווייל אין דעם פולן לאג זענען דא אסאך סטעפס און אסאך שורות, און אז מען וויל זען דעם לאג פון איין באשטימטן שטאפל, איז דאס דער ריכטיגער אָרט.

די שטאפלען וואס מען זעט אין דעם פייפליין זענען:

- Checkout Code
- Build
- אויספירן טעסט-קעיסעס
- SonarQube-אנאליז
- Archive Artifacts
- דיפלוימענט
- נאטיפיקאציע

דער גאנצער פייפליין ענדיגט זיך מיט הצלחה (Success) אין דער Stage View. דאס artifact קען מען אראפלאדן: א WAR-פייל, דאס הייסט א וועב-אפליקאציע. דער קורס באטאנט אז אזוי זעט אויס א טיפישער פייפליין און א פולע אויטאמאציע. מיט כמעט די זעלבע סטעפס קען מען אויטאמאטיזירן יעדן סארט פייפליין.

וואס איז ?MLOps (41:13:16)

דא הייבט זיך אן א נייער טייל וועגן MLOps. דער פלאן איז: וואס איז MLOps, וואס איז דער נוצן דערפון, באמת'ע אנווענדונגען, די קאמפאנענטן, און צום סוף דער אונטערשייד צווישן MLOps און DevOps.

MLOps, א קורצע פארם פון "Machine Learning Operations", איז א זאמלונג פון מעטאדן, געצייג און פרינציפן. דער ציל איז צו מאכן גלאטער און בעסער דאס ארויסשטעלן (deployment), פירן און אויפפאסן אויף מאשין-לערנען מאדעלן אין א פראדוקציע-סביבה. עס נעמט אינספיראציע פון DevOps, דאס הייסט פון "Development and Operations", און וועגדט עס אָן ספעציעל אויף די שווערִיגקייטן פון ארויסשטעלן און אויפהאלטן ML-מאדעלן.

דער נוצן פון MLOps (41:14:17)

דער קורס רעכנט אויס פינף צוועקן:

- עפעקטיוע אנטוויקלונג: MLOps אויטאמאטיזירט דעם deployment-פראצעס פון מאדעלן. דאדורך ווערט ער שנעלער, מען קען אים איבערחזרן, און עס זענען דא ווייניגער טעותים. אן ארגאניזאציע קען אזוי שנעל אריבערגיין פון אנטוויקלונג צו פראדוקציע, און דער מאדעל קומט שנעלער צום סוף-באניצער.
- מיטארבעט: MLOps מוטיגט מיטארבעט צווישן פארשידענע מאנשאפטן: דאטא-סייענטיסטן, מאשין-לערנען אינזשעניערן, אפעראציע, און ביזנעס-שטעיקהאלדערס. ווען די קאמוניקאציע איז בעסער, קען מען ליפערן לייזונגען וואס פאסן סיי צו די טעכנישע און סיי צו די ביזנעס-פאדערונגען.
- אויפפאסן און אויפהאלטן (MLOps :monitoring and maintenance) גיט געצייג צו קאנטראלירן ווי די מאדעלן ארבעטן אין דער ווירקליכקייט. מען כאפט אנאמאליעס, מען נאכפאלגט די הויפט-מעסטונגען, און מען פארריכט פראבלעמען שנעל. אזוי ליפערן די מאדעלן געניע רעזולטאטן אויך מיט דער צייט.
- שפארעוודיגקייט: מען נוצט די רעסורסן בעסער און מען אויטאמאטיזירט פראצעסן, און דאס שפארט געלט. מען רעדוצירט די אפעראציע-קאסטן און די פארשוועדונג, און מאשין-לערנען פראיעקטן ווערן מער כדאי.
- זיכערהייט: אין אסאך אינדוסטריעס איז compliance און זיכערהייט הויפט-זארגן. MLOps גיט מעטאדן פאר צוטריט-קאנטראל, דאטא-פריוואטקייט, און א רעגולירטער סביבה. אזוי פאלגן די מאדעלן נאך די נויטיגע סטאנדארטן.

אין סך הכל, זאגט דער קורס, שפילט MLOps א קריטישע ראלע צו ארויסקריגן די מערסטע ווערט פון מאשין-לערנען. מאדעלן ווערן עפעקטיוו ארויסגעשטעלט, מען פאסט אויף זיי אויף, מען האלט זיי אויף און מען פארבעסערט זיי. ארגאניזאציעס באקומען אזוי א קאנסיסטענטן, פארלעסליכן און ווייטער-וואקסנדיגן נוצן פון זייער אינווסטירונג.

באמת'ע אנווענדונגען פון MLOps (41:15:50)

- פרעדיקטיוע אנאליטיק: מאדעלן אנאליזירן היסטארישע דאטא כדי פאראויסצוזאגן קומענדיגע געשעענישן. מען נוצט זיי אין פינאנץ, מארקעט, פארקויף און אנדערע אינדוסטריעס, כדי צו זען טרענדס און צו נעמען בעסערע באשלוסן.
- בילדער- און ווידעא-אנאליז: בילדער-קלאסיפיקאציע, דערקענען אביעקטן, און דערקענען פנימער. מען נוצט דאס אין זיכערהייט-סיסטעמען, אין אויטאנאמע מאשינען, און אין מעדיצינישע בילדער.

• געזונטהייט און מעדיצינישע דיאגנאז: מאדעלן פאר אנאליז פון מעדיצינישע בילדער, דיאגנאז פון קרענק, געפינען נייע מעדיצינען, און אפשאצן דעם ריזיקע פון פאציענטן.

• אויטאנאמע סיסטעמען: אויטאס, דראנס און ראבאטיק. MLOps זארגט אז די AI-מאדעלן זאלן ארבעטן פארלעסליך און זיך צופאסן צו טוישנדיגע באדינגונגען.

דאס זענען נאר עטליכע ביישפילן, און די אנווענדונגען וואקסן אלץ ווייטער. (דא קומט אויך א קורצע רעקלאמע פאר א קורס.)

די קאמפאנענטן פון MLOps (41:17:54)

ווי ווייט MLOps גייט אין א פראיעקט, ווענדט זיך אין די באדערפענישן פונעם פראיעקט. אמאל דעקט עס אלעס, פון צוגרייטן די דאטא ביזן לעצטן מאדעל. אמאל איז עס נאר דאס ארויסנשטעלן דעם מאדעל אין ארבעט. רוב פירמעס נוצן MLOps פאר:

• Exploratory Data Analysis: דורכקוקן די דאטא כדי זי צו פארשטיין.

• צוגרייטן די דאטא און בויען ווערטפולע features: Data Preparation און Feature Engineering.

• טריינירן דעם מאדעל און fine-tuning.

• איבערקוקן דעם מאדעל און governance.

• Model Inference און Serving.

• deployment און monitoring פונעם מאדעל.

• אויטאמאטיש איבער-טריינירן דעם מאדעל ווען עס איז נויטיג.

דער אונטערשייד צווישן MLOps און DevOps (41:18:25)

• פאקוס: MLOps פאקוסירט זיך ספעציעל אויף deployment, monitoring און פירן מאשין-לערנען מאדעלן. DevOps פאקוסירט זיך בעיקר אויף מיטארבעט און אינטעגראציע צווישן אנטוויקלונג און אפעראציע.

• פעלד: MLOps איז געבויט ארום ML- און AI-אפליקאציעס, מיט זייערע ספעציעלע שווערigקייטן. DevOps גילט פאר סאפטווער-אנטוויקלונג בכלל, אויף אלע זייטן פון סאפטווער-אינזשעניערינג און deployment.

• אויטאמאציע: MLOps אויטאמאטיזירט דעם גאנצן מאשין-לערנען לעבנס-ציקל, פון פאר-באארבעטן די דאטא ביז monitoring און איבער-טריינירן. DevOps וויל מאכן גלאטער דעם deployment-ציקל: אויטאמאטישע פייפליינס, בעסערע מיטארבעט, און שנעלערע און פארלעסליכערע סאפטווער-ארויסגאבעס.

• געצייג: MLOps נוצט געצייג פאר ווערסיעס פון מאדעלן, איינפאקן מאדעלן, monitoring, אויטאמאטיש איבער-טריינירן, און ערקלערן מאדעלן. DevOps נוצט ווערסיע-קאנטראל-סיסטעמען ווי Git, CI/CD-פייפליינס, קאנפיגוראציע-פירונג, און Infrastructure as Code.

• ציל-רעזולטאטן: ביי MLOps איז דער ציל עפעקטיווער deployment פון מאדעלן, שטענדיגע קאנטראל אויף זייער פערפארמענס, די מעגליכקייט איבערצוחזרן רעזולטאטן (reproducibility), מיטארבעט צווישן דאטא-סייענטיסטן און אינזשעניערן, און צופאסן זיך צו טוישנדיגע דאטא און פאדערונגען. ביי DevOps איז

דער ציל שנעלערע אנטוויקלונג-ציקלען, Continuous Integration , שנעלער און אויטאמאטישער deployment, בעסערע מיטארבעט אין דער מאנשאפט, און מער פארלעסליכע סאפטוועיר.

אין סך הכל טיילן ביידע די באגריפן פון אויטאמאציע, מיטארבעט און שטענדיגער פארבעסערונג. DevOps האט אבער א ברייטערן פאקוס אויף סאפטוועיר, בשעת MLOps איז צוגעפאסט צו די ספעציפישע שווערigקייטן פון ארויסשטעלן און פירן מאשין-לערנען מאדעלן אין פראדוקציע. ביידע ווילן פארבעסערן עפעקטיווקייט, מיטארבעט און פארלעסליכקייט.

אינטערוויו-פראגע: וואס איז אן AI-אגענט? (41:20:28)

דער קורס הייבט אן מיט אינטערוויו-פראגעס און ענטפערס פאר אנהייבער אויף Agentic AI. די ערשטע פראגע וואס מען קען ערווארטן: "וואס איז אן AI-אגענט, און וואס איז דער אונטערשייד צווישן אים און א פשוט'ן LLM-רוף?"

דער פארגעשלאגענער ענטפער: אן AI-אגענט איז אן אינטעליגענטע סיסטעם וואס נוצט א שפראך-מאדעל אלס איר "מוח" פאר חשבונות. זי פארברייטערט אבער זיינע כוחות: זי פארשטייט צילן, נעמט באשלוסן, נוצט דרויסנדיגע געצייג, האלט זכרון, און פירט אויס אקציעס אליין.

א פשוטער LLM-רוף גייט אין א גלייכן שטייגער. דער באניצער גיט א פראמפט, און דער מאדעל ענטפערט לויט זיין טריינירונג און לויטן קאנטעקסט. דער מאדעל באשליסט נישט אליין וועלכע סטעפס צו נעמען, און ער רעדט נישט מיט דרויסנדיגע סיסטעמען.

אן AI-אגענט, אנדערש, קען צעטיילן א קאמפליצירטן ציל אויף קלענערע אויפגאבעס. ער באשטימט וועלכע אקציע איז נויטיג, רופט APIs אדער געצייג, אנאליזירט די רעזולטאטן, און ארבעט וועיטער ביז דער ציל איז דערגרייכט.

א ביישפיל: אז א באניצער בעט א LLM צו פלאנירן א רייזע, גיט ער פארשלאגן. אן AI-אגענט קען זוכן פליגערס, פארגלייכן האטעלן, קאנטראלירן צי עס איז פארן פלאץ, צוזאמשטעלן א רייזע-פלאן, און דעם פלאן טוישן לויט וואס דער באניצער וויל.

דער הויפט-אונטערשייד: א LLM שאפט ענטפערס, בשעת אן AI-אגענט קען טראכטן, האנדלען און פארענדיגן אויפגאבעס. דערנאך גייט דער קורס איבער צו דער קומענדיגער פראגע, וועגן א טראדיציאנעלן LLM.

אינטערוויו-פראגעס: יסודות פון אגענטישן AI (41:22:00)

א געווענליכער LLM קעגן אן אגענטישע AI סיסטעם (41:22:00)

דער קורס ערקלערט דעם חילוק צווישן א געווענליכן LLM און אן אגענטישער AI סיסטעם. א געווענליכער LLM איז בעיקר געבויט צו פארשטיין און ארויסגעבן טעקסט לויט דעם באניצער'ס פראמפט. ער ארבעט אויפן שיטה פון "בקשה און ענטפער": דער באניצער גיט אן אנווייזונג, און דער מאדעל גיט צוריק א תשובה. געווענליך האלט ער נישט קיין מצב (state), ארבעט נישט צוזאמען מיט דרויסנדיגע אפליקאציעס, און באשליסט נישט אליין וואס דער קומענדיגער שריט דארף זיין.

אן אגענטישע AI סיסטעם בויט זיך אויף א LLM און לייגט צו פלאנירן, זכרון (memory), באנוץ פון געצייג (tools) און די פעאיגקייט צו באשליסן. אנשטאט נאר צו ענטפערן פראגעס, קען זי דורכפירן גאנצע ארבעטס-פראצעסן מיט אסאך שריט.

א משל: א געווענליכער LLM קען ערקלערן ווי אזוי מען שטעלט צוזאם א מארקעטינג-קאמפיין. אן אגענטישע AI סיסטעם אבער קען אויספארשן די קאנקורענטן, אנאליזירן דאטא פון קונים, ארויסגעבן תוכן, איינפלאנירן די קאמפיינס און מעסטן ווי זיי האבן געווירקט, אלעס מיט פארשידענע געצייג.

אגענטישע סיסטעמען ווערן געטריבן פון א ציל, נישט פון א פראמפט. זיי באטראכטן דעם רעזולטאט, פאסן אן דעם צוגאנג, און טוען ווייטער שריט ביז מען קומט אן צום געווענטשענעם רעזולטאט.

די הויפט-טיילן פון אן AI אגענט (41:23:03)

די דריטע פראגע וואס מען קען ערווארטן איז: וואס זענען די הויפט-טיילן פון אן AI אגענט? אן אגענט באשטייט פון עטליכע טיילן וואס ארבעטן צוזאמען, כדי ער זאל קענען אליין באשליסן און דורכפירן ארבעט:

- דער "טראכט-מאדעל" (reasoning model). געווענליך איז דאס א גרויסער שפראך-מאדעל, וואס איז דער מוח פונעם אגענט. ער פארשטייט דעם באניצער'ס ציל, אנאליזירט די לאגע און באשליסט וועלכע שריט מען דארף טון.
- די אנווייזונג-שיכט (instruction layer). זי באשטימט דעם אגענט'ס ראלע, זיין אויפפירונג, זיינע גרעניצן און זיין ציל.
- געצייג (tools). זיי לאזן דעם אגענט ארבעטן מיט דרויסנדיגע סיסטעמען, ווי API'ס, דאטאבאזעס, זוכמאשינען און אפליקאציעס.
- זכרון (memory). ער העלפט דעם אגענט האלטן ניצליכע אינפארמאציע פון פריערדיגע שמועסן, און האלטן דעם קאנטעקסט ווען מען גייט פון איין ארבעט צו דער אנדערער.
- דער אגענט-לופ (agent loop). דער לעצטער טייל: די סיסטעם גייט אלעמאל ווידער דורך א ציקל פון טראכטן, טון א שריט, באטראכטן דעם רעזולטאט און דערהיינטיגן איר מצב.

א ביישפיל: א קונה-שטיצע אגענט ניצט זיין טראכט-פעאיגקייט צו פארשטיין דעם קונה'ס פראבלעם. ער נעמט ארויס אינפארמאציע פון דאטאבאזעס דורך געצייג, ער געדענקט דעם קונה'ס היסטאריע, און ער רעדט ווייטער מיטן קונה ביז דער פראבלעם ווערט געלייזט.

דער ReAct מוסטער: טראכטן און טון (41:24:33)

די פערטע פראגע: וואס איז דער ReAct מוסטער (?Reasoning plus Acting דער מוסטער פאראייניגט טראכטן מיט טון, כדי אן AI סיסטעם זאל קענען בעסער לייזן קאמפליצירטע פראבלעמען. דער אגענט גיט נישט גלייך ארויס א לעצטן ענטפער. ער גייט דורך א פראצעס פון שריט:

- ער טראכט קודם איבער דעם פראבלעם.
- ער באשליסט וועלכער שריט איז נויטיג.
- ער פירט אויס דעם שריט מיט די געצייג וואס ער האט.
- ער באטראכט דעם רעזולטאט.
- ער טראכט ווייטער לויט דער נייער אינפארמאציע.

אזוי שאפט זיך א שטענדיגער קרייז צווישן טראכטן און טון. א ביישפיל: א באניצער פרעגט "וואס איז דאס וועטער יעצט אין ?New York" דער אגענט כאפט קודם אז ער דארף אינפארמאציע פון יעצט, ווייל די

טרייניר‌ונג-דאטא פונעם LLM האט נישט די היינטיגע וועטער-פרטים. ער רופט דעריבער א וועטער-API, באקומט די לעצטע דאטא, און ניצט זי צו געבן א פינקטליכן ענטפער.

דער ReAct מוסטער מאכט די סיסטעם מער פארלעסליך, ווייל דער אגענט פארלאזט זיך נישט נאר אויף זיין אינערליכע וויסן. ער קען נאכקוקן אינפארמאציע, ארבעטן מיט דרויסנדיגע סיסטעמען, און ענדערן זיין צוגאנג לויט דעם וואס ער זעט אין דער וועלט. דערפאר איז דער מוסטער ניצליך פאר פארש-געהילפן, קונה-שטיצע סיסטעמען און אויטאמאטישע ארבעטס-פראצעסן.

דער Plan-and-Execute מוסטער (41:25:33)

די פינפטע פראגע: וואס איז דער Plan-and-Execute אגענט מוסטער? דאס איז א בוי-אופן וואו די AI סיסטעם טיילט אפ דאס פלאנירן פון א ארבעט פונעם טאקע דורכפירן. ערשט אנאליזירט דער פלאנירונג-טייל דעם באניצער'ס ציל, און שטעלט צוזאם א רייע קלענערע שריט וואס מען דארף טון כדי אנצוקומען צו אים. נאכדעם וואס דער פלאן איז פארטיג, פירט אן אויספיר-טייל אויס יעדן שריט מיט די געצייג און מיטלען וואס ער האט. די אפטיילונג מאכט די סיסטעם מער ארדנטליך און מער ווירקזאם פאר קאמפליצירטע ארבעטן.

א ביישפיל: דער ציל איז צוצושטעלן א געשעפט-באריכט. דער פלאנירער קען באשליסן אז מען דארף זאמלען דאטא, אנאליזירן טרענדס, ארויסברענגען איינזיכטן, און צוגרייטן דעם לעצטן דאקומענט. דער אויספירער טוט דאן יעדע זאך: ער גייט אריין אין דאטאבאזעס, לויפט די אנאליז, און שטעלט צוזאם דעם באריכט.

דער מוסטער האט עטליכע מעלות:

- מען קען דורכקוקן, ענדערן אדער פארבעסערן דעם פלאן נאך איידער מען הייבט אן דאס אויספירן.
- עס איז גרינגער צו געפינען פעלערן (debugging), ווייל די פראגראמירער קענען זען צי דער פראבלעם איז געשען בשעת'ן פלאנירן אדער בשעת'ן אויספירן.

דערפאר איז דער מוסטער באזונדערס ניצליך פאר לאנג-לויפנדיגע ארבעטס-פראצעסן, וואו מען דארף אסאך שריט וואס פאסן צוזאם איינס מיט'ן אנדערן.

געצייג און Function Calling (41:27:05)

די זעקסטע פראגע: וואס איז tool use אדער function calling, און פארוואס איז עס וויכטיג פאר אגענטן? מיט tool use אדער function calling קען אן AI אגענט ארבעטן מיט דרויסנדיגע סיסטעמען. ער בעט צו רופן פונקציעס וואס זענען פאראויס באשטימט, אנשטאט נאר צו שרייבן טעקסט.

א געוועלטיגער LLM האט גרעניצן: ער קען נישט גלייך באקומען אינפארמאציע פון יעצט, נישט דורכפירן טראנזאקציעס, נישט דערהיינטיגן דאטאבאזעס, און נישט ארבעטן מיט אפליקאציעס. function calling נעמט אוועק די גרעניצן, ווייל עס פארבינדט דעם שפראך-מאדעל מיט דרויסנדיגע פעאיגקייטן. למשל:

- א באנק-געהילף קען ניצן געצייג צו קוקן אויפן באלאנס פון א חשבון, נאכקוקן טראנזאקציעס, אדער באהאנדלען בקשות.
- א פארש-אגענט קען רופן זוך-געצייג צו זאמלען היינטיגע אינפארמאציע, ניצן קאוד-אויספיר געצייג פאר חשבונות, אדער פרעגן דאטאבאזעס פאר געשעפט-איינזיכטן.

געצייג זענען וויכטיג ווייל זיי מאכן פון א LLM, וואס גיט נאר ארויס אינפארמאציע, אן אקטיוועט סיסטעם וואס לייזט פראבלעמען. א גוט געבויטער געצייג האט א קלארן ציל, אינפוט וואס איז ארדנטליך אויסגעשטעלט, און אויטפוט וואס מען קען פאראויסזען. אזוי פארשטייט דער אגענט ווען און ווי אזוי ער זאל אים ניצן. אין

אגענטישע סיסטעמען זענען די געצייג א בריק צווישן דעם מאדעל'ס טראכט-פעאיגקייט און ממשות'דיגע שריט אין דער וועלט. אזוי קענען אגענטן דורכפירן פראקטישע ארבעטן, נישט נאר געבן ענטפערס.

ווי אזוי מען פלאנירט געצייג פאר אן אגענט (41:28:38)

די זיבעטע פראגע: ווי אזוי פלאנירט און באשטימט מען געצייג פאר אן AI אגענט? מען דארף שאפן פעאיגקייטן וואס זענען פשוט, ספעציפיש, און גרינג פארן מאדעל צו פארשטיין. יעדער געצייג דארף האבן א קלארן ציל, ריכטיגע פאדערונגען פארן אינפוט, און אויטפוט וואס איז ארדנטליך אויסגעשטעלט.

עס איז בעסער צו שאפן קלענערע געצייג, יעדער פאר א באשטימטער ארבעט, ווי איין גרויסער געצייג פאר אלעס. למשל, באזונדערע געצייג צו קוקן א קונה'ס פרטים, צו דערהיינטיגן אדרעסן, אדער צו באהאנדלען צוריקצאלונגען (refunds).

די באשרייבונג פון יעדן געצייג דארף קלאר ערקלערן ווען דער אגענט זאל אים ניצן. די פראגראמירער דארפן אויך צולייגן נאכקוקונג (validation), זיכערהייט-קאנטראל און באהאנדלונג פון פעלערן, כדי דער אגענט זאל נישט טון קיין אומריכטיגע שריט. גוט געבויטע געצייג העלפן אגענטן באשליסן בעסער, פארמינערן דורכפאלן, און ארבעטן זיכער מיט דרויסנדיגע סיסטעמען.

אגענט-זכרון און זיינע סארטן (41:29:08)

די אכטע פראגע: וואס איז אגענט-זכרון, און פארוואס איז ער וויכטיג? מיט זכרון קענען AI סיסטעמען אפהיטן און דערמאנען אינפארמאציע פון פריערדיגע שמועסן. דאס העלפט זיי האלטן דעם קאנטעקסט און געבן ענטפערס וואס זענען צוגעפאסט צום באניצער. אן זכרון באהאנדלט דער אגענט יעדן שמועס ווי א נייע סעסיע, און ער געדענקט נישט דעם באניצער'ס פארליבן אדער וואס איז פריער געטון געווארן. מיט זכרון געדענקט דער אגענט זאכן ווי דעם באניצער'ס פאדערונגען, פריערדיגע ארבעטן, אדער וויכטיגע געשעפט-אינפארמאציע.

- א פערזענליכער געהילף קען געדענקן וועלכע צייטן דער באניצער האט ליב פאר מיטינגס, און ווי אזוי ער האט ליב צו קאמוניצירן.
- ביי קונה-שטיצע קען דער אגענט מיט זכרון אריינקוקן אין פריערדיגע שמועסן און פארשטיין דעם קונה'ס היסטאריע.

א גוטער זכרון פארבעסערט די דערפארונג פונעם באניצער, פארמינערט פראגעס וואס וועלן ווידער און ווידער געפרעגט, און העלפט דעם אגענט באהאנדלען לאנג-טערמינדיגע ארבעטן. אבער מען דארף פארוואלטן דעם זכרון מיט זהירות, כדי צו היטן פריוואטקייט און זיכערהייט, און כדי די אינפארמאציע זאל בלייבן שייך צום ענין.

די ניינטע פראגע: וועלכע סארטן זכרון ניצן AI אגענטן?

- קורץ-טערמיניגער זכרון האלט דעם קאנטעקסט פונעם יעצטיגן שמועס, אריינגערעכנט די לעצטע מעסעדזשעס און צייטווייליגע אינפארמאציע וואס מען דארף פאר דער ארבעט.
- לאנג-טערמיניגער זכרון היט אפ אינפארמאציע איבער עטליכע סעסיעס, ווי דעם באניצער'ס פארליבן און פריערדיגע שמועסן.
- עפיזאדישער זכרון (episodic) העלפט געדענקן פריערדיגע דערפארונגען.
- סעמאנטישער זכרון (semantic) היט אפ פאקטן, ווי פירמע-כללים אדער אינפארמאציע וועגן פראדוקטן.

- פראצעדוראלער זכרון (procedural) היט אפ ארבעטס-פראצעסן, ווי די שריט וואס מען דארף טון כדי צו פארענדיגן א באשטימטע ארבעט.

א ביישפיל: א קונה-שטיצע אגענט קען ניצן סעמאנטישן זכרון צו פארשטיין די פירמע-כללים, און פראצעדוראלן זכרון צו גיין לויט'ן סדר פון צוריקצאלונגען. די סארטן זכרון מאכן דעם אגענט מער קאנסעקווענט, מער צוגעפאסט צום באניצער, און מער פעאיג צו באהאנדלען קאמפליצירטע ארבעטן. דער קורס רעקאמענדירט צו מאכן א סקרינשאט פון דער טאוול, כדי מען זאל קענען איבערחזרן איידער מען גייט צו אן אינטערוויו.

דער אגענט-לופ און ווען ער שטעלט זיך אפ (41:31:44)

די צענטע פראגע: וואס איז אן אגענט-לופ, און ווי אזוי באשליסט ער ווען אויפצוהערן? דער אגענט-לופ איז א שטענדיגער ציקל אין וועלכן דער אגענט טראכט, טוט שריט, באטראכט רעזולטאטן, און באשליסט וואס צו טון ווייטער. א פשוטער LLM גיט איין ענטפער; אן אגענט קען טון אסאך שריט ביז דער ציל איז דערגרייכט. דער געוועהנליכער סדר:

- פארשטיין דעם ציל;
- אויסקלייבן א שריט מיט געצייג;
- אנאליזירן דעם אויטפוט;
- ווייטער גיין אויב מען דארף נאך שריט.

א ביישפיל: א קאודירונג-אגענט קען אנאליזירן א פעלער, ענדערן דעם קאוד, לויפן טעסטס, און דאס איבערחזרן ביז דער פראבלעם איז פארריכט.

דער אגענט שטעלט זיך אפ אין דריי פעלער: ווען די ארבעט איז פארטיג, ווען ער גיט ארויס דעם לעצטן ענטפער, אדער ווען ער דערגרייכט גרעניצן וואס זענען פאראויס באשטימט, ווי צייט, קאסטן אדער א מאקסימום צאל שריט. ריכטיגע אפשטעל-באדינגונגען פארמיידן אומנויטיגע שריט און מאכן די סיסטעם מער פארלעסליך.

Model Context Protocol (MCP) (41:32:47)

די עלפטע פראגע: וואס איז MCP? Model Context Protocol (MCP) איז א סטאנדארט וואס לאזט AI מאדעלן זיך פארבינדן מיט דרויסנדיגע געצייג, אפליקאציעס און דאטא-קוואלן אויף א גלייכמעסיגן אופן. ער מאכט גרינגער דאס פארבינדן AI אגענטן מיט פארשידענע סיסטעמען, ווייל ער גיט א געמיינזאמען אופן ווי אזוי זיי זאלן קאמוניצירן. מיט MCP קענען אגענטן גרינג אריינגיין אין מיטלען ווי דאטאבאזעס, פיילס, API'ס און געשעפט-אפליקאציעס, דורך MCP סערווערס. למשל, אן AI געהילף קען ניצן MCP צו ארויסנעמען דאקומענטן פון דער פירמע'ס וויסן-באזע, אדער צו ארבעטן מיט וווייכווארג-געצייג. MCP מאכט די אנטוויקלונג פון אגענטן מער אויסברייטבאר, מער זיכער, און מער איבערניצבאר, ווייל עס גיט א סטאנדארטן אופן ווי אזוי צו פארבינדן AI מאדעלן מיט דרויסנדיגע פעאיגקייטן.

דערמיט ענדיגן זיך די אנהייבער-פראגעס. דער קורס רעקאמענדירט זיי איבערצוחזרן איידער מען גייט צו אן אינטערוויו, און מען קען לאזן אנדערע פראגעס אין די באמערקונגען.

מיטל-שטאפל: Context Engineering (41:34:19)

נאך א קורצע הפסקה גייט מען איבער צו פראגעס פונעם מיטל-שטאפל. די ערשטע: וואס איז context engineering אין אגענטישן context engineering? AI איז דער פראצעס פון פארוואלטן און פארבעסערן די אינפארמאציע וואס מען גיט אן AI אגענט, כדי ער זאל קענען באשליסן פינקטליך. מען קלייבט אויס די ריכטיגע אנווייזונגען, זכרון, ארויסגענומענע וויסן, און אויטפוט פון געצייג, אלעס צו דער ריכטיגער צייט. פראמפט-אינזשעניערינג קוקט אויף דעם פראמפט אליין; context engineering פארוואלט די גאנצע סביבה אין וועלכער דער אגענט ארבעט.

א ביישפיל: א קונה-שטיצע אגענט דארף נישט האבן צוטריט צו דער גאנצער דאטאבאזע פון דער פירמע. ער דארף נאר די פרטים פונעם קונה וואס זענען שייך, די אינפארמאציע וועגן דער באשטעלונג, און די כללים וואס זענען שייך צום פאל. מעטאדן ווי RAG, פארוואלטונג פון זכרון, צוזאמדריקן דעם קאנטעקסט, און דינאמישער ארויסנעמען פון אינפארמאציע העלפן געבן ניצליכע אינפארמאציע און פארמינערן אומנויטיגע דאטא. א גוטע context engineering פארבעסערט די פינקטלעכקייט, פארמינערט "האלוצינאציעס", באנוצט ווייניגער טאקענס, און לאזט אגענטן דורכפירן קאמפליצירטע ארבעטן מער פארלעסליך.

צוזאמדריקן דעם קאנטעקסט ביי לאנג-לויפנדיגע אגענטן (41:35:21)

די קומענדיגע פראגע: ווי אזוי ארבעט context compaction) צוזאמדריקן דעם קאנטעקסט) ביי אגענטן וואס לויפן לאנג? דאס איז א מעטאד צו פארוואלטן א גרויסע מאסע שמועס-היסטאריע און אינפארמאציע, אזוי אז מען זאל נישט אריבערגיין דעם מאדעל'ס קאנטעקסט-פענסטער. ווען דער אגענט ארבעט ווייטער, ווערן די מעסעדזשעס, די אויטפוט פון געצייג און די צווישן-רעזולטאטן זייער גרויס. דעריבער שיקט מען נישט יעדעס מאל די גאנצע היסטאריע. מען דריקט עס צוזאם: מען נעמט ארויס וואס איז נישט נויטיג, און מען היט אפ די וויכטיגע פרטים וואס מען דארף פאר צוקונפטיגע באשלוסן. דער אגענט קען:

- מאכן א קיצור פון עלטערע שמועסן;
- אוועקנעמען אויטפוט פון געצייג וואס איז נישט שייך;
- האלטן נאר די קריטישע אינפארמאציע: שריט וואס זענען שוין געטון געווארן, די יעצטיגע צילן, די באגרעניצונגען, און די ארבעטן וואס ווארטן נאך.

א ביישפיל: א קאודירונג-אגענט וואס ארבעט אויף א גרויסן פראיעקט דארף נישט געדענקן יעדן פריערדיגן באפעל. ער דארף אבער וויסן די יעצטיגע ענדערונגען אינעם קאוד, די פעלערן, און די ארבעטן וואס זענען נאך געבליבן. context compaction מאכט די ארבעט מער עפעקטיוו, באנוצט ווייניגער טאקענס, קאסט ווייניגער, און העלפט דעם אגענט האלטן זיין שטארקייט בשעת לאנגע ארבעטס-פראצעסן.

איין אגענט קעגן אסאך אגענטן (41:36:24)

די קומענדיגע פראגע: וואס איז דער חילוק צווישן א סיסטעם מיט איין אגענט און א סיסטעם מיט אסאך אגענטן? ביי א סיסטעם מיט איין אגענט באהאנדלט איין AI אגענט די גאנצע ארבעט. ער טראכט, פלאנירט און פירט אויס, אלעס אין דער זעלבער סיסטעם. אזא סיסטעם איז פשוטער צו בויען און גרינגער צו אונטערהאלטן, און זי ארבעט גוט פאר ארבעטן וואס פאדערן נישט קיין עטליכע פעלדער פון ספעציאליזאציע. א סיסטעם מיט אסאך אגענטן טיילט אויף א קאמפליצירטע ארבעט צווישן עטליכע ספעציאליזירטע אגענטן. יעדער אגענט קאנצענטרירט זיך אויף א באשטימטער אחריות, און ארבעט צוזאמען מיט די אנדערע כדי

אנצוקומען צו א געמיינזאמען ציל. דער ביישפיל וואס הייבט זיך דא אן איז א פראצעס פון ווייכווארג-אנטוויקלונג, וואו יעדער אגענט באקומט זיין טייל.

אינטערוויו-פראגעס: מולטי-אגענטן, זיכערהייט און אדוואנסירטע טעמעס (41:37:24)

איינציקער אגענט קעגן א סיסטעם פון אסאך אגענטן (41:37:24)

דער קורס ענדיגט די ערקלערונג פונעם אונטערשייד צווישן א סיסטעם מיט איין אגענט און א מולטי-אגענט סיסטעם. אין א מולטי-אגענט סיסטעם קען למשל איין אגענט שרייבן קאוד, א צווייטער קען איבערקוקן די קוואליטעט פונעם קאוד, און א דריטער קען דורכפירן טעסטס.

מולטי-אגענט סיסטעמען ערמעגליכן ספעציאליזאציע, און עס קענען לויפן עטליכע ארבעטן פאראלעל. אבער זיי ברענגען אויך צו מער קאמפליצירטקייט: מען דארף זיך ארומנעמען מיט קאמוניקאציע, קאארדינאציע און באהאנדלונג פון טעותים.

צי מען דארף א סיסטעם מיט איין אגענט אדער מיט אסאך אגענטן, הענגט אפ אין דריי זאכן: ווי קאמפליצירט דער פראבלעם איז, וועלכע מומחיות מען דארף, און וויפיל צוזאמענארבעט עס פעלט אויס צווישן די פארשידענע ארבעטן.

ווי אגענטן רעדן זיך אפ און ארבעטן צוזאמען (41:37:54)

די קומענדיגע פראגע איז ווי אסאך AI אגענטן קאמוניקירן און ארבעטן צוזאמען. עס ווערט ערקלערט אז אגענטן שיקן זיך איבער סטרוקטורירטע מעלדונגען, טיילן זיך מיט וויכטיגע אינפארמאציע, און קאארדינירן די ארבעטן דורך אן "ארקעסטראציע-שיכט" (orchestration layer). יעדער אגענט האט געווענליך זיין באשטימטע ראלע, למשל פארשונג, אנאליז, קאוד-שרייבן אדער באשטעטיגונג. זיי ארבעטן צוזאמען דורכדעם וואס זיי גיבן איבער רעזולטאטן און דערהיַנטיגונגען איינער דעם אנדערן.

כדי די צוזאמענארבעט זאל גיין גוט, דארפן די אגענטן קלארע קאמוניקאציע-פארמאטן, באשטימטע אחריות, און א קאנטראלירטן צוטריט צו געמיינזאמע אינפארמאציע.

דער ביישפיל אין דער לעקציע איז א פראצעס פון שאפן אינהאלט: איין אגענט פארשט די טעמעס, א צווייטער שרייבט דעם אינהאלט, און א דריטער קוקט איבער צי אלעס איז פונקטליך איידער מען פארעפנטלעכט עס. די סטרוקטורירטע מעלדונגען העלפן די אגענטן קלאר פארשטיין די אריינגאנג און ארויסגאנג. די ארקעסטראציע פירט דערווייל די פארטיילונג פון ארבעט און דעם גאנג פונעם פראצעס.

דער סיסטעם דארף אויך באהאנדלען סתירות. דאס טוט מען דורך באשטעטיגונג-כללים, און דורכדעם וואס מען באשליסט וועמענס ארויסגאנג באקומט פארצוג. א גוט-אויסגעטראכטע קאמוניקאציע העלפט דער מולטי-אגענט סיסטעם ארבעטן עפעקטיוו, מיט ווייניגער טעותים און אן איבעריגע אינטעראקציעס. דער קורס באמערקט אז מען קען זיכער ערווארטן אזא פראגע ביי אן אינטערוויו.

AI סאב-אגענטן און אגענט-סקילס (41:39:26)

די פראגע וואס ווערט באצייכנט אלס איינע פון די מערסט ערווארטע ביי אינטערוויוס: וואס זענען AI סאב-אגענטן, און ווען זאל מען זיי ניצן?

דער ענטפער: סאב-אגענטן זענען ספעציאליזירטע אגענטן וואס דער הויפט-אגענט שאפט אדער רופט, כדי זיי זאלן באהאנדלען באשטימטע טיילן פון א ארבעט. אנשטאט אז איין אגענט זאל טראגן אלע אחריות, ערמעגליכן סאב-אגענטן אז פארשידענע אגענטן זאלן זיך קאנצענטרירן אויף באשטימטע פאאיקייטן, געצייג אדער געביטן. מען ניצט זיי ווען א ארבעט איז קאמפליצירט, ווען זי פאדערט ספעציעלע וויסן, אדער ווען זי באשטייט פון עטליכע אומאפהענגיגע טעטיגקייטן.

דער ביישפיל: א אגענט פאר ווייכווארג-אנטוויקלונג קען ניצן באזונדערע סאב-אגענטן פאר שאפן קאוד, פאר זיכערהייט-איבערקוק, פאר טעסטס און פאר דאקומענטאציע.

סאב-אגענטן מאכן די ארבעט עפעקטיווער, ווייל ארבעטן קענען לויפן פאראלעל, און יעדער אגענט ארבעט מיט קאנצענטרירטע אנווייזונגען און א באגרעניצטן קאנטעקסט. זיי מאכן אויך דעם סיסטעם מער פארלעסלעך, ווייל א ספעציאליזירטער אגענט איז אפט בעסער אין זיין באשטימטער אחריות. אבער צו פיל סאב-אגענטן קענען מאכן דעם סיסטעם צו קאמפליצירט. דעריבער זאל מען זיי אריינברענגען נאר ווען ספעציאליזאציע אדער פאראלעלע ארבעט גיט א קלארע מעלה.

די קומענדיגע פראגע: וואס זענען אגענט-סקילס (?agent skills דאס זענען פאאיקייטן וואס מען קען ווידער און ווידער ניצן, און וואס א AI אגענט ניצט כדי אויסצופירן באשטימטע ארבעטן. א סקיל פאראייניגט אין איין פעקל: אנווייזונגען, געצייג, די ערווארטע אריינגאנג און די לאגיק פון אויספירן. דער אגענט קען דעם פעקל רופן ווען עס פעלט אויס. אנשטאט צו בויען די זעלבע פאאיקייט נאכאמאל און נאכאמאל, קען א פראגראמירער שאפן א סקיל איינמאל, און לאזן פארשידענע אגענטן עס ניצן.

ביישפילן פון סקילס: העלפן א AI געהילף זוכן אין א וויסן-באזע, שאפן באריכטן, אנאליזירן דאטא, אדער באהאנדלען בקשות פון קונים. סקילס מאכן די אגענטן מער מאדולאר און גרינגער אויסצובייטערן, ווייל מען קען צולייגן נייע פאאיקייטן אן איבערבויען דעם גאנצן סיסטעם. זיי מאכן אויך די ארבעט מער גלייכמעסיג, ווייל דער אגענט פאלגט א פריער-באשטימטן וועג. אין פראדוקציע-סביבות העלפן גוט-געבויטע סקילס ארגאניזאציעס בויען אגענטן וואס קענען וואקסן, מיט פונקציאנאליטעט וואס מען ניצט ווידער אין פארשידענע פראצעסן און אפליקאציעס.

פארוואלטונג פון "סטעיט" און ארקעסטראציע (41:41:30)

א פראגע פון מיטל-שטאפל: ווי פירט מען סטעיט-מענעדזשמענט (state management) אין קאמפליצירטע אגענט-פראצעסן?

עס ווערט ערקלערט אז דאס מיינט נאכפאלגן און אפהיטן די אינפארמאציע וואס א אגענט דארף כדי ווייטער צו פירן א ארבעט. האלטן די גאנצע סטעיט-אינפארמאציע אינעם קאנטעקסט פונעם מאדעל איז נישט עפעקטיוו, ווייל דער קאנטעקסט-פענצטער ווערט צו גרויס. דעריבער האלט מען די וויכטיגע אינפארמאציע געוועהנליך אינדרויסן, אין דאטא-באזעס אדער אין באזונדערע שפייכלער-סיסטעמען, און מען ברענגט זי צוריק ווען מען דארף.

דער ביישפיל: א געשעפט-אויטאמאציע אגענט וואס לויפט א לאנגע צייט קען אפהיטן די שריט וואס ער האט שוין דורכגעפירט. ווען עס פאסירט א דורכפאל, קען ער ווייטער פירן פונעם לעצטן "טשעקפוינט". א גוטע סטעיט-פארוואלטונג מאכט דעם סיסטעם מער פארלעסלעך, ערמעגליכט זיך אויפצוהייבן נאך א דורכפאל, און פארמיידט אז דער אגענט זאל איבערחזרן ארבעטן וואס זענען שוין פארטיג. זי העלפט אויך האלטן אלעס גלייכמעסיג ווען עטליכע אגענטן ארבעטן צוזאמען אויף דער זעלבער ארבעט.

די קומענדיגע פראגע: וואס איז אגענט-ארקעסטראציע, און ווי פירט מען עס דורך? ארקעסטראציע איז דער פראצעס פון פירן ווי פארשידענע AI אגענטן, געצייג און פראצעסן ארבעטן צוזאמען כדי אויספירן א ארבעט. זי באשליסט וועלכער אגענט זאל באהאנדלען א געוויסע טעטיגקייט, ווי מען גיט איבער אינפארמאציע צווישן די אגענטן, און ווי מען קאנטראלירט די גאנצע ארבעט. אין א מולטי-אגענט סיסטעם איז די ארקעסטראציע א קאארדינאטאר: זי טיילט אויס ארבעטן, פאלגט נאך דעם פארשריט, פירט די אפהענגיגקייטן צווישן ארבעטן, און באהאנדלט דורכפאלן.

דער ביישפיל: אין א AI סיסטעם פאר ווייכווארג-אנטוויקלונג קען דער ארקעסטראטאר געבן קאוד-שרייבן צו א "דעוועלאפער-אגענט", טעסטס צו א QA-אגענט, און זיכערהייט-קאנטראלן צו א איבערקוק-אגענט.

צו דורכפירן ארקעסטראציע דארף מען געווענליך באשטימען:

- די ראלעס פון די אגענטן;
- קאמוניקאציע-פראטאקאלן;
- די לאגיק ווי צו פארטיילן ארבעטן (task routing);
- מעכאניזמען פאר באהאנדלען טעותים.

א גוטע ארקעסטראציע זארגט אז די אגענטן זאלן עפעקטיוו צוזאמענארבעטן, נישט טון די זעלבע ארבעט צוויי מאל, און פארטיג מאכן די ארבעטן.

ווי מען אפשאצט און טעסט AI אגענטן (41:43:32)

די צוואנציגסטע פראגע: ווי אפשאצט און טעסט מען AI אגענטן? מען דארף טעסטן נישט נאר דעם לעצטן ענטפער, נאר אויך דעם גאנצן וועג וואס דער אגענט איז געגאנגען. אנדערש ווי ביי געווענליכע ווייכווארג-טעסטס, קוקט מען דא צי דער סיסטעם פארשטייט ריכטיג די צילן, קלייבט אויס פאסיגע שריט, ניצט ריכטיג די געצייג, און פירט מיט הצלחה אויס די ארבעטן.

וויכטיגע פונקטן ביים אפשאצן:

- וויפיל פראצענט פון די ארבעטן ווערן מיט הצלחה אויסגעפירט;
- די קוואליטעט פונעם טראכטן (reasoning);
- די פונקטלעכקייט ביים רופן געצייג (tool calling);
- די קוואליטעט פונעם ענטפער;
- זיכערהייט;
- לאטענסי (ווי לאנג עס נעמט);
- קאסטן.

א ריכטיגער אפשאצונג-פראצעס ניצט א געזאמלטן טעסט דאטא-סעט מיט פארשידענע מצבים: נארמאלע פעלער, עק-פעלער (edge cases), און אריינגאנג וואס איז מיט כוונה אויסגעשטעלט צו שאטן (adversarial). א פראגראמירער דארף אויך טעסטן ווי דער אגענט פירט זיך ווען געצייג פאלן דורך, אדער ווען מען גיט אים אומערוואַרטעטע אינפארמאציע. רעגרעסיע-טעסטס זענען וויכטיג יעדעס מאל וואס מען טוישט פראמפטס, מאדעלן, געצייג אדער פראצעסן. א שטענדיגע אויפזיכט העלפט טרעפן פראבלעמען מיט דער ארבעט און

פארבעסערן דעם אגענט מיט דער צייט. א גוט-געטעסטער אגענט דארף זיין פונקטליך, פארלעסלעך, עפעקטיוו און זיכער אין דער אמתער וועלט.

באהאנדלען דורכפאלן און זיך אויפהייבן דערפון (41:44:33)

די קומענדיגע פראגע וועלט באצייכנט אלס זייער ערווארטעט, און מען דארף זי גוט חזרן: ווי באהאנדלט מען דורכפאלן פון אגענטן, און ווי פירט מען דורך דאס זיך אויפהייבן פון טעותים?

מען דארף אויסטראכטן מעכאניזמען וואס לאזן דעם סיסטעם רעאגירן אויף א זיכערן אופן וועו עפעס גייט שלעכט. דער קורס חזרט איבער אז אגענטן קענען דורכפאלן צוליב פאלשן באניץ פון געצייג, סערוויסעס וואס זענען נישט פאראן, אומגילטיגן אריינגאנג, אדער אומערווארטעטע באשלוסן פונעם מאדעל.

די שריט פון א גוטער שטראטעגיע:

- באשטעטיגן דעם אריינגאנג און ארויסגאנג פון יעדן געצייג איידער מען גייט ווייטער צום קומענדיגן שריט;
- צייטווייליגע דורכפאלן באהאנדלט מען מיט נאכאמאל-פרובירן (retry), מיט טעכניקעס ווי exponential backoff;
- מען שטעלט ארויס צייט-גרעניצן (timeouts);
- מען שטעלט צו אן אלטערנאטיוו (fallback), למשל א אנדער געצייג אדער מאדעל וועו דער ערשטער פאלט דורך;
- ביי פראצעסן וואס לויפן לאנג, היט מען אפ וויכטיגע סטעיט-אינפארמאציע מיט טשעקפוינטס, אזוי אז דער אגענט קען ווייטער פירן פונעם לעצטן געלונגענעם שריט, אנשטאט אנהייבן פון אנפאנג;
- וועו דורכפאלן חזרן זיך איבער, דארף דער סיסטעם זיך אפשטעלן אויף א זיכערן אופן, ערקלערן דעם פראבלעם, און וועו נויטיג איבערגעבן די זאך צו א מענטש.

א גוטע באהאנדלונג מאכט דעם סיסטעם מער פארלעסלעך, און פארמיידט אז אגענטן זאלן מאכן פאלשע אדער געפערליכע באשלוסן.

מענטש-אין-דער-שלייף (HITL) און "גארדרעילס" (41:46:05)

פארוואס איז דער מוסטער פון Human in the Loop (HITL) וויכטיג? דאס איז א צוגאנג וואו מענטשן איבערקוקן, באשטעטיגן אדער פארריכטן די באשלוסן פון א AI אגענט, בפרט אין מצבים וואו א טעות קען האבן ערנסטע תוצאות. AI קען אויטאמאטיזירן אסאך ארבעטן, אבער געוויסע באשלוסן פאדערן א מענטשליכן שיקל-הדעת, א מענטש וואס נעמט אחריות, אדער פאך-וויסן.

HITL איז וויכטיג ביי באשלוסן מיט גרויסע השפעה, ביי שריט וואס מען קען נישט צוריקדרייען, ביי געלט-טראנזאקציעס, ביי ענדערונגען אין סענסיטיווע דאטא, ביי זיכערהייט-פראגעס, און ביי לעצטע באשטעטיגונג-פראצעסן. ביישפילן: א AI אגענט קען צוגרייטן א רעקאמענדאציע צי צו באשטעטיגן א הלוואה, אבער א מענטש זאל מאכן דעם לעצטן באשלוס. א קונה-סערוויס אגענט קען אליין באהאנדלען נארמאלע בקשות, אבער קאמפליצירטע טענות גיט ער איבער צו א מענטשליכן פארשטייער. HITL פארגרעסערט דאס צוטרוי, פארמינערט ריזיקע, און זארגט אז AI סיסטעמען ארבעטן אין די גרעניצן פון געשעפט און עטיק. עס שאפט א באלאנס צווישן דער עפעקטיווקייט פון אויטאמאציע און מענטשליכער אויפזיכט.

די קומענדיגע פראגע: ווי פארמיידן "גארדרעילס" (guardrails) געפערליכע שריט אין אגענטיק AI סיסטעמען? גארדרעילס זענען זיכערהייט-מעכאניזמען וואס קאנטראלירן צו וואס א אגענט האט צוטריט, וואס ער מעג באשליסן, און וואס ער מעג אויספירן. זיי העלפן פארמיידן שעדליכע, פאלשע אדער נישט-ערלויבטע שריט. מען קען זיי איינפירן אויף פארשידענע שטאפלען: באשטעטיגן דעם אריינגאנג, באגרענצן געצייג, קאנטראלירן ערלויבעניש, כללים פון פאליסי, פילטערן דעם ארויסגאנג, און פונקטן וואו א מענטש מוז באשטעטיגן.

ביישפילן: א פינאנץ-אגענט מעג אנאליזירן טראנזאקציעס, אבער ער דארף באשטעטיגונג פון א מענטש איידער ער שיקט איבער געלט. א קונה-סערוויס אגענט קען זען אינפארמאציע וועגן קונים, אבער נאר דעם מינימום דאטא וואס ער דארף פאר זיין ארבעט. נאך קאנטראלן, ווי באשטעטיגונג, גרעניצן אויף שריט, "אודיט-לאגס" און אויפזיכט, העלפן טרעפן און פארמיידן געפערליך אויפפירונג. גארדרעילס זענען נויטיג, ווייל AI אגענטן קענען אליין מאכן באשלוסן און ארבעטן מיט אמתע סיסטעמען. זיי זארגן אז די אגענטן זאלן בלייבן פארלעסלעך, זיכער, און לויט די כללים פון דער ארגאניזאציע.

רעפלעקציע און קאנטראל איבער טאקענס און קאסטן (41:48:06)

וואס איז אגענט-רעפלעקציע, און ווי פארבעסערט עס די ארבעט? רעפלעקציע איז דער פראצעס וואו דער אגענט קוקט איבער זיינע פריערדיגע שריט און ארויסגאנג איידער ער ענדיגט דעם ענטפער. אנשטאט גלייך אננעמען דעם ערשטן רעזולטאט, מאכט דער אגענט א צווייטן אפשאצונג-שריט, כדי צו טרעפן טעותים, פעלנדיגע אינפארמאציע אדער מעגליכע פארבעסערונגען. רעפלעקציע העלפט קאנטראלירן צי דער רעזולטאט ערפילט טאקע דעם ארגינעלן ציל, און צי דער טראכט-פראצעס איז געווען עפעקטיוו.

דער ביישפיל: א קאוד-אגענט קען איבערקוקן דעם קאוד וואס ער האט געשריבן, טרעפן מעגליכע באגס, און פארבעסערן די לעזונג איידער ער גיט עס אפ. רעפלעקציע קען אויך אנטפלעקן פאלשע הנחות, פעלנדיגע ראיות אדער טעותים פון געצייג. אבער שטענדיגע רעפלעקציע פארגרעסערט די קאסטן און די לאטענסי, דעריבער זאל מען זי ניצן מיט חשבון, וועו די ארבעט פאדערט העכערע פונקטלעכקייט. מיט דער פעאיקייט זיך אליין אפצושאצן, ווערן אגענטן מער פארלעסלעך און פעאיג צו באהאנדלען קאמפליצירטע פראבלעמען.

ווי פירט מען דעם באניץ פון טאקענס און די קאסטן אין פראצעסן וואס לויפן לאנג? דאס איז וויכטיג, ווייל אזעלכע פראצעסן שאפן גרויסע מאסן פון שמועס-היסטאריע, ארויסגאנג פון געצייג, און צווישן-שריט פון טראכטן. די עצות:

- ניצן קלענערע און שנעלערע מאדעלן פאר פשוטע ארבעטן, ווי פארטיילן (routing), קלאסיפיצירן אדער ארויסנעמען אינפארמאציע, און האלטן די שטארקע מאדעלן פאר קאמפליצירטע טראכטן;
- טעכניקעס פון קאנטעקסט-פארוואלטונג, ווי צוזאמענפאסן (summarization) און צוזאמענדריקן דעם קאנטעקסט (context compression), כדי נישט צו שיקן איבעריגע אינפארמאציע צום מאדעל;
- אפהיטן אין "קעש" (cache) פראמפטס וואס חזרן זיך איבער, אראפגעברענגטע דאקומענטן, און רעזולטאטן פון געצייג, כדי נישט צו טון די זעלבע ארבעט צוויי מאל;
- באגרעניצן איבעריגע שריט פונעם אגענט, קאנטראלירן דעם באניץ פון געצייג, און שטעלן א בודזשעט פאר אויספירן;
- נאכפאלגן ווי פיל טאקענס מען פארברויכט, די לאטענסי און די ארבעט פונעם מאדעל, כדי צו אפטימיזירן.

אזוי קענען ארגאניזאציעס בויען אגענטן וואס קענען וואקסן, מיט גוטע ארבעט און קאנטראלירטע הוצאות. דערמיט ענדיגט זיך דער טייל פון מיטל-שטאפל פראגעס. דער קורס רעקאמענדירט זיי אלע גוט צו חזרן פאר יעדן אינטערוויו, און דערנאך גייט מען איבער צו אדוואנסירטע פראגעס.

אדוואנסירטע פראגעס: קאוד-אגענטן און מולטי-מאדאלע סיסטעמען (41:51:08)

די ערשטע אדוואנסירטע פראגע: אין וואס אונטערשיידן זיך אגענטן וואס שרייבן קאוד פון אגענטן וואס רופן געצייג? דער הויפט-אונטערשייד איז ווי זיי פירן אויס די ארבעט. א קאוד-אגענט שאפט קאוד וואס מען קען לויפן, כדי צו לעזן פראבלעמען לויט די באדערפענישן פון יענעם מאמענט. ער איז נוצליך פאר בייגיגע ארבעטן וואו די לעזונג איז נישט פריער באשטימט, ווי בויען אפליקאציעס, אנאליזירן דאטא אדער פאררעכטן וועכווארג. אזעלכע אגענטן לויפן געווענליך דעם קאוד אין א קאנטראלירטן "סענדבאקס" צוליב זיכערהייט.

א געצייג-רופנדיגער אגענט, אנדערש, קלייבט אויס פריער-באשטימטע פונקציעס אדער APIs, מיט קאנטראלירטן אריינגאנג און ערלויבענישן. דער ביישפיל: א קונה-סערוויס אגענט קען רופן א API פאר צוריקגעבן געלט, אנשטאט שרייבן קאוד כדי דאס דורכצופירן. געצייג-רופנדיגע אגענטן זענען גרינגער צו קאנטראלירן און צו פארזיכערן. קאוד-אגענטן געבן מער בייגיגקייט, אבער זיי פאדערן שטארקערע זיכערהייט-מיטלען, ווי סענדבאקסינג, באשטעטיגונג, און גרענעצן אויף ניצן רעסורסן. די ברירה הענגט אפ צי די ארבעט פאדערט בייגיגקייט אדער שטרענגע קאנטראל.

די קומענדיגע פראגע: ווי באהאנדלט מען מולטי-מאדאלן אריינגאנג און ארויסגאנג און אגענטיק סיסטעמען? מולטי-מאדאלע סיסטעמען קענען פארשטיין און באארבעטן פארשידענע סארטן אינפארמאציע: טעקסט, בילדער, קול, ווידעאס און דאקומענטן. יעדער סארט דאטא ווערט קודם באארבעט מיט פאסיגע מאדעלן אדער "פארסערס", און איבערגעמאכט צו א פארמאט וואס דער אגענט פארשטייט. דערנאך פאראייניגט דער אגענט די אינפארמאציע מיט זיין יעצטיגן קאנטעקסט, און באשליסט וואס צו טון ווייטער. דער ביישפיל: א געזונטהייט-אגענט קען אנאליזירן מעדיצינישע בילדער צוזאמען מיט די נאטיצן וועגן דעם פאציענט, כדי צו געבן בעסערע פארשטענדעניש.

פראגראמירער דארפן זארגן אז מולטי-מאדאלע סיסטעמען האלטן אן די באווייזן פון וואו די אינפארמאציע קומט, היטן אפ וויכטיגע פרטים, און באהאנדלען סתירות צווישן די פארשידענע סארטן דאטא. דאס אפשאצן דארף נעמען אין באטראכט פונקטלעכקייט, צי דער ענטפער איז געגרינדעט אויף די מקורות (grounding), זיכערהייט, לאטענסי און קאסטן, ווייל טעותים קענען פאסירן סיי ביים אויפכאפן די אינפארמאציע און סיי ביים טראכטן. א ריכטיגע באהאנדלונג לאזט אגענטן לעזן מער קאמפליצירטע פראבלעמען פון דער אמתער וועלט.

די קומענדיגע פראגע וואס הייבט זיך אן: ווי וואלט מען געבויט א קונה-סערוויס אגענט מיט לאגיק פון איבערגעבן צו א מענטש (?escalation)

אינטערוויו-פראגעס און א וועג-קארטע פאר אגענטיק AI (41:53:42)

א קונה-סערוויס אגענט, און א קאוד-אגענט אין א "sandbox" (41:53:42)

דער קורס ערקלערט ווי אזוי מען בויט א קונה-סערוויס אגענט. אזא אגענט פאדערט א צוזאמשטעל פון שפראך-פארשטאנד, אויסזוכן וויסן פון א מקור, געצייג (tools), און א מעכאניזם וואס פירט די זאך ווייטער צו א מענטש (escalation). די סיסטעם הייבט אן מיט א "intent classifier". דאס איז א טייל וואס פארשטייט

וואס דער קונה וויל, און שיקט די בקשה אהין וואו זי געהערט. אלס פארלעסלעכער מקור פון וויסן קען דינען א RAG-סיסטעם. זי גיט פונקטליכע אינפארמאציע פון די פירמע'ס דאקומענטן און כללים (policies). דער אגענט קען ניצן געצייג כדי צוצוקומען צו פרטים פון א באשטעלונג, צו אינפארמאציע פון א חשבון, צו צוריקצאָלונגען (refunds), אדער צו סופּאָרט-טיקעטס.

מען דארף צולייגן א לאגיק פון escalation פאר געוויסע פעלער: וועז דער אגענט איז נישט זיכער, וועז ער טרעפט אן א זייטיגן אדער שארפן ענין, אדער וועז א בקשה פאדערט א הסכמה פון א מענטש. א ביישפיל: א בקשה פאר א צוריקצאָלונג וואס פאסט אריין אין די פירמע'ס כללים קען מען לאזן אויטאמאטיש. אבער קאמפליצירטע טענות, אדער פעלער וואס זענען אן אויסנאם פון די כללים, ווערן איבערגעגעבן צו א מענטשלעכן אגענט. דריי זאכן זארגן אז דער סופּאָרט-אגענט גיט פונקטליכע און זיכערע הילף: קאנטראלן פון זיכערקייט (confidence checks), די היסטאריע פונעם שמועס, און ריכטיגע גדרים (guardrails).

די ווייטערדיגע פראגע איז ווי אזוי מען בויט א זיכערן קאוד-אויספירנדיגן אגענט מיט א "sandbox". עס ווערט באטאנט אז דאס איז איינע פון די פראגעס וואס מען ערווארט מערסטנס ביי אינטערוויוס אויפן אַוואַנסירטן לעוועל. אזא אגענט דארף לויפן דעם קאוד וואס ער שאפט אינעם א פארזיכערטן "sandbox", כדי צו פארמיידן שעדלעכע אקציעס און צו באשיצן די סיסטעמען. דער sandbox דארף באגרעניצן:

- צוטריט צו די טעקעס (file system)
- נעץ-פארבינדונגען
- באניץ פון CPU
- באניץ פון זכרון (memory)
- די צייט פון אויספירן

נאר באשטעטיגטע ביבליאטעקן און ריסורסן טארן זיין צוגעשטעלט פארן אגענט. דער געשאפענער קאוד דארף ווערן געפרואווט (validated) פארן אויספירן. דעם רעזולטאט דארף מען נאכקוקן צי עס זענען דא טעותים אדער אומזיכערע אויפפירונגען. זייטיגע אינפארמאציע, ווי פאסווערטער (credentials) און פריוואטע דאטא, טאר קיינמאל נישט זיין אנטפלעקט אינעם sandbox. מען דארף פארשרייבן יעדע אויספירונג אין א לאג. דאס העלפט צו געפינען טעותים (debugging), צו קאנטראלירן (auditing), און צו פארבעסערן די פארלעסלעכקייט. א ביישפיל: א דאטא-אנאליז אגענט קען אויספירן Python קאוד אינעם א sandbox, און ער ווערט אפגעהאלטן פון צוקומען צו טעקעס אדער אינדרויסנדיגע סיסטעמען וואס ער האט נישט קיין רשות צו זיי. א פארזיכערטער sandbox לאזט פירמעס הנאה האבן פון די מעגלעכקייטן צו שאפן קאוד, און גלייכצייטיג פארמינערט עס די סכנות פון זיכערהייט.

"Computer use" אגענטן, און LangChain מיט LangGraph (41:55:44)

דער קורס ערקלערט אז "computer use" אגענטן זענען AI-סיסטעמען וואס קענען ארבעטן מיט א גראפישן אינטערפעיס (GUI) פונקט ווי א מענטשלעכער באניצער. זיי פארשטייען וואס עס שטייט אויפן עקראן, די ויזועלע עלעמענטן, און אין וועלכן מצב די אפליקאציע געפינט זיך. דערנאך באשליסן זיי וועלכע אקציע צו טון: דריקן א קנעפל, שרייבן טעקסט, סקראלן, אדער אויסקלייבן אן אפציע.

דער אגענט גייט אין א שטענדיגן קרייז (loop):

- ער באטראכט דעם עקראן

- ער טראכט וואס דער נעקסטער טריט דארף זיין
- ער טוט די אקציע
- ער קוקט נאך דעם נייעם מצב

א ביישפיל: אזא אגענט קען עפענען א וועבזייטל, אויספילן פארמען, זיך באוועגן אין אפליקאציעס, און פארענדיגן אן ארבעט, אלעס אן קיין דירעקטן צוטריט דורך אן API. אזעלכע אגענטן זענען נוצלעך ווען די סאפטווער-סיסטעמען האבן נישט קיין API, אדער ווען די אויטאמאציע פאדערט ארבעטן מיט די אינטערפעיסעס וואס זענען שוין דא. אבער מען דארף זיי האלטן מיט שטארקע קאנטראלן פון זיכערהייט, ווייל א פאלשע אקציע קען אנרירן פאקטישע אפליקאציעס.

די לעצטע פראגע פונעם אַוואַנסירטן לעוועל, וואס ווערט אויך באצייכנט אלס זייער ערווארטעט, איז: ווי אזוי שטיצן LangChain און LangGraph אגענטיק AI-אפליקאציעס? ביידע זענען פריימווערקס וואס העלפן פראגראמירער בויען אפליקאציעס וואס לויפן אויף גרויסע שפראך-מאדעלן און AI-אגענטן.

LangChain גיט צו טיילן וואס מען קען נוצן אויפסניי. זיי פארבינדן מאדעלן מיט פראמפטס, מיט געצייג, מיט אויסזוך-סיסטעמען (retrieval), מיט זכרון, און מיט אינדרויסנדיגע פארבינדונגען. דאס העלפט שאפן די געווינטלעכע LLM-ארבעט-פלוסן (workflows), ווי טשאטבאטס, RAG-אפליקאציעס, און אגענטן וואס ניצן געצייג.

LangGraph פארברייטערט די מעגלעכקייט. עס שטעלט פאר די ארבעט-פלוסן פונעם אגענט ווי גראַפן וואס האלטן א מצב (stateful graphs). יעדער טריט איז א "node", און דער פלוס צווישן די טריט ווערט קאנטראלירט דורך "edges". דאס איז נוצלעך פאר קאמפליצירטע ארבעט-פלוסן וואס פאדערן זכרון-פונקטן (checkpoints), נאכאמאל פרואוון (retries), הסכמה פון א מענטש, און באדינגטע ריכטונגען (conditional routing). א ביישפיל: א קונה-סערוויס אגענט וואס איז געבויט מיט LangGraph קען באשליסן לויטן מצב צי צו ענטפערן גלייך, צי אויסזוכן אינפארמאציע, אדער איבערגעבן דעם ענין ווייטער. צוזאמען העלפן LangChain און LangGraph בויען אגענטיק AI-סיסטעמען וואס זענען גרייט צו וואקסן, פארלעסלעך, און אונטער קאנטראל.

סצענאר-פראגעס: אן אומענדלעכער קרייז (41:58:19)

נאך די טעאריע-פראגעס גייט דער קורס איבער צו פראגעס וואס זענען געבויט אויף סצענארן, ווייל אזעלכע קען מען אויך ערווארטן ביי אן אינטערוויו. דער ערשטער פאל איז אן "infinite loop": דער AI-אגענט חזר'ט כסדר איבער די זעלבע רופן צו געצייג, און קומט קיינמאל נישט צו א לעצטן ענטפער. ווי אזוי דערקענט מען דאס, און ווי אזוי צעברעכט מען דעם קרייז?

דער ענטפער: אן אגענט פאלט אריין אין אן אומענדלעכן קרייז ווען ער טוט ווידער און ווידער די זעלבע אקציעס, און מאכט נישט קיין פארשריט צום ציל. כדי עס צו דערקענען דארף די סיסטעם נאכקוקן די שפורן פונעם אגענט (traces), דעם מוסטער פון ווי ער ניצט די געצייג, מצבים וואס חזר'ן זיך איבער, און גלייכע אריינגאבן וואס ווערן געשיקט עטליכע מאל. פראגראמירער קענען לייגן גרעניצן, כדי צו פארמיידן א פראצעס אן א סוף:

- מאקסימום צאל טריט
- א צייט-גרעניץ
- א בודזשעט פון טאקענס

דער אגענט דארף אויך נאכגיין דעם פארשריט נאך יעדער אקציע, און פרואוון צי דער לעצטער טריט האט געבראכט נייע אינפארמאציע. וועו ער זעט אז ער פעלט ווידער און ווידער, אדער אז עס איז נישט דא קיין פארשריט, קען ער מאכן א נייעם פלאן, טוישן די סטראטעגיע, זיך אפשטעלן אויף א זיכערן אופן, אדער איבערגעבן צו א מענטש. א ביישפיל: אויב א פארש-אגענט (research agent) רופט כסדר דאס זעלבע זוך-געצייג און געפינט נישט קיין נוצלעכע רעזולטאטן, דארף ער דערקענען דעם קרייז און פרואוון אן אנדערן וועג.

סתירות צווישן געצייגן, און א פאלשע אויסוואל פון געצייג (41:59:52)

דער צווייטער פאל: צוויי פארלעסלעכע געצייג געבן צוויי פארשידענע ענטפערס אויף דער זעלבער פאקט. ווי אזוי דארף דער אגענט פארגלייכן די סתירה? דער אגענט טאר נישט גלייך אויסקלייבן איין ענטפער אן נאכפרואוון. ער דארף פארגלייכן עטליכע פאקטארן:

- ווי פארלעסלעך דער מקור איז
- ווי פריש די אינפארמאציע איז
- דעם לעוועל פון זיכערקייט
- דעם ציל פון יעדן געצייג

ער קען נאכפרואוון ביי נאך א פארלעסלעכן מקור, אדער אנווענדן געשעפט-כללים וואס זענען פריער באשטימט, כדי צו באשטימען דעם ריכטיגן רעזולטאט. א ביישפיל: אויב צוויי דאטא-באזעס ווייזן פארשידענע באלאנסן אויף א קונה'ס חשבון, דארף דער אגענט נאכקוקן וועלכע סיסטעם ווערט געראכנט אלס דער אפיציעלער מקור פון אמת ("source of truth"). דער אגענט דארף אויך אויסמיידן צו שאפן א פאלשע זיכערקייט ווען די ראיות בלייבן נישט קלאר. ביי החלטות מיט גרויס השפעה, ווי געלט-טראנזאקציעס אדער ענדערונגען אין א סערווים, דארף מען די סתירה פארשרייבן און איבערגעבן צו א מענטשלעכן קאנטראלירער. א ריכטיגע באהאנדלונג פון סתירות פארבעסערט די פארלעסלעכקייט און פארמיידט פאלשע החלטות.

עס ווערט געגעבן אן עצה פארן אינטערוויו: קודם פארשטיין דעם פאל, וואס איז די פראבלעם און ווי אזוי מען קען זי לייזן. דערנאך טיף אטעמען, אדורכטראכטן אלע פונקטן אין קאפ, און ערשט דערנאך ענטפערן. מען זאל נישט אריינשפרינגען גלייך אין דער ערקלערונג.

דער דריטער פאל איז א פעלער אין אויסוואל פון געצייג. עס ווערט באטאנט אז די פראגעס זענען בלויז ביישפילן פון דעם סארט פראגעס, נישט די פונקטליכע פראגעס וואס מען וועט פרעגן. דער פאל: דער אגענט האט אסאך געצייג, און ער קלייבט אפט אויס דעם פאלשן, אדער גיט איבער פאלשע פאראמעטערס. ווי פאררעכנט מען דאס?

אזא פראבלעם קומט געווענליך פון אומקלארע באשרייבונגען פון די געצייג, צו פיל אפציעס, אדער א שוואכע קאנטראל פון די אריינגאבן. די לייזונגען:

- דער ערשטער טריט איז צו פארבעסערן די דעפיניציעס פון די געצייג: קלארע באשרייבונגען, ביישפילן פון באניץ, און ספעציפישע אנווייזונגען ווען מען דארף ניצן יעדן געצייג.
- די געצייג דארפן ניצן שטרענגע סכעמעס (schemas, enums), און קאנטראל פון פאראמעטערס, כדי צו פארמיידן פאלשע אריינגאבן.

- מען קען צולייגן א שיכט פון ריכטונג (routing layer). זי פילטערט און קלייבט אויס א קורצע רשימה פון שייכותדיגע געצייג, איידער דער LLM מאכט די לעצטע אויסוואל.
- פראגראמירער דארפן אנאליזירן די שפורן פון די רופן צו געצייג, כדי צו פארשטיין דעם מוסטער פון די פעלערן, און דערנאך פארבעסערן די פראמפטס אדער די כללים.

א ביישפיל: אנשטאט אנטפלעקן 20 געצייג וואס האבן נישט קיין שייכות איינס צום אנדערן אויף איינמאל, קען די סיסטעם קודם דערקענען וואס דער באניצער וויל, און צושטעלן בלויז די מערסט שייכותדיגע געצייג.

אקציעס וואס מען קען נישט צוריקדרייען (42:02:59)

דער לעצטער פאל: אן אויטאנאמער אגענט קען אויסמעקן דאטא, שיקן געלט, אדער ארויסגעבן אינהאלט. ווי אזוי פארמיידט מען אומזיכערע אקציעס וואס מען קען נישט צוריקדרייען? דאס פאדערט שטארקע קאנטראלן פון זיכערהייט, מעכאניזמען פון הסכמה, און באגרעניצטע רשותן. דער אגענט דארף גיין לויטן כלל פון "least privilege", דאס הייסט ער זאל האבן בלויז די מינימאלע מעגלעכקייטן וואס ער דארף פאר זיין ארבעט. אקציעס מיט גרויס השפעה, ווי אויסמעקן דאטא, איבערפירן געלט, אדער ארויספובליקירן אינהאלט, דארפן פאדערן א קלארע הסכמה פון א מענטש פארן אויספירן.

פראגראמירער קענען אויך צולייגן א "preview" אדער "dry run" מאדע, וואו דער אגענט ווייזט די אקציע וואס ער פלאנירט, איידער ער טוט זי. נאך שיצונגען:

- גרעניצן אויף טראנזאקציעס
- טריט פון באשטעטיגונג
- קאנטראלן פון) idempotencyאז די זעלבע אקציע זאל נישט ווערן געטון צוויי מאל)
- אפציעס צו צוריקדרייען (rollback)
- פרטימדיגע קאנטראל-לאגס (audit logs)

א ביישפיל: א פינאנץ-אגענט קען צוגרייטן א בקשה פאר א צאָלונג, אבער ער פאדערט א באשטעטיגונג פונעם באניצער איידער ער פארענדיגט די טראנזאקציע. אזעלכע שיצונגען זארגן אז AI-אגענטן בלייבן נוצלעך, און גלייכצייטיג פארמינערן זיי די סכנה פון אומגליקן אדער שעדלעכע אקציעס אין דער פאקטישער וועלט. צום סוף ווערט געראטן איבערצוחזור'ן די פעלער און אדורכצוגיין נאך פעלער, ווייל דאס זענען געווען בלויז עטליכע ביישפילן.

א וועג-קארטע פאר אגענטיק AI: פון ענטפערן צו טון (42:04:59)

דא הייבט זיך אן א נייער טייל: א פולע וועג-קארטע (roadmap) פאר אגענטיק AI. עס ווערט געזאגט אז ווער עס קוקט דאס אין 2026 גייט אריין אין איינע פון די וויכטיגסטע תקופות פון דער AI-רעוואלוציע. ביז לעצטנס האבן רוב מענטשן גענוצט AI ווי א זוך-שורה: מען שרייבט א פראגע און מען באקומט א ענטפער. מען בעט א בריוו, און עס שרייבט א בריוו. מען בעט א קיצור, און עס גיט א קיצור. די נעקסטע שטאפל איז אנדערש. AI גייט איבער פון בלויז ענטפערן אויף פראגעס צו פארענדיגן ארבעטן. עס קען פלאנירן טריט, ניצן געצייג, נאכקוקן אינפארמאציע, זיך פארבינדן מיט אפליקאציעס, און העלפן פארענדיגן ארבעט וואס האט פריער געפאדערט עטליכע מענטשן, עטליכע טאבס, און עטליכע טריט מיט די הענט.

א משל: ביי א געוועלטיגן טשאטבאט קען מען בעטן "שרייב א רייזע-פלאן", און עס גיט א פלאן. אן AI-אגענט קען גיין ווייטער. ער קען נאכקוקן וועלכע פליען זענען פאראן, פארגלייכן האטעלן, קוקן אויפן בודזשעט, שאפן

א רייזע-סדר (itinerary), און אפילו צוגרייטן א רשימה פאר באשטעלן. דער חילוק מאכט אגענטיק AI אזוי שטארק. מיט פשוטע ווערטער: אגענטיק AI איז AI וואס ענטפערט נישט נאר, נאר עס טוט. דערפאר גיבן אכטונג דערויף מקצוע-לייט, סטודענטן, פראגראמירער, מענעדזשערס, מארקעטינג-לייט, דאטא-אנאליסטן, און אפילו געשעפט-פירער.

דער World Economic Forum האט שוין אנגעוויזן אז דער גלאבאלער ארבעט-מארק טוישט זיך זייער שנעל. ביז 2030 ערווארט מען אז מיליאנען נייע פאזיציעס וועלן געשאפן ווערן, און אסאך אלטע פאזיציעס וועלן געשטערט ווערן. די וויכטיגסטע מעסעדזש איז: די וואס לערנען זיך פרי אויס נייע AI- און טעכנאלאגיע-קענטענישן וועלן זיין בעסער צוגעגרייט פאר דער צוקונפט. דער טייל פארשפרעכט צו ערקלערן אגענטיק AI אויף א פשוטן אופן, אן קאמפליצירטע שפראך: וואס עס איז, פארוואס עס איז וויכטיג, וועלכע קענטענישן מען דארף, וועלכע געציג מען דארף קענען, וועלכע פראיעקטן מען קען בויען, און ווי אזוי זיך צוגרייטן פאר דער נייער כוואליע פון AI-קאריערעס.

דער היינטיגער מצב: פון איין ענטפער צו א גאנצן ארבעט-פלוס (42:06:32)

אין 2023 און 2024 זענען רוב מענטשן געווען באגייסטערט פון געציג וואס קענען שרייבן אינהאלט, שאפן בילדער, מאכן קיצורים פון דאקומענטן, און ענטפערן פראגעס. דאס איז געווען די ערשטע גרויסע כוואליע. אין 2025 און 2026 האט זיך דער שמועס געטוישט, און פירמעס פרעגן א גרעסערע פראגע: קען AI פארענדיגן א געשעפט-ארבעט פון אנהייב ביז סוף? עס ווערן געברענגט ביישפילן פון אזעלכע פראגעס:

- קען AI באהאנדלען קונה-סערוויס?
- קען AI העלפן א סעילס-טיעם צוגרייטן פניות צו קונים (?outreach)
- קען AI העלפן פינאנץ-טיעמס לייענען רעכענונגען (?invoices)
- קען AI העלפן פראגראמירער פארריכטן טעותים (?bugs)
- קען AI העלפן אנגעשטעלטע זוכן פירמע-דאקומענטן און טון אן אקציע?

דא קומט אריין אגענטיק AI. א געווענליך AI-געציג איז נוצלעך, אבער עס הענגט נאך שטארק אפ פונעם באניצער. מען מוז אים כסדר געבן אנווייזונגען, קאפירן און אריינקלעבן אינפארמאציע, נאכקוקן דעם נעקסטן טריט, און באשליסן וואס צו טון נאך יעדן ענטפער. אגענטיק AI פרובירט צו פארמינערן די ארבעט מיט די הענט. מען גיט אים דעם ציל, און עס צעטיילט דעם ציל אויף טריט. א ביישפיל: אנשטאט זאגן "שרייב איין בריוו", קען מען זאגן "גרייט צו א נאכפאלג-בריוו פאר אלע קונים וואס האבן זיך באטייליגט נעכטן אין דעם וועבינאר, פערזענליך צוגעפאסט צו זייערע אינטערעסן, און שאף א קיצור-באריכט פארן סעילס-טיעם". דאס איז נישט בלויז איין ענטפער, נאר א גאנצער ארבעט-פלוס. דערפאר זענען געשעפטן באגייסטערט: אגענטיק AI קען שפארן צייט און איבערנעמען ארבעטן וואס חזר'ן זיך איבער.

וואס איז אגענטישע AI און דער וועג צו לערנען (42:08:04)

א מעכטיגער כלי וואס פאדערט פארזיכטיגקייט (42:08:04)

דער קורס הייבט אן דעם אפשניט מיט'ן ערקלערן אז אגענטישע AI קען פארבעסערן די גיכקייט פון ארבעט און לאזן די ארבעטער זיך קאנצענטרירן אויף בעסערע באשלוסן. אבער עס ווערט באטאנט א וויכטיגע ווירקליכקייט: אגענטישע AI איז מעכטיג, און מען דארף זי באנוצן זייער פארזיכטיג. אויב AI קען טון אקציעס, מוזן פירמעס זיכער מאכן אז זי טוט נישט קיין פאלשע אקציע. דערפאר זענען טעמעס ווי זיכערקייט (safety),

באשטעטיגונג (approval), נאכשפירן (tracking) און מענטשליכע איבערקוק (human review) זייער וויכטיג היינטיגע צייטן.

לויט דעם קורס איז די אמת'ע געלעגנהייט נישט נאר צו לערנען ווי צו באנוצן AI, נאר צו לערנען ווי אזוי צו בויען און פירן AI סיסטעמען וואס ארבעטן זיכער, נוצליך און פאראנטוואַרטליך.

די פשוטסטע דעפיניציע, מיט א ביזנעס-ביישפיל (42:08:34)

אגענטישע AI מיינט א AI סיסטעם וואס ענטפערט נישט נאר אויף פראגעס. זי קען פארשטיין א ציל, אים צעטיילן אין טריט, באנוצן געצייג, מאכן באשלוסן אין באשטימטע גרענעצן, געדענקען נוצליכע אינפארמאציע און ארבעטן צו פארענדיגן א שטיק ארבעט.

דער קורס ברענגט א ביזנעס-ביישפיל: א פירמע לאנצירט א נייעם אנליין קורס. די מאנשאפט האט איין קלארן ציל: באקומען מער איינשרייבונגען אין די קומענדיגע 30 טעג.

אויב זיי פרעגן א געוועהנליכע AI, למשל "גיב מיר א מארקעטינג אידעע פאר דעם קורס", קען די AI ענטפערן מיט א רשימה פון קאמפיין-אידעעס, אימעיל-פארשלאגן, אונטערשריפטן פאר סאשעל מידיע און אן אנאנס. דאס קען זיין נוצליך, אבער די ארבעט בלייבט נאך אלץ ביי די מאנשאפט. זיי דארפן נאך אלץ שטודירן דעם עולם, איבערקוקן די דאטא פון פריערדיגע קאמפיינס, אויסקלייבן די ריכטיגע קאנאלן, שרייבן די אימעילס, צוגרייטן דעם קאלענדאר, פארגלייכן ווי די אנאנסן ארבעטן, און באשליסן וואס צו פארבעסערן.

יעצט שטעלט זיך פאר אז די זעלבע פירמע באנוצט א AI אגענט. די מאנשאפט גיט איין ציל: פלאנירן און דורכפירן א 30-טאגיגע מארקעטינג קאמפיין פאר דעם קורס-לאנצירונג. דער אגענט קען אנהייבן מיט אנאליזירן די פרטים פון קורס און אויסגעפינען פאר וועמען דער קורס איז, למשל אנפאנגער (freshers), ארבעטנדע פראפעסיאנאלן אדער מענעדזשערס. דערנאך קען ער שטודירן ווי פריערדיגע קאמפיינס האבן געארבעט און געפינען וועלכע סארט מעסעדזש האט בעסער געווירקט. דערנאך קען ער שאפן פארשידענע קאמפיין-טעמעס, אנשרייבן ריי אימעילס, מאכן LinkedIn און Instagram פאסטס, צוגרייטן אנאנס-טעקסטן פאר פארשידענע גרופעס פון עולם, און בויען א וועכנטליכן אינהאלט-קאלענדאר.

אבער ער שטעלט זיך נישט אפ דארט. וועז די קאמפיין הייבט זיך אן, קען דער אגענט נאכשפירן וועלכע אימעילס ווערן מער געעפנט, וועלכע אנאנסן באקומען מער קליקס, און וועלכע מעסעדזשעס אויף די לאנדינג-בלאט ברענגען מער איינשרייבונגען. באזירט אויף די דאטא קען ער פארשלאגן ענדערונגען, איבערשרייבן שוואכע אנאנס-טעקסטן, רעקאמענדירן איבערצורוקן געלט פונעם בודזשעט, און צוזאמענפאסן וואס די מארקעטינג מאנשאפט דארף וויסן.

דער חילוק איז פשוט: א געוועהנליכע AI גיט דיר דעם רעזולטאט, אבער אגענטישע AI העלפט דיר פירן א ביזנעס-פראצעס פון ציל ביז אקציע.

פארוואס דאס וואַרט "אגענט" (42:10:07)

א אגענט איז עפעס וואס האנדלט אין דיין נאמען לויט געגעבענע אנווייזונגען. דער קורס ברענגט א משל פון א טראוועל-אגענט: ער ערקלערט נישט נאר וואס רייזן איז, ער העלפט פארגלייכן אפציעס, פלאנירן די רייזע, בוקן בילעטן און ארגאניזירן דעם וועג. די זעלבע זאך, א AI אגענט גיט נישט נאר אינפארמאציע; ער קען העלפן פלאנירן, דורכפירן, נאכקוקן און פארבעסערן אמת'ע ארבעט.

דערפאר ווערט אגענטישע AI וויכטיג: ביזנעסער דארפן נישט נאר ענטפערס, זיי דארפן סיסטעמען וואס העלפן זיי פארענדיגן ארבעט גיכער, פארמינערן האנט-ארבעט, און פארוואנדלען אידעעס אין רעזולטאטן.

אגענטישע AI קען מען באנוצן אין אסאך געביטן: קאסטומער סערוויס, חינוך, ווייכווארג-אנטוויקלונג, מארקעטינג, פארקויף, געזונט-אדמיניסטראציע, פינאנץ-אפעראציעס, פארשונג, HR און פערזענליכע פראדוקטיוויטעט.

א וויכטיגער פונקט: אגענטישע AI מיינט נישט אינגאנצן פארביטן מענטשן. די בעסטע באנוץ איז צו העלפן מענטשן ארבעטן גיכער און קלוגער. עס איז ווי צו האבן א דיגיטאלן מיטארבעטער וואס באהאנדלט איבערגעחזר'טע ארבעט, בשעת מענטשן מאכן די וויכטיגע באשלוסן.

דריי טערמינען: generative AI, AI אגענט, אגענטישע AI (42:11:07)

כדי נישט צו פארמישן, ערקלערט דער קורס דריי באגריפן:

- Generative AI – שאפט אינהאלט: טעקסט, בילדער, קאוד, צוזאמענפאסונגען, אידעעס, אימעילס, סקריפטן און דיזיינס. למשל, ChatGPT וואס שרייבט א בלאג-ארטיקל.
- AI אגענט – א סיסטעם וואס באנוצט AI צו טון ארבעט. ער שאפט נישט נאר טעקסט, נאר באנוצט אויך געצייג: ער קען פארשן אינפארמאציע, רופן אן API, ליינען א דאקומענט, איבערקוקן א דאטאבעיס, אדער אנצינדן א workflow.
- אגענטישע AI – א ברייטערער צוגאנג וואו AI סיסטעמען קענען פלאנירן, האנדלען, באנוצן געצייג, געדענקען קאנטעקסט און פארענדיגן צילן מיט עטליכע טריט. דאס איז נישט איין מעסעדזש און איין ענטפער, נאר א פולער פלוס פון ארבעט.

א קליין ביישפיל: מען וויל צוגרייטן א באריכט. א generative AI כלי קען שרייבן דעם באריכט. א AI אגענט קען זאמלען אינפארמאציע, עס צוזאמענפאסן און צוגרייטן דעם באריכט. אן אגענטישע סיסטעם, פארקערט, קען פארשטיין דעם ציל, זאמלען דאטא, פארגלייכן מקורות, שאפן דעם באריכט, איבערקוקן די קוואליטעט, בעטן באשטעטיגונג, און צום סוף צוגרייטן די ענדגילטיגע ווערסיע. דער קורס פאסט עס צוזאמען: generative AI שאפט, AI אגענטן האנדלען, אגענטישע AI פארענדיגט גאנצע workflows.

פארוואס ביזנעסער נעמען עס ערנסט (42:12:39)

יעדע פירמע האט איבערגעחזר'טע ארבעט: קאסטומער-סופּפּארט מאנשאפטן ענטפערן די זעלבע פראגעס, HR שטודירט פראפילן, מארקעטינג מאכט קאמפיין-פלענער, פארקויפער פאלגן נאך leads, פינאנץ גרייט צו באריכטן, און אפעראציעס שפירט נאך ארבעט, ליפערער און באשטעטיגונגען.

דער קורס ברענגט ווידער א ביישפיל פון א וואקסנדע edtech פירמע וואס לאנצירט א נייעם קורס. פריער פלעגט די מארקעטינג מאנשאפט מיט די הענט איבערקוקן עולם-דאטא, שרייבן אימעילס, שאפן אנאנס-טעקסטן, פלאנירן סאשעל מידיע פאסטס, נאכשפירן רעזולטאטן און צוגרייטן וועכנטליכע באריכטן. אפילו מיט א נארמאלן AI כלי האט מען באקומען נאר פארשלאגן, און די מאנשאפט האט נאך אלץ געדארפט דורכפירן. מיט אגענטישע AI ענדערט זיך דער workflow: מען גיט דעם אגענט א קלארן ציל, און ער שטודירט דעם קורס, אידענטיפיצירט דעם ציל-עולם, שרייבט אימעילס, שאפט פאסט-אידעעס, אנאנס-טעקסטן, א קאלענדאר, שפירט נאך רעזולטאטן און שלאגט פאר פארבעסערונגען. למשל, אויב איין אנאנס באקומט קליקס אבער נישט קיין איינשרייבונגען, קען דער אגענט אנווייזן אויפן פראבלעם, פארשלאגן

א בעסערן לאנדינג-בלאט מעסעדזש, איבערשרייבן דעם טעקסט, און רעקאמענדירן וואו די מאנשאפט זאל זיך קאנצענטרירן.

דער אמת'ער וועלט פון אגענטישע AI איז נישט נאר שאפן אינהאלט, נאר דורכפירן workflows.

צאלן און קאריערע-געלעגנהייטן (42:13:40)

דער קורס ברענגט צאלן: גלאבאלע AI אנשטעלונג איז געוואקסן מיט מער ווי 300% אין די לעצטע 8 יאר, און 30% גיכער ווי אלגעמיינע אנשטעלונג זינט לעצטן הערבסט. לויט Indeed's Hiring Lab, ביז דעצעמבער 2025 האבן ארבעט-אנאנסן וואס דערמאנען AI דערגרייכט א הויכפונקט פון 4.2%, מער ווי 130% העכער ווי פאר דער מגפה; בערך 45% אין דאטא-אנאליטיק, בערך 15% אין מארקעטינג און 9% אין HR.

דאס ווייזט אז AI שאפט נישט נאר טעכנישע פאזיציעס ווי AI אינזשעניר, AI קאנסולטאנט, AI/ML פארשער אדער דאטא-אנאטאטאר, נאר ענדערט אויך נישט-טעכנישע ראלעס אין מארקעטינג, HR, קאסטומער סופפארט, פארקויף און אפעראציעס. אין דער צוקונפט וועלן פירמעס זוכן מענטשן וואס קענען בויען, פירן און ארבעטן מיט AI אגענטן. מען וועט דארפן פראפעסיאנאלן וואס פארשטייען די סיסטעמען און קענען דיזיינען workflows, אויסקלייבן געציג, צוגרייטן אנווייזונגען, פארבינדן סיסטעמען, טעסטן רעזולטאטן, פירן ריזיקעס און פארבעסערן פערפארמענס.

מען דארף נישט זיין א טיפער AI פארשער און נישט בויען גרויסע מאדעלן פון אנהייב, אבער מען דארף פארשטיין ווי AI געצייג ווערן פארבונדן מיט דער וועלט. דערפאר ווערן פאזיציעס ווי AI agent developer, GenAI application developer, AI automation specialist, product builder און AI workflow designer אזוי רעלעוואנט. דער מארק איז נאך פריש, אסאך פירמעס עקספערימענטירן נאך, און עס פעלן נאך באהאוונטע מענטשן; ווער עס בויט יעצט א יסוד, קען זיך שטעלן פאר אנדערע.

די מאפע מיט 10 טריט (42:15:12)

- טריט 1: פארשטיין די יסודות פון AI און גרויסע שפראך-מאדעלן.
- טריט 2: לערנען Python יסודות.
- טריט 3: פראמפט-שרייבן און סטרוקטורירטע רעזולטאטן.
- טריט 4: RAG – העלפן AI באנוצן דרויסנדיגע אינפארמאציע.
- טריט 5: tool calling – AI פארבינדט זיך מיט רעכנמאשינען, זוכמאשינען, דאטאבעיסעס, APIs און אפליקאציעס.
- טריט 6: זכרון (memory) – AI זאל קענען באנוצן פריערדיגן קאנטעקסט.
- טריט 7: AI אגענט דיזיין – פלאנירן ווי א אגענט טראכט, האנדלט, קאנטראלירט און פארענדיגט ארבעט.
- טריט 8: פאפולערע פרעימווארקס ווי LangGraph, CrewAI, AutoGen און OpenAI Agents SDK.
- טריט 9: MCP – וואס העלפט AI זיך פארבינדן מיט דרויסנדיגע מקורות און דאטא-מקורות אויף א סטאנדארט אופן.
- טריט 10: בויען אמת'ע פראיעקטן און שאפן א פארטפאליא.

דער קורס ווארנט: לערנט נישט אלעס אויפ'ן גראדעוואל, און הייבט נישט אן מיט אדוואנסירטע פרעימווארקס פונעם ערשטן טאג. אנהייבן מיט די יסודות, דערנאך workflows, דערנאך פראיעקטן. אגענטישע AI ווערט גרינגער ווען מען זעט עס ווי א רייזע, נישט ווי איין כלי.

טריט 1: יסודות פון AI און שפראך-מאדעלן (42:16:44)

א גרויסער שפראך-מאדעל איז דער מאטאר הינטער אסאך AI געצייג: ער ליינט דיין אריינגאב, פארשטייט מוסטערן און שאפט נוצליכע אינפארמאציע. מען דארף נישט וויסן די גאנצע מאטעמאטיק אין אנהייב, נאר עטליכע פשוטע באגריפן:

- פראמפט – די אנווייזונג וואס מען גיט די AI.
- קאנטעקסט – די אינפארמאציע וואס AI באנוצט כדי בעסער צו ענטפערן.
- טאקענס – קליינע שטיקלעך טעקסט וואס AI ליינט און פראצעסירט.
- באגרעניצונגען – AI קען אמאל קלינגען זיכער און פונדעסטוועגן זיין פאלש; דערפאר דארף מען איבערקוקן מקורות און מענטשליכע איבערקוק.
- אויסוואל פון מאדעל – פארשידענע מאדעלן זענען גוט פאר פארשידענע ארבעט: עטליכע בעסער פאר שרייבן, עטליכע פאר קאוד, עטליכע גיכער, און עטליכע ביליגער.

דער יסוד איז וויכטיג ווייל אגענטן פארלאזן זיך אויף מאדעלן צו פארשטיין צילן און באשליסן דעם קומענדיגן טריט. ווען די יסודות זענען שוואך, וועלן פרעימווארקס זיך פילן פארמישט; ווען זיי זענען קלאר, ווערט יעדע אדוואנסירטע טעמע גרינגער. דער ערשטער צייל: פארשטיין ווי AI ליינט אנווייזונגען, באנוצט קאנטעקסט און שאפט רעזולטאטן.

טריט 2: פראקטישע Python (42:17:44)

Python איז וויכטיג ווייל אסאך AI געצייג, פרעימווארקס און ביישפילן באנוצן עס. מען דארף נישט גלייך ווערן אן אדוואנסירטער פראגראמירער; פאר אגענטישע AI דארף מען פראקטישע Python: וואריאבלען, פונקציעס, ליסטעס, דיקשאנעריס, פיילס, APIs, JSON, באהאנדלען פעלערן, און בעיסיק פעקעדזשעס.

פארוואס? ווען א AI אגענט באנוצט א כלי, ארבעט דער כלי אפט ווי א פונקציע. א רעכנמאשין-כלי נעמט נומערן און גיט צוריק אן ענטפער; א זוך-כלי נעמט א פראגע און גיט צוריק רעזולטאטן; א דאטאבעיס-כלי נעמט א בקשה און גיט צוריק רעקארדס. Python העלפט שאפן און פארבינדן אזעלכע געצייג.

אן API איז ווי א בריק צווישן צוויי אפליקאציעס. למשל, אויב א אגענט דארף וויסן דאס וועטער, קען ער רופן א וועטער-API; צו שיקן אן אימעיל – אן אימעיל-סערוויס; פאר קאסטומער-דאטא – א CRM סיסטעם.

מען דארף נישט אלעס באהערשן מיט אמאל. אנהייבן מיט א קליינע פראגראם: מאכן א רעכנמאשין, ליינען א טעקסט-פייל, רופן אן API. די קליינע פעאיקייטן וועלן שפעטער זיין זייער נוצליך. פאר נישט-טעכנישע לערנער: טראכט פון Python ווי א שפראך צו פארבינדן AI מיט אמת'ע ארבעט.

טריט 3: פראמפט-שרייבן און סטרוקטורירטע רעזולטאטן (42:19:15)

אסאך מיינען אז פראמפטינג מיינט שרייבן פיינע אנווייזונגען, אבער גוטע פראמפטינג איז אין עיקר קלארע קאמוניקאציע. אומקלארע אנווייזונגען ברענגען אומקלארע רעזולטאטן; קלארע אנווייזונגען מיט ביישפילן און גרענעצן פארבעסערן דעם רעזולטאט. ביי אגענטישע AI איז עס נאך וויכטיגער, ווייל א אגענט קען נעמען

עטליכע טריט; אויב די ערשטע אנווייזונג איז שוואך, קען דער גאנצער workflow גיין אין דער פאלשער ריכטונג.

א גוטער פראמפט דארף אנטהאלטן: דעם ציל, די ראלע, דעם קאנטעקסט, די טריט, דעם פארמאט פונעם רעזולטאט, און די גרענעצן. למשל, אנשטאט זאגן "מאך א באריכט", קען מען זאגן: מאך א באריכט פון איין בלאט פאר אן אנפאנגער-עולם, באנוץ פשוטע ענגליש, נעם אריין דריי הויפט-פונקטן, איין ביישפיל און א קורצן סוף, און באנוץ נישט קיין טעכנישע זשארגאן. דאס איז אסאך בעסער.

סטרוקטורירטע רעזולטאטן מיינט בעטן די AI צוריקצוגעבן אינפארמאציע אין א פעסטן פארמאט, למשל א טאבעלע, א טשעקליסט, פונקטן, JSON, אדער א פעסטע מוסטער. דאס איז וויכטיג ווייל אגענטן גיבן אפט איבער אינפארמאציע פון איין טריט צום צווייטן.

א וועג-קארטע צו לערנען Agentic AI (42:20:17)

סטרוקטורירטער אויסגאנג און RAG (42:20:17)

דער קורס ענדיגט דעם דריטן שריט: אויב דער אויסגאנג פון AI איז א בלאָנדזשענדיש, קען דער נעקסטער שריט אין דעם ארבעטס-פלוס דורכפאלן. אויב דער אויסגאנג איז סטרוקטורירט, ווערט דער גאנצער פראצעס פיל מער פארלעסליך. דעריבער דארף מען אין דעם שריט לערנען שרייבן קלארע פראמפטס, און מען דארף בעטן ריינעם, אָרגאניזירטן אויסגאנג.

דער פערטער שריט איז RAG, וואס שטייט פאר Retrieval Augmented Generation. אין דער לעקציע ווערט עס ערקלערט אויף א פשוט'ן אופן: RAG מיינט צו העלפן AI ענטפערן מיט אינפארמאציע פון דרויסן. פארוואס דארף מען דאס? AI-מאדעלן ווייסן נישט פון זיך אליין די דאקומענטן פון אייער פירמע, די לעצטע פאליסיס, פרטים פון קורסן, רעקארדס פון קונים, אינערליכע באריכטן אדער פריוואטע טעקעס. ווען מען וויל אז AI זאל ענטפערן פון די מקורות, דארף מען האבן א וועג ווי אזוי צו געבן די אינפארמאציע צו AI, און דאס איז פונקט דער ארט פון RAG.

דער ביישפיל: שטעלט אייך פאר אז מען האט 100 פירמע-דאקומענטן. א באניצער פרעגט: "וואס איז די רעפאנד-פאליסי (צוריקגעבן געלט) פאר דעם געווינען קורס?" אנשטאט צו בעטן AI אז ער זאל טרעפן, זוכט די סיסטעם אין די דאקומענטן, געפינט דעם פאסיגן אפטייל, און גיט די אינפארמאציע צו AI איידער ער ענטפערט. אזוי ווערט דער ענטפער פיל מער באגרינדעט און נוצליך.

פאר Agentic AI איז RAG זייער וויכטיג, ווייל אגענטן דארפן האבן געניוע אינפארמאציע איידער זיי נעמען א שריט. אן אגענט טאר נישט טרעפן די פירמע-פאליסי. א קונים-סערוויס אגענט טאר נישט טרעפן די כללים פון צוריקגעבן געלט; א פינאנץ-אגענט טאר נישט טרעפן די פרטים פון אן אינוואיס; און א פארשונגס-אגענט טאר נישט אויסטראכטן מקורות. צום סוף: RAG איז דער ווערג ווי אזוי AI לייענט דעם ריכטיגן דאקומענט איידער ער ענטפערט.

ניצן געצייג – Tool Calling (42:21:17)

דער פינפטער שריט איז Tool Calling, איינער פון די וויכטיגסטע טיילן פון Agentic AI. אן AI-אגענט ווערט שטארק ווען ער ניצט געצייג. א געצייג קען זיין סיי וואס: א קאלקולאטאר, א זוך-מאשין, א דאטא-באזע, א געצייג וואס שיקט אימעילס, א טעקע-לייענער, א CRM-סיסטעם, א באצאל-סיסטעם, אדער אן אינערליכע אפליקאציע פון דער פירמע.

די ביישפילן אין דער לעקציע:

- וועו מען פרעגט "וויפיל איז 17 × 43?" קען AI גלייך ענטפערן, אבער כדי עס זאל זיין געוי, קען ער פונדעסטוועגן ניצן א קאלקולאטאר-געצייג.
- "וועלכע מיטינגס האב איך מארגן?" – דא דארף ער א קאלענדאר-געצייג.
- "וואס איז דער סטאטוס פון מיין באשטעלונג?" – דא דארף ער צוטריט צו א דאטא-באזע פון באשטעלונגען.
- "שיק אן אימעיל צו דעם קונה וואס איז אינטערעסירט (lead)" – דא דארף ער אן אימעיל-געצייג.

דאס מאכט AI-אגענטן אזוי נוצליך אין דער אמת'ער וועלט. Tool Calling מיינט אז AI פארשטייט וועלכער געצייג וועלט געברויכט, קלייבט אויס דעם ריכטיגן געצייג, שיקט אריין די ריכטיגע אינפארמאציע, באקומט דעם רעזולטאט, און דערנאך גיט ער דעם לעצטן ענטפער. דאס איז דער מאמענט וועו AI גייט אריבער פון שמועס צו טאט. דער קורס באטאָנט: וועו עס וויל לערנען Agentic AI, פאר דעם איז Tool Calling נישט קיין ברירה, נאר איינע פון די יסוד-פעאיגקייטן.

אויסשטעלן אן אגענט – Agent Design (42:22:18)

דער זיבעטער שריט איז Agent Design, וואס מיינט באשליסן וו אזוי אן אגענט וועט ארבעטן. דער קורס זאגט אז א גוטער אגענט איז נישט סתם א טשעטבאט מיט א שיינעם נאמען. א גוטער אגענט האט א קלארן ציל, קלארע געצייג, קלארע גרעניצן, קלארע קאנטראל-פונקטן און קלארן אויסגאנג. די טיילן פון א יסודות'דיגן אגענט:

- ערשטנס, דער ציל – וואס דארף דער אגענט דערגרייכן?
- צווייטנס, א פלאנירער (planner) – וועלכע שריט דארף דער אגענט נעמען?
- דריטנס, געצייג – וועלכע אפליקאציעס אדער פונקציעס קען דער אגענט ניצן?
- פערטנס, זכרון – וועלכן קאנטעקסט דארף ער געדענקען?
- פינפטנס, א בודק (checker) – וו אזוי וועט ער באשטעטיגן דעם רעזולטאט?
- זעקסטנס, באשטעטיגונג פון א מענטש – וואו דארף א מענטש איבערקוקן די אקציע?

ביישפילן: אן אגענט וואס שיקט אימעילס צו מעגליכע קונים טאר נישט שיקן אימעילס פון זיך אליין אן באשטעטיגונג. ער דארף צוערשט אנגרייטן דעם אימעיל, אים ווייזן דעם באניצער, און בעטן באשטעטיגונג. א פינאנץ-אגענט טאר נישט באשטעטיגן באצאלונגען אויטאמאטיש; ער דארף אנגרייטן א רעקאמענדאציע און ווארטן אויף באשטעטיגונג פון א מענטש. א קונים-סערוויס אגענט טאר נישט צוזאגן קיין געלט צוריק סיידן די פאליסי ערלויבט עס. דערפאר איז Agent Design וויכטיג: דער ציל איז נישט אז AI זאל ארבעטן אן קאנטראל, נאר אז AI זאל ארבעטן אינערהאלב א זיכערן און נוצליכן פראצעס.

פריימווארקס פאר אגענטן (42:24:19)

דער אכטער שריט איז לערנען פריימווארקס פאר AI-אגענטן. א פריימווארק איז א פארטיגע סטרוקטור וואס העלפט פראגראמירער בויען שנעלער; אנשטאט אלעס צו בויען פון אנהייב, גיבן פריימווארקס נוצליכע בוי-שטיינער. פאפולערע פריימווארקס אין Agentic AI זענען LangGraph, CrewAI, AutoGen, OpenAI Agents SDK, Semantic Kernel און LlamaIndex. מען דארף נישט לערנען אלע מיטאמאל – מען הייבט אן מיט איינס אדער צוויי:

- LangGraph איז נוצליך ווען מען וויל מער קאנטראל איבער ארבעטס-פלוסן מיט אסאך שריט.
- CrewAI איז נוצליך צו פארשטיין אגענטן לויט ראלעס, וואו פארשידענע אגענטן אקטירן ווי פארשידענע מיטגלידער פון א טיעם.
- AutoGen איז נוצליך פאר שמועסן און צווזאמענארבעט צווישן עטליכע אגענטן.
- OpenAI Agents SDK איז נוצליך צו בויען אגענטן מיט געצייג, איבערגאבעס (handoffs), זיכערהייט-געלענדערס (guardrails) און סטרוקטורירטן אויסגאנג.
- LlamaIndex איז נוצליך ווען דער אגענט דארף שטארק ארבעטן מיט דאקומענטן און וויסן-מקורות.

דער עצה פאר אנהייבער: נישט פארלוירן ווערן אין פארגלייכן געצייג. צוערשט פארשטיין דעם באגריף, דערנאך אויסקלייבן איין פריימווארק און בויען קליינע פראיעקטן. א גוטער לערן-וועג: צוערשט בויען א פשוטן אגענט אליין מיט די הענט, דערנאך ניצן א פריימווארק עס צו מאכן ריינער און גרינגער צו פארגרעסערן.

MCP – א נאָרמאלער "שטעקער" פאר AI (42:25:21)

דער ניינטער שריט איז MCP, וואס שטייט פאר Model Context Protocol. פשוט געזאגט: MCP איז א נאָרמאלער וועג פאר AI-אפליקאציעס זיך צו פארבינדן מיט געצייג און דאטא-מקורות. דער משל אין דער לעקציע: א געמיינזאמער לאדנס-פארט. פריער האט יעדע מאשין געהאט אן אנדערן לאדער, וואס האט געשאפן צומישענישן; דערנאך האבן נאָרמאלע פארטן געמאכט די פארבינדונגען גרינגער. MCP פרובירט טון עפעס ענליכס פאר AI-געצייג. אנשטאט צו בויען א באזונדערע פארבינדונג פאר יעדע AI-אפליקאציע מיט יעדן געצייג, גיט MCP א מער נאָרמאלע מעטאד, אזוי אז אן AI-אפליקאציע קען זיך פארבינדן מיט טעקעס, דאטא-באזעס, זוך-געצייג, אנטוויקלונג-געצייג און ארבעטס-פלוסן דורך סערווערס.

פארוואס איז דאס וויכטיג אין 2026? ווייל Agentic AI הענגט אפ אין פארבינדונגען. אגענטן דארפן צוטריט צו דאטא, געצייג און אפליקאציעס, און אויב יעדע פארבינדונג ווערט געבויט באזונדער, ווערט די אנטוויקלונג זייער פאמעליך און בלאָנדזשעדיג. MCP מאכט די גאנצע סביבה מער ארגאניזירט. ביישפילן: אן AI-געהילף פאר קאוד קען זיך פארבינדן מיט GitHub; א ביזנעס-געהילף מיט פירמע-דאקומענטן; א דאטא-געהילף מיט א דאטא-באזע; א פראדוקטיוויטעט-געהילף מיט קאלענדאר און טעקעס. אבער מען דארף געדענקען: MCP פאדערט אויך זיכערהייט. אויב AI האט צוטריט צו געצייג, מוזן פירמעס קאנטראלירן צו וואס ער האט צוטריט, וועלכע אקציעס ער מעג נעמען, און ווען מען דארף באשטעטיגונג פון א מענטש.

זיכערהייט-געלענדערס – Guardrails (42:26:53)

דער צענטער שריט איז Guardrails: כללים און זיכערהייט-קאנטראלן וואס האלטן אפ AI פון טון א פאלשע זאך. ווען AI שרייבט נאר טעקסט, איז די ריזיקע באגרעניצט; אבער ווען AI קען שיקן אימעילס, דערהיינטיגן רעקארדס, רופן APIs, טוישן טעקעס און אנצינדן ארבעטס-פלוסן, ווערט די ריזיקע זייער גרויס. דערפאר דארף יעדער ערנסטער AI-אגענט האבן Guardrails. א Guardrail קען:

- קאנטראלירן צי די בקשה פון באניצער איז זיכער;
- קאנטראלירן צי דער אויסגאנג פאלגט די כללים פון דער פירמע;
- אפהאלטן דעם אגענט פון טיילן פריוואטע אינפארמאציע;
- בעטן באשטעטיגונג פון א מענטש פאר אן אקציע וואס איז שפירעוודיג;
- באגרעניצן וועלכע געצייג דער אגענט מעג ניצן;

• האלטן א רעקארד פון וואס דער אגענט האט געטון.

ביישפילן: א קונים-סערוויס אגענט קען ענטפערן יסודות'דיגע פראגעס אויטאמאטיש, אבער וען א קונה בעט געלט צוריק, דארף ער קוקן אין דער פאליסי און פרעגן א מענטש פאר ער באשטעטיגט עפעס. א פינאנץ-אגענט קען צונויפנעמען אינוואיסן, אבער טאר נישט ארויסלאזן באצאלונגען אן באשטעטיגונג. אן אגענט פאר אננעמען ארבעטער קען אויסקלייבן א קורצע רשימה פון רעזומעס, אבער די לעצטע באשלוסן דארפן אריינציען מענטשן. דאס, זאגט דער קורס, איז די פראפעסיאנעלע זייט פון Agentic AI: עס איז נישט גענוג צו בויען עפעס איינדרוקספול – עס מוז אויך זיין זיכער, פארלעסליך און פאראנטוואורטליך.

פראיעקטן צו בויען (42:27:53)

פראיעקטן זענען דער וויכטיגסטער טייל פון דעם וועג אין Agentic AI. וידאס קוקן איז נוצליך, אבער פראיעקטן בויען איז וואס מאכט איינעם גרייט פאר א שטעלע. מען הייבט אן מיט פשוטע פראיעקטן:

• AI לערן-פלאנירער: דער אגענט פרעגט דעם לערנער'ס ציל, וויפיל צייט ער האט און זיין יעצטיגע מדרגה, און שאפט א וואכעדיגן לערן-פלאן.

• AI פארשונגס-געהילף: נעמט א טעמע, זוכט אדער לייענט די געגעבענע מקורות, נעמט צונויף די הויפט-פונקטן, און גרייט אן א פשוטן באריכט.

• געהילף צו פארבעסערן א רעזומע: לייענט א רעזומע, פארגלייכט עס מיט דער באשרייבונג פון א שטעלע, שלאגט פאר פארבעסערונגען און שאפט א בעסערע צוזאמענפאסונג.

• קונים-סערוויס געהילף: לייענט די FAQ פון דער פירמע, ענטפערט פראגעס פון קונים, און גיט ווייטער קאמפליצירטע ענינים צו א העכערער שטאפל.

• פארקויף-נאכפאלג געהילף: נעמט אינפארמאציע וועגן א מעגליכן קונה און שאפט נאכפאלג-אימעילס.

• דאקומענט-געהילף: לייענט PDFs אדער פירמע-דאקומענטן און ענטפערט פראגעס מיט אנווייזונגען צו די מקורות.

• מיטינג-געהילף: נעמט צונויף נאטיצן פון א מיטינג, ציט ארויס די אויפגאבעס, באשטימט ווער איז פאראנטוואורטליך, און גרייט אן נאכפאלג-אימעילס.

• א טיעם פון עטליכע אגענטן פאר אינהאלט: איין אגענט פארשט, איינער שרייבט, איינער רעדאקטירט, און איינער שאפט א לעצטן אינהאלט-קאלענדאר.

די פראיעקטן העלפן פארשטיין דעם אמת'ן נוצן פון Agentic AI. דער קורס ווארנט: בויט נישט פראיעקטן נאר פאר'ן אויג; בויט פראיעקטן וואס לייזן אמת'ע פראבלעמען. א פשוטער, נוצליכער פראיעקט איז בעסער ווי א קאמפליצירטער פראיעקט וואס קיינער פארשטייט נישט.

א דריי-חדש'דיגער לערן-פלאן (42:29:55)

• ערשטער חודש – יסודות: לערנען AI -יסודות, די באזע פון גרויסע שפראך-מאדעלן, שרייבן פראמפטס, Python-יסודות, APIs, JSON און פשוטע אויטאמאציע. בויען קליינע פראיעקטן ווי א קאלקולאטאר-געצייג, א טעקע-לייענער און א פשוטער טשעטבאט.

• צווייטער חודש – וויסן און געצייג: לערנען RAG, embeddings, וועקטאר-דאטא-באזעס, Tool Calling, סטרוקטורירטן אויסגאנג און זכרון. בויען א געהילף וואס ענטפערט פראגעס פון דאקומענטן, און א פשוטן קונים-סערוויס אגענט.

- דריטער חודש – אגענטן און פארטפאליא: לערנען LangGraph אדער CrewAI, MCP-יסודות, Guardrails, באשטעטיגונג פון א מענטש, אפשאצונג (evaluation) און יסודות פון ארויסשטעלן (deployment). בויען צוויי פראיעקטן פאר דער פארטפאליא.

ביים סוף פון דריי חדשים זאל מען האבן א קלארן פארשטאנד ווי Agentic AI ארבעט, און כאטש דריי ביז פינף פראיעקטן צו וועזן. דער ציל איז נישט צו ווערן פערפעקט, נאר צו ווערן פראקטיש.

קאריערע-שטעלעס און די נויטיגע פעאיגקייטן (42:30:56)

Agentic AI שאפט פאדערונג פאר נייע סארטן שטעלעס:

- AI Agent Developer – בויט אגענטן וואס ניצן געצייג צו פארענדיגן אויפגאבעס.
- Gen AI Application Developer – בויט אפליקאציעס מיט AI-מאדעלן, APIs און ארבעטס-פלוסן.
- AI Automation Specialist – פארבינדט AI מיט ביזנעס-פראצעסן צו פארמינערן איבערגעחזר'טע ארבעט.
- LLM Application Engineer – בויט אפליקאציעס ארום גרויסע שפראך-מאדעלן.
- AI Product Builder – פארשטייט סיי ביזנעס-פראבלעמען סיי AI-לייזונגען.
- AI Workflow Designer – פלאנירט ווי AI זאל זיך באוועגן דורך א ביזנעס-פראצעס.
- AI Governance Specialist – קאנצענטרירט זיך אויף זיכערהייט, באשטעטיגונג, ריזיקע און פאראנטוואַרטליכן נוצן.

דער וויכטיגער פונקט: Agentic AI איז נישט נאר פאר שווערע פראגראמירער. עס וועט זיין פלאץ פאר פראגראמירער, אנאליטיקער, פראדוקט-מאנאדזשערס, ביזנעס-לייט, טריינערס, קאנסולטאנטן און אפעראציע-טיעמס. ווער עס פארשטייט דעם ביזנעס-פראבלעם און ווי AI-אגענטן קענען אים לייזן, האט שוין אן אויסנוץ.

צום סוף נעמט דער קורס צונויף די פעאיגקייטן וואס מען דארף: AI -יסודות, פעאיגקייט צו שרייבן פראמפטס, Python-יסודות, פארשטאנד פון APIs, RAG און זוכן אין דאקומענטן, Tool Calling, טראכטן אין ארבעטס-פלוסן, פריימווארקס פאר אגענטן, באקאנטשאפט מיט MCP, Guardrails און אפשאצונג, פעאיגקייט צו בויען פראיעקטן, און נאך – מיט א הינווייז אז עס איז דא נאך איין פעאיגקייט וואס וועט אפט [די לעקציע גייט ווייטער אין דעם נעקסטן טייל].

דער סוף: פראבלעם, ניצן, צוקונפט און וועג-מאפע (42:32:28)

וואס פאר א פראבלעם לייזט מען? (42:32:28)

דער קורס וועט לייזן אן אויף א זאך וואס וועט אסאך מאל איגנאָרירט, און דאס איז אויסקלייבן דעם ריכטיגן פראבלעם. מענטשן בויען אגענטן אן וויסן וועלכן פראבלעם זיי לייזן אייגנטליך, און דאס איז א גרויסער גרייז.

פאר מען בויט אן AI-אגענט, דארף מען זיך אלעמאל פרעגן:

- וואס איז די אויפגאבע?
- ווער איז דער באניצער?
- וועלכע געצייג דארף מען?

- וועלכע אינפארמאציע דארף מען?
- וואס קען שיף גיין?
- וואו דארף א מענטש געבן א הסכמה (?human approval)
- ווי אזוי וועט מען מעסטן הצלחה?

לויט דעם קורס איז דאס די טראכטונג וואס טיילט אפ א בעגינער פון א פראפעסיאנאל. Agentic AI איז נישט נאר ניצן געצייג, נאר אויסשטעלן נוצליכע סיסטעמען.

ביזנעס-ניצן אין פארשידענע געביטן (42:32:50)

דער קורס רעכנט אויס קאנקרעטע ביישפילן:

- קאסטומער-סופּאָרט: אגענטן קענען ענטפערן געווענליכע פראגעס, נאכקוקן דעם סטאטוס פון א באשטעלונג, צוואמענפאסן טיקעטס, און שיקן קאמפליצירטע פעלער צו מענטשן.
- פארקויפן (sales): פארשן לידס, צוגרייטן פערזענליכע אימעילס, דערהיינטיגן נאטיצן אין CRM, און פארשלאגן די קומענדיגע טריט.
- מארקעטינג: אויסשטעלן קאמפיין-פלענער, ארויסברענגען אידייען פאר אינהאלט, אנאליזירן רעזולטאטן, און צוגרייטן באריכטן.
- HR: ענטפערן פראגעס פון ארבעטער איבער פאליסיס, אריינשטעלן אינטערוויוס אין קאלענדאר, און צוגרייטן onboarding-ליסטעס.
- פינאנצן: לייענען אינווויסעס, צייכענען אומגעוועליכע איינשרייבונגען, צוואמענפאסן הוצאות, און צוגרייטן approval-workflows.
- בילדונג: אויסשטעלן לערן-פלענער, ענטפערן פראגעס איבער א קורס, רעקאמענדירן לערן-וועגן, און נאכפאלגן דעם פארשריט.
- IT: באהאנדלען בקשות פון service desk, זוכן אין knowledge bases, פארריכטן פשוטע פראבלעמען, און ווייטערגעבן קאמפליצירטע פראבלעמען העכער.

דערפאר איז agentic AI נישט באגרעניצט צו איין אינדוסטריע. יעדע אינדוסטריע מיט איבערגעחזר'טע workflows קען דערפון פראפיטירן. די גרעסטע געלעגנהייט ליגט אין צוואמשטעלן AI מיט אמת'ע ביזנעס-פראצעסן.

די צוקונפט און די אחריות (42:33:50)

לויט דעם קורס וועט די צוקונפט נישט זיין איין "סופער-אפליקאציע" וואס טוט אלעס, נאר ווארשיינליך אסאך ספעציאליזירטע אגענטן: איינער פאר פארשונג, איינער פארן שרייבן, איינער פאר קאוד, איינער פאר קאסטומער-סופּאָרט, איינער פאר אפעראציעס און איינער פארן לערנען. די אמת'ע כח וועט קומען ווען די אגענטן ארבעטן צוואמען אויף א זיכערן אופן.

אבער די צוקונפט פאדערט אויך אחריות. ווען אגענטן ווערן מער פעאיג, וועלן פירמעס דארפן שטרענגערע כללים: קאנטראל איבער צוטריט צו דאטא, צוטריט צו געצייג, הסכמות, אויפזיכט (monitoring) און זיכערהייט. דערפאר געהערט די צוקונפט צו די וואס פארשטייען ביידע זייטן: די כח פון agentic AI און די אחריות עס צו ניצן זיכער.

Agentic AI ווערט נישט נאר א טעכנישע שליטה, נאר א שליטה פאר יעדן ארבעטס-פלאץ. פראפעסיאנאלן וואס וועלן קענען ארבעטן מיט AI-אגענטן וועלן האבן אן איבערהאנט: זיי וועלן קענען אויטאמאטיזירן רוטינע-ארבעט, פארבעסערן פראדוקטיוויטעט און אויסשטעלן בעסערע workflows.

צוזאמענפאסונג פון דער וועג-מאפע (42:34:52)

די צען טריט:

- איינס: פארשטיין די יסודות פון AI און large language models.
- צוויי: לערנען פראקטישן Python.
- דריי: באהערשן קלארע פראמפטס און סטרוקטורירטע אויסגאבעס (structured outputs).
- פיר: לערנען RAG, כדי AI זאל קענען ניצן דרויסנדיגע וויסן.
- פינף: לערנען tool calling, כדי AI זאל קענען טון נוצליכע אקציעס.
- זעקס: פארשטיין זכרון (memory).
- זיבן: לערנען אויסשטעלן AI-אגענטן.
- אכט: אויספארשן פריימווערקס ווי LangGraph, CrewAI, AutoGen און OpenAI Agents SDK.
- נײַן: פארשטיין MCP, ווייל עס ווערט זייער וויכטיג פאר פארבינדן AI מיט געצייג און דאטא.
- צען: בויען פראיעקטן און שאפן א פארטפאליא.

פאר א בעגינער ווערט גראטן: נישט ווערן איבערגעשלאגן, אנהייבן קליין. בויען איין פשוטן אסיסטענט, דערנאך צוליגן איין געצייג, דערנאך דאקומענטן, דערנאך זכרון, דערנאך זיכערהייט, און צום סוף א פולשטענדיגן workflow.

סוף-וואָרט (42:35:38)

Agentic AI איז איינע פון די מערסט אויפרעגנדע ענדערונגען אין טעכנאלאגיע, ווייל עס קען טוישן ווי מען טראכט ווען ארבעט. אמאל האט סאפטווער געווארט אז א מענטש זאל קליקן קנעפלעך, דערנאך האט AI אנגעהויבן ענטפערן פראגעס, און היינט באוועגט זיך agentic AI צו פארענדיגן אויפגאבעס.

דאס מיינט נישט אז מענטשן ווערן ווייניגער וויכטיג, פארקערט, זיי ווערן וויכטיגער, ווייל עמיצער דארף באשטימען דעם ציל, אויסשטעלן דעם workflow, נאכקוקן די רעזולטאטן און מאכן אחריותדיגע באשלוסן. די וואס וועלן מצליח זיין זענען נישט די וואס קענען בלויז ניצן א געצייג, נאר די וואס פארשטייען ווי צוזאמצושטעלן AI, דאטא, געצייג, workflows, זיכערהייט און ביזנעס-טראכטן. די עצה: אנהייבן מיט די יסודות, לערנען טריט נאך טריט, בויען אמת'ע פראיעקטן, בלייבן נייגעריג, און דער עיקר, זיך קאנצענטרירן אויף לייזן אמת'ע פראבלעמען.

צום שלוס ווערט איבערגעחזר'ט דער גאנצער קורס: די יסודות פון AI און generative AI, large language models, prompt engineering, LLM-אפליקאציעס, embeddings, וועקטאר-דאטאבאזעס און RAG; דערנאך AI-אגענטן, agentic ארכיטעקטורן, געצייג און function calling , זכרון, ריזאנינג, פלאנירן, workflows און פריימווערקס ווי LangGraph און CrewAI; ווי אויך MCP, multi-agent סיסטעמען, אפשאצונג (evaluation), responsible AI, אמת'ע אגענט-פראיעקטן, deployment און LLMOps. דער עיקר צו געדענקען: agentic AI באדייט אריבערגיין פון AI וואס ענטפערט בלויז, צו AI וואס ארבעט צו א ציל

און טוט אקציעס, א שטארקער יסוד פאר פראגראמירער, דאטא-פראפעסיאנאלן און יעדן וואס וויל בויען א קאריערע אין AI.

טייל AI: 5 אינזשעניר קורס 2026, לייוו – Simplilearn

נאמען: AI Engineer Full Course 2026 | Complete AI Engineer Course | Simplilearn

קאנאל: Simplilearn

דאטום: לייוו געשטרימט, 16 אקטאבער 2026

לענג: 11:05:03

לינק: https://www.youtube.com/live/gdVxQZbHXA8

וועגן דעם ווידעא

וואס שטייט אין דער באשרייבונג

דער ווידעא הייסט "AI Engineer Full Course 2026 | Complete AI Engineer Course | Simplilearn". ער איז ארויסגעגעבן געווארן אויפן קאנאל Simplilearn אלס א לייוו-שטראם דעם 16טן אקטאבער 2026. ער איז זייער לאנג, 11 שעה, 5 מינוט און 3 סעקונדעס. דער ווידעא האט נישט קיין איינגעטיילטע קאפיטלען אויף YouTube.

לויט דער באשרייבונג גיט דער קורס א פולשטענדיגן וועג-ווייזער פון אנפאנג ביז סוף, ווי אזוי צו ווערן אן AI אינזשעניר וואס איז גרייט צו ארבעטן אויף עכטע פראדוקציע-סיסטעמען. ער דעקט Python, מאשין-לערנען, דיפ-לערנען, געניראטיווע AI (LLMs, RAG און אגענטן), און MLOps, דאס הייסט ארויסשטעלן סיסטעמען וון זיי נאכקוקן. דער קורס מישט צוזאם די יסודות מיט פראקטישע פראיעקטן, כדי דער לערנער זאל קענען פלאנירן, בויען, טריינירן, ארויסשטעלן און פארגרעסערן AI סיסטעמען אין 2026.

דער סדר פונעם קורס, לויט דער באשרייבונג, איז אזוי:

1. Python פאר AI: דאטא-סטרוקטורן, OOPאביעקט-אריענטירטע פראגראמירונג) און API's.

2. קלאסישע מאשין-לערנען: supervised און) unsupervised learningלערנען מיט און אן אנגעצייכנטע דאטא),) feature engineeringצוגרייטן די אייגנשאפטן פון דער דאטא), און אפשאצן ווי גוט א מאדעל ארבעט.

3. דיפ-לערנען: CNNs, RNNs און Transformers.

4. געניראטיווע AI (GenAI): prompt engineering (שרייבן גוטע פראמפטן), אפליקאציעס געבויט אויף LLMs, RAG, און) fine-tuningצופאסן א מאדעל צו א באזונדערע ארבעט).

5. MLOps און LLMOps: וועדסיע-פירונג, CI/CD, קאנטיינערס, ארויסשטעלן אויפן קלאוד (AWS, Azure, GCP), און observability, דאס הייסט נאכקוקן ווי דער סיסטעם ארבעט, כדי ארויסצוגעבן פארלעסליכע AI לייזונגען.

די באשרייבונג זאגט צו, אז ביים סוף וועט דער לערנער האבן א זאמלונג פראיעקטן (portfolio), די פעאיגקייטן צו דורכגיין אינטערוויוס פאר AI און ML שטעלעס, און א קלארן וועג צו שטעלעס ווי AI Engineer, ML Engineer, GenAI Engineer אדער MLOps Engineer.

באצאלטע פראגראמען וואס דער קאנאל אדווערטייזט:

אויבן אין דער באשרייבונג שטייען פינף באצאלטע פראגראמען:

- Microsoft Applied Agentic AI: Systems Design & Impact, א פראגראם אויף אגענטישע AI סיסטעמען.
- Microsoft Applied Generative AI and Agentic AI Specialization, א ספעציאליזאציע אין גענעראטיוו און אגענטישע AI.
- Virginia Tech Applied Agentic AI: Systems Design & Impact, א פראגראם אויף אגענטישע AI, אונטערן נאמען Virginia Tech.
- Advanced Executive Program In Applied Generative AI - IITM Pravartak, אן עקזעקיוטיוו פראגראם אין גענעראטיוו AI.
- Professional Certificate Program in Agentic AI & Multi-Agent Systems - Vishlesan IIT Patna, א סערטיפיקאט-פראגראם אין אגענטישע AI און סיסטעמען מיט עטליכע אגענטן.

א גרעסערער טייל פון דער באשרייבונג באשרייבט דעם ערשטן פון זיי, Applied Agentic AI: Systems Design & Impact. לויט דער באשרייבונג גיט ער א לערן-וועג פון אנפאנג ביז סוף: יסודות פון AI, גענעראטיוו AI, ווי LLMs ארבעטן אינעווייניג, און סיסטעמען מיט עטליכע אגענטן. מען לערנט אויך פלאנירן אגענטן, planning frameworks, און א מארקעט-סטראטעגיע פאר AI פראדוקטן.

די הויפט-פונקטן וואס דער קאנאל רעכנט אויס פאר דעם פראגראם:

- עס איז א פראגראם פון 10 וואכן, אויפן שטאפל פון א פראקטיקער, אויף אגענטישע AI סיסטעמען, planning frameworks, סיסטעמען מיט עטליכע אגענטן און פראדוקט-סטראטעגיע.
- מען בלייבט מיטגעהאלטן מיט נייע AI ריכטונגען דורך Simplilearn's AI Impact Studio. מען לערנט אויסגעפינען געלעגנהייטן פאר AI, אויטאמאטיזירן ארבעטס-פראצעסן און בויען קלוגע AI לייזונגען.
- מען לערנט agentic frameworks, סיסטעמען מיט עטליכע אגענטן, RAG, MCP, planning systems, workflow automation און נאך.
- מען ארבעט מיט די געצייג LangChain, AutoGen, CrewAI, n8n, Lovable, Miro, Figma, LangSmith און Jupyter.
- פארן הויפט-מאטעריאל איבערחזרט מען די יסודות פון Python און ארבעטס-פראצעסן וואס זענען גרייט פאר AI.
- עס זענען דא לייוו קלאסן אנליין, א שמועס צווישן די לערנער אויף Slack, שמועסן מיט מענטארן, און שטיצע פאר דער גרופע.
- מען באקומט א געמיינזאמען סערטיפיקאט פון Microsoft און Simplilearn.
- מען באקומט Microsoft Learn badges אויפן Microsoft Learn פארטאל פאר די קורסן מיטן נאמען פון Microsoft.

- Simplilearn's JobAssist, לויט דער באשרייבונג, העלפט אז גרויסע פירמעס וואס דינגען זאלן באמערקן דעם לערנער.

די פעאיגקייטן וואס דער פראגראם זאגט צו צו לערנען זענען: Workflow Automation, Agentic Frameworks, MultiAgent Systems, Intelligent Automation, GTM for Agentic AI (ארויסגיין אויפן AI), Ethics and Transparency, LLM Function Calling, Planning Systems, מארקעט מיט אגענטישע Prompt Engineering, UX and UI with Agentic AI, Copilots, MCP און RAG.

לינקס און רעסורסן וואס ווערן דערמאנט

- Microsoft Applied Agentic AI: Systems Design & Impact, באצאלטער פראגראם: https://www.simplilearn.com/agentic-ai-course-training) דער זעלבער לינק ווערט ווידער געגעבן צום סוף, ביי "Enroll Now")
- Microsoft Applied Generative AI and Agentic AI Specialization, באצאלטער פראגראם: https://www.simplilearn.com/microsoft-applied-ai-course
- Virginia Tech Applied Agentic AI: Systems Design & Impact, באצאלטער פראגראם: https://www.simplilearn.com/vt-applied-agentic-ai-course
- Advanced Executive Program In Applied Generative AI - IITM Pravartak, באצאלטער פראגראם: https://www.simplilearn.com/applied-generative-ai-course
- Professional Certificate Program in Agentic AI & Multi-Agent Systems - Vishlesan IIT Patna, באצאלטער פראגראם: https://www.simplilearn.com/agentic-ai-professional-certificate-course
- פינף ווידאס וואס די באשרייבונג רופט "פארבונדענע ווידאס": https://youtu.be/aTE0Vx-rXmA , https://youtu.be/DqPmwTzwipU , https://youtu.be/uWueU6hcgLE , https://youtu.be/ u29qvwRWGWk , https://youtu.be/SGSIoEDX9mE
- א רשימה פון נאך AI ווידאס פון Simplilearn, מיטן נאמען "Artificial Intelligence [2027 Updated]". אין דער באשרייבונג שטייט נישט קיין פולער לינק דערצו.
- פון סאציאלע קאנאלן ווערט נאר דערמאנט א לינק זיך צו אבאנירן אויפן YouTube קאנאל פון Simplilearn: https://bit.ly/2VT4WtH

הקדמה צום קורס און ווי אזוי צו באהערשן AI געצייג (0:00)

וואס דער קורס וועט לערנען (0:26)

דער קורס הייבט אן מיט א פראגע: וואס וואלט געווען ווען מען קען בויען קלוגע אפליקאציעס, שאפן לייזונגען וואס ארבעטן מיט AI, אויטאמאטיזירן קאמפליצירטע ארבעטן, און אנטוויקלען די פעאיגקייטן וואס מען דארף פאר איינע פון די שנעלסט-וואקסנדע טעכנאלאגישע קאריערעס? דאס איז פונקט וואס "AI Engineering" העלפט אויפטוען.

עס ווערט ערקלערט אז היינט טוישט AI כמעט יעדע אינדוסטריע: געזונטהייט, פינאנצן, אויטאמאציע, קונה-סערוויס און ווייכווארג-אנטוויקלונג. פירמעס זוכן שוין נישט נאר מענטשן וואס באנוצן AI געצייג. זיי זוכן פראפעסיאנאלן וואס קענען פלאנירן, בויען און ארויסשטעלן קלוגע AI סיסטעמען.

אן AI אינזשעניר שפילט א וויכטיגע ראלע אין עטליכע געביטן: מען אנטוויקלט מאשין-לערנען מאדעלן, מען בויט Generative AI אפליקאציעס, מען שאפט AI אסיסטענטן, מען ארבעט מיט גרויסע שפראך-מאדעלן (LLM), און מען לייזט אמת'ע ביזנעס פראבלעמען. מיט די פארשריטן אין Gen AI, Agentic AI און אויטאמאציע, קען דאס לערנען AI אינזשעניר‌ינג צוגרייטן פאר שיינע קאריערע געלעגנהייטן.

דער קורס וועט דורכגיין די פאלגנדע טעמעס:

- הקדמה צו AI.
- AI אינזשעניר‌ינג, דער "וועג-מאפע" פון אן AI אינזשעניר, פעאיגקייטן, געצייג און קאריערע-וועג.
- Python פראגראמירן פאר AI.
- די יסודות פון דאטא-וויסנשאפט און דאטא-אנאליז.
- מאשין-לערנען אלגאריטמען און אנטוויקלען מאדעלן.
- דיפ-לערנען און נוירא‌ל-נעצן.
- Natural Language Processing און גרויסע שפראך-מאדעלן.
- Generative AI און פראמפט-אינזשעניר‌ינג.
- בויען AI אפליקאציעס און chatbots.
- Agentic AI און AI אויטאמאציע.
- AI געצייג און frameworks וואס אינדוסטריעס ניצן.
- קאריערע געלעגנהייטן פאר AI אינזשענירן.

דער קורס איז געמאכט פאר א אנהייבער, א סטודענט, א developer, א דאטא-פראפעסיאנאל, אדער איינער וואס וויל אריבערגיין אין AI. אינעם סוף פונעם קורס זאל מען פארשטיין ווי אזוי AI סיסטעמען ווערן געבויט, ווי אזוי מאדערנע AI אפליקאציעס ארבעטן, און ווי אזוי מען אנטוויקלט די פעאיגקייטן וואס מען דארף כדי צו ווערן אן AI אינזשעניר. (עס ווערט אויך קורץ דערמאנט א סערטיפיקאט-פראגראם פון Simplilearn אין AI און מאשין-לערנען.)

א קורצע פרעגע-שפיל (3:09)

פארן אנהייבן ווערט געשטעלט א פראגע: וועלכע AI טעכנאלאגיע דערמעגלעכט אן AI סיסטעם צו פארשטיין און שאפן טעקסט ווי א מענטש? די ברירות זענען: (1) א סיסטעם צו פארוואלטן דאטא-באזעס, (2) א גרויסער שפראך-מאדעל (LLM), (3) אן אפערירנדע סיסטעם, (4) נעץ-פראטאקאלן. דער צוקוקער ווערט געבעטן צו שרייבן זיין תשובה אין די קאמענטארן.

AI געצייג ווערן א טעגליכע ארבעטס-פעאיגקייט (3:27)

די ערשטע לעקציע באהאנדלט ווי אזוי צו באהערשן AI געצייג אויף א פשוט'ן, פראקטישן וועג, צוגעפאסט פאר אנהייבער. עס ווערט ערקלערט אז AI געצייג זענען שוין נישט עפעס וואס נאר טעכנישע מענטשן אדער גרויסע פירמעס ניצן. זיי ווערן טעגליכע ארבעטס-שותפים פאר סטודענטן, לערער, שאפער פון אינהאלט, מארקעטערס, דיזיינערס, ביזנעס-מאנשאפטן, פרילענסערס און ארבעטנדע פראפעסיאנאלן. די אלע ווילן שפארן צייט, פארבעסערן די קוואליטעט און פארטיק ווערן מיט ווייניגער מי.

עס ווערן געברענגט עטליכע ציפערן:

- אין Microsoft און LinkedIn'ס "Work Trend Index" פון 2024 שטייט אז 75% פון די "וויסן-ארבעטער" אויף דער וועלט האבן שוין גענוצט AI ביי דער ארבעט.
- Microsoft India האט באריכטעט אז 92% פון די אינדישע וויסן-ארבעטער ניצן AI ביי דער ארבעט. דאס ווייזט אז דאס איז שוין נישט קיין פעאיגקייט פאר דער צוקונפט, נאר א טעגליכע ארבעטס-פעאיגקייט.
- McKinsey'ס וועלט-אנקעטע פון 2025 האט געפונען אז 78% פון די געפרעגטע האבן געזאגט אז זייער ארגאניזאציע ניצט AI אין כאטש איין ביזנעס-פונקציע. דאס הייסט אז פירמעס ניצן AI אין אמת'ע אפטיילונגען, ווי IT, מארקעטינג, פארקויף, קונה-סערוויס און אפעראציעס, און נישט נאר פאר עקספערימענטן.
- Stanford'ס "AI Index" פון 2025 האט באריכטעט אז דער ביזנעס-באנוץ פון AI איז געשטיגן צו 78% אין 2024, פון 55% דאס יאר פארדעם. דאס ווייזט ווי שנעל ארבעטס-ערטער נעמען אן די געצייג.
- OpenAI האט מיטגעטיילט אז ChatGPT איז געוואקסן צו איבער 700 מיליאן אקטיווע באניצער א וואך. דאס מאכט עס איינס פון די מערסט-גענוצטע דיגיטאלע געצייג אויף דער וועלט, פאר לערנען, שרייבן, פלאנירן, פארשן און לייזן טעגליכע פראבלעמען.

באנוצן קענן באהערשן, און די כח פון יעדן געצייג (4:52)

די וויכטיגסטע נקודה איז אז "באנוצן" AI געצייג און "באהערשן" זיי זענען צוויי גאר באזונדערע זאכן. אסאך מענטשן עפענען ChatGPT, Gemini, Claude אדער אן אנדער געצייג, טייפן איין פשוט'ע פראגע, באקומען א מיטלמעסיגע תשובה, און מיינען דערנאך אז דער געצייג איז נישט שטארק. אבער דער פראבלעם ליגט געוועניליך נישט אינעם געצייג. ער ליגט אין דעם ווי מען פרעגט און ווי מען פירט עס.

יעדער געצייג האט זיין אייגענע כח:

- ChatGPT העלפט אויספארשן אידייען, לייזן פראבלעמען און לערנען שנעלער.
- Gemini העלפט מיט שרייבן, פלאנירן, "brainstorming", און ארבעטן איבער די Google אפליקאציעס.
- Claude איז געבויט צו העלפן מיט שרייבן, פארשן, קאוד און טיפערע פראבלעם-לייזונג.
- Perplexity ארבעט ווי אן "ענטפער-מאשין" וואס גיט תשובות אין עכטער צייט, מיט מקורות.

די געצייג זענען שוין אויך נישט באגרעניצט צו טעקסט:

- Gamma קען פארוואנדלען אידייען אין פאלירטע פרעזענטאציעס, און זיי ארויסגעבן אלס PPT, PDF, PNG אדער Google Slides.
- Canva'ס "Magic Design" קען שאפן דיזיין-מוסטערן פון איין טעקסט-בקשה אדער א בילד.
- ElevenLabs קען פארוואנדלען טעקסט אין רעאליסטישע "voiceovers" (געראדטע קול-אויפנאמען) אין אסאך שפראכן.
- Runway העלפט שאפן פראפעסיאנאלע וידעא-בילדער.
- Zapier פארבינדט מער ווי 9,000 אפליקאציעס, כדי צו פארמינערן איבערגעחזר'טע האנט-ארבעט.
- Make לאזט באניצער בויען וויזועלע "workflows" (ארבעטס-שטראמען) אן געוועניליכן קאוד.

- Envato Elements גיט "stock" וידאס, וידאא-מוסטערן, מוזיק, שריפטן, גראפיקס און פרעזענטאציע-מוסטערן. דאס קען מאכן לערן-אינהאלט אויסזען מער פראפעסיאנאל, און עס איז ספעציעל נוצלעך פאר שאפער פון וידאס און טריינערס.

פונעם ערשטן אידיי ביזן לעצטן רעזולטאט (6:11)

עס ווערט געבעטן זיך פארצושטעלן: אמאל, וועו מען האט געוואלט שאפן א פולן לערן-וידאא, האט מען געדארפט אליין טון אלעס: פארשן די טעמע, שרייבן דעם סקריפט, שאפן סליידס, דיזיינען "thumbnails" (קליינע פארשטעל-בילדער), זוכן "stock" בילדער, אויפנעמען דאס קול, צוגרייטן ביישפילן, און ארגאניזירן אלעס. היינט קענען ChatGPT, Gemini, Claude, Perplexity, NotebookLM, Gamma, Canva, Runway, ElevenLabs, Zapier, Make און Envato Elements שטיצן כמעט יעדן שטאפל פון דעם וועג, פונעם ערשטן אידיי ביזן לעצטן רעזולטאט.

עס ווערט אונטערגעשטראכן אז די געצייג פארביטן נישט די מענטשליכע שעפערישקייט. זיי מאכן די ארבעט שנעלער, שארפער און גרינגער אויסצופירן. דער מענטש באשליסט נאך אלץ דעם אידיי, ער קאנטראלירט די פאקטן, ער פארבעסערט דעם לעצטן רעזולטאט און ער לייגט צו דעם מענטשליכן "טאטש". די געצייג נעמען אוועק דעם קאמף מיט דער "ליידיגער בלאט", און זיי געבן א שטארקן אנהייב-פונקט. עס ווערט דערמאנט אז Canva'ס אפיציעלע AI זייטל באשרייבט געצייג צו שאפן דיזיינס און לייאאוטס וואס מען קען רעדאגירן. Gamma באשרייבט שנעלע פרעזענטאציעס און דאקומענטן. ElevenLabs באשרייבט קול-שאפונג פאר צוועקן ווי דערציילונג, "audio books" און voiceovers.

די לעקציע וועט נישט ווייזן א צופעליגע ליסטע פון געצייג. זי וועט לערנען ווי אזוי זיי ריכטיג צו ניצן, ווי אזוי צו שטעלן בעסערע פראגעס, ווי אזוי צו נעמען דעם ריכטיגן געצייג פאר דער ריכטיגער ארבעט, און ווי אזוי אויסצומיידן געווענליכע טעותים. דער ציל איז אז מען זאל קוקן אויף AI געצייג נישט ווי צעמישטע וועבזייטלעך אדער איינמאליגע קיצורי-דרך, נאר ווי גאנצע סיסטעמען פאר פראדוקטיווקייט. זיי העלפן לערנען שנעלער, שאפן בעסערן אינהאלט, פארשן קלוגער, דיזיינען שנעלער, פרעזענטירן פראפעסיאנאל, און ארבעטן מיט מער זעלבסטבטחון אין 2026 און ווייטער.

AI געצייג זענען קלוגע אסיסטענטן, נישט קיין צויבער-קנעפל (8:08)

די לעקציע פרעגט: צי האט מען אמאל געעפנט אן AI געצייג, געשטעלט א פראגע, באקומען א מיטלמעסיגע תשובה, און געטראכט אז דער געצייג איז נישט אזוי נוצלעך? אויב יא, איז מען נישט אליין. רוב אנהייבער טרעפן דעם זעלבן פראבלעם. דער פראבלעם איז נישט שטענדיג דער געצייג, נאר געווענליך ווי מען ניצט אים. AI געצייג זענען נישט קיין "צויבער-קנעפלעך". זיי זענען מער ווי קלוגע אסיסטענטן. גיט מען זיי אומקלארע אנווייזונגען, קענען זיי געבן אומקלארע תשובות. גיט מען זיי א קלארן קאנטעקסט, א קלארע אויפגאבע און א קלארן פארמאט פאר דער תשובה, קענען זיי העלפן בעסער שרייבן, שנעלער פארשן, שאפן אינהאלט, און אפילו צוגרייטן פרעזענטאציעס.

דער פלאן איז אנצוהייבן פון די יסודות, דערנאך צו לערנען ווי אזוי צו שרייבן בעסערע פראמפטס, און דערנאך אויספארשן די געצייג. מען קען ניצן ChatGPT, Gemini, Claude, Perplexity און NotebookLM. פאר א פרעזענטאציע ניצט מען Gamma אדער Canva. פאר א voiceover ניצט מען ElevenLabs. פאר קורצע וידאא-בילדער ניצט מען Runway. און כדי צו פארבינדן פארשידענע אפליקאציעס צוזאמען ניצט מען Zapier אדער Make.

יעדער געצייג פאר זיין אויפגאבע: דער משל פון א געצייג-קאסטן (9:18)

די ערשטע וויכטיגע נקודה איז אז נישט יעדער AI געצייג איז געמאכט פאר יעדער ארבעט. דא ווערן אסאך אנהייבער צעמישט, ווייל זיי פרובירן צו ניצן איין געצייג פאר אלעס. דער בעסערער צוגאנג איז צו פארשטיין די כח פון יעדן געצייג. דער משל איז פון א געצייג-קאסטן: א האמער, א שרויפציער, א מעסטער און א בויער זענען אלע נוצלעך, אבער יעדער פאר אן אנדער ארבעט. אזוי אויך ChatGPT, Gemini און די אנדערע: אלע זענען נוצלעך, אבער יעדער לייזט א ביסל אן אנדערן פראבלעם.

א ביישפיל: מען שאפט א וידעא ווען מען וועגן "cyber security" (קאמפיוטער-זיכערהייט) פאר אנהייבער. מען קען דאס צעטיילן אזוי:

- ChatGPT שאפט די סטרוקטור.
- Perplexity זאמלט דערהיינטיגטע פאקטן און מקורות.
- Claude פארבעסערט די שפראך פונעם סקריפט.
- Gamma שאפט די פרעזענטאציע.
- ElevenLabs שאפט דעם voiceover.
- Runway שאפט א קליין וידעא-בילד.
- Zapier אדער Make אויטאמאטיזירן פשוט'ע אויפגאבעס. למשל, זיי שפייכלען אידייען אין א "spreadsheet" (טאבעלע), אדער שיקן א מעלדונג ווען א פארעם ווערט אויסגעפילט.

דערפאר זאל מען נישט פרעגן "וועלכער איז דער בעסטער AI געצייג?" נאר "וועלכער AI געצייג איז דער בעסטער פאר די אויפגאבע?" דער איין טויש מאכט די ארבעט אסאך בעסער.

ווי אזוי די געצייג ארבעטן, און פארוואס א פראמפט איז וויכטיג (10:19)

אן צו גיין צו טיף אין טעכנישע זאכן, ווערט ערקלערט ווי די געצייג ארבעטן אויף א יסודות'דיגן שטאפל. מען זאל זיך פארשטעלן אז מען רעדט מיט א זייער שנעלן אסיסטענט וואס האט געזען אסאך ביישפילן: שרייבן, קאוד, פראגעס און תשובות, דאקומענטן און מוסטערן. ווען מען טייפט עפעס, פרובירט דער געצייג צו פארשטיין וואס מען פרעגט, און גיט די מערסט-פאסיגע תשובה לויט דער אנווייזונג. אבער דער געצייג ווייסט נישט אליין דעם געניעם ציל, סיידן מען ערקלערט עס קלאר.

א ביישפיל: מען טייפט "Write about cyber security", און די תשובה איז גאר אלגעמיין. דערנאך טייפט מען א מער פרטימ'דיגן פראמפט: "שרייב א הקדמה צו cyber security פאר א YouTube לעקציע, צוגעפאסט פאר אנהייבער; ניץ פשוט'ע ענגליש, גיב איין עכטן ביישפיל, און האלט עס אונטער 120 ווערטער." די תשובה ווערט אסאך בעסער, ווייל איצט ווייסט דער געצייג די טעמע, דעם עולם, דעם פארמאט, דעם סטיל און די לענג. פראמפטן מיינט פשוט צו געבן קלארע אנווייזונגען. א גוטער פראמפט האט נישט צו טון מיט שיינע ווערטער. ער האט צו טון מיט געבן נוצלעכע פרטים.

א פארגלייך פון צוויי פראמפטס: דער ערשטער, "Explain AI tools", איז צו בריט, און מען זעט אז די תשובה איז אלגעמיין. דער צווייטער: "ערקלער AI געצייג פאר אן אנהייבער וואס האט זיי קיינמאל נישט געזוצט; ניץ פשוט'ע ענגליש, גיב פינף ביישפילן, און ערקלער ווען צו ניצן יעדן געצייג." דער צווייטער איז אסאך בעסער, ווייל ער זאגט געזוי וועלכע סארט תשובה מען וויל. דער כלל איז: אומקלארער אריינגאנג גיט אומקלארן ארויסגאנג, און קלארער אריינגאנג גיט נוצלעכן ארויסגאנג.

א ווארענונג: AI קען זיך טועה זיין (12:15)

AI געצייג קענען מאכן טעותים. זיי קענען געבן אלטמאדישע אינפארמאציע, אומפולשטענדיגע תשובות, אדער פאלשע תשובות וואס הערן זיך זיכער אן. מען זאל נישט בלינד גלייבן אלעס. מען דארף קאנטראלירן וויכטיגע פאקטן, ספעציעל ביי דאטא, נייעס, קאריערע-אינפארמאציע, געזעצליכע אינפארמאציע, מעדיצינישע אינפארמאציע, אדער אלעס וואס ווירקט אויף אמת'ע באשלוסן. דערפאר זענען Perplexity און NotebookLM נוצלעך, ווייל זיי ארבעטן מיט מקורות. אבער אפילו דאן דארף מען אליין איבערקוקן די מקורות.

די מעטאד: ראלע, קאנטעקסט, אויפגאבע, פארמאט (12:44)

איצט ווערט געלערנט די וויכטיגסטע פעאיגקייט: שרייבן בעסערע פראמפטס. מען לערנט א פשוט'ע מעטאד וואס ארבעט ביי אסאך געצייג: Role, Context, Task, Format.

- ראלע (Role): ווער דער געצייג זאל זיך אויפפירן ווי.
- קאנטעקסט (Context): די הינטערגרונט-אינפארמאציע.
- אויפגאבע (Task): וואס מען וויל אז דער געצייג זאל טון.
- פארמאט (Format): ווי אזוי מען וויל די תשובה.

דער ערשטער ביישפיל: אנשטאט צו שרייבן "מאך א וידעא-סקריפט", שרייבט מען: "פיר זיך אויף ווי א YouTube טריינער וואס איז צוגעפאסט פאר אנהייבער. איך שאף א וידעא פאר Simplilearn ווען די יסודות פון cyber security. שאף א סקריפט פון 60 סעקונדעס וואס ערקלערט וואס cyber security איז, פארוואס עס איז וויכטיג, און ווי אזוי אנהייבער קענען אנהייבן עס צו לערנען. ניץ פשוט'ע ענגליש, קורצע זאצן, און א שמועס-טאן." דער צעטייל איז אזוי: "פיר זיך אויף ווי א YouTube טריינער" איז די ראלע; "איך שאף א וידעא פאר Simplilearn ווען cyber security" איז דער קאנטעקסט; "שאף א סקריפט פון 60 סעקונדעס" איז די אויפגאבע; "פשוט'ע ענגליש, קורצע זאצן און שמועס-טאן" איז דער פארמאט און סטיל. די תשובה ווען זיין אסאך בעסער ווי פון א פשוט'ן פראמפט.

דער צווייטער ביישפיל איז פאר א פרעזענטאציע. א פשוט'ער פראמפט וואלט געווען "מאך PPT אינהאלט ווען AI געצייג". א בעסערער פראמפט איז: "פיר זיך אויף ווי א שאפער פון קורסן. איך מאך א לעקציע פאר אנהייבער ווען באהערשן AI געצייג. שאף אינהאלט פאר סליידס. האלט די שפראך פשוט, ניץ נישט צופיל טעכנישע ווערטער. יעדער סלייד זאל האבן א קלארן קעפל, דריי-פיר קורצע פונקטן, און איין ביישפיל." דער פראמפט איז בעסער, ווייל ער גיט א קלארע ריכטונג.

דער דריטער ביישפיל איז פאר פארשונג. א פשוט'ער פראמפט וואלט געווען "בעסטע AI געצייג". א בעסערער פראמפט איז: "איך מאך א לעקציע פאר אנהייבער ווען AI געצייג. גיב מיר א פארגלייך פון געצייג פאר שרייבן, פארשן, פרעזענטאציעס, voiceover, וידעא-שאפונג און אויטאמאציע. פאר יעדע קאטעגאריע דערמאן דעם בעסטן צוועק, איין טעות וואס אנהייבער מאכן, און איין פשוט'ן פראמפט." דאס זאגט דעם געצייג ענוי וואס מען וויל.

צום סוף ווערט דערמאנט נאך א נוצלעכע טעכניק: מען בעט דעם געצייג ער זאל אליין פארבעסערן דעם פראמפט, אזוי אז ער זאל געבן א מער פרטימ'דיגע תשובה וואס איז צוגעפאסט פאר אנהייבער. דאס איז זייער נוצלעך, ווייל אנהייבער ווייסן אפט נישט פון אנפאנג ווי אזוי צו שרייבן דעם פערפעקטן פראמפט. דערמיט ענדיגט זיך די טעאריע פון פראמפטן, און מען גייט איבער צו פראקטישן באנוץ.

ChatGPT פאר שרייבן, פלאנירן און אינהאלט-שאפונג (15:00)

נאך דעם וואס מען האט פארשטאנען די יסודות פון פראמפטן, עפנט דער קורס דאס ערשטע געצייג, ChatGPT , און ווייזט ווי מען באניצט עס פאר שרייבן, פלאנירן און שאפן אינהאלט. עס ווערט ערקלערט אז ChatGPT איז נוצליך ווען מען וויל פלאנירן אידייען, שרייבן אינהאלט, צוזאמענפאסן אינפארמאציע, פארבעסערן א טעקסט, שאפן ביישפילן, מאכן אן אויסשטעל (outline), און אפילו פארשטיין קאוד.

פאר דער דעמאנסטראציע ניצט מען איין ביישפיל דורכאויס, כדי נישט צו שפרינגען פון איין צופעליגן ביישפיל צום אנדערן: מען שאפט א YouTube לעקציע וועגן סייבער-זיכערהייט (cybersecurity) פאר אנהייבער. אזוי זעט מען ווי פארשידענע געצייג העלפן ביי פארשידענע שטאפלען פון דער זעלבער ארבעט.

דער ערשטער פראמפט וואס ווערט אריינגעשריבן אין ChatGPT: "שפיל די ראלע פון א YouTube קורס-שאפער וואס איז פריינטליך פאר אנהייבער. איך שאף א פשוט'ע לעקציע וועגן סייבער-זיכערהייט פאר אנהייבער. גיב מיר א קלארן אויסשטעל פאר א 45-מינוט לעקציע. האלט די שפראך פשוט, און נעם אריין: הקדמה, סדר-היום (agenda), הויפט-באגריפן, ביישפילן פונעם לעבן, געצייג, געוועהנליכע טעותים, א קוויז, און א סיום."

ChatGPT גיט צוריק אן אויסשטעל. דא ווארנט דער קורס אז מען טאר נישט בלינדערהייט קאפירן אלעס. מען דארף איבערקוקן די סטרוקטור, באטראכטן צי עס פאסט פאר דעם עולם, און אויב דער אויסשטעל איז צו פארגעשריטן, קען מען בעטן ChatGPT עס צו פארפשוט'ן. דער נעקסטער פראמפט: "מאך דעם אויסשטעל מער פריינטליך פאר אנהייבער, פארמינער טעכנישע שפראך, און לייג צו ביישפילן פונעם לעבן נאך יעדן הויפט-באגריף."

דער קורס באטאנט אז דאס איז א וויכטיגער שריט: אסאך אנהייבער האלטן אויף נאכן ערשטן ענטפער, אבער דער אמת'ער וועלט קומט פון פארפיינערן דעם ענטפער.

דערנאך בעט מען ChatGPT צו שרייבן א קורצן סקריפט פאר איין אפטייל: "שרייב נאר די הקדמה ווי א שמועס-סקריפט פאר YouTube. הייב אן מיט א 'האק' (hook) וואס כאפט אן דעם צוקוקער, האלט עס פשוט און נאטירליך, און זאל עס נישט קלינגען צו פיל ווי רעקלאמע." דאס גיט א הקדמה וואס מען קען ניצן.

דערנאך פארבעסערט מען דעם טאן: "שרייב איבער די הקדמה ווי א טרעינער וואס רעדט צו אנהייבער, ניץ קורצע זאצן און גלאטע איבערגאנגען." אזוי, ערקלערט מען, פארוואנדלט מען א רויען ערשטן אנשטריך אין א בעסערע ענדגילטיגע ווערסיע.

איבערניצן אינהאלט, א קוויז, און א ווארענונג וועגן פאקטן (16:41)

דער נעקסטער שימוש פון ChatGPT איז "איבערניצן" אינהאלט (content repurposing). למשל, מען האט שוין א לאנגן פאראגראף. מען פאסט אריין א ביישפיל-פאראגראף און שרייבט: "פארוואנדל דעם פאראגראף אין פינף פשוט'ע פונקטן (bullet points) פאר א פרעזענטאציע, יעדער פונקט אונטער 12 ווערטער." דאס איז נוצליך ווייל איין לאנגער סקריפט קען פארוואנדלט ווערן אין סליידס, קורצע קליפס, באשרייבונגען, קוויז-פראגעס און פאסטס פאר סאציאלע מעדיע.

דערנאך שאפט מען א קוויז: "שאף דריי MCQ קוויז-פראגעס (פראגעס מיט אויסוואל פון ענטפערס) פאר אנהייבער פון דער הקדמה וועגן סייבער-זיכערהייט, גיב פיר אפציעס און צייכן אן דעם ריכטיגן ענטפער."

אין סך-הכל האט ChatGPT געהאלפן שאפן א אויסשטעל, א הקדמה, פונקטן פאר סליידס, און קוויז-פראגעס. דערפאר איז ChatGPT נוצליך פאר פלאנירן און אינהאלט-שאפונג. אבער דער קורס ווארנט: כאטש ChatGPT איז זייער בעהילפיג, דארף מען נאך אלץ קאנטראלירן די פאקטן. אויב דאס טעמע נעמט אריין לויפנדיגע אינפארמאציע — נייע געזעצ, געהאלטן (salaries), נייעס, סטאטיסטיק — דארף מען שטענדיג באשטעטיגן די מקורות. דערפאר ווערן געזעצ ווי Perplexity נוצליך, וואס מען וועט באלד זען.

Gemini: ערקלערונגען, טאבעלעס און ארבעט מיט פיילס (17:35)

פאר Perplexity גייט מען איבער צו Gemini , וואס קען העלפן מיט פארשונג, פיילס, און ארבעט אין א Google-סטיל סביבה. עס ווערט ערקלערט אז Gemini איז נוצליך פאר שרייבן, פלאנירן, אויסטראכטן אידייען (brainstorming), ארבעטן מיט פיילס, און אויספארשן אינפארמאציע אין א Google-סטיל סביבה. מען האלט ווייטער ביים זעלבן ביישפיל פון דער סייבער-זיכערהייט לעקציע.

דער פראמפט אין Gemini: "איך שאף א לעקציע פאר אנהייבער וועגן סייבער-זיכערהייט. גיב מיר א פשוט'ע ערקלערונג פון סייבער-זיכערהייט מיט א ביישפיל פון שוץ פון א הויז. האלט עס פאסיג פאר סטודענטן און ארבעטנדע מענטשן." דער קורס באמערקט אז דער פראמפט איז פשוט, אבער ער גיט Gemini א קלארן עולם און א קלארן סטיל פון ערקלערונג. Gemini קען ערקלערן סייבער-זיכערהייט דורך שלעסער, קאמערעס, פאסווערטער, אלארמען און וועכטער — און דאס מאכט דאס טעמע גרינגער פאר אנהייבער.

דערנאך בעט מען א פארגלייך: "שאף א פשוט'ע טאבעלע וואס פארגלייכט סייבער-זיכערהייט באגריפן: סכנה (threat), שוואכקייט (vulnerability), אטאקע (attack), ריזיקע (risk), און שוץ (protection). ניץ איין ביישפיל פונעם לעבן פאר יעדן איינעם." טאבעלעס, ערקלערט דער קורס, מאכן קאמפליצירטע טעמעס גרינגער צו ערקלערן.

דערנאך ניצט מען Gemini פאר ארבעט מיט פיילס. למשל, מען האט א PDF אדער דאקומענט וועגן יסודות פון סייבער-זיכערהייט. מען לאדנט ארויף די פייל (מען זעט אז א צוגעלייגטע פייל איז אטעטשט) און שרייבט: "פאס צוזאם דעם דאקומענט אין פשוט'ן ענגליש. גיב מיר די שפיץ 10 פונקטן וואס אן אנהייבער דארף געדענקען. שאף אויך פינף קוויז-פראגעס דערפון." דאס איז נוצליך וועז מען וויל נישט ליינען א לאנגן דאקומענט מיט די הענט. אבער א וויכטיגער פונקט: אפילו וועז דאס געזעצ פאסט צוזאם דעם דאקומענט, דארף מען נאך אלץ קאנטראלירן די ארגינעלע פייל פאר וויכטיגע פרטים.

Gemini פאר וויזועלע אידייען און איבערשרייבן (19:07)

דערנאך ניצט מען Gemini פאר אויסטראכטן בילדער: "שלאג פאר פינף וויזועלע אידייען פאר זיכערהייט אין א YouTube לעקציע. מיידט אויס איבערגעפילטקייט, האלט די בילדער גרויס און פריינטליך." דאס קען געבן אידייען ווי: א לאפטאפ מיט א שלאס, א שילד וואס באשיצט דאטא, א העקער וואס ווערט אפגעשטעלט דורך א firewall, א טויער מיט א פאסווארט, אדער א באשיצטער וואלקן (cloud). די אידייען קען מען שפעטער ניצן אין Canva אדער Runway.

נאכדעם ניצט מען Gemini פאר איבערשרייבן. מען בעט איבערצושרייבן אין א מער שמועסדיגן סטיל די ערקלערונג: "סייבער-זיכערהייט איז די פראקטיק פון באשיצן סיסטעמען, נעצן און דאטא פון דיגיטאלע אטאקעס." Gemini קען עס איבערשרייבן אזוי: "טראכט פון סייבער-זיכערהייט ווי א דיגיטאלע וועלט פון גאס-חכמה (street smarts) — עס איז בעצם אלעס וואס מיר טוען כדי צו האלטן אונזערע מכשירים, אונזער

פערזענליכע אינפארמאציע און אונזער אינטערנעט-פארבינדונג זיכער פון מענטשן וואס דארפן נישט האבן צוטריט דערצו." דאס איז אסאך גרינגער פאר אנהייבער.

דער הויפט-פונקט: Gemini קען העלפן ערקלערן טעמעס, פארפשוט'ן אינהאלט, ארבעטן מיט פיילס, שאפן טאבעלעס, און ארויסגעבן וויזועלע אידייען.

Claude: פארפיינערן שריפט (20:00)

נאך ChatGPT און Gemini גייט מען איבער צו Claude. עס ווערט ערקלערט אז Claude איז נוצליך ווען מען וויל פאלירטע שריפט, לאנג-פארמיגע רעדאקטירונג, פארבעסערן דאקומענטן, און ריינע ערקלערונגען. עס איז אויך נוצליך פאר שטיצע מיט קאוד, באזונדערס ווען מען וויל אז קאוד זאל ערקלערט ווערן אין פשוט'ער שפראך. אין דעם אפטייל ווערן געמאכט צוויי דעמאנסטראציעס: ערשטנס, פארבעסערן א רויען פאראגראף; צווייטנס, שאפן און לויפן א פשוט'ן Python סקריפט.

דער פראמפט פאר שרייבן: "שפיל די ראלע פון א טרעינער וואס איז פריינטליך פאר אנהייבער, פארבעסער דעם פאלגנדן פאראגראף פאר א YouTube לעקציע. האלט עס פשוט, נאטירליך און שמועסדיג, און זאל עס נישט קלינגען צו ראבאטיש." דער פאראגראף איז: "סייבער-זיכערהייט איז וויכטיג ווייל אסאך מענטשן ניצן דעם אינטערנעט, און אטאקירער קענען גנב'ענען אינפארמאציע אויב די סיסטעמען זענען נישט באשיצט."

Claude שרייבט עס איבער אויף א גלאטערן אופן; מען זעט דעם פארבעסערטן פאראגראף, וואס איז גרינג צו פרעזענטירן וויל ער קלינגט מער נאטירליך. דערנאך בעט מען עס נאך מער צו פארפשוט'ן: "מאך דאס פאסיג פאר א גאנץ נייעם אנהייבער, און לייג צו איין ביישפיל פונעם לעבן." דער קורס באטאנט אז דער שריט איז וויכטיג ווייל מען שאפט נישט סתם אינהאלט — מען פאסט עס צו צום עולם.

Claude: א Python סקריפט וואס פאסט צוזאם א טעקסט (21:04)

דערנאך דער קאוד-טייל: מען שאפט א פשוט'ע Python פראגראם וואס נעמט א פאראגראף און גיט ארויס וויכטיגע שליסל-ווערטער און א קורצע צוזאמפאסונג. עס ווערט דערמאנט אז כדי צו לויפן דעם קאוד דארף מען האבן צוויי זאכן אינסטאלירט אויפן קאמפיוטער: Python און Visual Studio Code. אבער דערווייל לויפט מען עס גלייך אין Claude אליין.

דער פראמפט: "שאף א פשוט'ן Python סקריפט וואס פאסט צוזאם א פאראגראף אן ניצן קיין דרויסנדיגע לייברעריס. דער סקריפט זאל ארויסנעמען געוועניכע שליסל-ווערטער און דרוקן די צוויי וויכטיגסטע זאצן. ערקלער אויך דעם קאוד אין פשוט'ן ענגליש." Claude שאפט דעם קאוד, און מען ניצט די פשוט'ע וועריע. די ערקלערונג, שריט נאך שריט:

- די ערשטע שורה אימפארטירט) Counterפון Python'ס אייגענע ביבליאטעק). Counter העלפט ציילן וויפל מאל יעדעס וואָרט באווייזט זיך.
- די נעקסטע שורה אימפארטירט re, וואס העלפט געפינען ווערטער און זאצן דורך מוסטערן (patterns).
- מען שאפט א וועריאבל text, וואס אנטהאלט דעם פאראגראף וואס מען וויל צוזאמפאסן.
- מען צעטיילט דעם פאראגראף אין זאצן, כדי די פראגראם זאל פארשטיין יעדן זאץ באזונדער.
- מען נעמט ארויס די ווערטער פונעם טעקסט — נאר ווערטער מיט פיר אדער מער אותיות, וויל גאר קלייַנע ווערטער ווי "is", "to" און "of" זענען געווענליך נישט נוצליך אלס שליסל-ווערטער.

• מען שאפט א קליינע ליסטע פון "stop words" — געוועגליכע ווערטער וואס מען וויל נישט ציילן אלס וויכטיגע שליסל-ווערטער — און מען נעמט זיי ארויס פון דער ווערטער-ליסטע.

• Counter ציילט ווי אפט די איבערגעבליבענע ווערטער באווייזן זיך; די מערסט-געוועגליכע ווערן די שליסל-ווערטער.

• מען גיט יעדן זאץ א חשבון (score): א זאץ וואס אנטהאלט מער וויכטיגע שליסל-ווערטער באקומט א העכערן חשבון.

• צום סוף דרוקט די פראגראם די שפּיץ שליסל-ווערטער און די צוויי וויכטיגסטע זאצן אלס א קורצע צוזאמפאסונג.

דער רעזולטאט ווייזט די שפּיץ שליסל-ווערטער און א קורצע צוזאמפאסונג. דער וויכטיגער פונקט, לויט דעם קורס: AI העלפט שאפן א ארבעטנדיגן אנהייב-פונקט, און מען פארשטייט וואס יעדער טייל פונעם קאוד מיינט.

אזוי קענען אנהייבער ניצן AI געצייג פאר קאוד. די עצות: קאפירט נישט בלינדערהייט דעם קאוד; בעט דאס געצייג עס צו ערקלערן; פרעגט וואס יעדע שורה טוט און ווי אזוי עס צו לויפן; מען קען אויך פרעגן וועלכע פעלערן (errors) קענען פאסירן. און אויב עס קומט א פעלער, קאפירט דעם פעלער-מעלדונג און בעט דאס געצייג עס צו ערקלערן אין פשוט'ער שפּראך.

Perplexity: פארשונג מיט מקורות (23:06)

נאך שרייבן, פלאנירן און קאוד-שטיצע, גייט מען איבער צו פארשונג — דא ווערט Perplexity זייער נוצליך. Perplexity איז בעהילפיג ווען מען וויל דירעקטע ענטפערס מיט מקורות. דאס איז וויכטיג, ווייל ווען מען שאפט לערן-אינהאלט טאר מען זיך נישט פארלאזן נאר אויפן זכרון אדער אויף צופעליגע ענטפערס; מען דארף באשטעטיגן די וויכטיגע פונקטן.

אין Perplexity שרייבט מען: "וואס זענען די וויכטיגסטע יסודות פון סייבער-זיכערהייט, און פארוואס זאל מען עס לערנען אין 2026? גיב פשוט'ע ערקלערונגען און נעם אריין מקורות." Perplexity גיט א ענטפער מיט מקורות. דער שליסל-פונקט, באטאנט דער קורס, איז נישט סתם ליינען דעם ענטפער, נאר קאנטראלירן די מקורות: אויב Perplexity זאגט או עפעס איז וויכטיג, דארף מען עפענען דעם מקור און זען צי ער שטיצט טאקע דעם פונקט. דאס איז נוצליך פאר לעקציעס וויל עס העלפט אויסמיידן פאלשע אדער פארעלטערטע אינפארמאציע.

א בעסערער פראמפט: "שאף א ליסטע פון סייבער-זיכערהייט טעמעס פאר אנהייבער פאר א 45-מינוט YouTube לעקציע. נעם אריין פאסווערטער, phishing, malware, זיכער בלעטערן (safe browsing), באקאפס, און יסודות פון נעץ-זיכערהייט. האלט די ערקלערונג פשוט." דאס גיט א סטרוקטורירטן ענטפער.

דערנאך בעט מען Perplexity צו פארגלייכן מקורות: "פארגלייך רעקאמענדאציעס פאר אנהייבער אין סייבער-זיכערהייט פון פארלעסליכע מקורות, און גיב מיר די געמיינזאמע פונקטן וואס קומען ווידער און ווידער." דאס איז נוצליך ווייל אנשטאט זיך צו פארלאזן אויף איין מקור, זעט מען די איבערגעחזרט'ע אידייען איבער מערערע מקורות.

פאקטן-קאנטראל מיט Perplexity, און א געוועלטליכער טעות (24:30)

מען ניצט Perplexity אויך פאר פאקטן-קאנטראל. למשל, ChatGPT אדער Claude גיט א באהויפטונג ווי: "Phishing איז איינע פון די מערסט געווענליכע סייבער סכנות." מען קען גיין צו Perplexity און פרעגן: "געפין פארלעסליכע מקורות וואס קענען שטיצן אדער פאררעכטן די באהויפטונג: Phishing איז איינע פון די מערסט געווענליכע סייבער-זיכערהייט סכנות." דאס העלפט באשטעטיגן.

א געווענליכער טעות: אן אנהייבער קען קאפירן דעם גאנצן Perplexity ענטפער און אים גלייך ניצן. דאס איז נישט דער בעסטער וועג. א בעסערער וועג איז צו ניצן Perplexity פאר פארשונג, און דערנאך שרייבן די ערקלערונג אין אייער אייגענעם פשוט'ן לערן-סטיל. דער ארבעטס-סדר (workflow) איז:

- פארשן מיט Perplexity.
- קאנטראלירן די מקורות.
- פארצייכענען די וויכטיגע פונקטן.
- דערנאך ניצן ChatGPT אדער Claude צו פארוואנדלען די פונקטן אין א סקריפט אדער אינהאלט פאסיג פאר אנהייבער.

NotebookLM: לערנען פון אייגענע דאקומענטן (25:11)

דערנאך גייט מען איבער צו NotebookLM, וואס איז נוצליך ווען מען האט אייגענע דאקומענטן — ווי PDFs, נאטיצן אדער לערן-מאטעריאל — און מען וויל זיי שנעלער פארשטיין. דער שליסל-חילוק: Perplexity איז נוצליך פאר זוכן און פארשן איבער מקורות, בשעת NotebookLM איז נוצליך פאר ארבעטן מיט מקורות וואס מען גיט אליין צו.

א ביישפיל: מען האט א PDF מיטן נאמען "Cybersecurity Basic Notes" און מען לאדנט עס ארויף אין NotebookLM; ווען דער מקור איז צוגעלייגט, קען מען פרעגן פראגעס באזירט אויף דעם דאקומענט. אין דער דעמאנסטראציע האט מען נאך נישט געשאפן א דאקומענט וועגן סייבער-זיכערהייט, דערפאר ניצט מען א צופעליגע פייל requirement.txt, מיט די פאלגנדע פראמפטן:

- "פאס צוזאם דעם דאקומענט אין פשוט'ן ענגליש פאר אן אנהייבער" — דאס פאסט צוזאם דעם אינהאלט פונעם ארויפגעלאדנטן מקור.
- "שאף א לערן-וועגווייזער (study guide) פון דעם דאקומענט מיט שליסל-פונקטן, פשוט'ע דעפיניציעס און ביישפילן."
- "שאף 10 קוויז-פראגעס פון דעם דאקומענט, מאך זיי פריינטליך פאר אנהייבער און נעם אריין ענטפערס." דאס איז נוצליך פאר טרעינערס און אינהאלט-שאפער.

פאר שאפן א סקריפט: "באזירט אויף דעם ארויפגעלאדנטן דאקומענט, שאף א פינף-מינוטיגע ערקלערונג פון phishing פאר אנהייבער. ניץ פשוט'ע ערקלערונגען." דער קורס באטאנט דעם אויסדרוק "באזירט נאר אויף דעם ארויפגעלאדנטן דאקומענט" — ער זאגט דעם געצייג צו בלייבן נאענט צום מקור.

פאר א פארגלייך: "פון דעם דאקומענט, מאך א ליסטע פון די חילוקים צווישן malware, phishing און ransomware אין איין טאבעלע." דאס גיט אינהאלט גרייט צו לערנען. פאר חזרה: "גיב מיר די שפיץ פינף זאכן וואס איך אלס אנהייבער דארף געדענקען פון דעם טעמע." דאס העלפט ווייל לאנגע דאקומענטן קענען זיך פילן איבערוועלטיגנד.

א וויכטיגער פונקט: NotebookLM איז נוצליך, אבער מען דארף נאך אלץ קאנטראלירן דעם מקור אויב דאס טעמע איז וויכטיג. דאס געצייג העלפט בעסער פארשטיין, אבער מען איז נאך אלץ פאראנטוואָרטליך איבערצוקוקן דעם ענדגילטיגן רעזולטאט.

Gamma און Canva: פרעזענטאציעס און דיזיין (27:01)

דערנאך גייט מען איבער צו שעפערישע געצייג, וואס העלפן מאכן פרעזענטאציעס, דיזיינס, בילדער, וויידעאס און קול-איבערשפילונגען (voiceovers). מען הייבט אן מיט Gamma, וואס איז נוצליך וועט מען וויל שנעל פארוואנדלען אן אידיי אין א פרעזענטאציע אדער א וויזועלן דאקומענט. דער פראמפט: "שאף א פרעזענטאציע פאר אנהייבער וועגן סייבער-זיכערהייט. האלט עס פשוט און ריין. נעם אריין סליידס וועגן וואס סייבער-זיכערהייט מיינט און פארוואס עס איז וויכטיג, געוועליכע סכנות, פאסווארט-זיכערהייט, phishing, malware, זיכער בלעטערן, באקאפס, און צום סוף א חזרה. ניט קורצע פונקטן און פשוט'ע ביישפילן."

Gamma שאפט א ערשטן אנשטריך פון דער פרעזענטאציע. אבער ווידער טאר מען נישט בלינדערהייט ניצן דעם ערשטן אנשטריך: מען דארף איבערקוקן דעם פלוס פון די סליידס, אראפנעמען איבעריגן טעקסט, און זיכער מאכן אז די שפראך איז פשוט. א גוטע פרעזענטאציע, זאגט דער קורס, דארף שטיצן דעם טרעינער און נישט פארבייטן אים.

דערנאך Canva, וואס איז נוצליך פאר דיזיינען thumbnails, פאסטערס, סאציאלע מעדיע פאסטס, פרעזענטאציעס און ווידעא-מאטעריאלן. דער דיזיין-פראמפט: "שאף א קאנצעפט פאר א YouTube thumbnail פאר 'סייבער-זיכערהייט פאר אנהייבער'. ניט א ריינעם טעכנישן הינטערגרונט, א שילד-אייקאן, א לאפטאפ, און דיקן לייענבארן טעקסט. האלט דעם דיזיין פשוט און פראפעסיאנעל." ביים שאפן thumbnails דארף מען אויסמיידן צופיל איבערגעפילטקייט; דער צוקוקער דארף פארשטיין דאס טעמע אין צוויי סעקונדעס. אנשטאט לאנגע טעקסטן ווי "Complete Cybersecurity Tutorial for Beginners in 2026", ניצט מען קורצערע ווי "Cybersecurity Basics" אדער "Learn Cybersecurity".

Runway פאר ווידעא און ElevenLabs פאר קול (28:29)

Runway איז נוצליך וועט מען וויל שאפן קורצע ווידעא-בילדער. דער פראמפט: "שאף א קורצן קינאמאטישן בילד פון א לאפטאפ-סקרין באשיצט דורך א לייכטנדיגן דיגיטאלן שילד, האלט עס פראפעסיאנעל, ריין, און פאסיג פאר סייבער-זיכערהייט לעקציעס." דאס קען העלפן שאפן קליינע הקדמה-בילדער, הינטערגרונט-קליפס אדער ערקלער-סצענעס. אבער מען דארף ניצן בילדער מיט באזונענקייט: זיי דארפן שטיצן די ערקלערונג און נישט אפלענקען דערפון. אין דער דעמאנסטראציע לאדנט מען ארויף א בילד וואס פאסט צו סייבער-זיכערהייט און מען שאפט דעם ווידעא; מען זעט דעם געשאפענעם ווידעא, און עס ווערט באמערקט אז די בילדער שטיצן די ערקלערונג און לענקען נישט אפ.

צום סוף ElevenLabs, וואס איז נוצליך פאר שאפן קול-איבערשפילונגען פון טעקסט. למשל, מען פאסט אריין א קורצע דערציילונג: "סייבער-זיכערהייט איז די פראקטיק פון באשיצן אייערע מכשירים, קאנטעס און דאטא פון אומגעוואונטשענעם צוטריט." דערנאך קלייבט מען א קול און מען שאפט די אודיא. דער וועג: מען גייט צו Voices, דערנאך Create Voice, דערנאך Voice Design, און Generate Voice. אין דער דעמאנסטראציע ניצט מען א געוויסן פראמפט מיט א קול וואס מען האט אויסגעקליבן, און מען הערט דעם רעזולטאט: "היי אלע, לאמיר שמועסן וועגן סייבער-זיכערהייט פאר א רגע. עס איז בעצם וועגן האלטן אייער גאנצע דיגיטאלע זאכן זיכער. טראכט פון עס ווי אייער פערזענליכער אנליין באדיגארד. מיר רעדן וועגן באשיצן אייערע

מכשירים, אייערע קאנטעס, און אלע טייערע דאטא פון יעדן וואס דארף נישט האבן צוטריט. עס איז זייער וויכטיג היינטיגע צייטן."

פון קרעאטיוו געצייג ביז Generative AI (30:00)

צוזאמשטעלן די געצייג און גוטע פראקטיקן (30:00)

דער קורס ענדיגט די באהאנדלונג פון 11 Labs: דאס געצייג איז נוצליך פאר שאפער וואס דארפן א נאראציע (א שטימע וואס דערציילט) פאר קורצע ווידעאס, ערקלערונג-ווידעאס אדער טריינינג-מאטעריאל. אבער עס ווערט ארויסגעשטעלט א וויכטיגע וואַרענונג: מען טאר נישט נאכמאכן אדער באנוצן א צווייטנס שטימע אן רשות. מען דארף שטענדיג באנוצן שטימעס מיט אחריות.

דערנאך ווערן אלע קרעאטיווע געצייג צוזאמגעשטעלט: Gamma העלפט שאפן א פרעזענטאציע; Canva העלפט שאפן דיזיינס; Runway העלפט שאפן וויידעא-בילדער; און 11 Labs העלפט שאפן א שטימע-באגלייטונג (voiceover). וועג מען באנוצט זיי צוזאמען, קען מען זיך באווענגן פון א געדאנק ביז א פארטיגן אינהאלט אסאך שנעלער. מען קען זיי פארבינדן אין איין ארבעטס-גאנג (workflow) פאר א פירמע.

זעקס גוטע פראקטיקן (30:31)

נאכדעם וואס מען האט באנוצט עטליכע געצייג, ערקלערט דער קורס די בעסטע פראקטיקן:

- ערשטנס, זיין קלאר מיט די אנווייזונגען. נישט שרייבן פראמפטס פון איין וואַרט, נאר געבן דעם געצייג א ראלע, דעם קאנטעקסט, די אויפגאבע, און דעם פארמאט.
- צווייטנס, פארבעסערן די ענטפערס שריט ביי שריט. דער ערשטער ענטפער איז געווענליך נישט דער לעצטער. מען קען בעטן דעם געצייג עס צו פארפשוטן, פארקירצן, אויסברייטערן, צולייגן ביישפילן, טוישן דעם טאן, אדער אפילו איבערמאכן דעם ענטפער אין א טאבעלע.
- דריטנס, באשטעטיגן וויכטיגע פאקטן. וועג מען רעדט וועגן לויפיגע דאטא, געהאלטן ביי דזשאבס, אייגנשאפטן פון געצייג, געזעצליכע כללים, געזונט-עצות אדער זיכערהייט-אינפארמאציע, דארף מען קאנטראלירן אין פארלעסליכע מקורות.
- פערטנס, היטן פריוואטע אינפארמאציע. נישט אריינלייגן פאסווערטער, פריוואטע דאקומענטן, קונה-דאטא, פערזענליכע אדרעסן אדער געהיימע פירמע-אינפארמאציע אין די געצייג, סיידן מען פארשטייט קלאר די פריוואטקייט-כללים פון יענעם געצייג און די ארגאניזאציע ערלויבט עס.
- פינפטנס, באנוצן דעם ריכטיגן געצייג פאר דער ריכטיגער אויפגאבע: ChatGPT אדער Claude פאר שרייבן און ערקלערונגען; Perplexity פאר פארשונג; (און א געצייג פאר ארויפגעלאדענע דאקומענטן); Gamma און Canva פאר פרעזענטאציעס און דיזיינס; 11 Labs פאר voiceover; Runway פאר וויידעא-בילדער; און Zapier און Make פאר איינפאכע אויטאמאציע.
- זעקסטנס, האלטן דאס אייגענע טראכטן אקטיוו. AI געצייג קענען העלפן, אבער זיי דארפן נישט שטענדיג פארביטן דעם אייגענעם שכל. מען דארף שטענדיג איבערקוקן דעם לעצטן רעזולטאט.

געווענליכע טעותים (31:44)

דער קורס רעכנט אויס זעקס געווענליכע טעותים:

- ניצן א גאר פשוטן פראמפט.
- קאפירן ענטפערס אן קאנטראלירן.
- ניצן צופיל געצייג אן פארשטיין גוט כאטש איינעם.
- אריינלייגן צופיל טעקסט אין סליידס און דיזיינס.
- ניצן געשאפענע בילדער אדער שטימעס אן קאנטראלירן די באניץ-רעכטן און ערלויבענישן.
- צוטרויען קאוד אן אים לויפן און פארשטיין. וועמען ניצט AI געצייג פאר קאוד, דארף מען שטענדיג פרעגן: וואס טוט דער קאוד? ווי לויפט מען אים? ווי קענען פאסירן גרייזן, און ווי קען מען אים אויספרובירן?

עס ווערט ארויסגעשטעלט איין פשוטער כלל: AI געצייג זענען נוצליך וועמען ניצט זיי ווי געהילפן און נישט ווי פארבייטער. דער מענטש באשליסט נאך אלץ דעם ציל, קאנטראלירט דעם רעזולטאט, פארבעסערט די קוואליטעט, און נעמט אחריות פאר דעם לעצטן רעזולטאט.

אנהייב: AI יסודות פאר יעדן איינעם (32:31)

דא הייבט זיך אן א נייער טייל: "AI Foundation for Everyone". אין א וועלט וואס ווערט שנעל געשטאלטעט דורך קינסטליכע אינטעליגענץ, ווערט פארשטיין AI נויטיג פאר יעדן, נישט נאר פאר מומחים. דער טייל איז א נישט-טעכנישער וועגווייזער צו קלארמאכן AI און געבן פראקטישע פעאיקייטן פאר דער ארבעט. מען וועט באהאנדלען די יסוד-באגריפן פון AI, דעם נייעם פראנט פון Generative AI, ווי אזוי צו רעדן מיט אים דורך prompt engineering, און צום סוף, בויען שטארקע AI chatbots, אלעס אן פראגראמירן.

וואו AI ווערט שוין באנוצט (33:11)

AI איז שוין נישט קיין חלום פון דער צוקונפט; עס טוישט אסאך אינדוסטריעס און טיילן פונעם לעבן:

- געזונט: AI העלפט ביי מער פינקטליכע דיאגנאזן, פארשנעלערט דאס געפינען נייע מעדיצינען, און מאכט מעגליך פערזענליכע באהאנדלונג-פלענער.
- פינאנצן: עס ווערט באנוצט צו כאפן שווינדל (fraud), טרייבן אלגאריטמישע האנדל, און העלפן מיט ריזיקע-אפשאצונג.
- טראנספארט: מען זעט דעם אויפקום פון זעלבסט-פארנדיגע קארס און AI-געטריבענע טראפיק-סיסטעמען.
- קראמען: נוצן AI פאר פערזענליכע רעקאמענדאציעס און צו אפטימירן דעם סופליי-קייט.
- פארוויילונג: AI שטאלטעט וואס מען קוקט און הערט דורך רעקאמענדאציע-סיסטעמען, און אפילו שאפט נייעם אינהאלט.

מעלות און חסרונות (33:55)

ווי יעדע שטארקע טעכנאלאגיע, האט AI סיי מעלות סיי באגרעניצונגען. אויף דער פלוס-זייט: AI קען שטארק פארגרעסערן עפעקטיווקייט און פראדוקטיוויטעט, אויטאמאטיזירן לאנגווייליגע און איבערגע'חזר'טע אויפגאבעס, פארבעסערן די פינקטלעכקייט פון באשלוסן, און טרייבן נייע אנטדעקונגען און חידושים וואס זענען פריער געווען אוממעגליך. אויף דער אנדערער זייט: אנטוויקלען AI סיסטעמען קען זיין טייער, און זיי דארפן אפט ריזיגע מאסן דאטא כדי צו ארבעטן גוט. היינטיגע AI סיסטעמען פעלט צומאל דער "קאמאן

סענס" (פשוטער מענטשליכער שכל) וואס מענטשן פארמאגן, און עס איז דא א בארעכטיגטע זארג וועגן פארלירן דזשאבס ווען אויטאמאציע וואקסט.

AI אין טאג-טעגליכן לעבן און ביזנעס (34:38)

AI איז שוין טיף איינגעוועבט אין טאג-טעגליכן לעבן און אין דעם אפעראציע פון ביזנעסער. מען באגעגנט AI יעדעס מאל מען ניצט א ווירטועלן געהילף ווי Siri אדער Google Assistant, פארלאזט זיך אויף נאוויגאציע-אפליקאציעס, סקראלט דורך סאשעל-מידיע פידס וואס ווערן אויסגעקליבן דורך אלגאריטמען, אדער ניצט אימעיל סערוויסעס וואס פילטערן ארויס ספאם.

אין דער ביזנעס-וועלט ווערט AI באנוצט צו פארבעסערן קונה-סערוויס דורך chatbots, פאראויסזאגן מאשין-צוברעכונגען כדי צו מאכן פאראויס-אנהאלטונג (predictive maintenance), אויטאמאטיזירן מארקעטינג-קאמפיינס, און אנאליזירן ריזיגע מאסן דאטא צו באקומען ווערטפולע איינזיכטן. עס ווערט אויך דערמאנט אז שפראך-מאדעלן שאפן און באארבעטן מענטשליכע שפראך; זיי ווערן טריינירט אויף גרויסע דאטא-סעטס פון טעקסט כדי צו לערנען די סטאטיסטישע מוסטערן ווי ווערטער, פראזן און זאצן זענען געבויט.

עטישע זארגן (35:28)

ווען AI ווערט שטארקער און מער פארשפרייט, איז נויטיג באהאנדלען די עטישע פראגן וואס עס שטעלט ארויס. א הויפט-פראבלעם איז פארטיישקייט (bias) אין AI סיסטעמען, וואס קען פארלענגערן און אפילו פארשטארקן עקזיסטירנדע געזעלשאפטליכע פאראורטיילן. פריוואטקייט און דאטא-זיכערהייט זענען אויך העכסט וויכטיג, ווייל AI סיסטעמען פארלאזן זיך אפט אויף גרויסע מאסן פערזענליכע דאטא. דאס פארלירן דזשאבס צוליב אויטאמאציע איז א באדייטנדע געזעלשאפטליכע שווערייקייט. און עס זענען דא נאך זארגן, ווי די אנטוויקלונג פון אויטאנאמע וואפן, און דער באדארף אין ערקלערבארקייט און דורכזיכטיגקייט ווי אזוי AI סיסטעמען קומען צו זייערע באשלוסן.

פארטיישקייט אין AI (36:11)

א קריטישע עטישע שווערייקייט איז AI bias. סיסטעמען לערנען פון דער דאטא אויף וועלכער זיי ווערן טריינירט, און אויב די דאטא שפיגלט אפ עקזיסטירנדע געזעלשאפטליכע פארטיישקייטן, וועט דער AI זיי מסתמא פארלענגערן און אפילו פארשטארקן. ביישפילן וואס מען האט שוין געזען: פנים-דערקענונג סיסטעמען וואס ארבעטן וויניגער פינקטליך ביי מענטשן מיט טונקעלערע הויט, און אויפנעם-אלגאריטמען (hiring algorithms) וואס האבן אן כוונה פארגעצויגן קאנדידאטן פון געוויסע הינטערגרונטן. צו באהאנדלען דאס דארף מען אכטונג געבן אויף דער דאטא וואס ווערט באנוצט צו טריינירן AI, זיכער מאכן אז זי איז פארשידנארטיג און רעפרעזענטאטיוו, און א התחייבות צו יושר און אחריות דורכאויס דעם גאנצן אנטוויקלונג-פראצעס.

פריוואטקייט און דאטא-זיכערהייט (36:58)

אסאך AI סיסטעמען, באזונדערס די וואס געבן פערזענליכע סערוויסעס, פארלאזן זיך אויף ריזיגע מאסן דאטא, אפט אריינגערעכנט סענסיטיווע פערזענליכע אינפארמאציע. דאס שטעלט ארויס ערנסטע זארגן וועגן פריוואטקייט און זיכערהייט. די ריזיקע פון דאטא-איינברוכן און מיסברויך איז גאנץ רעאל. עס איז נויטיג איינפירן שטארקע דאטא-שוץ מיטלען, געבן פריאריטעט צו דורכזיכטיגקייט ווי דאטא ווערט געזאמלט און באנוצט, און געבן באניצער קאנטראל איבער זייער אייגענע דאטא.

פאראנטוואַרטליכע AI אנטוויקלונג (37:30)

כדי צו פארמינערן די ריזיקעס און זיכער מאכן אז AI ברענגט נוצן פאר דער גאנצער געזעלשאפט, דארף מען אננעמען פאראנטוואַרטליכע AI אנטוויקלונג. דאס מיינט זיך האלטן צו עטישע פרינציפן:

- יושר (fairness) – אז AI סיסטעמען זאלן נישט דיסקרימינירן.
- דורכזיכטיגקייט (transparency) – אז מען פארשטייט ווי AI סיסטעמען מאכן באשלוסן.
- אחריות (accountability) – פעסטשטעלן ווער איז פאראנטוואַרטליך ווען עפעס גייט שלעכט.
- זיכערקייט (safety) – אז AI סיסטעמען ארבעטן פארלעסליך.
- און פארשטייט זיך, שוץ פון פריוואטקייט.

אנטוויקלען עטישע אנווייזונגען און רעגולאציעס איז קריטיש, און דאס איז א געמיינזאמע אחריות פון AI אנטוויקלער, ארגאניזאציעס און פאליסי-מאכער.

טרענדן פאר דער צוקונפט (38:16)

דאס פעלד פון AI אנטוויקלט זיך זייער שנעל, און עטליכע טרענדן שטאלטעווען זיין צוקונפט:

- ווייטערדיגע פארשריטן אין דיפ-לערנען, וואס פירן צו נאך מער פיינע AI סיסטעמען.
- Generative AI, וואס קען שאפן נייעם אינהאלט ווי טעקסט, בילדער און מוזיק, איז גרייט צו רעוואלוציאנירן די קרעאטיווע אינדוסטריעס.
- Explainable AI אדער XAI ווערט אלץ וויכטיגער, ווייל מען וויל פארשטיין ווי AI סיסטעמען קומען צו זייערע באשלוסן.
- AI at the edge – א בייט צו פראצעסירן נענטער צום מקור פון דער דאטא.
- אויף לענגערע צייט, דער צוזאמגאנג פון quantum computing און AI קען עפענען גאנץ נייע מעגליכקייטן.

קאריערע-וועגן (39:04)

דער שנעלער ווּוקס פון AI שאפט א שטארקע פראגע אויף באהאוונטע פאכלייט. עס עפענען זיך א ברייטע ריי קאריערע-וועגן:

- מאשין-לערנען אינזשענירן – וואס פלאנירן און בויען AI סיסטעמען.
- דאטא-סייענטיסטן – וואס אנאליזירן דאטא צו ארויסנעמען איינזיכטן.
- AI פארשער – וואס שטופן די גרעניצן פון AI וויסן.
- AI עטיקער – וואס העלפן זיכער מאכן פאראנטוואַרטליכע אנטוויקלונג.
- NLP ספעציאליסטן – וואס ארבעטן אויף שפראך-באזירטע AI.
- Computer vision אינזשענירן – וואס אנטוויקלען סיסטעמען וואס קענען "זען".

די ראלעס פאדערן געוועלדיג א שטארקן יסוד אין קאמפיוטער-וויסנשאפט, מאטעמאטיק און סטאטיסטיק, ווי אויך ספעציאליזירטע וויסן אין AI געביטן.

וואס איז ?Generative AI (39:50)

דער קומענדיגער טעמע איז Gen AI אדער Generative AI. דאס איז א סארט קינסטליכע אינטעליגענץ וואס שאפט נייעם, ארגינעלן אינהאלט ווי טעקסט, בילדער, קול און ווידעא. עס לערנט מוסטערן פון ריזיגע מאסן עקזיסטירנדע דאטא כדי צו שאפן נייע רעזולטאטן וואס זענען ענליך צו דער דאטא אויף וועלכער עס איז טרייניירט געווארן.

ווי אזוי גרייכן די מאדעלן אזא זאך? עס גייט אין צוויי הויפט-שטאפלען:

- ערשטנס, דער שטאפל פון לערנען די פארטיילונג (distribution). מען שטופט אריין אין מאדעל ריזיגע מאסן דאטא, למשל מיליאנען בילדער, ארטיקלען אדער קול-שטיקלעך. ער אנאליזירט די דאטא גאר גענוי כדי צו פארשטיין אירע סטאטיסטישע אייגנשאפטן, באהאלטענע כללים, און די יסודותדיגע מוסטערן וואס באשרייבן זי. למשל, ווען ער לערנט וועגן מענטשליכע פנימער, פארשטייט ער זאכן ווי דאס געווענליכע ארט פון די אויגן, נאז און מויל, און ווי זיי שטייען איינס צום צווייטן.
- צווייטנס, דער שטאפל פון שאפן נייע ביישפילן. נאכן לערנען קען דער מאדעל "ציען" (sample) פון זיין אינערליכער פארשטעלונג. ער קאפירט נישט סתם; ער ניצט זיין געלערנטע פארשטאנד צו שאפן נייע דאטא וואס איז סטאטיסטיש ענליך צו דעם אויף וואס ער איז טרייניירט, און דאס מאכט זיי אומגלויבליך רעאליסטיש און נייארטיג.

Generative קעגן Discriminative AI (41:21)

כדי באמת צו פארשטיין Generative AI, דארף מען וויסן ווי ער איז אנדערש פון discriminative AI. דער קורס מאכט א פארגלייך:

- לערן-ציל: discriminative AI צילט צו פארבינדן אריינגאבן מיט באשטימטע לעיבלס אדער ווערטן; ער לערנט די גרענץ צווישן פארשידענע קאטעגאריעס. Generative AI צילט צו מאדעלירן די יסודותדיגע פארטיילונג פון דער דאטא; ער לערנט דעם גאנצן דאטא-רוים.
- סארט רעזולטאט: ביי discriminative AI איז דער רעזולטאט געווענליך א פאראויסזאגונג, א קלאסיפיקאציע אדער א צייכן (score) – למשל יא אדער ניין, קאץ אדער הונט, אדער א נומערישער ווערט. ביי Generative AI איז דער רעזולטאט נייע דאטא אליין – א נייע בילד, א נייער טעקסט, א נייער קול, אדער נייער קאוד.
- די פראגע: discriminative AI פרעגט "וואס איז דאס?" אדער "איז דאס X אדער Y?"; Generative AI פרעגט "וואס קען דאס זיין?" אדער גלייכער: "שאף עפעס ווי דאס".
- הויפט-אויפגאבעס: ביי discriminative – פאראויסזאגונג, קלאסיפיקאציע און רעגרעסיע; ביי generative – שאפונג, סינטעז און) data augmentationפארמערן דאטא).

דער יסודותדיגער אונטערשייד ווייזט זייערע פארשידענע צילן און צוגאנגען.

קאנקרעטע ביישפילן (42:47)

פאר discriminative AI: א בילד-דערקענונג סיסטעם וואס זאגט "דאס איז א קאץ אין בילד" – ער דערקענט וואס איז שוין דארט. א ספאם-פילטער קלאסיפיצירט אן אנקומענדיגן אימעיל אלס ספאם אדער נישט – ער אונטערשיידט צווישן צוויי קאטעגאריעס. א שווינדל-כאפן סיסטעם זאגט פאראויס צי א קרעדיט-קארטל טראנזאקציע איז א שווינדל, באזירט אויף היסטארישע מוסטערן.

פאר Generative AI: אנשטאט נאר דערקענען צי עס איז א קאץ אדער א הונט, קען Generative AI שאפן א גאנץ נייע בילד פון א קאץ וואס האט קיינמאל נישט עקזיסטירט – למשל א קאץ מיט א קליין היטל. אנשטאט נאר פילטערן ספאם, קען ער שרייבן א גאנץ נייעם אימעיל-אנטוווורף פון עטליכע שליסל-וועדטער. ביי שווינדל-כאפן קען Generative AI סינטעזירן נייע, רעאליסטישע אבער פאלשע טראנזאקציע-דאטא, כדי צו טריינירן א discriminative שווינדל-מאדעל מער עפעקטיוו, באזונדערס ווען עכטע שווינדל-דאטא איז זעלטן. די ביישפילן ווייזן קלאר דעם בייט פון אנאליז צו שאפונג.

די פעאיקייטן פון Generative AI (43:56)

די פעאיקייטן פון Generative AI זענען ריזיג און וואקסן וויייטער. עס גייט וויייט אריבער נאר פאראויסזאגן רעזולטאטן, צו אקטיוו שטאלטעווען און שאפן די דיגיטאלע וועלט ארום אונז. עס קען שאפן מענטשן-ענליכן טעקסט, אומגלויבליך רעאליסטישע בילדער, פונקציאנירנדן קאוד, נאטירלעך-קלינגענדע רייד און קול, און אפילו לעבעדיגע ווידעא. אויסער דירעקטע אינהאלט-שאפונג איז עס אויך שטארק פאר data augmentation, וואס איז קריטיש פאר טריינירן אנדערע AI מאדעלן.

דער ערשטער פעאיקייט וואס ווערט באהאנדלט איז טעקסט-שאפונג, אדער natural language generation (NLG). דאס איז איינע פון די מערסט באקאנטע פעאיקייטן היינט, און עס טרייבט א גרויסן טייל פון דעם עולם'ס באגעגעניש מיט AI. Generative AI מאדעלן קענען שאפן מענטשן-ענליכן טעקסט אויף פארשידענע טעמעס און אין פארשידענע סטילן, מיט א בריייטע ריי באניצונגען, אנגעהויבן פון אינהאלט-שאפונג.

געפֿערלעכקייטן פון AI אנווענדונגען און NLP יסודות (45:00)

טעקסט, בילדער און קאוד (45:00)

דער קורס פירט וויייטער מיט די אנווענדונגען פון געפֿערלעכקייט AI. ביים שאפן טעקסט ווערט דערמאנט אז די מאדעלן שרייבן ארטיקלען, בלאג-פאסטס, מארקעטינג-טעקסטן און סאשעל-מידיע מעלדונגען. זיי ווערן אויך געניצט פאר סומירונג, דאס הייסט צו פארקירצן לאנגע דאקומענטן. זיי שטייען הינטער די קליגערע chatbots און ווירטועלע געהילפן, וואס קענען דורכדעם פירן מער נאטירלעכע און צוזאמענהענגעדיקע שמועסן. זיי זענען אויך א שטארקער געצייג פאר שפּעריישן שרייבן, ווי לידער, מעשיות אדער סקריפטן.

דערנאך קומט דאס שאפן בילדער, אדער וויזועלע שאפונג. עס ווערט ערקלערט אז דאס איז אפשר דער פונקט וואו געפֿערלעכקייט AI האט אמערסטנס פארכאפט דעם עולם'ס פאנטאזיע. די מאדעלן קענען שאפן גאנץ נייע בילדער, פאטאגראפיעס און קונסט-ווערק פון א פשוטער טעקסט-באשרייבונג, אדער ענדערן בילדער וואס זענען שוין פארהאן. די אנווענדונגען וואס ווערן אויסגערעכנט:

- קונסט און דיזיין: שאפן אייגנארטיגע קונסט-ווערק, אילוסטראציעס אדער concept art.
- פראדוקט-פראטאטייפן: גיך זען פארשידענע ווערסיעס פון א פראדוקט, אדער ארכיטעקטור-דיזיין.
- פאטא-באארבעטונג: איבערמאלן (inpainting), דאס הייסט אויספילן פעלנדיקע טיילן פון א בילד; style transfer; און העכערן די רעזאלוציע (upscaling).
- שאפן רעאליסטישע טעקסטורן און סביבות פאר ווירטועלע וועלטן אין גיימס און סימולאציעס.

וויייטער ווערט ערקלערט אז געפֿערלעכקייט AI ווערט אלץ שטארקער אין שאפן קאוד און אויטאמאטיזירט טיילן פונעם אנטוויקלונגס-פראצעס. AI קען דינען ווי א פראגראמירונגס-געהילף: עס שרייבט שטיקלעך קאוד,

פונקציעס אדער אפילו פשוטע פראגראמען, לויט א באשרייבונג אין נאטירלעכער שפראך אדער לויטן קאנטעקסט פונעם קאוד וואס איז שוין פארהאן. די אנוועדונגען:

- code completion: עס שלאגט פאר שורות קאוד בשעת מען טייפט, און דאס פארגיכערט באדייטנד די ארבעט.
- פאררעכטן באגס: עס געפינט פעלערן און שלאגט פאר א תיקון.
- boilerplate: עס שאפט סטאנדארט סטרוקטורן פון פונקציעס, class-דעפיניציעס און געווייַנטלעכע קאנפיגוראציעס.
- איבערמאכן נאטירלעכע שפראך אויף קאוד: דער פראגראמירער באשרייבט אין פשוטן ענגליש וואס ער ווייל, און די AI מאכט דאס איבער אויף ארבעטנדיקן קאוד.

קול, מוזיק, ווידעא און דאטא-אוגמענטאציע (47:03)

דער קורס רעדט ווייטער וועגן שאפן רייד און אודיא. די פעאיקייט לאזט AI שאפן רעאליסטישע מענטשלעכע רייד, קאמפאנירן ארגינעלע מוזיק און שאפן פארשידענע קלאנג-עפעקטן. די אנוועדונגען:

- text-to-speech: איבערמאכן יעדן געשריבענעם טעקסט אויף נאטירלעך-קלינגענדע רייד, פאר ווירטועלע געהילפן, אודיא-ביכער אדער דערציילונגען.
- voice cloning: נאכמאכן א באשטימט קול.
- שאפן מוזיק: ארגינעלע מעלאדיעס, הארמאניעס אדער אפילו גאנצע אינסטרומענטאלע שטיקער.
- שאפן באזונדערע קלאנג-עפעקטן פאר גיימס, פילמען אדער באניצער-אינטערפעיסעס.

עס ווערט געבעטן מען זאל זיך פארשטעלן א AI וואס דערציילט די סליידס פון א פרעזענטאציע, אדער וואס קאמפאנירט אן אייגנארטיגע הינטערגרונט-מוזיק פאר א ווידעא. דאס איז די נייע כח פון אודיא-שאפונג.

די קאמפליצירטסטע פעאיקייט, וואס גייט אבער זייער גיך פאראויס, איז שאפן ווידעא. דאס מיינט פראדוצירן נייעם ווידעא-אינהאלט, באלעבן שטילע בילדער, אדער ענדערן ווידעא-פוטידזש וואס איז שוין פארהאן, כדי צו שאפן לעבעדיקע און פארכאפנדע סצענעס. די אנוועדונגען:

- video synthesis: שאפן קורצע ווידעא-קליפס פון א טעקסט-באשרייבונג אדער פון בילדער.
- אנימאציע: העלפן אנימאטארן דורך שאפן די באוועגונגען פון די פיגורן און lip syncing.
- ווירטועלע אוואטארן: שאפן רעאליסטישע "רעדנדיקע קעפ" ווידעאס פון טעקסט.
- פארגעשריטענע ספעציעלע עפעקטן: פארבעסערן אדער ענדערן ווידעא-אינהאלט אויף אופנים וואס מען האט פריער געהאלטן פאר אוממעגלעך.

עס ווערט באמערקט אז AI ווידעא-שאפונג איז נאך אלץ א נייע גרענעץ, אבער דער פארשריט איז פארבלופנדיק און גיט א שעפערישע קאנטראל וואס מען האט נישט ערווארט.

צום סוף ווערט באהאנדלט דאטא-אוגמענטאציע. דאס איז א וויכטיגע פעאיקייט וואס ארבעט הינטער די קוליסן און פארבעסערט די לייסטונג פון אנדערע AI מאדעלן. דאטא-אוגמענטאציע איז דער פראצעס פון פארגרעסערן די מאס און די פארשידנקייט פון די טריינירונגס-דאטא. געווארטיווע AI שפילט דא א וויכטיגע ראלע: עס שאפט געענדערטע ווערסיעס פון דאטא וואס איז שוין פארהאן. נאך שטארקער, עס קען שאפן גאנץ

נייע, רעאליסטישע דאטא וואס האלט זיך צו די מוסטערן וואס מען האט אויסגעלערנט פון דער אמת'דיקער דאטא. פארוואס איז דאס וויכטיג?

- עס באהאנדלט דעם מאנגל אין דאטא וועז אמת'דיקע דאטא איז באגרעניצט.
- עס מאכט די מאדעלן שטארקער (robustness), ווייל זיי ווערן וויניגער שפירעוודיק צו קליינע ענדערונגען.
- עס העלפט קעגן אומבאלאנסירטע דאטא, אזוי אז מאדעלן זאלן גוט לערנען אויך פון קאטעגאריעס וואס זענען שוואך פארטראטן.
- עס איז וויכטיג פאר פריוואטקייט, ווייל מען קען נוצן סינטעטישע דאטא וועז די אמת'דיקע דאטא איז סענסיטיוו.

דער קורס פאסט עס אזוי צוזאם: AI לערנט אנדערע AI מאדעלן, דורכדעם וואס עס שאפט מער פארשידנארטיגן און פולשטענדיקן לערן-מאטעריאל. עס ווערט אנגעזאגט אז מען וועט ווייזן ביישפילן פון שאפן בילדער און ווידעאס פון א פראמפט.

וואס איז NLP? (50:08)

דא הייבט זיך אן א נייער חלק, וואס זאל ערקלערן NLP אין ערך צען מינוט. דער פלאן: קודם א יסודות'דיקער באגריף פון NLP, פארוואס מען דארף עס און וואס דאס איז פונקטלעך. דערנאך NLP pipelines, וואס באהאנדלען די יסוד-פראצעסן פון באארבעטן טעקסט. דאס הייסט איבערמאכן טעקסט אויף א פארעם וואס מאשינען קענען פארשטיין, אזוי ווי מענטשן פארשטייען עס. צום סוף, אנווענדונגען פון NLP אינעם עכטן לעבן, וואס זענען ארום אונז און פיל מער געווייַנטלעך ווי מען מיינט.

קודם ווערט באטראכט ווי מענטשן רעדן און האבן צו טון איינער מיטן אנדערן. מען קאמוניקירט מערסטנס דורך רייד אדער טעקסט. כדי אנצוזאגן פאר איינעם וואס מען וויל, רעדט מען גלייך מיט אים אדער מען שיקט אים א מעסעדזש. דער מענטשלעכער אופן פון קאמוניקירן הייסט "נאטירלעכע שפראך". איבער דער וועלט רעדט מען אסאך שפראכן, און כמעט יעדן טאג טיילט מען זיך מיט מעסעדזשעס. אלע די שמועסן, אפרופן און מעסעדזשעס זענען אליין דאטא. די דאטא איז זייער ווערטפול, ווייל זי קען געבן אינפארמאציע וועגן קונים און אן איינזיכט אין מענטשלעכע געפילן. אבער פאר קאמפיוטערס איז די דאטא נישט ניצלעך, ווייל זי איז נישט אין א פארעם וואס מאשינען קענען פארשטיין. מאשינען קאמוניקירן מיט איינסן און נולן, נישט מיט ווערטער. זיי פארשטייען נישט קיין ענגליש, פראנצויזיש אדער שפאניש, נאר binary. און דא קומט אריין NLP.

NLP באדייט Natural Language Processing, באארבעטן נאטירלעכע שפראך. דאס איז א צווייג פון AI וואס באהאנדלט די קאמוניקאציע צווישן מענטשן און קאמפיוטערס דורך נאטירלעכער שפראך. דער לעצטער ציל פון NLP איז צו לייענען, אויפטייטשן, פארשטיין און כאפן דעם זין פון מענטשלעכע שפראכן אויף א ניצלעכן אופן. און צו בויען סיסטעמען וואס פארשטייען טעקסט און טוען ארבעטן ווי איבערזעצונג, קאנטראלירן גראמאטיק אדער סארטירן לויט טעמע. בקיצור, NLP באארבעט די געשריבענע פארעם פון שפראכן און מאכט אז א קאמפיוטער זאל זיי פארשטיין. פירמעס נוצן אלץ מער געצייג מיט NLP, כדי צו באקומען איינזיכטן פון דאטא און צו אויטאמאטיזירן רוטין ארבעטן. למשל, א sentiment analyzer קען העלפן פירמעס דערקענען געפילן אין טעקסט, ווי נעגאטיוע קאמענטארן אויף סאשעל מידיע. ביים סוף פונעם טאג איז NLP נישט מער ווי א צוזאמשטעל פון AI, קאמפיוטער-וויסנשאפט און מענטשלעכער שפראך.

NLP pipeline: סעגמענטאציע און טאקעניזאציע (52:47)

א pipeline איז א רייע עלעמענטן וואס באארבעטן דאטא און זענען פארבונדן איינס נאכן אנדערן, אזוי אז דער ארויסגאנג פון איין עלעמענט איז דער אריינגאנג פונעם קומענדיקן. ער שטעלט מיט זיך פאר די שריט וואס מען דארף מאכן אין דער רעכענונג, און אין וועלכן סדר זיי קומען. אין דעם pipeline טוט מען parsing, דאס הייסט איבערמאכן די דאטא פון איין פארעם אויף א צווייטע. דא מאכט מען איבער אריינגעגעבענע טעקסט-דאטא אויף א פייל וואס א קאמפיוטער קען פארשטיין.

אט איז דער סדר: דער טעקסט ווערט קודם צעטיילט אויף סעגמענטן. די סעגמענטן ווערן דערנאך צעטיילט אויף נאך קלענערע tokens. מען געפינט דעם word stem פון יעדן token, און דערנאך דאס וואָרצל-ווארט פון די stems, אין א פראצעס וואס הייסט lemmatization. נאכדעם דערקענט מען צו וועלכן חלק פון רייד א וואָרט געהערט, ווי verb, noun אדער pronoun. צום סוף זונדערט מען אפ די באקאנטע entities פון די ווערטער, כדי בעסער צו פארשטיין וואס דער זאץ וויל זאגן. דער לעצטער פייל איז איינער וואס א קאמפיוטער פארשטייט.

סעגמענטאציע איז דער ערשטער שריט. sentence segmentation, אדער text segmentation, הייסט צעטיילן א טעקסט אויף לאגיש פארשטענדליכע איינהייטן אינפארמאציע. מען צעטיילט דעם זאץ אויף זיינע אונטער-זאצן, געוועלך לויט די פונקטואציע, ווי פונקטן אדער קאמעס, אדער לויט שורה-ברעכן און בלאט-טיילן ביי HTML פיילס. וועלן מען צעטיילט א דאקומענט אויף זיינע זאצן, קען מען אים באארבעטן אן צו פארלירן זיין עיקר און די נויטיקע אינפארמאציע. דער ביישפיל: "The lemonade quenched her thirst but not her longing". נאך דער סעגמענטאציע ווערן דערפון צוויי: "The lemonade quenched her thirst" און "but not her longing".

טאקעניזאציע איז נישט מער ווי צעטיילן א זאץ אויף זיינע ווערטער. יעדעס וואָרט פונעם זאץ ווערט אפגעהיטן באזונדער. מען טוט דאס כדי צו פארשטיין די סינטאקטישע און סעמאנטישע אינפארמאציע אין יעדן זאץ. מען אנאליזירט אים וואָרט ביי וואָרט, און אזוי ווערט קיין אינפארמאציע נישט פארלוירן. ווייל דער קאמפיוטער פארשטייט נישט פונקטואציע און ספעציעלע צייכנס, קען מען זיי אראפנעמען. ביים ביישפיל "The lemonade quenched her thirst" באקומט מען פינף tokens: The, lemonade, quenched, her, thirst. יעדער ווערט באהאנדלט ווי א באזונדערע זאך.

stemming, lemmatization, part of speech און NER (56:11)

נאך טאקעניזאציע קומט stemming: געפינען דעם word stem פון א וואָרט, וואס הייסט אויך די יסוד-פארעם פונעם וואָרט. צו דער יסוד-פארעם קען מען צולייגן affixes און שאפן נייע ווערטער, אין א פראצעס וואס הייסט inflection. stemming דערקענט די stems דורך אראפנעמען affixes ווי ing, s, ed א.א.וו. ביישפיל: jump איז דער stem פון jumping, jumped און jumps. נעמט מען אראפ די affixes, בלייבט jump. דאס איז וואס מען ווייל באקומען ביים סוף פון stemming.

lemmatization געפינט דאס וואָרצל-וואָרט, די מערסט יסוד'דיקע פארעם, וואס הייסט אויך lemma, פון יעדן וואָרט אין זאץ. עס איז זייער ענלעך צו stemming, וואו מען נעמט אויך אראפ affixes. דער חילוק איז אז דאס וואָרצל-וואָרט איז שטענדיק א וואָרט וואס געפינט זיך אינעם ווערטערבוך, וועלן דער stem קען אמאל נישט זיין. lemmatization נוצט א וויסנס-באזע וואס הייסט WordNet. ביישפיל: went, going און gone שטאמען אלע פון איין וואָרט, go. דאס איז די lemma, און אלע אנדערע ווערטער פירן זיך צוריק צו איר.

part of speech tagging מאכט איבער א זאץ אויף פארשידענע פארמען, ווי א ליסטע ווערטער אדער א ליסטע tuples. דער tag צייגט אן צי דאס ווארט איז א noun, adjective, verb א.א.וו. מען צעטיילט די ווערטער לויט זייערע גראמאטישע טיילן. דאס איז א זייער וויכטיגער שריט כדי צו פארשטיין דעם טייטש פון א זאץ, ארויסנעמען באציאונגען און בויען א knowledge graph, ווייל דאס זעלבע ווארט קען זיין א פארשידענער חלק פון רייד אין פארשידענע זאצן. אין "give me your answer" איז answer א noun, אבער אין "answer the question" איז עס א verb. ביים ביישפיל-זאץ: The איז א determiner, lemonade א noun, quenched א verb, her א pronoun, thirst א noun.

דער לעצטער שריט וואס ווערט דא באהאנדלט איז named entity recognition (NER) . עס הייסט אויך named entity identification, entity chunking אדער entity extraction. דאס איז אן אונטער-ארבעט פון אינפארמאציע-ארויסנעמען, וואס זוכט צו געפינען און סארטירן באנאמענטע entities אין אומסטרוקטורירטן טעקסט, לויט פריער באשטימטע קאטעגאריעס. געפינען די הויפט-entities אין א טעקסט העלפט סארטירן אומסטרוקטורירטע דאטא און דערקענען וויכטיגע אינפארמאציע. דאס איז קריטיש ווען מען האט צו טון מיט גרויסע דאטא-סעטס. די קאטעגאריעס דא זענען: א מענטש (person); א כמות, ווי קילאגראם; אן ארט; אן ארגאניזאציע; דער נאמען פון א פילם; אדער א געלט-ווערט, ווי דאלארן אדער יוראס.

NLP-אנווענדונגען, AI-ווידעא און פראמפט-אינזשעניריע (1:00:00)

וואו ווערט NLP געניצט: טשעטבאטס (1:00:00)

דער קורס הייבט אן מיט א פראגע: מען האט שוין געזעדט וויאזוי מען טוט natural language processing (NLP), אבער וואס זענען די אנווענדונגען דערפון, און וואו ווערט עס גענוצט אין דער עכטער וועלט?

די ערשטע אנווענדונג איז טשעטבאטס. א טשעטבאט קען העלפן אויפלייזן פראבלעמען, און דערביי פירט ער אויס natural language generation, דאס הייסט ער קען פירן א שמועס אין פשוט'ן ענגליש. א טשעטבאט איז נישט מער ווי א ווייכווארג-אפליקאציע וואס פירט אן אנליין שמועס, סיי דורך טעקסט און סיי דורך רעדן, אנשטאט דעם וואס דער באניצער זאל האבן א דירעקטן קאנטאקט מיט א לעבעדיגן מענטשלעכן אגענט. דער קורס ברענגט א ביישפיל: אויף וועבזייטלעך זעט מען אפט א טייל וואו עס שטייט "רעדט מיט איינעם פון אונזערע אגענטן", און דאס זענען געווענליך טשעטבאטס. אסאך פירמעס ניצן אויך טשעטבאטס אויף WhatsApp, כדי דער גאנצער פראצעס זאל אויסזען ווייניגער מעכאניש.

רייד-דערקענונג און אויטאמאטישע קארעקטור (1:00:46)

די צווייטע אנווענדונג איז רייד-דערקענונג (speech recognition). לויט דעם קורס זענען די באקאנטסטע ביישפילן פון NLP אין אקציע די ווירטועלע געהילפן, ווי Google Assistant, Siri און Alexa. NLP פארשטייט די מענטשלעכע שפראך און איבערזעצט זי אויף נומערן, אזוי אז א מאשין קען זי גרינג פארשטיין. דער ביישפיל אין דער לעקציע: מען פרעגט Siri "וואו איז די נאענטסטע גאז-סטאנציע?". די געהילפן כאפן אויף ווען מען רעדט, זיי פארוואנדלען דאס רעדן אויף טעקסט, און זיי פארשטייען וואס מען האט געבעטן. במשך די יארן זענען די ווירטועלע געהילפן געווארן אזוי גלאט, אז זיי קענען נאכמאכן מענטשלעכע רייד-מוסטערן כמעט אן קיין פעלער.

די דריטע אנווענדונג איז אויטאמאטישע קארעקטור (autocorrect). מען רופט עס אויך "טעקסט-פארבייטונג" (text replacement) אדער "פארבייטן בשעת'ן טייפן" (replace as you type). דאס

איז אן אויטאמאטישע פונקציע וואס באשטעטיגט צי די דאטא איז ריכטיג, און מען טרעפט זי אין ווארט-פראצעסארן און אין טעקסט-רעדאקטירונג אויף סמארטפאונס און טאבלעטן. זי ארבעט ווי א ספעל-טשעק און פארריכט אויסלייג- און גראמאטיק-פעלער וואס קענען אויפקומען בשעת מען טייפט. געווויסע שפראך-פראגראמען, ווי Grammarly, Paperrater, Reverso און אנדערע, קענען אפילו קאנטראלירן ווי אייגנארטיג און ווי אינטערעסאנט אן ארטיקל איז. און דאס אלעס ווערט געטון מיט NLP.

א ווידעא פון א פראמפט מיט invideo AI (1:02:01)

דערנאך ווערט געוויזן די היים-בלאט פון invideo AI. דער קורס באשרייבט עס ווי א וואונדערליכע געצייג צו שאפן ווידעאס פון הויכער קוואליטעט, סיי פון טעקסט-פראמפטס און סיי פון בילדער, און זאגט אז עס איז זייער פיל-זייטיג. אויף דער נאוויגאציע-באר זעט מען אפציעס ווי "טויזנטער שטימען", "טעקסט צו קליפ" און "טעקסט צו בילד". די טריט אין דער דעמא:

- מען קליקט אויף "טעקסט צו קליפ", ווייל מען וויל שאפן א ווידעא.
- מען קליקט אויף "create AI video from scratch", דאס הייסט שאפן אן AI-ווידעא פון אנהייב.
- מען דארף אריינגעבן א טעמע אדער א פרטימדיגע אנווייזונג וועלכן ווידעא מען וויל האבן. עס זענען דא אויך אנדערע אפציעס: א קורצער ווידעא, אן ערקלער-ווידעא, אן אנימירטער פילם, אדער ניצן א שוין-געשריבענעם סקריפט.
- דער פראמפט אין דער דעמא באשרייבט א קאץ וואס זיצט ארום אין שטוב און גייט ארויס אויספארשן די שטאט פון איר אייגענעם בליק-ווינקל. די וויזועלע סצענעס: די קאץ קומט אן צום פיש-מארק, און דערנאך קומט זי צוריק אהיים פונקט בזמנו.
- מען וויל אן אנימירטן פילם, אבער די פאסיגע ווערסיע איז נישט פאראן, דעריבער קליקט מען אויף "generate video" און מען ווארט עטליכע סעקונדעס.
- מען זעט ווי די אפליקאציע אנאליזירט, און זי גיט אליין א טיטל, בערך "די געהיימע לעבן פון א קאץ: מיסיע...". דער קורס רופט דאס זייער איינדרוקספול.
- מען קלייבט א וויזועלן סטיל: רעאליסטיש, אנימאציע אדער אנימע. מען קלייבט "רעאליסטיש", א לענג פון אונטער 30 סעקונדעס, און די פלאטפארמע YouTube Shorts.
- מען קען אויך נעמען די Plus ווערסיע, וואו מען קען אויסקלייבן יעדן וויזועלן סטיל, פאר 35 דאלער א חודש. אין דער דעמא ווערט אבער גענוצט די פרייע ווערסיע, אזוי אז מען גייט ווייטער מיט stock media, דאס הייסט פארטיגע מעדיע-מאטריאלן, און מען ווארט ביז דער ווידעא ווערט געשאפן.

דערנאך ווערט דער ווידעא געשפילט. א שטימע דערציילט וועגן "Whiskers", אן אומדערשראקענער קאץ, וואס גנב'עט זיך ארויס אויף א "סטראג-געהיימע מיסיע" צו אויספארשן די בעהאלעמדיגע גאסן פון Bengaluru. ערשט שפרינגט ער פון איין דאך צום אנדערן ווי א סופער-העלד, שטעלט זיך אפ מיט א צאפלדיגן עק און שפיאנירט די וועלט אונטן. דערנאך דרייט ער זיך דורך דעם טראפיק ווי א פראפעסיאנאל. צום סוף שמעקט ער און טרעפט דעם פיש-מארק. דער קורס זאגט אז דער רעזולטאט איז זייער איינדרוקספול: מיט בלויז איין פראמפט, נאר א טעקסט, באקומט מען א גאנצן ארבעטנדיגן ווידעא.

וואס איז פראמפט-אינזשעניריע? (1:05:09)

דערנאך ווערט ערקלערט וואס פראמפט-אינזשעניריע איז. אין די פשוטסטע ווערטער איז עס די קונסט און די ווייסנשאפט פון צוגרייטן איינגאבעס, דאס הייסט פראמפטס, כדי צו קאמוניצירן אויף אן עפעקטיוע אופן מיט

AI-מאדעלן ווי E·GPT-3, DALL און אנדערע. מען דארף זיין געווי, שעפעריש און סטראטעגיש, כדי ארויסצוברענגען די בעסטע מעגליכע ענטפערס. פארוואס איז דאס וויכטיג? מיט דעם ריכטיגן פראמפט קען מען ארויסנעמען דעם פולן פאטענציאל פון AI, פון שאפן קינסטלערישע מייסטערווערק ביז אויפלייזן קאמפליצירטע פראבלעמען.

די יסודות: א גוטער פראמפט דארף זיין קלאר, קורץ און צילגעריכטעט. מען דארף אויסמיידן צווייטייטשיגקייט, און מען דארף באשרייבן אזויפיל ווי נויטיג. מען דארף געדענקען אז מען פירט דעם AI צו פארשטיין פונקט וואס מען וויל. און דא קומט אריין דער "כישוף" פון פראמפט-אינזשעניריע: אז מען טוישט נאר עטליכע ווערטער, קען דער רעזולטאט זיך דראסטיש טוישן.

די פיר באשטאנדטיילן פון א פראמפט (1:06:00)

ווען מען קוקט אויף פארשידענע ביישפילן פון פראמפט-אינזשעניריע, ווערט קלאר אז א פראמפט באשטייט פון עטליכע הויפט-טיילן. א פראמפט קען אנטהאלטן יעדע קאמבינאציע פון די פאלגנדע:

- אנווייזונג (instruction): א געניע אקציע אדער באפעל וואס מען וויל אז דער מאדעל זאל אויספירן.
- קאנטעקסט (context): הינטערגרונט-אינפארמאציע אדער נאך צוגאב-קאנטעקסט, וואס קען פירן דעם מאדעל צו מער געניע ענטפערס.
- איינגאב-דאטא (input data): די ספעציפישע פראגע אדער דאטא אויף וועלכע מען זוכט א ענטפער.
- ארויסגאב-אנווייזער (output indicator): די געוואונטשענע סטרוקטור אדער נאטור פונעם ענטפער.

נישט אלע פיר טיילן זענען נויטיג אין יעדן פראמפט, און ווי מען ניצט זיי וועגדט זיך אין דער ספעציפישער אויפגאבע.

עצות פון פראפעסיאנאלן: ניצט ספעציפישע און פרטימדיגע שפראך, עקספערימענטירט מיט פארשידענע פראזעס, און נעמט אין באטראכט די באגרעניצונגען פונעם מאדעל. און מען טאר נישט פארגעסן צו איבערחזר'ן (iterate): די בעסטע פראמפט-אינזשעניר פארבעסערן כסדר זייערע פראמפטס לויט די ענטפערס פונעם AI.

עס ווערט אויך געווארנט אז אפילו עקספערטן מאכן טעותים. אויב מען מאכט דעם פראמפט צו קאמפליצירט, אדער מען איז צו אומקלאר, אדער מען פארשטייט נישט ריכטיג וואס דער AI קען, קען מען באקומען אומגעריכטע רעזולטאטן, אמאל אפילו לעכערליכע. מען דארף לערנען פון די טעותים כדי צו פארפולקומען די טעכניק. די דעמא ווערט געמאכט מיט ChatGPT, אין דער GPT-4 ווערסיע, און דער צוקוקער ווערט פארבעטן מיטצומאכן מיט זיין אייגענער AI-געצייג.

ערשטער פראמפט: טעקסט-צוזאמענדריקונג (1:07:52)

דער ערשטער פראמפט הייסט "טעקסט-צוזאמענדריקונג און פארפיינערונג". דאס איז א וויכטיגער כשרון אין NLP, און מען ניצט עס אין א ברייטן קרייז פון געביטן. איינע פון די מערסט אינטערעסאנטע אנווענדונגען פון שפראך-מאדעלן איז די מעגליכקייט צו דיסטילירן קאמפליצירטע ארטיקלען און באגריפן אין קורצע, גרינג-פארשטענדליכע סיכומים.

דער ביישפיל: מען איז נייגעריג ווען דער ראלע פון אנטיאקסידאנטן אינעם קערפער, און מען שרייבט: "summarize the role of antioxidants". ChatGPT גיט א ענטפער. כדי עס נאך מער צו פארפיינערן, קען מען בעטן דעם מאדעל עס איינצופאסן אין א קירצערן פארמאט, למשל: "ניץ דעם קאנטעקסט פון אויבן און

פאס עס צוזאם אין 100 וועלט", אדער 200 וועלט, אדער "דריק עס צוזאם אין א קורצע ערקלערונג". אין דער דעמא נעמט ChatGPT דעם פראמפט און דריקט צוזאם דעם אינהאלט פון פריער אין א קורצע ערקלערונג. אזוי קען מען ניצן פארשידענע פארמען.

א בילד מיט E·DALL (1:09:53)

דערנאך גייט מען איבער צו E·DALL, און מען ניצט דעם זעלבן צוזאמגעדריקטן פארמאט פאר א פראמפט: "generate a highly detailed image of a futuristic city", דאס הייסט א זייער פרטימדיגע בילד פון א צוקונפטיגער שטאט. מען קען ספעציפיצירן די צייט פונעם טאג, "at dusk" (בין השמשות), מיט הויכע וואלקן-קראצערס מיט שיינענדיגע פענצטער, פליענדיגע קארס, און א הימל פול מיט וואלקנס אין נעאן-פארבן. מען קען צולייגן וואס נאר מען דארף. E·DALL שאפט די בילדער און ענטפערט אז דאס זענען די בילדער פון דער צוקונפטיגער שטאט בין השמשות, מיט אלע פרטים וואס מען האט געבעטן. אויף דעם זעלבן אופן קען מען געבן פראמפטס צו E·DALL אדער צו יעדן אנדערן AI-געצייג און באקומען דעם ערווארטעטן רעזולטאט, און מען קען איבערארבעטן די פראמפטס לויט וואס דער AI קען.

אינפארמאציע-סינטעז און אריסנעמען (1:12:16)

צוריק אין ChatGPT וועלט געוויזן א נייער סארט פראמפט: "אינפארמאציע-סינטעז און אריסנעמען". שפראך-מאדעלן זענען נישט בלויז גוט אין שאפן שפראך, נאר אויך אין אריסנעמען און צוזאמשטעלן אינפארמאציע פון א טעקסט. מען גיט א טעקסט וואס זאגט: פארשער ניצן אפט פארשידענע AI-געצייג אין זייערע שטודיעס, אריינגערעכנט LLMs ווי OpenAI'ס GPT. אן ערליכע אנטפלעקונג פון די געצייג העלפט היטן די אמונה פון א פובליקאציע. מחברים דארפן אנטפלעקן אין וועלכן קאנטעקסט און אין וועלכע מאס זיי האבן גענוצט AI-הילף, און זשורנאלן דארפן זיין אפן וועגן זייער AI-באנוץ אינעם רעצענזיע-פראצעס. די פראגע אויף דעם טעקסט: "identify the specific LLM mentioned in the text", דאס הייסט וועלכער LLM (large language model) וועלט דערמאנט אינעם טעקסט. ChatGPT ענטפערט ריכטיג: OpenAI'ס GPT.

פארבעסערטער ענטפער אויף פראגעס (1:13:42)

דער לעצטער פראמפט אין דעם אפשניט הייסט "enhanced query response". כדי צו פארבעסערן ווי מאדעלן ענטפערן אויף פראגעס, דארף מען שרייבן מער פיינע פראמפטס, וואס פירן דעם AI צו געבן גענויע און נוצליכע ענטפערס. דער פראמפט: "גיב א ענטפער באזירט אויפן געגעבענעם קאנטעקסט. זיי קורץ און קלאר. אויב דו ביסט נישט זיכער, ענטפער 'cannot confirm'". דער קאנטעקסט: די נייע מעדיצין "azumab" איז ערשט אנטוויקלט געווארן אלס רעזולטאט פון א שותפות'דיגער ארבעט פון עטליכע ביאטעק-פירמעס. די מעדיצין איז געצילט צו באהאנדלען אויטאימיונע קראנקייטן, און זי איז א באדייטנדיגער דורכברוך אין ביאמעדיצין. די פראגע: "what is the purpose of azumab?" ChatGPT ענטפערט אז דער ציל פון azumab איז צו באהאנדלען אויטאימיונע קראנקייטן. דאס איז דער סארט פראמפט וואס מען רופט enhanced query response.

פראמפט-סארטן און טשעטבאטס אן קאוד (1:15:00)

פארגעשריטענע טעקסט-קאטעגאריזאציע: סענטימענט-אנאליז (1:15:00)

דער קורס הייבט אן מיט א קורצן סך-הכל פונעם פריערדיגן טייל: פראמפט-אינזשעניריע אפטימירט ווי אזוי מאדעלן ענטפערן אויף פראגעס, דורכן שרייבן מער אויסגעארבעטע פראמפטס וואס פירן דעם AI צו געבן געוויע און נוצליכע ענטפערס.

דערנאך גייט מען איבער צום נעקסטן סארט פראמפט: פארגעשריטענע טעקסט-קאטעגאריזאציע. עס ווערט ערקלערט אז ביי מער קאמפליצירטע אויפגאבעס קען מען שטארק פארבעסערן די ארבעט פונעם מאדעל, אויב מען פארבינדט דעטאלירטע אנווייזונגען מיט קאנטעקסט און ביישפילן.

אין דער לעקציע ווערט געוויזן ווי מען שרייבט א פראמפט וואס בעט דעם מאדעל אפצושאצן דעם "סענטימענט" פון געוויסע זאצן, דאס הייסט צו סארטירן יעדן זאץ אלס פאזיטיוו, נעגאטיוו אדער נייטראל. דער פראמפט אין ChatGPT לויטעט בערך אזוי: "שאץ אפ דעם סענטימענט פונעם פאלגנדן זאץ אלס פאזיטיוו, נעגאטיוו אדער נייטראל". נאכדעם גיט מען דעם זאץ: "די אויסגעצייכנטע בילדער פונעם פילם פארשאטענען זיין שוואכע געשיכטע".

מען זעט אז ChatGPT ענטפערט אז דער סענטימענט פון דעם זאץ איז "געמישט". עס ווערט ערקלערט סיי די פאזיטיווע זייט, די אויסגעצייכנטע בילדער, סיי די נעגאטיווע זייט, די שוואכע געשיכטע. דער קורס ציט דערפון אראפ אז מען קען ניצן ChatGPT פאר סענטימענט-אנאליז: מען גיט אריין זאצן און מען פרעגט צי זיי זענען פאזיטיוו, נייטראל אדער נעגאטיוו.

דינאמישע שמועסן (1:17:07)

דער נעקסטער סארט פראמפט הייסט "דינאמישע שמועסן". עס ווערט ערקלערט אז צו בויען שמועס-סיסטעמען וואס פאסן זיך צו פארשידענע טענער און שווערigkייטן, איז א זייער אינטערעסאנטע אנווענדונג פון פראמפט-אינזשעניריע. דער ציל איז א סיסטעם וואס גיט פשוטע, פריינטליכע ענטפערס.

מען הייבט אן א שמועס מיטן מאדעל מיט דער פראגע: "שלום, וואס איז דיין ראלע?" ChatGPT ענטפערט אז זיין ראלע איז צו העלפן דעם באניצער: צו געבן אינפארמאציע, צו ענטפערן אויף פראגעס און צו העלפן מיט א ברייטע סארט אויפגאבעס, און עס פרעגט: "מיט וואס קען איך דיר היינט העלפן?"

דערנאך פירט מען וויטער דעם שמועס און מען פרעגט צי עס ווייסט וואס איז פאטאסינטעז, און אויב יא, זאל עס דאס ערקלערן. ChatGPT ענטפערט אז יא, און ערקלערט אז דאס איז א פראצעס וואס פלאנצן, אלגעס און געוויסע באקטעריעס ניצן כדי איבערצודרייען ליכט-ענערגיע, געווענליך פון דער זון, אין כעמישע ענערגיע. עס גיט אויך ביישפילן. דער קורס באטאנט אז מען קען אזוי וויטער פירן דעם שמועס מיטן AI-מאדעל.

קאוד-שרייבן מיט א פראמפט (1:19:02)

דער נעקסטער סארט פראמפט איז "אינאוואטיוו קאוד-שאפונג". עס ווערט ערקלערט אז דאס איז א הרחבה פון די מעגליכקייטן וואס LLMs, דאס הייסט גרויסע שפראך-מאדעלן, האבן צו שרייבן קאוד, און מען וועט זיי בעטן שרייבן קאוד פאר אונז.

דער פראמפט לויטעט: "שרייב א Python פונקציע וואס רעכנט אויס דעם factorial פון א נומער און דרוקט אויס דעם רעזולטאט." מען זעט אז ChatGPT ווייזט ערשט "Analyzing", דאס הייסט עס אנאליזירט דעם פראמפט, און נאכדעם שאפט עס דעם קאוד. דער קאוד קומט שוין מיט באמערקונגען וואס ערקלערן אים.

דער קורס גיט אן עצה: אויב מען וויל בעסער פארשטיין דעם קאוד, קען מען בעטן ChatGPT עס זאל אים ערקלערן שורה ביי שורה. מען זעט אז ChatGPT טוט דאס און ערקלערט יעדע שורה, צוזאמען מיט אלע נויטיגע באגריפן.

אלס סך-הכל ווערט געזאגט אז די אלע פארבעסערטע פראמפטס און ביישפילן וועלן ווי ווייטגרייכנדיג און שטארק שפראך-מאדעלן קענען זיין, וועלן מען גיט זיי גוט-סטרוקטורירטע און דורכגעטראכטע אנווייזונגען. וואס מער מען פרובירט און עקספערימענטירט מיט די פראמפטס, אלץ קלארער ווערט די מעגליכקייט צו שאפן אויסגעארבעטע, פיינע און זייער פונקציאנעלע אפליקאציעס וואס ארבעטן מיט AI. דאס זענען די סארטן פראמפטס וואס מען קען אליין פרובירן.

AI-טשעטבאטס אן פראגראמירן: פארוואס ביזנעסער ווילן זיי (1:21:16)

דער נעקסטער טעמע איז ווי אזוי מען בויט AI-טשעטבאטס אן פראגראמירן. עס ווערט ערקלערט אז פילע מיינען, וועלן זיי הערן "AI-טשעטבאט", אז עס פאדערט קאמפליצירטן קאוד און ספעציעלע פראגראמירונג-שפראכן. אבער דער מצב האט זיך גרונטיק געטוישט: היינטיגע געניראטיווע AI און "no-code" מכשירים מאכן אז אויסגעארבעטע טשעטבאטס זענען מער נישט נאר א זאך פאר פראגראמירער. מען קען בויען מער ווי פשוטע פראגע-ענטפער באטס, נאר דינאמישע, קלוגע געהילפן.

דער קורס זאגט אז דער פאקוס איז אויף פעאיגקייטן וואס ביזנעסער זוכן טאקע: צו פארשטיין וואס דער באניצער דארף, צו פלאנירן לאגישע שמועס-פלוסן, און צו פארבינדן פארשידענע AI-טיילן, אלעס אן שרייבן איין שורה קאוד.

פארוואס ווילן ביזנעסער אזוי שטארק אריינברענגען AI-טשעטבאטס? די מעלות זענען:

- זיי זענען צוגענגליך 24/7, אזוי אז קונים קענען באקומען ענטפערס אדער הילף צו יעדער צייט, אין וועלכן צייט-זאנע זיי זאלן נאר זיין.
- מער עפעקטיווקייט: געוועלטיגע פראגעס און אויפגאבעס ווערן אויטאמאטיש, און דאס באפרייט די מענטשליכע קונה-סערוויס ארבעטער צו באהאנדלען מער קאמפליצירטע אדער עמפינדליכע ענינים.
- א בעסערע איבערלעבונג פארן קונה: שנעלערע ענטפערס, גלייכמעסיגע אינפארמאציע און קירצערע וואַרט-צייט.
- וויניגער הוצאות: דער ביזנעס קען פארגרעסערן זיין קונה-סערוויס אן מווון אויפנעמען פראפארציאנעל מער ארבעטער.

רעקאמענדאציע-סיסטעמען אין א טשעטבאט (1:23:00)

עס ווערט ערקלערט אז איינער פון די שטארקסטע וועגן צו פארבעסערן א טשעטבאט איז אריינצושטעלן אין אים א רעקאמענדאציע-סיסטעם מיט AI. דאס ברענגט דעם טשעטבאט פון אלגעמיינע ענטפערס צו א פערזענליכע קונה-סערוויס.

וואס זענען רעקאמענדאציע-סיסטעמען? דאס זענען אלגאריטמען וואס אנאליזירן דאטא כדי פארצושלאגן פאסיגע פראדוקטן, סערוויסעס אדער אינפארמאציע פאר יעדן באניצער באזונדער. אין א no-code

טשעטבאט-סביבה מיינט דאס אז מען שטעלט איין דעם טשעטבאט צו כאפן סימנים פונעם באניצער, למשל זיינע אויסגעדריקטע פרעפערענצן, זיין בלעטער-היסטאריע (אויב דאס איז פארבונדן), אדער פריערדיגע שמועסן. דערנאך, מיט פארויס-באשטימטע כללים אדער פשוטע פארבינדונגען, ווייזט די טשעטבאט-פלאטפארמע צוגעפאסטע פארשלאגן פונעם פראדוקט- אדער סערוויס-קאטאלאג.

די מעלות זענען קלאר: מער פארקויפן דורך געצילטע רעקאמענדאציעס, צופרידענערע באניצער ווייל זיי פילן אז מען פארשטייט זיי, און די מעגליכקייט צו אנבאטן הילף פון פאראויס.

קאמפליצירטע ווארקפלאוס: אינטענטס, ענטיטיס, אקציעס (1:24:01)

דערנאך רעדט דער קורס וועגן בויען קאמפליצירטע "workflows". דא ווערט דער טשעטבאט טאקע לעבעדיג און קען באהאנדלען פארשידענע באדערפענישן פון באניצער, נישט נאר זיין א פשוטער באט.

א workflow איז אין גרונט דער סטרוקטורירטער וועג פון א שמועס, באזירט אויף וואס דער באניצער שרייבט און אויף דער לאגיק פונעם סיסטעם. אין no-code פלאטפארמעס ארבעט מען מיט עטליכע הויפט-עלעמענטן:

- Intents: זיי העלפן דעם AI פארשטיין וואס דער באניצער וויל. למשל: "איך וויל קאנטראלירן דעם סטאטוס פון מיין באשטעלונג."
- Entities: וויכטיגע פרטים וואס מען ציט ארויס פון דער בקשה, ווי למשל א באשטעלונג-נומער.
- אקציעס: לויט די צוויי טוט דער טשעטבאט אן אקציע, למשל אינפארמאציע וויזן, א וויטערדיגע פראגע פרעגן, אדער זיך פארבינדן מיט א דרויסנדיגן סיסטעם.
- באדינגונג-לאגיק: אלעס ווערט געפירט לויט דעם כלל "אויב דער באניצער זאגט X, טו Y."

דער קורס באטאנט אז דורכאויס דעם גאנצן פראצעס איז גרינטליכע טעסטירונג גאר וויכטיג, כדי דער שמועס זאל גיין גלאט, לאגיש און אן טעותים.

פלאנירן און טעסטירן א workflow (1:25:07)

עס ווערט געזאגט אז פלאנירן א קאמפליצירטן workflow איז א קונסט. מען הייבט געווענליך אן מיט אויסצייכענען דעם שמועס: מען צייכנט אויס די מעגליכע וועגן וואס א באניצער קען גיין, און ווי דער טשעטבאט וועט ענטפערן ביי יעדן שריט. דערנאך גייט מען איבער צו פראטאטיפן: מען בויט פריע ווערסיעס כדי צו טעסטירן די הויפט-פונקציעס.

וועגן טעסטירן גיט דער קורס עצות, און ווארנט אז מען זאל זיך נישט פארלאזן נאר אויף זיך אליין:

- זעלבסט-טעסטירונג: גיין דורך יעדן מעגליכן סצענאר וואס מען קען זיך פארשטעלן.
- טעסטירונג דורך חברים: בעטן קאלעגעס זיי זאלן פרובירן "צוברעכן" דעם באט, מיט אומגעווענליכע פראגעס אדער אומגעריכטע וועגן.
- אויב מעגליך, UAT (User Acceptance Testing) מיט אמת'ע ענד-באניצער, כדי צו באקומען עכטע רעאקציעס.

דער גאנצער פראצעס גייט אין רונדן: מען פארבעסערט כסדר דעם טשעטבאט לויט וואס מען האט זיך אויסגעלערנט פון די טעסטן, ביז ער איז פעסט און באקוועם פארן באניצער.

ארויסשטעלן דעם טשעטבאט אין דער וועלט (1:26:04)

צום סוף קומט די גרויסע מינוט: דעם טשעטבאט ארויסשטעלן אין א פאקטישער סיטואציע. דאס פאדערט עטליכע הויפט-שריטן:

- אויסקלייבן דעם ריכטיגן קאנאל, דארט וואו די ציל-באניצער רעדן: דער וועבזייטל, Facebook Messenger, WhatsApp אדער אנדערע פלאטפארמעס. No-code פלאטפארמעס מאכן די פארבינדונג גאר פשוט, אפט נאר מיט קאפירן און אריינלייגן דעם embed-קאוד, אדער מיט איינשטעלן אן API-שליסל.
- נאכן ארויסשטעלן, נאכקוקן ווי דער טשעטבאט ארבעט: נאכפאלגן די שמועסן מיט באניצער, דערקענען די געווענליכע פראגעס און כאפן טעותים.
- דער קורס ווארנט אז טשעטבאטס זענען נישט "איינשטעלן און פארגעסן". זיי פאדערן כסדר'דיגע אויפהאלטונג, דערהיינטיגן דעם אינהאלט און שטענדיגע פארבעסערונג לויט אמת'ע דאטא פון באניצער.
- שטענדיג געדענקען זיכערהייט און פריוואטקייט, בפרט וועז מען האט צו טון מיט עמפינדליכע קונה-אינפארמאציע.

פאפולערע no-code טשעטבאט-פלאטפארמעס (1:27:01)

דער קורס ברענגט עטליכע פלאטפארמעס וואס האבן א באקוועמען drag-and-drop אויסשטעל, וואו מען קען בויען שמועס-פלוסן, זיי פארבינדן מיט דאטא-קוואלן, און ארויסשטעלן דעם טשעטבאט אויף וועבזייטלעך, אפליקאציעס אדער מעסעדזש-פלאטפארמעס. יעדע איינע האט אירע מעלות און חסרונות:

- Tidio: גוט פאר קונה-סערוויס און צו געווינען נייע קונים (lead generation). קען זיך פארבינדן מיט וועבזייטלעך, Messenger, Instagram און נאך. גיט אויך מוסטערן און ענטפערס מיט AI.
- Chatfuel: פאפולער פאר Facebook Messenger באטס, פאסיג פאר e-commerce און קונה-באטיילגונג.
- ManyChat: קאנצענטרירט זיך אויף מארקעטינג- און פארקויף-באטס, ארבעט גוט מיט Messenger, Instagram און WhatsApp.
- Landbot: האט א שטארקן drag-and-drop בויער, קען שאפן טשעטבאטס פאר וועבזייטלעך און WhatsApp באטס, מיט א ריינעם אויסשטעל פארן פלאנירן די שמועס-איבערלעבונג.
- א לעצטע פלאטפארמע, וואס איר נאמען איז נישט קלאר אין דער אויפנאמע, גיט AI-מעגליכקייטן און שטיצע פאר פיל שפראכן, פאסיג פאר מער קאמפליצירטע workflows מיט פארבינדונגען.

דעמא: א טשעטבאט אין Tidio מיט Lyro (1:28:08)

אין דער לעקציע וועלן מיר גענוצן Tidio מאכט מעגליך א טשעטבאט אן קאוד. בשעת'ן איינשרייבן פרעגט עס פאר אן email, און וואס איז דער ציל: צי מען וויל שאפן א טשעטבאט, א Messenger פארבונדן מיט א טשעטבאט, Instagram, אדער WhatsApp שמועסן.

אין Tidio איז דא אן איינגעבויטער AI-אגענט מיטן נאמען Lyro, און עס ווייזט אן די שריטן צו שאפן א טשעטבאט:

- ערשטנס לייגט מען צו "וויסן": מען גיט די URL פונעם וועבזייטל אויף וועלכן דער טשעטבאט וועט זיין. אינעם ביישפיל גיט מען אריין דעם Simplilearn וועבזייטל.

- מען דריקט "Import knowledge", און דאס ציט ארויס די אינפארמאציע פונעם זייטל. דאס נעמט א ביסל צייט: עס גייט דורך אלע לינקס און בלעטער אונטער simplilearn.com, למשל דעם קאנטאקט-בלאט און אנדערע בלעטער, יעדע אינפארמאציע וואס דער באט דארף וויסן.
- בשעת דעם ווערט דער AI-טשעטבאט טריינירט. נאכדעם קען מען טעסטירן Lyro און מיט אים שמועסן.
- אויב מען וויל נישט אז א געוויסער לינק זאל אריינגיין אין די דאטא, קען מען דארט קליקן און אים אויסמעקן, ווי מען זעט טאקע אינעם דעמא.

א פלאן ווי צו ווערן אן AI אינזשעניר (1:30:00)

פראבירן דעם טשעטבאט Lido און דער סוף פון "AI Foundation for everyone" (1:30:00)

אין דער לעקציע ווערט ווייטער געוויזן ווי מען פרובירט אויס דעם טשעטבאט וואס מען האט פריער אויפגעבויט, און וואס הייסט Lido. ווען מען עפנט אים, זאגט ער "hi, how can I help you", און מען זעט אז עס זענען דא פארגעשטעלטע פראגעס (pre-built questions) וואס דער באניצער קען אויסקלייבן, כדי אויסצופרובירן די וויסן וואס דער טשעטבאט האט. מען זעט אז ער גיט פראגעס און ענטפערס.

דערנאך ווערט אריינגעשריבן א ספעציפישע פראגע, כדי צו זען צי די וויסן איז טאקע דא: מען פרעגט אים צי ער קען פארשלאגן א קורס וואס וועט העכערן די סקילס פאר דער איצטיגער פאזיציע. דער טשעטבאט ענטפערט אז כדי צו רעקאמענדירן דעם פאסיגסטן קורס, זאל מען אים זאגן וואס איז די איצטיגע פאזיציע אדער פעלד פון ארבעט, ווייל מען האט פארשידענע ספעציאליזאציעס. דאס הייסט, ער בעט אינפוט פונעם באניצער.

מען ענטפערט אים: "מיין איצטיגע פאזיציע איז data analyst". דער טשעטבאט שלאגט דאן פאר עטלעכע קורסן פאר א data analyst: ער רעקאמענדירט א business analyst קורס, מיט טריינירן אין Excel און Power BI און א Tableau סערטיפיקאציע, און ער זאגט אז די קורסן וועלן פארבעסערן די סקילס אין דאטא-וויזואליזאציע און אנאליז. ער גיט אויך אינפארמאציע וועגן די געהאלטן און די סערטיפיקאציע, אלע אינפארמאציע ווערט צוגעשטעלט.

דער קורס באטאנט אז אזא טשעטבאט קען מען אריינלייגן אין דעם אייגענעם וועבסייט, און מען האט א טשעטבאט אן צו שרייבן אפילו איין שורה קאוד.

דערמיט ווערט פארענדיגט דער טייל "AI Foundation for everyone". עס ווערט צוזאמגעפאסט וואס מען האט געלערנט: א יסודות'דיגער פארשטאנד וואס AI איז; די שעפערישע כח פון generative AI; ווי אזוי צו רעדן ווירקזאם מיט די מאדעלן דורך prompt engineering; און ווי צו בויען פראקטישע טשעטבאטס וואס לויפן אויף AI, אלעס אן שרייבן א שורה קאוד. די וועלט פון AI טוישט זיך כסדר, אבער מען האט איצט א פעסטן יסוד זיך דארט צו געפינען מיט בטחון. מען דערמאנט אז AI איז א געצייג, און מיט די וויסן וואס מען האט באקומען קען מען בעסער אויסנוצן איר כח, אויף א פאראנטווארטליכן און שעפערישן אופן. מען מוטיגט ווייטער צו פארשן, עקספערימענטירן און אנווענדן וואס מען האט געלערנט.

פארוואס AI אינזשענירן, און וואס זיי טוען (1:32:33)

דא הייבט זיך אן א נייער טייל: א "ראוד-מעפ" ווי אזוי צו ווערן אן AI אינזשעניר. עס ווערט ערקלערט אז ווען AI מאכט א רעוואלוציע אין פארשידענע אינדוסטריעס, שטייען די AI אינזשענירן אין די פראנט פון דער

טעכנאלאגיע. זיי זענען נויטיג כדי צו בויען אינטעליגענטע סיסטעמען וואס לייזן קאמפליצירטע ביזנעס-פראבלעמען.

AI פראיעקטן פאלן אפט דורך צוליב שוואכע פלאנירונג פון דעם deployment, נישט-גוטע ארכיטעקטור אדער פראבלעמען מיט) scalabilityוואקסן צו גרויסע מאסן). AI אינזשענירן שפילן א הויפט-ראלע צו איבערקומען די מניעות, דורך פאראייניגן די נייסטע AI טעכנאלאגיעס מיט סטראטעגישע פארשטאנד.

וואס טוט אן AI אינזשעניר? ער בויט AI מאדעלן מיט מאשין-לערנען אלגאריטמען און דיפ-לערנען נוראל-נעצן. די מאדעלן זענען וויכטיג צו ארויסברענגען ביזנעס-אינפארמאציע (insights) וואס באווירקט ווי אן ארגאניזאציע נעמט באשלוסן. די ארבעט איז ברייט: פון אנטוויקלען אפליקאציעס וואס ניצן sentiment) analysisאנאליז פון געפילן אין טעקסט) פאר) contextual advertisingרעקלאמע לויט דעם קאנטעקסט), ביז סיסטעמען פאר וויזועלע דערקענונג און איבערזעצן שפראכן. כדי צו מצליח זיין דארף מען א מישונג פון טעכנישע פעאיגקייטן און) soft skillsמענטשליכע פעאיגקייטן).

חודש איינס: יסודות פון computer science און אנהייב Python (1:33:49)

דער קורס שטעלט פאר אן אכט-חדשים פלאן. דער ערשטער חודש איז computer science יסודות און Python פאר אנהייבער. איידער מען גייט אריין אין AI, דארף מען האבן א שטארקן יסוד אין computer science. די טעמעס:

- דאטא-פארשטעלונג: פארשטיין bits און bytes, ווי אזוי טעקסט און נומערן ווערן אפגעהיטן, און די בינארישע נומער-סיסטעם. דאס איז דער יסוד פון אלעס אין קאמפיוטינג, און עס העלפט פארשטיין ווי קאמפיוטערס טייטשן און פראצעסירן דאטא.
- קאמפיוטער-נעצן: די יסודות, אריינגערעכנט IP adresses און internet routing protocols. מען דארף פארשטיין ווי דאטא פארט איבער נעצן מיט UDP, TCP און HTTP, וואס זענען דער רוקנביין פון אינטערנעט און דעם World Wide Web.
- יסודות פון פראגראמירן: variables, strings, נומערן,) conditionalsתנאים), loops און די יסודות פון אלגאריטמען, כדי צו קענען שרייבן און פארשטיין פשוטע פראגראמען.
- צוגלייך הייבט מען אן מיט Python, די שפראך וואס מען באפרעפערט פאר AI: variables, numbers, strings, lists, dictionaries, sets, tuples, און קאנטראל-סטרוקטורן ווי if conditionals און for loops.
- פונקציעס און מאדולן: ווי צו שאפן פונקציעס, אריינגערעכנט lambda פונקציעס, און ארבעטן מיט מאדולן, מיט pip install כדי צוצולייגן פונקציאנאליטעט.
- file handling און exceptions: לייענען פון און שרייבן אין טעקעס, און באהאנדלען) exceptionsפעלערן) כדי די פראגראמען זאלן זיין מער פעסט.
- צום סוף: די יסודות פון classes און objects, וואס זענען וויכטיג פאר שרייבן ארגאניזירטן און עפעקטיוון קאוד.

דער איבערבליק גרייט צו פאר קאמפליצירטערע אויפגאבן אין די קומענדיגע חדשים.

חודש צוויי: דאטא-סטרוקטורן, אלגאריטמען און העכערער Python (1:35:25)

אין צווייטן חודש בויט מען אויפן יסוד און מען גייט אריין אין דאטא-סטרוקטורן און אלגאריטמען:

- big O notation, כדי צו פארשטיין די עפעקטיווקייט פון פארשידענע אלגאריטמען און דאטא-סטרוקטורן.

- דאטא-סטרוקטורן: arrays, linked lists, hash tables, stacks, queues, trees און graphs. וועז מען באהערשט זיי, קען מען אפהיטן און באארבעטן דאטא אויף א גוטן אופן.
- אלגאריטמען: binary search, bubble sort, quick sort, merge sort און recursion. זיי זענען וויכטיג כדי צו אפטימירן דעם קאוד.
- פאראלעל דערצו פארשטארקט מען די Python סקילס: inheritance, generators, iterators, list comprehensions, decorators, multi-threading און multi-processing. די טעמעס לאזן שרייבן מער עפעקטיוון און scalable קאוד.

דער חודש גרייט צו צו באהאנדלען קאמפליצירטע דאטא-אפאראציעס און מען וועלט עפעקטיוואר אין קאוד.

חודש דריי: version control, SQL און באארבעטן דאטא (1:36:17)

אין דריטן חודש גייט דער פאקוס צו מיטארבעט און פירן דאטא:

- version control: פארשטיין די וויכטיגקייט פון version control סיסטעמען, באזונדערס Git און GitHub. מען לערנט די יסוד-באפעלן add, commit און push; ווי צו באהאנדלען branches, צוריקדרייען ענדערונגען, און באגריפן ווי head, diff און merge. די סקילס זענען זייער וועלטפול צו נאכפאלגן ענדערונגען און ארבעטן צוזאמען מיט אנדערע פראגראמירער.
- pull requests: באהערשן ווי צו שאפן און פירן pull requests כדי ביישטייערן צו געמיינזאמע פראיעקטן.
- SQL פאר פירן דאטאבעיסעס: אנהייבן מיט די יסודות, relational databases און פשוטע queries; דערנאך העכערע queries מיט טעכניקן ווי CTE, sub-queries און window functions.
- joins און דאטאבעיס-פירונג: פארשידענע סארטן joins, ווי left, right, inner און full join; אויך ווי צו שאפן דאטאבעיסעס, פירן indexes און שרייבן stored procedures.
- דערצו ניצט מען NumPy און pandas צו באארבעטן דאטא, און מען לערנט פשוטע דאטא-וויזואליזאציע.

די סקילס וועלן זיין וויכטיג וועז מען גייט אריבער צו העכערע data science טעמעס.

חודש פיר: מאטעמאטיק און סטאטיסטיק פאר AI (1:37:28)

מאטעמאטיק און סטאטיסטיק זענען דער רוקנביין פון AI, און דער חודש איז געווידמעט צו זיי:

- ערשט: descriptive קעגן inferential סטאטיסטיק; continuous קעגן discrete דאטא; nominal קעגן ordinal דאטא; מאסן פון central tendency ווי mean, median, mode; און מאסן פון dispersion ווי variance און standard deviation.
- דערנאך: די יסודות פון probability, און normal distribution, correlation און covariance.
- העכערע באגריפן: central limit theorem, hypothesis testing, P values, confidence intervals און אזוי ווייטער.
- פאראלעל: linear algebra: vectors, matrices, eigenvalues און eigenvectors. אין calculus: די יסודות פון integral און differential calculus.

דער מאטעמאטישער יסוד איז נויטיג צו אנטוויקלען און פארשטיין AI מאדעלן, און ער גרייט צו פארן אריבערגיין צו מאשין-לערנען.

חודש פינף: EDA און מאשין-לערנען (1:38:23)

אינעם פינפטן חודש קומט exploratory data analysis (EDA) און מאשין-לערנען:

- pre-processing: באהאנדלען פעלנדיגע (null) ווערטן, באהאנדלען outliers, data normalization , און feature engineering. אויך encoding טעכניקן, ווי one-hot און label encoding.
- supervised און unsupervised learning, מיט א פאקוס אויף regression און classification. לינעארע מאדעלן ווי linear און logistic regression, און נישט-לינעארע מאדעלן ווי decision tree און random forest.
- אפשאצן מאדעלן מיט metrics ווי mean squared error, mean absolute error , און פאר classification: accuracy, precision, recall א.א.וו.
- hyperparameter tuning: טעכניקן ווי GridSearchCV און RandomizedSearchCV צו אפטימירן די מאדעלן.
- unsupervised learning: clustering טעכניקן ווי K-means און hierarchical clustering , און dimensionality reduction מיט PCA.

דער פאקוס אויף EDA און בויען מאדעלן גרייט צו פאר קאמפליצירטערע AI אפליקאציעס, און דערנאך הייבט מען אן צו ארבעטן אויף deploy'ען די מאדעלן אין עכטע סיטואציעס.

חודש זעקס און זיבן: MLOps און דיפ-לערנען (1:39:30)

אין זעקסטן חודש קומט MLOps און מאשין-לערנען פראיעקטן, די אפעראציאנעלע זייט פון מאשין-לערנען, צוזאמען מיט פראקטישע פראיעקטן:

- MLOps יסודות: לערנען ווען APIs, באזונדערס מיט FastAPI פאר Python און סערווער-אנטוויקלונג.
- DevOps יסודות, אריינגערעכנט CI/CD pipelines, און containerization מיט Docker און Kubernetes.
- באקאנט ווערן מיט כאטש איין cloud פלאטפארמע, ווי AWS אדער Azure.

אין זיבעטן חודש גייט מען אריין אין דיפ-לערנען:

- נויראל-נעצן: forward און backward propagation, און בויען multi-layer perceptrons.
- העכערע ארכיטעקטורן: convolutional נויראל-נעצן (CNNs) פאר בילדער-דאטא, און sequence מאדעלן ווי RNNs און LSTMs.

די וויסן ווערט זיין וויכטיג פארן גיין אין ספעציאליזירטע געביטן פון AI אינעם לעצטן חודש.

חודש אכט: NLP אדער computer vision, און צוזאמפאסונג (1:40:23)

אין אכטן און לעצטן חודש קען מען זיך ספעציאליזירן אין natural language processing (NLP) אדער computer vision:

- NLP ווען: regex, מעטאדן פון טעקסט-פארשטעלונג ווי CountVectorizer, TF-IDF, bag of words, Word2Vec, embeddings, און טעקסט-קלאסיפיקאציע מיט Naive Bayes. באקאנט ווערן מיט ביבליאטעקן ווי spaCy און NLTK, און ארבעטן אויף אן end-to-end NLP פראיעקט.

- computer vision וועג: פשוטע טעכניקן פון בילדער-פראצעסירן, ווי filtering, edge detection, image scaling און rotation; ניצן ביבליאטעקן ווי OpenCV, און בויען אויף די CNN וויסן פונעם פריערדיגן חודש; פראקטיצירן data processing און augmentation.

ביזן סוף פונעם חודש זאל מען האבן א פעסטן יסוד אין דער אויסגעקליבענער ספעציאליזאציע, גרייט אנצוהייבן א קאריערע ווי אן AI אינזשעניר. צום שלוס זאגט דער קורס אז AI אננעמען איז נישט נאר א מאדע, נאר א סטראטעגישער שריט וואס קען טוישן ווי אן ארגאניזאציע צוגייט צו מאשין-לערנען. (עס ווערט אויך קורץ רעקלאמירט א סערטיפיקאט-פראגראם פון Simplilearn אין AI און מאשין-לערנען, מיט באדינגונגען ווי א bachelor's degree מיט 50% דורכשניט, בעסער צוויי יאר ארבעטס-ערפארונג, און יסוד-וויסן אין פראגראמירן און מאטעמאטיק.)

אנהייב "Machine Learning Tutorial Part 1": פארוואס מאשין-לערנען (1:41:52)

דא הייבט זיך אן א נייער טייל: דער ערשטער חלק פון א מאשין-לערנען סעריע. דער פלאן: א קורצע ערקלערונג פארוואס מאשין-לערנען און וואס עס איז; די סארטן מאשין-לערנען; אלגאריטמען ווי linear regression, decision trees און support vector machine; און צום סוף א use case: מען וועט קלאסיפיצירן צי א רעצעפט איז פון א cupcake אדער א muffin, מיט SVM (support vector machine).

פארוואס זאל מען זיך אינטערעסירן? ערשטנס, ווייל מאשינען קענען שוין היינט פירן דעם קאר. דאס איז נאך אין די ערשטע שטאפלען, אבער עס וואקסט זייער שנעל, ווי מען זעט ביי Google'ס Waymo, און Uber האט געהאט א פראגראם וואס איז ליידער געקראכט. מען ווייסט אז דאס וועט זיין א ריזיגע אינדוסטריע וואס וועט טוישן די גאנצע טראנספארטאציע-אינפראסטרוקטור.

צווייטנס, מאשין-לערנען ווערט שוין געוצט צו דערקענען איבער 50 אויגן-קרענק. עס ווערט אונטערגעשטראכן ווי וואונדערליך עס איז צו האבן א קאמפיוטער וואס קאנטראלירט נאך פאר דעם דאקטער זאכן וואס ער קען פארפעלן, א ריזיגע זאך אין געזונט-אינדוסטריע. אין געוויסע ערטער ניצט מען שוין, אפשר נישט פאר אויגן נאר פאר אנדערע קרענק, די קאמערע פונעם טעלעפאן כדי צו מאכן א פאר-דיאגנאז איידער מען גייט צום דאקטער.

דריטנס, א מאשין קען שוין אויפשליסן דעם טעלעפאן מיטן פנים. אויך קען זי דערקענען דעם פנים אדער די שטימע און אנצינדן אדער אויסלעשן זאכן לויט וואו מען איז און וואס מען דארף; א שטיק אויטאמאציע אין דער וועלט וואו מיר לעבן.

ביישפיל: Facebook קעגן "engagement bait" (1:43:34)

א ביישפיל פון Facebook: מען זעט א פאסט וואס בעט "comment yes" אדער לייגט אריין א "order here". קיינער האט נישט ליב spam פאסטס וואס טשעפען אים צו טון likes, shares, comments און אנדערע אקציעס. די ערשטע אזעלכע פאסטס זענען געווען אין סגנון פון "אויב דו קליקסט נישט, וועסטו האבן שלעכט מזל", א מין שרעק-פאקטאר. דאס איז א גרויסע פראבלעם אין סאשעל מידיע.

די טאקטיק הייסט engagement bait, און זי ניצט אויס דעם newsfeed אלגאריטם פון Facebook: מען יאגט זיך נאך engagement כדי צו דערגרייכן מער מענטשן. כדי עס אפצושאפן, האט די פירמע איבערגעקוקט און קאטעגאריזירט הונדערטער טויזנטער פאסטס, כדי צו טריינירן א מאשין-לערנען מאדעל וואס דערקענט פארשידענע סארטן engagement bait. דאס איז נישט נאר ביי Facebook; אלע סאשעל מידיע בויען אזעלכע געצייג.

מען זעט אויפן בילד ווי דער פאסט נעמט איבער א טייל פון דער Facebook סטרוקטור און ווערט ערזעצט, ווי א ווירוס וואס קומט אריין. עס זענען דא קאטעגאריעס ווי vote baiting, react baiting, share baiting; דאס זענען אלגעמיינע נעמען, אבער עס זענען דא אסאך וועגן צו צוציען דעם באניצער צו קליקן. די גאנצע דאטא איז אריינגעפיטערט געווארן אין דער מאשין, און וועט א נייער פאסט קומט, וואס נעמט איבער א טייל פון דער Facebook סטרוקטור, ווערט ער דערקענט.

פון ספאם-פילטערס ביז ליניארע רעגרעסיע (1:45:00)

ווי Facebook פילטערט ספאם, און AlphaGo (1:45:00)

דער קורס ענדיגט דעם ביישפיל פון Facebook און ספאם. כדי אויסצורייניגן דעם ספאם, האט Facebook אנגעהויבן אויסצוזוכן געוויסע ווערטער און פראזעס וואס קומען פאר אין ספאם-מעלדונגען. דערנאך קוקט מען אויף דעם "click-through rate", הייסט ווי אפט מען דריקט אויף אזא לינק. מען זוכט מענטשן וואס דריקן דורך אן צו קוקן וואס עס איז, אדער אויף זאכן וואס נארמאל וואלט מען נישט געדריקט.

וועו Facebook האט שוין דורכגעזוכט די ווערטער און פראזעס, קען עס דערקענען דעם ספאם וואס קומט אריין. דאס מאכט דעם באניצער'ס לעבן גרינגער. עס ווערט פארגליכן מיט ארומגיין אין אן עראפארט אין אסאך לענדער, וואו הונדערטער מענטשן פרובירן דיר צו פארקויפן "time share" און רופן "קומט זיך אנשליסן, שרייבט זיך איין". דער פילטער באגרעניצט די צרה, און מען קען ענדליך הנאה האבן פון Facebook און פון די בילדער פון דער פאמיליע.

א צווייטער גוטער ביישפיל איז DeepMind פון Google'ס AlphaGo. דאס איז א קאמפיוטער-פראגראם וואס שפילט דאס ברעט-שפיל Go , און עס האט באזיגט דעם וועלט'ס נומער-איינס Go-שפילער, Ke Jie. דער לעצטער מאטש, דריי פון דריי, איז פארגעקומען דעם 27טן מאי 2017. אין דער לעקציע ווערט עס דערמאנט ווי "ערשט פארגאנגענעם יאר".

פארוואס איז דאס אזוי וויכטיג? Go איז נאר א שפיל, נישט ווי פירן א קאר אין דער אמת'ער וועלט. אבער מען ניצט שפילן כדי צו לערנען ווי אזוי א מאשין-לערנען פראגראם זאל לערנען. מען וויל אז עס זאל לערנען "ווי אזוי צו לערנען", און דאס איז א ריזיגער שריט.

עס ווערט באטאנט אז אסאך פון דעם איז נאך אין זיין קינדהייט-שטאפל. מען זעט עס ביי די Uber קארס וואס זענען פריער דערמאנט געווארן: Uber האט פארלוירן זייער גאנצע אפטיילונג, ווייל זיי זענען צו שנעל פאראויסגעשפרונגען. פונדעסטוועגן איז דאס ערשט דער אנהייב פון א וואונדערליכער וועלט, אויטאמאטיזירט אויף אופנים וואס מען קען זיך היינט נישט פארשטעלן, ווי אזוי מארגן וועט אויסזען.

וואס איז מאשין-לערנען? א קלארע דעפיניציע (1:46:41)

נאך אזויפיל ביישפילן גיט דער קורס א מער קאנקרעטע דעפיניציע: מאשין-לערנען איז די וויסנשאפט פון מאכן קאמפיוטערס לערנען און זיך פירן ווי מענטשן, דורך זיי צו געבן דאטא און אינפארמאציע, אן דעם וואס מען זאל זיי בפירוש פראגראמירן.

אויף א דיאגראם זעט מען א געוועלטיגע סיסטעם, א היינטיגער קאמפיוטער. היינט קען מען אסאך פון די זאכן אפילו לויפן אויף א טעלעפאן, ווייל טעלעפאנען זענען שטארק פארגעשריטן. מיט AI און מאשין-לערנען נעמט די סיסטעם די דאטא, לערנט פון דעם וואס איז פריער געשען, און דערנאך זאגט עס פאראויס וואס וועט קומען.

דער גרעסטער טייל וואס גייט היינט פאר אין מאשין-לערנען איז אז עס פארבעסערט זיך אויף דעם, און פרעגט: ווי אזוי געפינט מען א נייע לייזונג? מען גייט אזוי אריבער פון "דעסקריפטיוו" (לערנען און פארשטיין ווי זאכן פאסן צוזאמען), צו "פרעדיקטיוו" (פאראויסזאגן וואס עס וועט טון), ביז "פרעסקריפטיוו" (אויסטראכטן א נייע לייזונג).

די שריט פון א מאשין-לערנען פראיעקט (1:47:35)

פארשידענע מענטשן האבן ארויסגעגעבן פארשידענע דיאגראמען פאר די שריט פון א מאשין-לערנען פראיעקט. אסאך פון זיי זענען זייער ספעציפיש צו דעם געביט: ביי בילדער-אידענטיפיקאציע, שפראך, מעדיצין אדער פיזיק ווערן געווויסע שריט איבערגעביטן אדער ס'קומען צו נייע. דער קורס ווייזט א גאנץ אלגעמיינעם דיאגראם:

- ערשטנס, באשטימען דעם ציל. עס איז זייער וויכטיג צו וויסן פונקט וואס מען וויל פאראויסזאגן.
- צווייטנס, זאמלען די דאטא וואס פאסט צום ציל. אין דאטא-וויסנשאפט פארברענגט מען אסאך צייט מיט זאמלען דאטא.
- דריטנס, צוגרייטן די דאטא, און זיכער מאכן אז זי איז ריין. עס ווערט געבראכט דער אלטער ווערטל: "שלעכטע דאטא אריין, שלעכטער ענטפער ארויס".
- פערטנס, אויסקלייבן דעם אלגאריטם וואס מען וועט ניצן.
- פינפטנס, טריינירן דעם אלגאריטם. אין דעם קורס וועט מען ארבעטן מיט SVM, דער "support vector machine".
- זעקסטנס, טעסטן דעם מאדעל: ארבעט ער? איז עס א גילטיגער מאדעל פאר דעם וואס מען טוט?
- זיבעטנס, לויפן די פרעדיקציע, אדער די ברירה, אדער וועלכן רעזולטאט עס זאל נאר ארויסגעבן.
- אכטנס, ווען אלעס איז גרייט און מען האט שוין אסאך געטעסט, "deploy"-ן דעם מאדעל, הייסט אים ארויסשטעלן צום באנוץ.

עס ווערט ווידער באטאנט אז דאס איז זייער אלגעמיין. ביי אסאך מאדעלן קומט מען אן אויפן האלבן וועג און מען זעט אז עס פעלט עפעס אין דער דאטא. מען דארף גיין זאמלען נייע דאטא, ווייל א טעסט ערגעץ אויפן וועג וויזט: "איך באקום נישט די ענטפערס וואס איך דארף". אסאך געביט-ספעציפישע זאכן ווערן אזוי א טייל פונעם פראצעס. דאך איז דאס א זייער גוטער מאדעל מיט וועלכן אנצוהויבן.

די הויפט-טיילן: קלאסיפיקאציע, רעגרעסיע, אנאמאליע, קלאסטערינג (1:49:19)

עס זענען פאראן געווויסע גרונט-אפטיילונגען פון וואס מאשין-לערנען טוט, וואס מען דארף קענען:

- ווילסטו פאראויסזאגן א קאטעגאריע? דאס הייסט קלאסיפיקאציע (classification). א ביישפיל: צי דער פרייז פון א סטאק וועט ארויפגיין אדער אראפגיין. מען זוכט א "יא-ניין" ענטפער: גייט עס ארויף, איז דאס "true"; גייט עס נישט ארויף, איז דאס "false", הייסט עס גייט אראפ. אזוי ווערט עס א יא-ניין, נול-איינס.
- ווילסטו פאראויסזאגן א כמות? דאס הייסט רעגרעסיע (regression). דאס זענען די צוויי הויפט-אפטיילונגען. א ביישפיל: אפשאצן דעם עלטער פון א מענטש לויט זיין הייך, וואג, געזונט און אנדערע פאקטארן.
- ווילסטו ענטדעקן אן אנאמאליע? דאס הייסט "anomaly detection", און עס איז היינט זייער פאפולער. א ביישפיל: דערקענען אנאמאליעס ביי געלט-ארויסנעמונגען, כדי צו וויסן ווען איינער נעמט ארויס געלט פון

א חשבון וואס איז אפשר נישט זיינער. ביים סטאק-מארקעט: אויב מען זאגט פאראויס צי צו קויפן א סטאק, וויל מען וויסן צי דאס וואס גייט פאר אין מארקעט איז אן אנאמאליע. אויב יא, דארף מען א אנדערן פרעדיקציע-מאדעל, ווייל עפעס אנדערש גייט פאר און מען מוז ארויסציען נייע אינפארמאציע. אדער איז דאס נאר דער נארמאלער גאנג, און מען וועט באקומען דעם געווענליכן רווח אויף דעם אינוועסטירטן געלט? אנאמאליעס דערקענען איז היינט זייער גרויס אין דאטא-וויסנשאפט.

- א פראגע וואס פאלט אריין אונטער "אומטריינירטע דאטא": ווילסטו ענטדעקן א סטרוקטור אין אום-אויסגעפארשטע דאטא? דאס הייסט קלאסטערינג (clustering). א ביישפיל: געפינען גרופעס קונים מיט ענליכע אויפפירונג, פון א גרויסער דאטאבאזע מיט קונים'ס דעמאגראפיע און זייערע פריערדיגע איינקויפן. מען קען אפשר באמערקן אז ווער עס טראגט א געוויסן סארט שיך, אדער גייט איינקויפן אין געוויסע געשעפטן, וועט מאכן געוויסע איינקויפן. די אינפארמאציע העלפט צו מארקעטירן, אדער צו גרופירן מענטשן כדי אויספארשן די גרופע און געפינען וואס מען וויל זיי פארקויפן.

דאס ארבעט אין כמעט יעדן פעלד. למשל, א באנק וויל אפשר גרופירן מענטשן לויט זייערע געגנטער, אינוועסטמענטס און פינאנציעלן הינטערגרונט, צי זיי וועלן באקומען א הלוואה אדער נישט, נאך איידער מען הייבט אן אונטערזוכן צי זיי זענען גילטיגע קונים פאר דער באנק. מען וויסט נישט וואס די דאטא וועט זאגן, נאר מען וויל צונויפנעמען מענטשן וואס געהערן צוזאמען.

א קוויז (1:51:49)

דער קורס מאכט א קורצע אפשטעל פאר א קוויז. די ענטפערס וועלן ארויסגעגעבן ווערן אין טייל צוויי פונעם לערנונג. דער לייענער ווערט געבעטן צו טראכטן וואס עס גייט פאר אין יעדן פון די פאלגנדע פעלער, ווי איינער וואס זיצט מיט א טאס קאווע און א פען און טראכט אויס זיין קומענדיגן שריט אין דער דאטא-אנאליז:

- א) גרופירן דאקומענטן אין פארשידענע קאטעגאריעס לויט דעם טעמע און אינהאלט פון יעדן דאקומענט. דאס איז היינט זייער גרויס: לעגאלע דאקומענטן, ספארט-דאקומענטן, אנאליז פון צייטונג-ארטיקלען. דאס אויטאמאטיזירן איז א ריזיגע זאך היינט.
- ב) דערקענען ריכטיג האנטגעשריבענע ציפערן אין בילדער: וויסן וואס א מענטש שרייבט מיט זיין האנטשריפט, א קליין אדער גרויס A, B, C.
- ג) די אויפפירונג פון א וועבזייטל וואס וויזיט אן אז דער זייטל ארבעט נישט ווי עס איז געפלאנט.
- ד) פאראויסזאגן די געהאלט פון א מענטש לויט זיינע יארן ערפארונג, אזוי ווי ביי HR דינגען.

מען קען ענטפערן ביים טייל צוויי, דורך שרייבן אונטן און שיקן א מעלדונג צו Simplilearn, און זיי וועלן זיך מעלדן.

סופערווייזד, אנסופערווייזד און רעינפארסמענט-לערנען (1:53:21)

די לעצטע ביישפילן פירן צו א נייע טעמע: נאך א וועג אפצוטיילן די סארטן מאשין-לערנען: סופערווייזד (supervised), אנסופערווייזד (unsupervised) און רעינפארסמענט (reinforcement) לערנען.

סופערווייזד לערנען איז א מעטאד וואס מאכט מעגליך פאר מאשינען צו קלאסיפיצירן און פאראויסזאגן אביעקטן, פראבלעמען אדער מצבים לויט באצייכנטע (labeled) דאטא וואס מען גיט דער מאשין. אויפן בילד זעט מען א געמיש פון דאטא: קרייזן, דרייעקן און פירעקן. מען באצייכנט זיי: וואס איז א קרייז, וואס איז א דרייעק, וואס איז א פירעק. דער מאדעל טריינירט זיך דערויף. דאס וויכטיגסטע ביי סופערווייזד לערנען: מען וויסט שוין דעם ענטפער פון א גרויסן טייל פון דער אינפארמאציע וואס קומט אריין.

נאכן טרייניגן ווייסט דער מאדעל דעם חילוק צווישן א קרייז, א פירעק און א דרייעק. וועז מען שיקט אריין נייע דאטא, למשל א פירעק און א קרייז, זאגט ער פאראויס אז די אויבערשטע זענען פירעקן און די קומענדיגע זענען קרייזן. אזוי קען מען פאראויסזאגן צי איינער וועט נישט קענען צוריקצאלן א הלוואה, לויט דעם וואס איז פריער געדעדט געווארן וועגן באנקען. מען קען דאס אויך ניצן ביים סטאק-מארקעט, צי מען וועט פארדינען אדער נישט.

דא קומט א ווארנונג: וועד עס וויל מאכן א פארמעגן אויפן סטאק-מארקעט, זאל געדענקען אז עס איז זייער שווער צו באקומען אלע דאטא ריכטיג. דער מארקעט שוואנקט אויף אופנים וואס זענען זייער שווער פאראויסצוזאגן. עס איז ווי א "roller coaster" רייד, און מען כאפט זיך אז מען דארף טאקע גראבן נאך נייע דאטא.

אנסופערווייזד לערנען: אזא מאדעל געפינט דעם באהאלטענעם מוסטער אין אומבאצייכנטער דאטא. אנשטאט אים צו זאגן וואס איז א קרייז, א דרייעק אדער א פירעק, קוקט ער זיך אליין צו און גרופירט זיי צוזאמען. אפשר גרופירט ער לויט די צאל עקן: ער באמערקט אז א טייל האבן דריי עקן, א טייל פיר עקן, און א טייל קיין עקן נישט, און ער פילטערט און גרופירט זיי אזוי.

דאס איז ענליך צום פריערדיגן ביישפיל פון א גרופע מענטשן וואס גייען איינקויפן: מען וויל זיי גרופירן כדי צו געפינען וואס זיי האבן געמיינזאם. נעמט מען אן אז פינף פון זיי זענען קונים אין דיין געשעפט, און זיי האבן אסאך געמיינזאם מיט פינף אנדערע וואס זענען נאך נישט קיין קונים. ווי אזוי מארקעטירט מען צו יענע פינף? זיי פאסן אריין אין דער דעמאגראפיע פון די וואס קויפן ביי דיר, און דו ווילסט אז זיי זאלן קויפן ביי דיר און נישט ביים געשעפט דערנעבן. דאס איז פארשטייט זיך א פארפשוטערטע וועדסיע: דעם חילוק צווישן א דרייעק און א קרייז זעט מען גרינג, אבער ביים מארקעטינג איז עס אפשר נישט אזוי גרינג.

רעינפארסמענט לערנען איז א וויכטיגער סארט מאשין-לערנען, וואו אן אגענט לערנט ווי אזוי זיך צו פירן אין א סביבה, דורך טון אקציעס און זען די רעזולטאטן. אויפן בילד זעט מען א קליין קינד. עס פאסט, ווייל רעינפארסמענט לערנען איז אויך נאך אין זיין קינדהייט-שטאפל, אבער מסתמא איז דאס דער גרעסטער באדארף אין מאשין-לערנען היינט און אין די קומענדיגע יארן: ווי אזוי מען זאל דאס מאכן ארבעטן פאר אונז.

די אקציע: דאס קינד גייט אריין אין פייער (מען האפט אז עס איז נאר א קליין ליכט און נישט א ריזיגע פייער-גרוב ווי אויפן בילד). דער נייער מצב: דאס קינד איז טרויעריג און ווייניט, ווייל עס האט זיך פארברענט. דאס קינד הייסט דער "אגענט", ווייל עס טוט די אקציעס. דערנאך טוט עס אן אנדערע אקציע, גייט אין אן אנדער ריכטונג און נישט צום פייער, און יעצט איז עס פרייליך, לאכט און שפילט.

רעינפארסמענט לערנען איז גרינג צו פארשטיין, ווייל אזוי לערנען מענטשן אויך: מען פארברענט זיך אויפן אויוון און מען רירט אים מער נישט אן. וועז א מאשין-לערנען פראגראם אדער AI קען דאס טון, איז דאס ריזיג, ווייל עס הייבט אן לערנען ווי אזוי צו לערנען. דאס איז א גרויסער שפרונג אין דער וועלט פון קאמפיוטערס און מאשין-לערנען.

איבערחזרן: סופערווייזד קעגן אנסופערווייזד (1:57:13)

דער קורס חוז'ט איבער דעם חילוק צווישן סופערווייזד און אנסופערווייזד לערנען, ווייל דאס וועט ארויסקומען אין יעדן פראיעקט.

ביי סופערווייזד לערנען איז די דאטא באצייכנט, און עס איז פאראן דירעקטער פידבעק: איינער האט שוין געזאגט "יא, דאס איז א דרייעק" אדער "ניין, דאס איז נישט קיין דרייעק". דערנאך זאגט מען פאראויס א רעזולטאט: וועלכן א נייע דאטא קומט אריין, ווייסט מען וואס עס וועט זיין.

ביי אנסופערווייזד לערנען איז די דאטא נישט באצייכנט, מען ווייסט נישט וואס עס איז, און עס איז נישטא קיין פידבעק. מען זאגט נישט צי עס איז ריכטיג אדער פאלש, צי עס איז א דרייעק אדער א פירעק, צי מען זאל גיין רעכטס אדער לינקס. מען געפינט נאר א באהאלטענע סטרוקטור אין דער דאטא, און מען גרופירט זי כדי צו זען וואס איז פארבונדן מיט וואס.

מען קען די צוויי אויך ניצן צוזאמען. למשל, מען האט א בילד און מען ווייסט נישט וואס מען זוכט. מען נעמט די אומסטרוקטורירטע דאטא און געפינט אלע זאכן וואס זענען פארבונדן צוזאמען. דערנאך קוקט א מענטש אויף זיי און באצייכנט זיי. יעצט קען מען נעמען די באצייכנטע דאטא און פראגראמירן עפעס וואס זאל פאראויסזאגן וואס איז אין בילד. אזוי גייט מען הין און צוריק, און מען פארבינדט פארשידענע געציג צוזאמען צו א גרעסערן בילד.

אלגאריטמען: ליניארע רעגרעסיע (1:58:29)

עס זענען פאראן אסאך אינטערעסאנטע מאשין-לערנען אלגאריטמען. דער קורס גיט א טעם פון עטליכע פון די וויכטיגסטע וואס ווערן היינט גענוצט: ליניארע רעגרעסיע (linear regression), "decision tree" (באשלוס-בוים), און דער "support vector machine".

מען הייבט אן מיט ליניארע רעגרעסיע. דאס איז אפשר איינער פון די מערסט באוואוסטע און בעסט פארשטאנענע אלגאריטמען אין סטאטיסטיק און מאשין-לערנען. ליניארע רעגרעסיע איז א ליניארער מאדעל, הייסט א מאדעל וואס נעמט אן אז עס איז פאראן א ליניארע באציאונג צווישן די אריינגאנג-ווערייבלס x און דעם איינציגן ארויסגאנג-ווערייבל y. מען קען זיך דערמאנען פון די אלגעברע קלאסן: $y = mx + c$.

א ביישפיל: מען וויל פאראויסזאגן די דיסטאנץ וואס מען פארט (y) לויט דער שנעלקייט (x). די ליניארע רעגרעסיע מאדעל-פארשטעלונג פאר דעם פראבלעם וואלט געווען $y = m \times$ ..., און דער אויסטייטש גייט ווייטער אין דעם קומענדיגן טייל.

פון 2:00:00 ביזן סוף: וואס דער עקראן ווייזט (2:00:00)

פארוואס פון דא ווייטער נאר דאס בילד

אויף YouTube איז דער קול פון דעם ווידעא שטיל פון בערך 2:00:00 ביזן סוף פונעם ווידעא, 11:05:03. אין יענעם טייל הערט מען גארנישט, און דערפאר זענען די געדעטע וועדטער פון די לעצטע ארום 9 שעה נישטא.

דאס בילד גייט אבער ווייטער: סליידס, Jupyter נאטבוקס מיט קאוד און רעזולטאטן, גראפן און דיאגראמען. דער רעשט פון דעם טייל באשרייבט דערפאר, לויט דער צייט, וואס דער עקראן ווייזט, לויט בילדער פון יעדע האלבע מינוט. וואס מען האט דערביי געדעטע, ווייסט מען נישט, און עס ווערט נישט צוגעטראכט.

לינעארע רעגרעסיע און באשלוס-ביימער (לויטן עקראן) (2:00:00)

צוויי ראוועריפארער: גיכקייט, מהלך און צייט (2:00:00)

אויפן עקראן ווערט געוויזן א בילד פון צוויי וועגן, און אויף יעדן וועג פארט א ראוועריפארער. אויבן שטייט די אויפשריפט "Time is constant", ד.ה. די צייט בלייבט די זעלבע.

- אויפן ערשטן וועג: Speed = 10m/s, און דער מהלך איז Distance = 36 km.
- אויפן צווייטן וועג: Speed = 20m/s, און דער מהלך איז Distance = 72 km.

Linear Regression: א positive באציאונג (2:00:30)

דער סלייד מיטן טיטל "Linear Regression" ווייזט א גראפיק. אויף דער האריזאנטאלער אקס שטייט Speed, און אויף דער ווערטיקאלער אקס Distance. א גרינע געשטריכלטע ליניע גייט ארויף.

- m = +ve slope of the line, ד.ה. m איז דער positive שיפוע פון דער ליניע.
- c = y-intercept of line, ד.ה. c איז דער פונקט וואו די ליניע שניידט די y־אקס.
- די פארמל איז y = mx + c. דערביי איז y "Distance travelled in fixed interval of time" (דער מהלך אין א פעסטער צייט), און x איז "Speed of the person".

אונטן שטייט: "As the speed increases, distance also increases, hence the variables have a positive relationship". דאס הייסט: ווען די גיכקייט וואקסט, וואקסט אויך דער מהלך, און דערפאר האבן די צוויי פאריאבלען א positive באציאונג. דער סלייד בלייבט אויפן עקראן ביז 2:01:00.

ווען דער מהלך איז פעסט (2:01:30)

אויפן עקראן ווערן געוויזן דריי וועגן, אונטער דער אויפשריפט "Distance is constant", ד.ה. דער מהלך בלייבט דער זעלבער.

- Speed = 10m/s, דערביי Time = 100 s.
- Speed = 20m/s, דערביי Time = 50 s.
- Speed = 30m/s, דערביי Time = ?

Linear Regression: א negative באציאונג (2:02:00)

אויבן אויפן סלייד שטייט: "If distance is assumed to be constant, let's see the relationship between speed and time". אויף דער גראפיק איז Speed אויף דער האריזאנטאלער אקס און Time אויף דער ווערטיקאלער אקס. א רויטע געשטריכלטע ליניע גייט אראפ.

- m = -ve slope of the line, ד.ה. דער שיפוע איז negative.
- y איז c + mx = y, "Time taken to travel a fixed distance", און x איז "Speed of the person".

אונטן שטייט: "As the speed increases, time decreases, hence the variables have a negative relationship". דאס הייסט: ווען די גיכקייט וואקסט, ווערט די צייט קלענער, און דערפאר איז די באציאונג negative.

א ביישפיל מיט פינף פונקטן (2:02:30)

אויפן עקראן ווערט געוויזן א טאבעלע מיט x און y:

- x=1, y=3
- x=2, y=2
- x=3, y=2
- x=4, y=4
- x=5, y=3

אונטער דער טאבעלע שטייט Mean(xi) = 3. נעבן דער טאבעלע, אונטער דער אויפשריפט "Let's plot these points!!", זענען די פינף פונקטן אויסגעצייכנט אויף א גראפיק.

ביי 2:03:00 קומט צו Mean(yi) = 2.8. אויף דער גראפיק איז דא א נייער, אראנזש פונקט ביי (3, 2.8), און געשטריכלטע ליניעס פירן פון אים צו ביידע אקסן.

געפינען די בעסטע ליניע (2:03:30)

דער סלייד זאגט: "Now, lets find regression equation to find the best fit line!", און ווייזט די פארמל y = mx + c. ווייטער שטייט: "To find this equation for our data, we need to find our slope (m) and coefficient (c)". דאס הייסט: כדי צו געפינען די גלייכונג פאר אונזער דאטא, דארף מען געפינען דעם שיפוע m און דעם קאעפיציענט c.

דער חשבון פון m (2:04:00)

אויפן עקראן שטייט די פארמל:

$$m = \Sigma(xi - x)(y - xi) / \Sigma(xi - x)^2$$

דערונטער איז א טאבעלע מיט די זעקס קאלומנעס x, y, x–xi, y–x, (xi–yi)², און (yi–xi)(y–x):

- 1, 3, -0.4, 4, 0.2, -2
- 2, 2, 0.8, 1, -0.8, -1
- 3, 2, 0, 0, 0, -0.8
- 4, 4, 1, 1.2, 1, 1.2
- 5, 3, 2, 0.2, 4, 0.4

דער סלייד בלייבט אויפן עקראן ביז 2:04:30.

דער חשבון פון c (2:05:00)

אויפן עקראן שטייט: m = 2/10 = 0.2. דערנאך, אונטער דער אויפשריפט "So, we can calculate the value of c", און מיט Mean values = (3, 2.8), גייט דער חשבון אזוי:

- c + 0.2x = y
- c + 3 * 0.2 = 2.8

- c + 0.6 = 2.8
- 0.6 – 2.8 = c
- 2.2 = c

דער סלייד בלייבט אויפן עקראן ביז 2:05:30.

פאראויסזאגן די ווערטן פון y (2:06:00)

דער סלייד זאגט: "Now, let's predict the values of y using x {1,2,3,4,5}! and plot the points". מיט דער פארמל y = (0.2*x) + 2.2 ווערן אויסגערעכנט די ווערטן:

- yp = (1*0.2) + 2.2 = 2.4
- yp = (2*0.2) + 2.2 = 2.6
- yp = (3*0.2) + 2.2 = 2.8
- yp = (4*0.2) + 2.2 = 3.0
- yp = (5*0.2) + 2.2 = 3.2

דערביי באדייט yp "Predicted values of y", ד.ה. די פאראויסגעזאגטע ווערטן פון y.

דער אונטערשייד צווישן פאראויסזאג און וירקלעכקייט (2:06:30)

דער סלייד זאגט: "Plot the predicted values along with the actual values to see the difference". דאס הייסט: מען צייכנט די פאראויסגעזאגטע ווערטן צוזאמען מיט די אמתע ווערטן, כדי צו זען דעם אונטערשייד.

אין דער טאבעלע שטייען x, y און yp. אויף דער גראפיק זענען די אמתע פונקטן בלויע, און די פאראויסגעזאגטע פונקטן רויטע, אויף א געלער ליניע. דער מהלך צווישן יעדן בלויען פונקט און דער ליניע איז אנגעצייכנט מיטן ווארט "Error".

מינימיזירן דעם מהלך (2:07:00)

אויפן סלייד שטייט: "Minimizing the Distance: There are lots of ways to minimize the distance between the line and the data points like Sum of Squared errors, Sum of Absolute errors, Root Mean Square error etc.". דאס הייסט: עס זענען דא אסאך ווענן צו מינימיזירן דעם מהלך צווישן דער ליניע און די דאטא־פונקטן.

אויף דער גראפיק זענען דא ניַין גרינע פונקטן, d1 ביז d9, און יעדער איז פארבונדן מיט א געשטריכלטער ליניע צו א רויטער ליניע. די פארמל איז:

$$D = {}^2d1 + {}^2d2 + \ldots + {}^2d9$$

אונטן שטייט: "We keep moving this line through the data points to make sure the best fit line has the least square distance between the data points and the regression line". דאס הייסט: מען רוקט די ליניע ווייטער צווישן די פונקטן, ביז די בעסטע ליניע האט דעם קלענסטן קוואדראטישן מהלך פון די דאטא־פונקטן.

ביי 2:07:30 ווערט די ליניע געוויזן אויף א שטיצערן שיפוע, און די פארמל ווערט $D = d1^2 + d2^2 + d3^2 + \ldots$

Decision Trees: די גאלף־טאבעלע (2:08:00)

אונטער דעם נייעם טיטל "Decision Trees" ווערט געוויזן א טאבעלע אין OpenOffice Calc, אין א פייל מיטן נאמען "Decision Tree- golf". די קאלומנעס זענען Outlook, Temp, Humidity, Windy, און Play Golf. עס זענען דא 14 שורות:

- Rainy, Hot, High, FALSE, No
- Rainy, Hot, High, TRUE, No
- Overcast, Hot, High, FALSE, Yes
- Sunny, Mild, High, FALSE, Yes
- Sunny, Cool, Normal, FALSE, Yes
- Sunny, Cool, Normal, TRUE, No
- Overcast, Cool, Normal, TRUE, Yes
- Rainy, Mild, High, FALSE, No
- Rainy, Cool, Normal, FALSE, Yes
- Sunny, Mild, Normal, FALSE, Yes
- Rainy, Mild, Normal, TRUE, Yes
- Overcast, Mild, High, TRUE, Yes
- Overcast, Hot, Normal, FALSE, Yes
- Sunny, Mild, High, TRUE, No

ביי 2:08:30 שטעלט דער סלייד די פראגע: "Let's determine if you should play golf when the day is sunny and windy?". דאס הייסט: לאמיר אויסגעפינען צי מען זאל שפילן גאלף ווען דער טאג איז זוניק און ווינטיק.

א פרובירטער בוים (2:09:00)

אויפן סלייד שטייט: "Suppose, we draw our tree like this!", און דערונטער א בוים:

- דער אויבערשטער קנופ איז Humidity.
- ביי Normal: Play.
- ביי High גייט מען צום קנופ Outlook:
- Sunny: Don't Play
- Overcast: Play
- Rainy: Don't Play

דער סלייד בלייבט אויפן עקראן ביז 2:10:00.

Entropy און Information Gain (2:10:30)

דער סלייד פרעגט: "But, is this the right decision tree? For that, we should calculate Entropy and Information Gain!". דאס הייסט: איז דאס טאקע דער ריכטיקער בוים? כדי דאס צו ווייסן, דארף מען אויסרעכענען Entropy און Information Gain.

- Entropy: "Entropy or the measure of randomness or 'impurity' in the dataset", ד.ה. א מאס פון צופעליקייט אדער אומריינקייט אינעם דאטא־סעט. "Entropy should be low!", ד.ה. Entropy דארף זיין נידעריג.
- Information Gain: "Information Gain or the measure of decrease in entropy after the dataset is split", ד.ה. א מאס ווי־פיל די Entropy וועט קלענער נאכדעם וואס מען צוטיילט דעם דאטא־סעט. עס וועט אויך גערופן Entropy Reduction. "Information Gain should be high!", ד.ה. Information Gain דארף זיין הויך.

די פארמל פון Entropy (2:11:00)

אונטער דער אויפשריפט "Let's look at entropy!" שטייט די פארמל:

Entropy = I(p,n) = −p/(p+n) × log2(p/(p+n)) − n/(p+n) × log2(n/(p+n))

דער סלייד בלייבט אויפן עקראן ביז 2:11:30.

די גאנצע Entropy (2:12:00)

דער סלייד וויזט: "(a) Entropy of target class of the dataset (whole entropy)", ד.ה. די Entropy פון דער צילקלאס פונעם גאנצן דאטא־סעט. א קליינע טאבעלע וויזט Play Golf: Yes = 9, No = 5, און Total = 14. דער חשבון:

- Entropy(Play golf) = E(5,9)
- = I(5/14, 9/14)
- = I(0.36, 0.64)
- = −(0.36 log2 0.36) − (0.64 log2 0.64)
- = 0.94

דער סלייד בלייבט אויפן עקראן ביז 2:12:30.

Outlook לויט Entropy (2:13:00)

אויפן סלייד שטייט א טאבעלע פון Predictors, מיט Play Golf צוטיילט לויט Outlook:

- Sunny: Yes 3, No 2, Total 5
- Overcast: Yes 4, No 0, Total 4
- Rainy: Yes 2, No 3, Total 5
- אינגאנצן: 14

דער חשבון:

- Entropy(Play golf, Outlook) = P(rainy)*E(2,3) + P(Overcast)*E(4,0) + P(sunny)*E(3,2)
- = 5/14*I(2,3) + 4/14*I(4,0) + 5/14*I(3,2)

- 0.693 =

אונטן שטייט: "Similarly, we can calculate the entropy of other predictors like Temperature, Humidity, Windy!". דאס הייסט: אויפן זעלבן אופן קען מען אויסרעכענען די Entropy פון די אנדערע פרעדיקטארן. דער סלייד בלייבט אויפן עקראן ביז 2:13:30.

Information Gain פון יעדע אייגנשאפט (2:14:00)

אונטער דער אויפשריפט "Now, let's look at Information Gain!" ווערט אויסגערעכנט:

Gain(Outlook) = Entropy(PlayGolf) − Entropy(PlayGolf, Outlook) = 0.940 − 0.693 = 0.247

דערנאך שטייט: "The information gain of the other three attributes can be calculated in the same way", ד.ה. די אנדערע דריי אייגנשאפטן רעכנט מען אויס אויפן זעלבן אופן:

- Gain(Temp) = 0.029
- Gain(Humidity) = 0.152
- Gain(Windy) = 0.048

בויען דעם ריכטיקן בוים (2:14:30)

אונטער דער אויפשריפט "Now, let's build the decision tree!" ווערט געווויזן דער בוים:

- דער Root Node) דער ווארצל־קנופ) איז Outlook.
- ביי Overcast: Play.
- ביי Rainy: Don't Play.
- ביי Sunny גייט מען צום Branch Node) צווייג־קנופ) Windy:
- False: Play
- True: Don't Play

We choose the" :אונטן שטייט .(Leaf Nodes בלאט־קנופן) וויdי ענד־קעסטלעך זענען אנגעצייכנט די attribute with largest information gain as the root node". דאס הייסט: מען קלייבט אויס די אייגנשאפט מיטן גרעסטן Information Gain אלס דעם ווארצל־קנופ.

פון Decision Tree צו SVM: מאפינס קעגן קאפקעיקס (לויטן עקראן) (2:15:00)

דער פארטיגער Decision Tree (2:15:00)

אויפן עקראן ווערט געווויזן א סלייד מיטן טיטל "Decision Trees" און דער אויפשריפט "Now, let's build the decision tree!". עס ווערט געווויזן א פארטיגער בוים:

- אויבן, ביים "Root Node", שטייט "Outlook".
- פון Outlook גייען ארויס דריי צווייגן: "Sunny" פירט צו א קנופ "Windy" (באצייכנט אלס "Branch Node"); "Overcast" פירט גלייך צו "Play"; "Rainy" פירט צו "Don't Play".
- פון Windy גייען ארויס צוויי צווייגן: "False" פירט צו "Play", און "True" פירט צו "Don't Play".

- די ענד-קעסטלעך (Don't Play / Play) זענען באצייכנט אלס "Leaf Nodes".

אונטן שטייט: "We choose the attribute with largest information gain as the root node", דאס הייסט, מען קלויבט פאר דעם root node דעם אטריבוט וואס האט דעם גרעסטן information gain.

דער נעקסטער סלייד ((2:15:30 שטעלט די פראגע: "So, we wanted to know if it's a good day to play golf when it's sunny and windy!". אויפן זעלבן בוים איז מיט א רויטער פונקטירטער ליניע ארומגעצויגן דער וועג Outlook → Sunny → Windy → True → "Don't Play". דער ענטפער לויטן בוים איז אלזא: וועט עס איז זוניג און ווינטיג, שפילט מען נישט. אויפן סלייד זעט מען אויך א בילד פון א גאלף-שפילער.

וואס איז א Support Vector Machine (2:16:00)

אן איבערשריפט "Support Vector Machine" עפנט אן א נייעם טייל. אין א קעסטל שטייט:

- "Support Vector Machine is a widely used classification algorithm!", א בריט-באניצטער קלאסיפיקאציע-אלגאריטם.
- דער געדאנק איז פשוט: דער אלגאריטם שאפט א טיילונג-ליניע וואס צעטיילט די קלאסן אויפן בעסטן מעגלעכן אופן.
- ביישפילן: הונט אדער קאץ, קראנקהייט אדער נישט קיין קראנקהייט.

דער ביישפיל פון הייך און וואג (2:16:30)

א גראף ווייזט "Height" (הייך, 155 ביז 205) אויפן הארוזאנטאלן אקס און "Weight" (וואג, 50 ביז 140) אויפן ווערטיקאלן אקס. פרויען-פיגורן ליגן אויף דער לינקער זייט (נידעריגער און לייכטער), מענער-פיגורן אויף דער רעכטער. עס זענען געצייכנט צוויי קרייצנדיגע ליניעס, "Line 1" און "Line 2", און א נייער פונקט (א גרינע פיגור אין א קרייזל) שטייט אויבן אין מיטן. דער טעקסט זאגט: "And if we consider decision line 2, then it will be a female!", לויט Line 2 וואלט דער נייער פונקט געווען א פרוי.

ביי 2:17:00 ווערט דערקלערט דער ציל: מען דארף אויסקלויבן א hyperplane מיט דעם גרעסטן מעגלעכן margin צווישן דער באשלוס-ליניע און דעם נאענטסטן פונקט אינעם טריינירן-סעט. אויפן גראף ווערט געוויזן Line 1 מיט פארלעלע ליניעס פון ביידע זייטן און דער אפשטאנד "Margin" צווישן זיי. די צוויי נאענטסטע פיגורן (איינע פון יעדע זייט) זענען ארומגעקרייזלט און באצייכנט אלס "Support Vectors". אונטן שטייט די דעפיניציע: "Distance Margin" איז דער אפשטאנד צווישן דעם hyperplane און דעם נאענטסטן דאטא-פונקט פון יעדן סעט.

ביי 2:17:30 קומט דער אויספיר: ווען מען צייכנט די hyperplanes, זעט מען אז Line 1 האט דעם מאקסימאלן distance margin, און דערפאר וועט זי ריכטיג קלאסיפיצירן דעם נייעם דאטא-פונקט. דער רעזולטאט: "New data point is male!", דער נייער פונקט איז א מאן.

די אויפגאבע: מאפינס קענן קאפקעיקס (2:18:00)

דער "Problem Statement" אויפן סלייד: קלאסיפיצירן רעצעפטן פון muffin און cupcake מיט support vector machines. מען זעט א בילד פון א קאפקעיק "VS" א מאפין.

ביי 2:18:30 ווערט געוויזן דער דאטא-סעט ("Let's have a look at our dataset:") אלס א טאבעלע מיט די קאלאמען: Type, Flour, Milk, Sugar, Butter, Egg, Baking Powder, Vanilla, Salt . עס זענען דא צען רייען

Muffin און צען רייען Cupcake. די ערשטע שורה איז Muffin : מעל 55, מילך 28, צוקער 3, פוטער 7, אייער 2 ,baking powder 5, וואַרניל 0, זאלץ 0. ביי די מאפינס גייט די מעל בערך פון 45 ביז 55; ביי די קאפקעיקס בערך פון 34 ביז 42, מיט מער צוקער (16 ביז 31) און פוטער. זאלץ איז אומעטום 0.

ביי 2:19:00 איז צוגעלייגט א וואלקן-קעסטל: וואס איז דער חילוק צווישן א מאפין און א קאפקעיק? עס ווייזט זיך ארויס אז מאפינס האבן מער מעל, בשעת קאפקעיקס האבן מער פוטער און צוקער.

די פעקלעך וואס מען דארף (2:19:30)

א סלייד ווייזט דעם קאוד פון אימפארטן:

- אונטער "Packages for analysis": מען אימפארטירט pandas אלס pd, numpy אלס np , און פון sklearn ווערט אימפארטירט svm.
- אונטער "Packages for visuals": מען אימפארטירט matplotlib.pyplot אלס plt, און seaborn אלס sns, מיט sns.set(font_scale=1.2) צו פארגרעסערן דעם שריפט אין די גראפיקן.
- די שורה matplotlib inline% לאזט די טשארטס זיך באווייזן גלייך אין נאטבוק.
- אונטער "Pickle package": import pickle.

Anaconda און Jupyter (2:20:00)

אויפן עקראן ווערט געוויזן Anaconda Navigator, מיט די אפליקאציעס: Jupyter notebook (5.5.0, מיט א "Launch" קנעפל), glueviz, jupyterlab, orange3, qtconsole, rstudio און spyder) מיט "Install" קנעפלעך). דער אויסגעקלייבענער סביבה הייסט keraspython36.

פון 2:20:30 ביז 2:23:00 ווערט אין א Jupyter נאטבוק (Python 3), וואס הייסט SimplilearnSVM_Muffin_Cupcake.ipynb, שריט ביי שריט אריינגעשריבן די ערשטע צעל:

- א קאמענטאר "# packages for data analysis", און דערנאך import numpy as np און import pandas as pd.
- from sklearn import svm.
- א קאמענטאר "# visual your data", import matplotlib.pyplot as plt, און import seaborn as sns; sns.set(font_scale=1.2).
- matplotlib inline%.

ביי 2:23:30 איז די צעל אויסגעפירט (In [1]) און א נייע ליידיגע צעל עפנט זיך.

אריינלייענען דעם דאטא-סעט (2:24:00)

אויפן עקראן עפנט זיך א פייל-פענצטער מיטן פארצייכענעש פון די פיילס, צווישן זיי "Cupcakes vs Muffins.csv" און "Decision Tree- golf.xlsx". מען עפנט דעם CSV פייל אין א "Text Import" פענצטער (Unicode UTF-8, צעטיילט מיט Comma), וואו מען זעט די זעלבע קאלאמען און ציפערן פונעם דאטא-סעט.

דערנאך (2:24:30 ביז (2:25:00 ווערט אין דער צווייטער צעל געשריבן:

pd.read_csv('Cupcakes vs Muffins.csv') = recipes

דאס לייענט דעם CSV פייל אריין אין א pandas טאבעלע וואס הייסט recipes. ביי 2:25:30 איז צוגעלייגט print(recipes.head()) , וואס דרוקט די ערשטע פינף רייען. דער רעזולטאט ווייזט רייען 0 ביז 4, אלע Muffin, למשל ריי 0: 55, 28, 3, 7, 5, 2, 0, 0, און ריי 4: 50, 25, 12, 6, 5, 2, 1, 0. דער עקראן בלייבט אויף דעם ביז 2:26:00.

א גראפיק פון מעל קעגן צוקער (2:26:30)

א נייע צעל מיטן קאמענטאר "# plot our data". ביי 2:27:00 שטייט דארט:

,'Sugar', data=recipes, hue='Type' ,'sns.lmplot('Flour

;({70 :"s"}palette='Set1', fit_reg=False, scatter_kws=

דער קאוד צייכנט א פונקט-גראפיק פון) קעגן Sugar) פון דער recipes טאבעלע, מיט Flourמעל) צוקער Type, אן א רעגרעסיע-ליניע (fit_reg=False), און מיט פונקטן פון גרייס 70.

פון 2:27:30 ביז 2:28:00 ווערט געוויזן דער רעזולטאט: דער הארוזאנטאלער אקס "Flour" (בערך 35 ביז 55) און דער ווערטיקאלער "Sugar" (בערך 5 ביז 30). די בלויע פונקטן (Cupcake) ליגן אויבן לינקס, מיט ווייניגער מעל און מער צוקער. די רויטע פונקטן (Muffin) ליגן אונטן רעכטס, מיט מער מעל און ווייניגער צוקער. די צוויי גרופעס זענען קלאר צעטיילט.

צוגרייטן די דאטא (2:28:30)

א נייע צעל מיטן קאמענטאר "# format or pre-process our data". ביז 2:29:30 ווערט אריינגעשריבן:

- type_label = np.where(recipes['Type']=='Muffin', 0, 1). דאס שאפט לעיבלס מיט np.where : א Muffin באקומט 0, און א Cupcake באקומט 1.
- recipe_features = recipe.columns.values[ ], א שורה וואס הייבט אן נעמען די נעמען פון די קאלאמען אלס features. ביים לעצטן פריים איז זי נאך נישט פארטיג געשריבן.

SVM: מאפין אדער קאפקעיק – פונעם קאוד ביזן גראף (לויטן עקראן) (2:30:00)

דער גראף פון מעל און צוקער, און א טעות אין קאוד (2:30:00)

אויפן עקראן ווערט געוויזן א Jupyter notebook מיטן נאמען SimplilearnSVM_Muffin_Cupcake.ipynb. אויבן זעט מען נאך די לעצטע שורה פון א טאבעלע מיט רעצעפטן (שורה 4, Muffin, מיט די צאלן 50, 25, 12, 6, 5, 2, 1, 0).

אין צעל In [3] מיטן קאמענטאר "# plot our data" רופט דער קאוד 'sns.lmplot('Flour ,'Sugar' data=recipes, hue='Type', palette='Set1', fit_reg=False, scatter_kws={"s": 70}). ער מאלט א פונקט-גראף פון די רעצעפטן: אויף דער האריזאנטאלער אקס שטייט) Flourמעל, פון בערך 35 ביז 55), און אויף דער ווערטיקאלער אקס) Sugarצוקער, פון בערך 5 ביז 30). די קאליר ווערט באשטימט לויט Type: רויטע פונקטן זענען Muffin, בלויע זענען Cupcake. די בלויע פונקטן ליגן לינקס און העכער (ווייניגער מעל, מער צוקער). די רויטע ליגן רעכטס און נידעריגער (מער מעל, ווייניגער צוקער). די צוויי גרופעס זענען קלאר אפגעטיילט.

אין צעל In [4], מיטן קאמענטאר "# format or pre-process our data", שטייען די שורות:

- type_label = np.where(recipes['Type']=='Muffin', 0, 1) – שאפט א רשימה פון לעיבלס: 0 פאר Muffin, 1 פאר Cupcake.
- recipe_features = recipes.columns.values[1:].tolist() – נעמט די נעמען פון די קאלומנעס, חוץ די ערשטע.
- recipe_features – ווייזט ארויס די רשימה.

דער ערשטער לויף גיט א NameError. אין דער טרעיסבעק זעט מען אז אין שורה 3 שטייט recipe.columns אנשטאט recipes.columns.

די אינגרידיענטן ווערן צוגעגרייט (2:30:30)

אין דער פאררעכטער צעל In [5] ווייזט דער אויסגאנג די רשימה פון קאלומנעס: ['Flour', 'Milk', 'Sugar', 'Butter', 'Egg', 'Baking Powder', 'Vanilla', 'Salt'].

דערנאך ווערט צוגעלייגט א שורה ingredients = recipes[['Flour','Sugar']].values. זי נעמט נאר די צוויי קאלומנעס מעל און צוקערן, אלס א מאטריצע. נאך דעם קומט print(ingredients).

אין צעל In [10] ווייזט דער אויסגאנג א מאטריצע פון פארן [[Flour Sugar: [55 3], [47 12], [47 18], [45 17], [50 12], [55 3], [54 7], [47 10], [50 17], [50 17], [39 26], [42 16], [34 20], [39 17], [38 23], [42 25], [36 21], [38 31], [36 24], [34 23].

דער מאדעל ווערט טרייניירט (2:32:00)

א נייע צעל הייבט זיך אן מיטן קאמענטאר "# fit model".

ביי 2:32:30 ווערט געווויזן די דאקומענטאציע-זייט פון scikit-learn, דער אפטייל "sklearn.svm: Support Vector Machines". דארט שטייט אז דער מאדול sklearn.svm אנטהאלט Support Vector Machine אלגאריטמען.

אונטער "Estimators" זענען אויסגערעכנט:

- Linear Support Vector Classification – svm.LinearSVC
- Linear Support Vector Regression – svm.LinearSVR
- Nu-Support Vector Classification – svm.NuSVC
- Nu Support Vector Regression – svm.NuSVR
- Unsupervised Outlier Detection – svm.OneClassSVM
- C-Support Vector Classification – svm.SVC
- Epsilon-Support Vector Regression – svm.SVR
- svm.l1_min_c – גיט צוריק די נידעריגסטע גרענעץ פאר C, אזוי אז דער מאדעל זאל זיכער נישט זיין ליידיג.

אונטער "Low-level methods" זענען אויסגערעכנט:

- svm.libsvm.cross_validation
- svm.libsvm.decision_function

- svm.libsvm.fit
- svm.libsvm.predict
- svm.libsvm.predict_proba

דערונטער הייבט זיך אן דער אפטייל sklearn.tree: Decision Trees.

צוריק אין notebook שרייבט דער קאוד model = svm.svc(kernel='linear') און model.fit(ingredients, type_label). דער לויף In [11]) גיט א טעות: AttributeError: module 'sklearn.svm' has no attribute 'svc'.

נאכן פאררעכטן צו) svm.SVCמיט גרויסע אותיות) ווייזט Out[12] דעם מאדעל מיט אלע זיינע פאראמעטערס: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False).

די צעטיילנדיקע היפערפלאך (2:35:00)

א נייע צעל, מיטן קאמענטאר "# get the separating hyperplane", רעכנט אויס די ליניע וואס טיילט צווישן די צוויי גרופעס:

- w = model.coef_[0] – די קאעפיציענטן פונעם מאדעל.
- a = -w[0] / w[1] – דער שיפוע פון דער ליניע.
- xx = np.linspace(30, 60) – ווערטן אויף דער X-אקס פון 30 ביז 60.
- yy = a * xx - (model.intercept_[0]) / w[1] – די Y-ווערטן פון דער ליניע.

דער עקראן בלייבט אויף דעם ביז בערך 2:36:30.

ביי 2:37:00 ווערט צוגעלייגט א צווייטער טייל, מיטן קאמענטאר "# Plot the parallels to the separating hyperplane that pass through the support vectors". דער טייל רעכנט אויס די צוויי פאראלעלע ליניעס וואס גייען דורך די support vectors:

- b = model.support_vectors_[0], און דערנאך yy_down = a * xx + (b[0] - a * b[1]) – די אונטערשטע ליניע.
- b = model.support_vectors_[-1], און דערנאך yy_up = a * xx + (b[0] - a * b[1]) – די אויבערשטע ליניע.

אין In [14] ווערט צוגעלייגט print(yy) . דער אויסגאנג איז א רשימה פון זייער גרויסע צאלן אין דער סדר פון e+16 ביז e+15. זיי הייבן זיך אן ביי 1.15779472e+16 און ענדיגן זיך ביי -1.41505904e+16.

דער גראף מיט די ליניעס (2:38:30)

In [16] מאלט נאכאמאל דעם זעלבן פונקט-גראף מיט sns.lmplot . דער אויסגאנג ווייזט seaborn.axisgrid.FacetGrid.

אין In [17] ווערט צוגעלייגט plt.plot(xx, yy, linewidth=2, color='black'). איצט איז די Sugar-אקס אין א מאסשטאב פון) 1e16פון בערך 1.5- ביז 1.0), און די Flour-אקס גייט פון 30 ביז 60. אלע פונקטן זעען אויס

ליגנדיג אויף 0. די בלויע זענען לינקס, די רויטע רעכטס, און א שווארצע שיפע ליניע גייט אראפ פון לינקס-אויבן ביז רעכטס-אונטן און שניידט דורך צווישן זיי.

אין In [18] ווערן צוגעלייגט plt.plot(xx, yy_down, 'k--') און plt.plot(xx, yy_up, 'k--'). זיי מאלן צוויי געשטריכלטע פאראלעלע ליניעס, איינע פון יעדער זייט פון דער שווארצער ליניע.

א פונקציע צו פאראויסזאגן (2:40:30)

א צעל מיטן קאמענטאר "# create a function to predict muffin or cupcake " דעפינירט די פונקציע muffin_or_cupcake(flour, sugar). אויב model.predict([[flour, sugar]])==0, דרוקט זי "You're looking at a muffin recipe!". אנדערש דרוקט זי "You're looking at a cupcake recipe!".

דערנאך, מיטן קאמענטאר "# Predict if 50 parts flour and 20 parts sugar ", ווערט גערופן muffin_or_cupcake(50, 20). דער אויסגאנג: "You're looking at a muffin recipe!".

אין In [20], מיטן קאמענטאר "# Let's plot this on the graph", ווערט נאכאמאל געמאלט דער גראף מיט דער שווארצער ליניע. אין In [21] ווערט צוגעלייגט plt.plot(50, 20, 'yo', markersize='9'). דאס מאלט א געלבלעכן פונקט ביי Flour 50, אויף דער רעכטער (מאפין) זייט פון דער ליניע.

סיכום, און דער צווייטער טייל (2:43:30)

דער סלייד "Key Takeaways" רעכנט אויס די טעמעס פונעם טייל, יעדע מיט א קליין בילד:

- What is Machine Learning?
- Supervised, Unsupervised, Reinforcement Learning – Types of Machine Learning
- Line of Best Fit – Regression
- Building a Decision Tree
- Classification using SVM

ביי 2:44:00 קומט דער טיטל "2 - MACHINE LEARNING TUTORIAL PART" פון Simplilearn.

ביי 2:44:30 ווייזט דער סלייד "What's in it for you?" וואס דער צווייטער טייל וועט אנטהאלטן:

- Clustering: What is Clustering?, K-Means Clustering, Flowchart to understand K-means Clustering (באצייכנט), DEMO: Clustering of cars based on brands.
- Logistic Regression: What is Logistic Regression?, Logistic Regression Curve & Sigmoid function, DEMO: Classify a tumor as malignant or benign based on features.

K-Means קלאסטערינג: טעאריע און א Python ביישפיל (לויטן עקראן) (2:45:00)

וואס איז Clustering (2:45:00)

דער סלייד מיטן טיטל "Clustering" ווייזט דריי פאליצעס מיט ספרים, מיט דער אויפשריפט "Now, we divide them into different groups like Fiction, Horror און Educational. די ספרים ווערן איינגעטיילט אין דריי גרופעס:

דער נעקסטער סלייד גיט די דעפיניציע: "Well, organizing objects into groups based on their similarity is Clustering!" אויף אידיש: איינטיילן זאכן אין גרופעס לויט ווי ענליך זיי זענען צו אנאנד, דאס הייסט Clustering.

K-Means אלס Unsupervised learning (2:46:00)

דריי וואלקנס, פארבונדן מיט פיילן, דערקלערן:

- K-Means Clustering איז א ביישפיל פון Unsupervised learning.
- מען ניצט עס וועו מען האט דאטא אן לעיבלס (unlabeled data).
- דער ציל איז צו געפינען קלאסטערס אין דער דאטא לויט ענלעכקייט פון די פיטשערס (feature similarity).

דער עקראן בלייבט אויף דעם סלייד בערך א מינוט.

די טריט פונעם אלגאריטם (2:47:00)

"STEP 1: Initialize Cluster Centroids": מען קלייבט אויס 'K' קלאסטערס און מען גיט די קלאסטערס צופעליגע צענטראידן. דערנאך רעכנט מען אויס די ווייטקייט פון יעדן אביעקט ביז די צענטראידן. א גראפיק ווייזט רויטע און גרינע פונקטן, מיט א רויטן שטערן און א גרינעם שטערן אלס צענטראידן.

"STEP 3: Assign Points to New Clusters": מען איבערחזר'ט די פריערדיגע צוויי טריט נאכאנאנד, ביז די צענטראידן הערן אויף זיך צו רירן און וועו סטאַטיש. א וואלקן מערקט אן: וועו די קלאסטערס וועו סטאַטיש, זאגט מען אז דער K-Means אלגאריטם איז "converged" (האט זיך אויסגעגליכן). אין דער גראפיק שטייען יעצט די שטערנס אינמיטן פון זייערע גרופעס.

א פלאוטשארט פון K-Means (2:48:00)

דער סלייד "Flowchart to understand K-Means" ווייזט די טריט, איינס נאכן אנדערן:

- START
- Choose K (Elbow Method), אויסקלייבן K דורך דער עלבאו-מעטאד
- Assign random centroids to clusters, צוטיילן צופעליגע צענטראידן
- Compute distance from objects to centroids, אויסרעכענען די ווייטקייטן ביז די צענטראידן
- Form new clusters based on minimum distance and calculate their centroids, פארמירן נייע קלאסטערס לויט דער קלענסטער ווייטקייט און אויסרעכענען זייערע צענטראידן
- Compute distance from objects to new centroids, אויסרעכענען די ווייטקייטן ביז די נייע צענטראידן

א רויטער קרייז מיט א געשטריכלטער פייל צוריק צום פארמירן-טריט זאגט: "Repeat until no observations change groups", חזר'ן ביז קיין איינציגע אבזערוואציע טוישט מער נישט איר גרופע. דער סלייד בלייבט אויפן עקראן אביסל מער ווי א האלבע מינוט.

א פראקטישער חשבון (2:49:00)

דער סלייד "K-Means Algorithm " נעמט א דאטא-סעט פון 7 מענטשן מיט זייערע פונקטן אין צוויי טעמעס, A און B:

- 1: A=1, B=1
- 2: A=1.5, B=2
- 3: A=3, B=4
- 4: A=5, B=7
- 5: A=3.5, B=5
- 6: A=4.5, B=5
- 7: A=3.5, B=4.5

דער נעקסטער סלייד ווייזט א גראפיק: יעדער פונקט ווערט צוגעטיילט צום נאענטסטן קלאסטער, לויט זיין ווייטקייט פון די צענטראידן. די פונקטן (1,1) און (5,7) זענען ארומגערינגלט אלס אנהייב-צענטראידן. Cluster 1 נעמט אריין די פונקטן 1, 2 און 3; Cluster 2 די איבעריגע.

דערנאך רעכנט מען אויס נאכאמאל די צענטראידן ("Now, we again calculate the centroids of each cluster"):

- Cluster 1: מענטשן 1, 2, 3, מיטל-וועקטאר (צענטראיד) (1.8, 2.3)
- Cluster 2: מענטשן 4, 5, 6, 7, צענטראיד (4.1, 5.4)

די נייע צענטראידן זענען אנגעצייכנט מיט גרינע פונקטן.

דער פונקט 3 גייט איבער (2:51:00)

א טאבעלע ווייזט די ווייטקייט פון יעדן מענטש ביז דעם צענטראיד פון Cluster 1 און פון Cluster 2:

- 1: 1.5 / 5.4
- 2: 0.4 / 4.3
- 3: 2.1 / 1.8 (אנגעצייכנט אין רויט)
- 4: 5.7 / 1.8
- 5: 3.2 / 0.7
- 6: 3.8 / 0.6
- 7: 2.8 / 1.1

נאר מענטש 3 איז נענטער צום צענטראיד פונעם אנדערן קלאסטער (Cluster 2) ווי צו זיין אייגענעם (Cluster 1). אויף דער גראפיק שטייט "Moving point 3 to new cluster", מען פירט איבער פונקט 3 צום נייעם קלאסטער.

פאר די נייע קלאסטערס געפינט מען די עכטע צענטראידן: Cluster 1 האט (1.25, 1.5), Cluster 2 האט (3.9, 5.1). אין דער טאבעלע שטייט ביי Cluster 1 נאר אלץ "1, 2, 3" און ביי Cluster 2 "4, 5, 6, 7". דער גראף ווייזט אבער א קליינעם Cluster 1 מיט צוויי פונקטן, און פונקט 3 (3.0, 4) ארויסגעפירט אין Cluster 2.

די Elbow מעטאד (2:52:00)

כדי צו געפינען די ריכטיגע צאל קלאסטערס אין א דאטא-סעט, ניצט מען די elbow method . א גראף ווייזט WSS אויף דער ווערטיקאלער אקס און "No. of clusters" (1 ביז 8) אויף דער אריזאנטאלער. די קורווע פאלט שטארק אראפ און ווערט דערנאך פלאך. די בייג, ארום 2–3, איז אנגעצייכנט אלס "Elbow point".

- Within sum of squares (WSS) איז די סומע פון די קוואדראטן פון די ווייטקייטן צווישן יעדן מיטגליד פונעם קלאסטער און זיין צענטראיד.
- געפינען די אפטימאלע צאל קלאסטערס דורך דעם "עלנבויגן" פונעם גראף הייסט די Elbow method.
- אויסקלייבן דעם ריכטיגן ווערט פון k העלפט צו דארפן ווייניגער איטעראציעס.

דער סלייד בלייבט אויף דעם עקראן אן ערך א מינוט.

Use Case: קלאסטערירן קארס (2:53:00)

דער סלייד "Use Case" שטעלט פאר א פראיעקט: ניצן K-means clustering צו איינטיילן קארס לויט פירמע (brand), מיט פאראמעטערס ווי horsepower, cubic inches, make year א.א.וו. די דאטא-סעט: קאר-דאטא מיט אינפארמאציע וועגן 3 פירמעס, Toyota, Honda און Nissan.

דער קאוד אין Jupyter (2:53:30)

אין א Jupyter notebook (Python 3) ווערט געשריבן א צעל מיטן קאמענטאר "# K-Means Clustering". זי אימפארטירט די ביבליאטעקן:

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd
```

דערנאך קומט %matplotlib inline , מיטן קאמענטאר אז דאס לאזט די גראפיקס זיך באווייזן אינעם notebook.

די דאטא-טעקע cars.csv (2:54:30)

דער עקראן עפנט די טעקע cars.csv אין א ספרעדשיט-פראגראם. די קאלאמען: mpg, cylinders, cubicinches, hp, weightlbs, time-to-60, year, brand. למשל, די ערשטע שורה: 14, 8, 350, 165, 4209, 12, 1972, Toyota; די צווייטע: 31.9, 4, 89, 71, 1925, 14, 1980, Honda. עס קומען אויך פאר Nissan-שורות, למשל 37.7, 4, 89, 62, 2050, 17, 1982. אין איין שורה (19.1, 6, 225, 90, ..., 19, 1981, Toyota) איז דער weightlbs ליידיג, א פעלנדיגער ווערט.

לאדן און צוגרייטן די דאטא (2:55:00)

די נעקסטע צעל ("# Importing the dataset") לייענט אריין די טעקע:

```
pd.read_csv('cars.csv') = dataset
```

דערנאך ווערט געשריבן dataset[:-1] = X און X.head(). דער רעזולטאט ווייזט די ערשטע 5 שורות פון אלע 8 קאלאמען, אריינגערעכנט brand.

א נייע צעל, "Eliminating null values #", גייט איבער יעדע קאלאמע אין X.columns און רופט X[i].fillna(int(X[i].mean()), כדי צו פארפילן ליידיגע ווערטן מיטן דורכשניט פון דער קאלאמע. נאך דעם קומט א צווייטע לופ איבער X.columns מיט print.

אפזונדערן די פיטשערס (2:57:30)

דער קאוד ווערט איבערגעשריבן. קודם X = dataset.iloc[:,:-1], דערנאך print(dataset.columns). דער אויסדרוק ווייזט:

Index(['mpg','cylinders','cubicinches','hp','weightlbs','time-to-60','year','brand'], dtype='object')

מען זעט אז די נעמען פון די קאלאמען, אויסער mpg, הייבן זיך אן מיט א ליידיגן פלאץ.

דערנאך print(dataset.columns[:-1]) און X = dataset[dataset.columns[:-1]]. דער אויסדרוק ווייזט שוין די קאלאמען אָן 'brand'.

צום סוף ווערט X.head() אויסגעפירט (מיט אן אנגעהויבענער שורה X. = X). דער רעזולטאט ווייזט די ערשטע 5 שורות מיט 7 קאלאמען: mpg, cylinders, cubicinches, hp, weightlbs, time-to-60, year. די brand-קאלאמע איז שוין אוועקגענומען.

K-Means קלאסטערינג אויף אויטא-דאטא (לויטן עקראן) (3:00:00)

צוגרייטן דעם דאטא-סעט cars.csv (3:00:00)

אויפן עקראן ווערט געוויזן א Jupyter נאטבוק (Python 3) מיטן טיטל "K-Means Clustering". אין דער ערשטער צעל ווערן אריינגעברענגט די ביבליאטעקן: numpy אלס np, matplotlib.pyplot אלס plt , און pandas אלס pd. די שורה %matplotlib inline ערמעגליכט אז די גראפיקס זאלן זיך ווייזן גלייך אינעם נאטבוק.

אין דער צווייטער צעל ווערט דער דאטא-סעט אריינגעלאדן:

dataset = pd.read_csv('cars.csv')

אין דער דריטער צעל ווערן אויסגעקליבן אלע קאלומנעס חוץ דער לעצטער: X = dataset[dataset.columns[:-1]] . דערנאך ווערט פראבירט די דאטא איבערצומאכן אויף נומערן מיט X.convertobjects(convert_numeric=True), און X.head() ווייזט די ערשטע שורות. דער ערשטער פרואוו גיט אן AttributeError , ווייל דער נאמען פון דער פונקציע איז נישט ריכטיג געשריבן. נאכן פארעכטן אויף convert_objects לויפט עס, אבער pandas ווייזט א FutureWarning: convert_objects איז deprecated, און מען זאל ניצן DataFrame.infer_objects, אדער די ספעציפישע פונקציעס pd.to_datetime, pd.to_timedelta און pd.to_numeric.

דער רעזולטאט (Out[23]) ווייזט א טאבעלע מיט די קאלומנעס mpg, cylinders, cubicinches, hp, weightlbs, time-to-60, year. די ערשטע פינף שורות:

- 0: 14.0, 8, 350.0, 165, 4209.0, 12, 1972
- 1: 31.9, 4, 89.0, 71, 1925.0, 14, 1980

- 2: 17.0, 8, 302.0, 140, 3449.0, 11, 1971
- 3: 15.0, 8, 400.0, 150, 3761.0, 10, 1971
- 4: 30.5, 4, 98.0, 63, 2051.0, 17, 1978

אויסמעקן ליידיגע ווערטן (3:00:30)

א נייע צעל מיטן קאמענטאר "Eliminating null values". דער קאוד גייט דורך יעדע קאלום אין X.columns, און פילט אן די פעלנדיקע ווערטן (null) מיט דעם דורכשניט פון יענער קאלום, איבערגעמאכט אויף א גאנצע צאל:

X[i].fillna(int(X[i].mean())) = X[i]

דערנאך גייט א צווייטע לופּ איבער אלע קאלומנעס און דרוקט אויס X[i].isnull().sum(), ד.ה. וויפיל ליידיגע ווערטן זענען נאך געבליבן. (אין צווישן איז דא א טיפּ-פעלער "X.coluns", וואס ווערט פארריכט אויף X.columns.) דער רעזולטאט: זיבן נולן, איינס פאר יעדע קאלום – קיין ליידיגע ווערטן זענען נישט געבליבן.

די "עלבויגן-מעטאד" (Elbow method) (3:03:30)

א נייע צעל הייבט זיך אן מיטן קאמענטאר "Using the elbow method to find the optimal number of clusters" און from sklearn ...

דאן ווערט געוויזן א סלייד מיטן טיטל "K-Means Algorithm":

- "To find appropriate number of clusters in a dataset, we use elbow method".
- א גראפיק: אויף דער Y-אַקס WSS, אויף דער X-אַקס "No. of clusters" (1 ביז 8). די ליניע פאלט שטארק אראפ און ווערט דערנאך כמעט פלאך; דער בייג-פונקט איז אנגעצייכנט "Elbow point".
- דעפיניציע: "Within sum of squares (WSS) is defined as the sum of the squared distance between each member of the cluster and its centroid" – די סומע פון די קוואדראטן פון דער וויטקייט צווישן יעדן מיטגליד פונעם קלאסטער און זיין צענטער (centroid).
- "Finding the optimal number of clusters using the elbow of the graph is called as the Elbow method".
- "Choosing the right value of k will help in less number of iterations" – דער ריכטיגער k מינדערט די צאל איבערחזרונגען.

דער קאוד פאר דער עלבויגן-גראפיק (3:04:30)

צוריק אינעם נאטבוק. דער קאוד:

- from sklearn.cluster import KMeans – ברענגט אריין דעם KMeans אלגאריטם.
- wcss = [] – א ליידיגע ליסטע פאר די WCSS ווערטן.
- א לופּ for i in range(0, 11), וואו יעדעס מאל ווערט געשאפן א מאדעל:

KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0) = kmeans

ד.ה. i קלאסטערן, אנהייב-מעטאד ++k-means , מאקסימום 300 איטעראציעס, 10 אנהייבן, און א פעסטער random_state.

- kmeans.fit(X) – טרייניירט דעם מאדעל אויף דער דאטא.
- wcss.append(kmeans.inertia_) – לייגט צו דעם inertia_ ווערט (די WCSS) צו דער ליסטע.

נאכדעם ווערט די גראפיק געצייכנט מיט) plt.plot(range(1,11), wcss)דער range ווערט שפעטער געטוישט פון 0,11 אויף 1,11), plt.title('The Elbow Method'), plt.xlabel('Number of clusters'), plt.ylabel('WCSS'), plt.show(.

דער רעזולטאט פון דער עלבויגן-גראפיק (3:08:00)

די גראפיק "The Elbow Method" ווייזט) WCSSסקאלע 1e8) קעגן Number of clusters פון 1 ביז 10. ביי 1 קלאסטער איז דער ווערט ארום 1.9, ביי 2 פאלט ער שארף אראפ צו ארום 0.5, ביי 3 ארום 0.25, ביי 4 ארום 0.13, און דערנאך ווערט די ליניע כמעט פלאך ביז נאענט צו 0.

אנוועגדן K-Means אויפן אויטא-דאטא-סעט (3:08:30)

א צעל מיטן קאמענטאר "Applying k-means to the cars dataset":

- KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state...) = kmeans – א מאדעל מיט דריי קלאסטערן.
- kmeans.fit_predict(X) = y_kmeans – טרייניירט און גיט פאר יעדע אויטא א קלאסטער-נומער.
- X.as_matrix(columns=None) = X – מאכט איבער X אויף א מאטריץ.

עס ווייזט זיך א FutureWarning: "Method .. will be removed in a future version. as_matrix use instead. values".

print(y_kmeans) דרוקט אויס א לאנגע רייע פון די נומערן 0, 1 און 2 – דער קלאסטער פון יעדער אויטא (די אנהייב: 2 1 2 0 1 2 2 2 1 2 0 1 1 1 0 2 2 2 1 2 ...).

print(X) ווייזט די מאטריץ אין וויסנשאפטלעכער נאטאציע, למשל [3.500e+02 8.000e+00 1.400e+01 ... 4.209e+03, 1.200e+01 1.972e+03] ד.ה. די זעלבע ווערטן ווי אין דער טאבעלע; די לעצטע שורות הייבן זיך אן מיט 2.200e+01, 1.800e+01, 2.200e+01 און 6 צילינדערס.

וויזואליזירן די קלאסטערן (3:11:30)

דער לעצטער קאוד צייכנט א scatter-גראפיק:

- plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s=100, c='red', label='Toyota') – די אויטאס פון קלאסטער 0, לויט דער ערשטער און צווייטער קאלום, אין רויט.
- די זעלבע שורה פאר קלאסטער 1 אין בלוי (label='Nissan') און פאר קלאסטער 2 אין גרין (label='Honda').
- plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=300, ...) – די צענטערס פון די קלאסטערן, מיט גרעסערע פונקטן.
- plt.title('Clusters of car make'), plt.legend(), plt.show().

די קלאסטער-גראפיק (3:14:30)

די גראפיק "Clusters of car make" ווייזט אויף דער X -אַקס ווערטן פון בערך 10 ביז 47 (די ערשטע קאלום, mpg) און אויף דער Y-אַקס 3 ביז 8. (cylinders) די לעגענדע: (Toyotaרויט), (Nissanבלוי), Honda)גרין), (Centroidsגעל).

- ביי 8 צילינדערס זענען מערסטנס גרינע פונקטן צווישן בערך 10 און 20, מיט א פאר רויטע; א געלער צענטער ביי ארום 15.
- ביי 6 צילינדערס זענען געמישטע גרינע, רויטע און בלויע פונקטן, בערך 15 ביז 38; א געלער צענטער ביי ארום 20.
- ביי 5 צילינדערס – א פאר רויטע פונקטן.
- ביי 4 צילינדערס – כמעט אלץ בלוי, פון בערך 18 ביז 47, מיט א געלן צענטער ביי ארום 29.
- ביי 3 צילינדערס – צוויי בלויע פונקטן.

לאגיסטישע רעגרעסיע: טעאריע און א ערשטער נאוטבוק (לויטן עקראן) (3:15:00)

דער סוף פון K-means: קלאסטערס פון קאר-מאַרקעס (3:15:00)

אויפן עקראן ווערט געווויזן א Jupyter נאוטבוק (Python 3) מיט די לעצטע שורות פון א קאוד-צעל. שורה 4 רופט `plt.scatter` מיט `kmeans.cluster_centers_[:,0]` און `kmeans.cluster_centers_[:,1]` מיט `s=300`, דאס הייסט עס צייכנט די צענטערס פון די קלאסטערס מיט גרויסע פינטלעך. שורה 5 שטעלט דעם טיטל 'Clusters of car make', שורה 6 לייגט צו א לעגענדע (`plt.legend()`), און שורה 7 ווייזט דעם גראף (`plt.show()`).

דער אויסגאנג איז א סקעטער-גראף מיטן טיטל "Clusters of car make". די לעגענדע ווייזט פיר סארטן: (Toyotaרויט), (Nissanבלוי), (Hondaגרין) און (Centroidsגעל). די X -אקס גייט בערך פון 10 ביז 45, די Y-אקס פון 3 ביז 8. די פינטלעך ליגן אין האריזאנטאלע שורות ביי די ווערטן 3, 4, 5, 6 און 8:

- ביי 8: מערסטנס גרינע (Honda) פינטלעך צווישן 10 און 20, מיט א פאר רויטע, און א געלער צענטער ביי בערך 15.
- ביי 6: א געמיש פון רויט, גרין און בלוי, מיט א געלן צענטער ביי בערך 20.
- ביי 5: נאר עטלעכע רויטע פינטלעך.
- ביי 4: א לאנגע שורה בלויע (Nissan) פינטלעך פון בערך 18 ביז 47, מיט א געלן צענטער ביי בערך 30.
- ביי 3: צוויי בלויע פינטלעך.

וואס איז Logistic Regression (3:15:30)

א נייער סלייד מיטן טיטל "Logistic Regression" ברענגט די דעפיניציע: דער Logistic Regression אלגאריטם איז דער איינפאכסטער קלאסיפיקאציע-אלגאריטם, און מען ניצט אים פאר בינארע אדער מולטי-קלאסיפיקאציע פראבלעמען.

דער ווייטערדיקער סלייד ((3:16:00 גייט צוריק צו דעם וואס איז געלערנט געווארן אין דער פריערדיקער לעקציע: Linear Regression, און דעפענדענטע און אינדעפענדענטע וואריאבלען. אלס א חזרה שטייט די פארמל mx+c = y. צוויי וואלקנס דערקלערן די באגריפן:

- דער דעפענדענטער וואריאבל איז דער ציל-קלאס וואריאבל וואס מען וויל פאָרויסזאגן.
- די אינדעפענדענטע וואריאבלען (x1...xn) זענען די אייגנשאפטן (features אדער attributes) מיט וועלכע מען וועט פאָרויסזאגן דעם ציל-קלאס.

פון א גלייכער ליניע צו דער Sigmoid קורווע (3:16:30)

דער סלייד ווייזט צוויי גראפן. ביידע האבן אויף דער X-אקס "No. of hours studied" (צאל שעה'ן געלערנט) און אויף דער Y-אקס "Marks" (צייכנס):

- לינקס: א גלייכע ליניע וואס גייט ארויף פון 0 ביז איבער 100, מיט דער פארמל y = m*x + c.
- אין מיטן: א פייל מיטן נאמען "Sigmoid Function" און דער פארמל p = 1 / (1 + e^-y).
- רעכטס: אן S-פארמיגע "Sigmoid Curve" וואס הייבט זיך אן ביי 0, שטייגט שטארק ארויף, און ווערט גלייך ביי 1. אונטן שטייט די פארמל ln(p / (1-p)) = m*x + c.

דער סלייד בלייבט אויפן עקראן ביז 3:17:00.

דער שוועל-ווערט (3:17:30)

א גראף מיטן טיטל "Logistic Regression" ווייזט א sigmoid קורווע. די X-אקס גייט פון 0 ביז 9, די Y-אקס פון 0 ביז 1.2. פינקטלעך ביי 1, 2 און 3 ליגן ביי 0, און פינקטלעך ביי 5, 6 און 7 ליגן ביי 1. דריי האריזאנטאלע שטרייפלדיקע ליניעס זענען באצייכנט:

- אויבן, ביי 0.82 (גרין): וועט די פראבאבילקייט איז מער ווי 0.50, ווערט דער ווערט אפגעקיַלעכט צו 1. דאס מיינט אז דער תלמיד וועט דורכגיין.
- אין מיטן, ביי 0.5 (שווארץ): "Threshold value", דער שוועל-ווערט.
- אונטן, ביי 0.30 (רויט): וועט די פראבאבילקייט איז ווייניגער ווי 0.50, ווערט דער ווערט אפגעקיַלעכט צו 0. דאס מיינט אז דער תלמיד וועט דורכפאלן.

די אויפגאבע: א בייזע אדער א גוטארטיגע געשוואולסט (3:18:00)

א סלייד מיט א גרויסן פראגע-צייכן שטעלט פאָר דעם "Problem statement": מען דארף קלאסיפיצירן צי א געשוואולסט (tumor) איז 'malignant' (בייזארטיג) אדער 'benign' (גוטארטיג). דער סלייד בלייבט אויפן עקראן ביז 3:19:00.

דער נאוטבוק: אימפארטן און לייענען די דאטא (3:19:30)

אין א נייעם Jupyter נאוטבוק אימפארטירט די ערשטע צעל די נויטיקע ביבליאטעקן:

- `numpy as np`, מיטן באמערק "linear algebra";
- `pandas as pd`, מיטן באמערק "data processing, CSV file I/O (e.g. pd.read_csv)";
- `seaborn as sns`;

- `matplotlib.pyplot as plt`;
- `%matplotlib inline`, כדי די גראפן זאלן זיך ווייזן גלייך אינעם נאוטבוק.

די צעל ווערט אויסגעפירט (In [1]). דערנאך (3:20:00–3:20:30) ווערט אין א נייער צעל אנגעשריבן:

`pd.read_csv('dataForP2.csv') = data`

דאס לייענט אריין דעם טעקע dataForP2.csv אין א pandas טאבעלע מיטן נאמען data.

דער דאטא-סעט אין א טאבעלע (3:21:00)

דער טעקע dataForP2.csv ווערט געעפנט אין OpenOffice Calc. די קאלומנעס זענען: id, diagnosis, radius_mean, texture_mean, perimeter_mean, area_mean, smoothness_mean, און ווייטער. אין די ערשטע שורות שטייט ביי diagnosis "M". למשל, די ערשטע שורה: id 842302, radius_mean 17.99, texture_mean 10.38, perimeter_mean 122.8, area_mean 1001, smoothness_mean 0.1184. די צווייטע שורה: id 842517, מיט 20.57, 17.77, 132.9, 1326 און 0.08474. די טאבעלע בלייבט אויפן עקראן ביז 3:21:30.

data.head() און א joint-plot (3:22:00)

צוריק אינעם נאוטבוק ווייזט צעל [2] (`data.head()`) די ערשטע 5 שורות. אויף דער רעכטער זייט זעט מען קאלומנעס ווי compactness_worst, concavity_worst, concave points_worst, symmetry_worst און fractal_dimension_worst. למשל, די ערשטע שורה האט 0.6656, 0.7119, 0.2654, 0.4601 און 0.11890. אויף דער לינקער זייט (3:22:30 זעט מען) id, diagnosis (M), radius_mean, texture_mean און ווייטער. אונטן שטייט "5 rows × 32 columns".

די נעקסטע צעל צייכנט א joint-plot פון צוויי קאלומנעס:

`sns.jointplot('radius_mean', data=data, 'texture_mean')`

דער אויסגאנג (3:23:00 איז א סקעטער-גראף מיט radius_mean אויף דער X -אקס (בערך 10 ביז 25) און texture_mean אויף דער Y-אקס (בערך 10 ביז 40). רוב פינטלעך זענען גענידיגט ארום radius 10–15 און texture 15–20. אויבן און אויף דער זייט שטייען היסטאגראמען. ביי 3:26:30 זעט מען דעם אויסגאנג אלס `seaborn.axisgrid.JointGrid`.

א heatmap פון קארעלאציעס (3:23:30)

די צעל `sns.heatmap(data.corr())` צייכנט א היטמאפ פון די קארעלאציעס צווישן אלע קאלומנעס. דער אויסגאנג איז `matplotlib.axes._subplots.AxesSubplot`. די אקסן ווייזן נעמען ווי id, texture_mean, area_mean, compactness_mean, concave points_mean, fractal_dimension_mean, texture_se, area_se, compactness_se, concave points_se, fractal_dimension_se, texture_worst, area_worst, compactness_worst, concave points_worst און fractal_dimension_worst. די פארב-סקאלע גייט פון 0.25- (טונקל) ביז 1.00 (העל). די היטמאפ בלייבט אויפן עקראן פון 3:24:30 ביז 3:27:00.

קאנטראל אויף פעלנדיקע ווערטן (3:26:00)

די צעל `data.isnull().sum()` טראגט דעם באמערק "#columns doesn't have any nulls". דער אויסגאנג (3:27:30 רעכנט אויס פאר יעדע קאלומנע ווי פיל ווערטן פעלן, אנגעהויבן פון concavity_se ביז fractal_dimension_worst. ביי יעדע קאלומנע שטייט 0, און דער טיפ איז dtype: int64.

אויסקלייבן X און y (3:28:00)

אינעם אויסגאנג ווערן אנגעצייכנט די קאלומנעס radius_worst ביז compactness_worst. אין א נייער צעל שרייבט מען אָן:

```
X = data[['radius_worst' ,'texture_worst' ,'perimeter_worst' ,'area_worst' ,...`
```

```
y = data['diagnosis']`
```

```
X.head()`
```

דאס הייסט: X באקומט די אויסגעקליבענע "worst" קאלומנעס אלס אייגנשאפטן, און y באקומט די קאלומנע diagnosis אלס דעם ציל-קלאס. `X.head()` זאל ווייזן די ערשטע שורות. ביי 3:29:30 ווערט די פערטע שורה, y, נאך אנגעשריבן.

לאגיסטישע רעגרעסיע אויף אַ שוואלסט-דאטא, און דער אנהייב פון) LLMלויטן עקראן) (3:30:00)

דער Jupyter נאטבוק: אויסקלייבן X און y (3:30:00)

אויפן עקראן ווערט געווויזן א Jupyter נאטבוק (Python 3, "Untitled1"). אויבן שטייט נאך דער אויסגאנג פון א פריערדיגע צעל: א רשימה פון קאלומען-נעמען, יעדער מיט דער צאל 0 דערנעבן. די נעמען זענען radius_worst, texture_worst, perimeter_worst, area_worst, smoothness_worst, compactness_worst, concavity_worst, concave points_worst, symmetry_worst און fractal_dimension_worst, און צום סוף dtype: int64. דאס וויזט אז אין די קאלומען פעלט נישט קיין אייגציגער ווערט.

אין צעל In [8] שטייען צוויי שורות. די ערשטע שורה נעמט פון דער טאבעלע data די קאלומען radius_worst, texture_worst, perimeter_worst, area_worst און נאך, און לייגט זיי אריין אין X. די רעשט פון דער שורה גייט ווייטער פון עקראן. די צווייטע שורה נעמט די קאלומע diagnosis און לייגט זי אריין אין y:

```
X = data[['radius_worst' ,'texture_worst' ,'perimeter_worst' ,'area_worst' ,...
```

```
y = data['diagnosis']
```

צעטיילן די דאטא אויף טרייניר און טעסטן (3:30:30)

אין דער נייער צעל ווערט שריטלעך געשריבן דער קאוד. ערשט ווערט אימפארטירט די פונקציע train_test_split פון sklearn.model_selection. דערנאך ווערט די דאטא צעטיילט אויף פיר טיילן: X_train, X_test, y_train, y_test. צום סוף ווערט צוגעלייגט או 30% פון דער דאטא גייט אויף טעסטן

(test_size=0.3, און עס הייבט זיך אן א ווייטערדיגער פאראמעטער מיט "rand…", וואס ווערט אפגעשניטן ביים ראנד פון עקראן:

…train_test_split(X, y, test_size=0.3, rand = X_train, X_test, y_train, y_test

די צעל ווערט לויפן געלאזט מיטן "Run" קנעפל און באקומט די נומער In [9].

בויען און טרייניירן דעם מאדעל (3:32:30)

אין צעל In [10] ווערט אימפארטירט LogisticRegression פון sklearn.linear_model. מען שאפט א מאדעל מיטן נאמען logModel, און מען טרייניירט אים אויף דער טרייניגג-דאטא:

LogisticRegression() = logModel

logModel.fit(X_train, y_train)

דער אויסגאנג Out[10] ווייזט אלע איינשטעלונגען פונעם מאדעל:

- C=1.0
- class_weight=None
- dual=False
- fit_intercept=True
- intercept_scaling=1
- max_iter=100
- 'multi_class='ovr
- n_jobs=1
- 'penalty='l2
- random_state=None
- 'solver='liblinear
- tol=0.0001
- verbose=0
- warm_start=False

פאראויסזאגן און דרוקן די רעזולטאטן (3:33:30)

אין צעל In [11] זאגט דער מאדעל פאראויס די ענטפערס פאר דער טעסט-דאטא:

logModel.predict(X_test) = y_pred

אין צעל In [12] ווערט y_pred אויסגעדרוקט מיט print(y_pred). דער אויסגאנג איז א לאנגע רשימה פון בלויז צוויי אותיות, 'B' און 'M'. אזוי ווערט יעדער פאל אין טעסט-סעט צוגעטיילט צו איינע פון די צוויי קלאסן. די רשימה פארנעמט ארום צען שורות.

דער classification_report (3:35:30)

אין צעל In [13] ווערט אימפארטירט classification_report פון sklearn.metrics, און מען דרוקט דעם באריכט וואס פארגלייכט y_test מיט y_pred. דער אויסגאנג איז א טאבעלע:

- B: precision 0.93, recall 0.94, f1-score 0.93, support 105
- M: precision 0.91, recall 0.88, f1-score 0.89, support 66
- total: precision 0.92, recall 0.92, f1-score 0.92, support 171 / avg

דאס הייסט אז אין טעסט-סעט זענען דא 171 פעלער: 105 פון קלאס B און 66 פון קלאס M.

סליידס "Use Case": באריכט און confusion matrix (3:36:30)

א סלייד מיטן טיטל "Use Case" ווייזט דעם זעלבן classification_report, נאר דארט הייסט דער צווייטער פאראמעטער predictions אנשטאט y_pred. די צאלן זענען די זעלבע. א בלאז-טעקסט אויפן סלייד זאגט: "Let's see the precision, recall using classification report".

דער נעקסטער סלייד אימפארטירט confusion_matrix פון sklearn.metrics , רעכנט אויס confusion_matrix(y_test, y_pred), און ווייזט דעם רעזולטאט:

array([[99, 6], [8, 58]], dtype=int64)

דער בלאז-טעקסט דערביי זאגט: "So, this model is able to predict the type of tumor with 91% accuracy!" (דער מאדעל קען פאראויסזאגן דעם סארט שוואלסט מיט 91% גענויקייט).

"Can you tell what's happening in the following cases?" (3:37:30)

א סעריע סליידס שטעלט פיר פראגעס, יעדע מיט אן ענטפער:

- A. איינטיילן דאקומענטן אין פארשידענע קאטעגאריעס לויט דעם טעמע און אינהאלט פון יעדן דאקומענט. ענטפער: דאס איז א ביישפיל פון Clustering. מען קען ניצן K-means clustering כדי צו גרופירן די דאקומענטן לויט טעמעס, מיט דעם bag-of-words צוגאנג.
- B. ריכטיג דערקענען האנט-געשריבענע ציפערן אין בילדער. ענטפער: דאס איז Classification. דער טראדיציאנעלער וועג איז ארויסצוציען אייגנשאפטן וואס הענגען אפ פון יעדן ציפער, למשל די קרומקייט פון די פארשידענע ציפערן, און דערנאך ניצן א קלאסיפיצירער ווי SVM כדי צו אונטערשיידן צווישן די בילדער.
- C . א וועבזייטל פירט זיך אויף אויף אן אופן וואס ווייזט אז עס ארבעט נישט ווי פלאנירט. ענטפער: דאס איז Anomaly Detection. דער אלגאריטם לערנט זיך אויס וואס איז "normal" און וואס איז "not normal", געווענליך דורך אנקוקן די logs פונעם וועבזייטל. דער סלייד בלייבט אויפן עקראן ארום א מינוט.
- D. פאראויסזאגן די געהאלט פון א מענטש לויט זיינע יארן ערפארונג. ענטפער: דאס איז Regression. מען קען דעם פראבלעם מאטעמאטיש דעפינירן ווי א פונקציע צווישן אן אומאפהענגיגן פאריאבל (יארן ערפארונג) און אן אפהענגיגן פאריאבל (די געהאלט).

"Summary" (3:40:00)

א סלייד מיטן טיטל "Summary" ברענגט פינף קליינע בילדער מיט כותרות:

- WHAT IS K-MEANS: א פלוס-דיאגראם. "Choose K (Elbow Method)", "Assign random centroids to clusters", "Compute distance from objects to centroids", "Form new clusters based on minimum distance and calculate their centroids", "Compute distance from objects to new centroids", און א שלייף "Repeat until no observations change groups".
- ELBOW METHOD TO CHOOSE K: א גראפיק פון WSS קעגן "No. of clusters", מיט אן "Elbow point".
- CLUSTERING CARS WITH K-MEANS: קאוד און א פונקט-גראפיק פון קלאסטערס.
- WHAT IS LOGISTIC REGRESSION: א גראפיק מיטן נאמען "Logistic Regression" מיט אן S-פארמיגער ליניע.
- CLASSIFYING TUMOR WITH LOGISTIC: א heatmap פון קארעלאציעס און א jointplot.

א נייער טייל: Large Language Models (3:40:30)

עס הייבט זיך אן א נייער טייל פון Simplilearn. ערשט ווערט געווייזן דער טיטל "Large Language Models". דערנאך קומען די פראגעס וואס דער טייל וועט באהאנדלען: "What LLMs Are?", "How They Work?", "Why Models Like Gemini Are Critical For The Future Of AI?".

ביי 3:42:30 ווערט געווייזן א רשימה פון ניצונגען: Virtual Assistants, Chatbots, Content Creation, Language Translation, Supporting Research, Decision Making.

"How Are Large Language Models Trained?" (3:43:00)

א סלייד מיט דעם טיטל ערקלערט: גרויסע שפראך-מאדעלן גייען געווענליך דורך pre-training אויף א ברייטן, אלעס-אומפאסנדיקן דאטא-סעט, וואס האט סטאטיסטישע ענלעכקייטן מיטן דאטא-סעט פון דער ספעציפישער ציל-אויפגאבע. דערנאך דערשיינט, שריט נאך שריט, א קייט פון זעקס קרייזן:

- Text Pre-processing
- Random Parameter Initialization
- Inputting Numerical Data
- Loss Function Calculation
- Parameter Optimization
- Iterative Training

פון גרויסע שפראך-מאדעלן צו אן MCP סערווער (לויטן עקראן) (3:45:00)

ווי ארבעטן גרויסע שפראך-מאדעלן (3:45:00)

אויפן עקראן ווערט געווייזן א סלייד מיטן טיטל "How Do Large Language Models Work?". אויבן שטייט די הויפט-דעפיניציע: גרויסע שפראך-מאדעלן (Large Language Models) ניצן טיפע נויראל-נעצן כדי צו שאפן רעזולטאטן לויט די מוסטערן וואס זיי האבן זיך אויסגעלערנט פון דער טריינירונג-דאטא.

אין א געלן קעסטל אויף דער לינקער זייט שטייט: LLM דערמעגלעכט דעם מאדעל צו דערקענען די באציאונגען צווישן ווערטער אין א זאץ, נישט קוקנדיג וואו זיי שטייען אין דער רייע.

אין מיטן איז א בילד פון א נויראל-נעץ אין דער פארעם פון א מוח, און דערונטער א בלאסער צעטל מיטן נאמען "Recurrent Neural Network (RNN)".

אנוועגדונגען פון גרויסע שפראך-מאדעלן (3:45:30)

דער ווייטערדיגער סלייד, "Applications of Large Language Models.", ווייזט קארטלעך וואס באווייזן זיך איינס נאכן אנדערן:

- Natural Language Processing – פאראלבעטן נאטירלעכע שפראך
- Chatbots & Virtual Assistants – טשעט-באטס און ווירטועלע געהילפן
- Machine Translation – מאשין-איבערזעצונג

א נייער טייל: ווי אזוי בויט מען אן MCP סערווער (3:46:30)

דערנאך הייבט זיך אן א נייער טייל. אויפן טיטל-בילד שטייט "How To Build MCP Server", צוזאמען מיטן MCP לאגא. נאכדעם קומט אן איינפירונג-בילד מיט א טשיפ וואס איז באצייכנט "AI".

וואס איז Model Context Protocol (3:47:30)

א סלייד מיטן טיטל "What is Model Context Protocol?" גיט די דעפיניציע: דאס איז אן אפענער סטאנדארט וואס Anthropic האט אנטוויקלט. זיין ציל איז צו מאכן איין איינהייטלעכן שטייגער ווי אזוי גרויסע שפראך-מאדעלן (LLMs) פארבינדן זיך מיט דרויסנדיגע דאטא, געצייג און סערוויסעס.

די אפיציעלע דאקומענטאציע (3:48:00)

דער בלעטערער עפנט דעם בלאט docs.anthropic.com/en/docs/mcp, מיטן טיטל "Model Context Protocol (MCP)". דארט שטייט אז MCP איז אן אפענער פראטאקאל וואס שטעלט א סטאנדארט ווי אזוי אפליקאציעס גיבן צו "קאנטעקסט" צו LLMs.

דער בלאט ברענגט א משל: מען קען טראכטן פון MCP ווי פון א USB-C פארט פאר AI אפליקאציעס. פונקט ווי USB-C גיט א סטאנדארטן וועג צו פארבינדן מכשירים מיט פארשידענע צוגעהערן, אזוי גיט MCP א סטאנדארטן וועג צו פארבינדן AI מאדעלן מיט פארשידענע דאטא-מקורות און געצייג.

אונטער "Building with MCP" זענען דא צוויי קארטלעך: "MCP Documentation" און "MCP in Claude Desktop". אין דער האלבער מינוט נאכדעם ווערן די צוויי פאראגראפן פון דער דעפיניציע אנגעצייכנט.

SDK-ס און דער Python SDK אויף GitHub (3:49:00)

אויפן בלאט modelcontextprotocol.io/introduction זעט מען א רשימה פון SDK-ס: Java, Kotlin, Python, Ruby, Swift און TypeScript. אויף דער רעכטער זייט שטייען צוויי אפטיילונגען:

- "For Server Developers" – ווי אזוי צו בויען אן אייגענעם סערווער פאר Claude for Desktop און אנדערע קליענטן
- "For Claude Desktop Users" – ווי אזוי צו ניצן סערווערס וואס זענען שוין פארטיג געבויט

דערנאך עפנט זיך דער GitHub רעפאזיטאריום modelcontextprotocol/python-sdk. אין זיין באשרייבונג שטייט: "The official Python SDK for Model Context Protocol servers and clients". דער רעפאזיטאריום האט 14.3k שטערנדלעך, 1.7k פארקס, 26 ,481 commits בראנטשעס און 30 טעגס, און זיין לייסענס איז MIT. צווישן די פאלדערס און טעקעס זעט מען: docs, examples, src/mcp, tests, CLAUDE.md און pre-commit-config.yaml.

אינסטאלירן דעם פעקל (3:50:00)

אין א Windows cmd פענסטער לויפט די אינסטאלאציע. יעדע שורה זאגט "Requirement already satisfied", דאס הייסט אז דאס פעקל איז שוין דא. די פעקלעך וואס ווערן ארויסגעצייגט זענען אנגעהאנגען אין mcp, למשל click, python-dotenv, httpx, sse-starlette, uvicorn, starlette, pydantic און colorama. אלע ליגן אין Python310 site-packages.

דער README (3:50:30)

דער אינהאלט-צעטל פונעם README.md (900 שורות) ווייזט די הויפט-טעמעס:

- What is MCP?
- Core Concepts: Server, Resources, Tools, Prompts, Images, Context
- Running Your Server: Development Mode, Claude Desktop Integration, Direct Execution, Mounting to an Existing ASGI Server
- Examples: Echo Server, SQLite Explorer
- Advanced Usage: Low-Level Server

דער ערשטער MCP קאוד אין Cursor (3:51:00)

מען שאפט א נייעם פאלדער מיטן נאמען MCPServer און עפנט אים אין Cursor . דארט מאכט מען א טעקע FirstMCP.py, און אין דעם טשעט-פענסטער שטייט אויסגעקליבן דער מאדעל gpt-4.1. דער קאוד זעט אויס אזוי:

```
from mcp.server.fastmcp import FastMCP

mcp = FastMCP("Demo")

@mcp.tool()
```

דער קאוד טוט אזוי:

- די ערשטע שורה ברענגט אריין FastMCP.
- די צווייטע שורה שאפט א סערווער מיטן נאמען "Demo".
- דער דעקאראטאר @mcp.tool() מאכט די פונקציע וואס קומט נאך אים פאר א "געצייג" (tool) פונעם סערווער.

די פונקציע הייסט usd_to_gbp. זי נעמט א צאל (amount א float) און גיט צוריק א float. איר באשרייבונג זאגט אז זי בייט איבער דאלארן אויף ענגלישע פונט. די פונקציע שטעלט EXCHANGE_RATE = 0.79 און גיט

צוריק round(amount * EXCHANGE_RATE, 2), דאס הייסט דעם סכום געמערט מיטן קורס און אפגערונדט אויף צוויי ציפערן נאכן פונקט.

די mcp.json קאנפיגוראציע (3:53:00)

די טעקע mcp.json אונטער .cursor באווייזט זיך ערשט מיט ביישפילן פון אנדערע סערווערס: "crypto-price-server" וואס לויפט דורך node, און "4o-mini-search" וואס לויפט דורך npx און האט אן API_KEY.

דערנאך ווערט די טעקע אויסגערייניגט, און עס בלייבט נאר איין סערווער מיטן נאמען "Demo". זיין "command" איז "python", און זיין "args" ווייזט אויף C:/Users/ravi.kiran/Downloads/FirstMCP.py.

דער טעסט אינעם טשעט (3:54:00)

אינעם טשעט ווערט געפרעגט: "can you convert 100$ to pounds". דער ענטפער ניצט די פונקציע usd_to_gbp און רעכנט אויס: 100 USD × 0.79 = 79 GBP. ער לייגט צו אז מען קען רופן usd_to_gbp(100), און זי וועט צוריקגעבן 79.0. דער עקראן בלייבט אויף דעם ביז 3:55:00.

Cursor איינשטעלונגען און מאדעלן (3:55:30)

מען זעט דעם אנהייב-עקראן פון Cursor מיט די לעצטע פראיעקטן: MCPServer, tic-tac-toe, WebCrawler, FreeAIWebCrawler. נאכדעם עפנט מען Cursor Settings ביי "Indexing & Docs". דארט זעט מען Codebase Indexing און אן אפציע "Index New Folders", וואס אינדעקסירט פאלדערס מיט ווינציגער ווי 50,000 טעקעס. אונטער "Docs" שטייט א דאקומענט מיטן נאמען "Simplilearn Python SDK Demo".

אין א נייעם פראיעקט מיטן נאמען sample עפנט זיך די מאדעל-רשימה: claude-4-sonnet (אויסגעקליבן), claude-3.5-sonnet, o3, gemini-2.5-pro, gemini-2.5-flash און gpt-4.1 . אויבן זענען דא שאלטערס פאר Auto און MAX Mode. ווען מען שרייבט "@s" אינעם טשעט, קומט ארויס דער דאקומענט "Simplilearn Python SDK Demo" און די אפציע "Add new doc".

אינסטאלירן Node.js און Cursor (3:58:30)

אויפן בלאט nodejs.org/en/download זעט מען די באפעלן צו באשטעטיגן די ווערסיעס: v- node דארף ווייזן v22.15.0, און v- npm דארף ווייזן 10.9.2. דערונטער שטייען די אפציעס "Windows Installer (.msi)" און "Standalone Binary (.zip)" פאר x64.

דערנאך באווייזט זיך דער Cursor עקראן "Customize Theme" מיט דריי אפציעס: Cursor Dark, Cursor Dark Midnight און Cursor Dark High Contrast.

צום סוף, אין אן Administrator PowerShell פענסטער, אינסטאלירט chocolatey די פעקלעך vcredist140 און vcredist2015, און הייבט אן אינסטאלירן python313 (3.13.3, 64-bit).

Cursor מיט MCP, און RNN נויראל-נעצן (לויטן עקראן) (4:00:00)

Cursor און זיינע איינשטעלונגען (4:00:00)

אויפן עקראן ווערט געווויזן דער פראגראם Cursor, אן אפגעשטעלטער אנהייב-עקראן מיט דעם נאמען "Cursor", דער פלאן "Free" און א לינק "Go unlimited". עס זענען דא דריי קנעפלעך: "Open project",

"Clone repo" און "Connect via SSH". אויף דער רעכטער זייט שטייט א פענצטער "New chat" מיט "Add context", דעם טעקסט "Plan, search, build anything", און א מאדוס "Agent" מיט דער אויסוואל "Auto".

דערנאך עפנט זיך "Cursor Settings". אויף דער לינקער זייט זענען דא די אפטיילונגען General, Features, Models, Rules, MCP און Beta. אין General זעט מען:

- Account: א קנעפל "Upgrade to Pro", "Manage" און "Log out".
- VS Code Import: מען קען גלייך אריינברענגען אלע extensions, איינשטעלונגען און keybindings, דורך "Import".
- Appearance: לינקס "Open editor settings" (font, auto-save, word wrap) און "Configure keyboard shortcuts"; מען קען ניצן Ctrl+Shift+P פאר דער command palette.
- Privacy mode: אויב ס'איז אנגעצונדן, ווערט קיין קאוד נישט אויפגעהאלטן; אויב נישט, ניצט מען פראמפטס, טעלעמעטריע און קאוד-באזע דאטא כדי צו פארבעסערן Cursor. דאס איז א machine-level איינשטעלונג. אויסגעוויילט איז "Enabled (all code remains private)".

די MCP Servers אפטיילונג (4:01:00)

אין דער אפטיילונג MCP שטייט: "Model Context Protocol is a way to offer new tools to Cursor Agent". עס זענען נאך נישטא קיין MCP סערווערס. מען קען צולייגן א גלאבאלן דורכן קנעפל "Add new global MCP server", אדער זיי איינשטעלן אין cursor/mcp.json./;gt&project-root;lt& פאר א באזונדער פראיעקט.

די ליסטע awesome-mcp-servers אויף GitHub (4:01:30)

אין בראוזער ווערט אריינגעשריבן "awes", און עס קומט ארויס א ליסטע פארשלאגן (awesome cursor rules, awesome mcp, awesome ai github א.א.). מען עפנט דעם בלאט github.com/punkpeye/awesome-mcp-servers. אינעם README שטייט אונטער "What is MCP?": MCP איז אן אפענער פראטאקאל וואס לאזט AI מאדעלן זיכער ארבעטן מיט לאקאלע און ווייטע מקורות דורך סטאנדארטיזירטע סערווערס. די ליסטע קאנצענטרירט זיך אויף סערווערס וואס זענען גרייט פאר פראדוקציע אדער עקספערימענטאל, און וואס פארברייטערן AI מיט צוטריט צו טעקעס, דאטא-באזעס, API פארבינדונגען און אנדערע דינסטן. אונטער "Clients" ווערט רעקאמענדירט awesome-mcp-clients און glama.ai/mcp/clients. דערצו שטייט אז Glama Chat איז א multi-modal AI קליענט מיט MCP שטיצע און AI gateway.

ווייטער אראפ ווערט אנגעצייכנט די קאטעגאריע "Cloud Platforms". דארט שטייען צווישן אנדערע:

- qiniu-mcp-server
- mcp-ipfs
- books-mcp-server
- aws-mcp-server: לאזט AI אסיסטענטן לויפן AWS CLI באפעלן אין א זיכערער Docker סביבה.
- k8s-mcp-server: פאר Kubernetes באפעלן, למשל kubectl, helm, istioctl און argocd.
- alibaba-cloud-ops-mcp-server

פריער אויפן בלאט זעט מען אויך web-search, א סערווער וואס זוכט אויפן וועב דורך Google אן קיין API שליסל, און mcp-server-apple-shortcuts.

Firecrawl און זיין API שליסל (4:03:00)

מען גייט צו firecrawl.dev. דארט שטייט "We handle the hard stuff", מיט דער ערקלערונג: rotating proxies, orchestration, rate limits און js-blocked content. עס זענען דא צוויי פעלדער:

- Crawling: זאמלט ריינע דאטא פון אלע צוגענגליכע אונטער-בלעטער, אפילו אן א sitemap.
- Media Parsing: ליינט אינהאלט פון PDF און docx אויפן וועב.

דערנאך ווערט געוויזן א "Sign Up" פארעם מיט Email און Password. א Google זוכונג "firecrawl api" ברענגט ארויס דעם GitHub repo mendableai/firecrawl, וואס ווערט באשריבן אזוי: "Turn entire websites into LLM-ready...". דאס איז אן API דינסט וואס נעמט א URL, קרייכט דורך דעם בלאט, און פארוואנדלט אים אין markdown אדער סטרוקטורירטע דאטא.

אויף mcp.so ווערט געוויזן דער "Firecrawl MCP Server" פון mendableai, וואס לייגט צו web scraping צו Cursor, Claude און אנדערע LLM קליענטן. דער Server Config איז א JSON מיט "mcpServers", "mcp-server-firecrawl", "command": "npx", "args": ["-y", "firecrawl-mcp"], און א env מיט FIRECRAWL_API_KEY.

אין דעם Firecrawl dashboard, אונטער "API Keys", זעט מען א שליסל מיטן נאמען "Default", וואס האט זיך אנגעהויבן מיט fc- און איז באשאפן געווארן דעם May 6, 2025.

אנווייזונגען צו איינשטעלן Firecrawl אין Cursor (4:06:30)

אויף mcp.so ווערט געוויזן:

- Manual Installation: npm install -g firecrawl-mcp
- Running on Cursor: מען דארף Cursor ווערסיע 0.45.6 אדער העכער. די טריט זענען: עפענען Cursor Settings, גיין צו Features > MCP Servers, און דריקן "+ Add New MCP Server". דארט גיט מען דעם Name "firecrawl-mcp", דעם Type "command", און דעם Command: env FIRECRAWL_API_KEY=your-api-key npx -y firecrawl-mcp.
- פאר Windows, אויב עס זענען דא פראבלעמען, ווערט פארגעשלאגן דער באפעל: cmd /c "set FIRECRAWL_API_KEY=your-api-key && npx -y firecrawl-mcp". דער באפעל ווערט אנגעצייכנט.
- מען דארף פארבייטן your-api-key מיט דעם שליסל פון firecrawl.dev/app/api-keys. נאכן צולייגן דארף מען איבערלאדן די ליסטע; דער Composer Agent וועט דאן ניצן Firecrawl.
- פאר Windsurf לייגט מען צו א ענליכן JSON אין codeium/windsurf/model_config.json/.

די טעקע mcp.json (4:07:30)

אין Cursor ווערט געעפנט די טעקע C:\Users\Honey\.cursor\mcp.json. דער קאוד באשטייט פון "mcpServers", דערין "firecrawl", מיט א "command" וואס איז:

```
"cmd /c set FIRECRAWL_API_KEY=your-api-key && npx -y firecrawl-mcp"
```

דער שליסל אליין איז פארשוומען.

אין cmd ווערט נאכגעקוקט וועלכע ווערסיעס זענען אינסטאלירט: v- npm גיט 10.9.2, און v- node גיט v22.15.0. דערנאך ווערט אריינגעשריבן winget install Schniz.fnm.

Firecrawl ארבעט אין Cursor (4:10:00)

אין MCP Servers איז "firecrawl" אנגעצונדן (גרין). עס ווערן אויסגערעכנט זיינע tools:

- firecrawl_scrape
- firecrawl_map
- firecrawl_crawl
- firecrawl_check_crawl_status
- firecrawl_search
- firecrawl_extract
- firecrawl_deep_research
- firecrawl_generate_llmstxt

אין chat ווערט געגעבן דער פראמפט "Search for Simplilearn on Google". דער אגענט ווייזט "Searched web for "Simplilearn"". דערנאך עפנט זיך א YouTube קאנאל און דער LinkedIn בלאט פון Simplilearn. אינעם Overview שטייט או דאס איז א פלאטפארמע פאר דיגיטאלע לערנונג אין פעלדער ווי Cyber Security, Cloud Computing, Project Management, Digital Marketing און Data Science. אונטער Funding via Crunchbase שטייט Series E, US$ 45M.

Feed Forward נויראל-נעץ (4:12:00)

א סלייד "Feed Forward Neural Network" ערקלערט: אין אוא נעץ גייט די אינפארמאציע נאר פאראויס, פון די input nodes, דורך די hidden layers (אויב עס זענען דא), ביז די output nodes. עס זענען נישטא קיין רינגען אדער לופס. דער דיאגראם ווייזט א Input Layer מיט 3 נאדן, צוויי Hidden Layers מיט 4 נאדן יעדער, און אן Output Layer מיט 2 נאדן. א פארפשוטע פארשטעלונג ווייזט: x input → Input Layers → Hidden Layers → Output Layer → ŷ (Predicted output).

פארוואס Recurrent נויראל-נעץ? (4:12:30)

דער סלייד "Issues in Feed Forward Neural Network" רעכנט אויס דריי פראבלעמען:

- 01: קען נישט באהאנדלען sequential דאטא.
- 02: באטראכט נאר דעם יעצטיגן input.
- 03: קען נישט געדענקען פריערדיגע inputs.

דער קענן-סלייד, "Solution to Feed Forward Neural Network", ווייזט א דיאגראם: x → (B) → h, מיט א לופ C צוריק צו h, און דערנאך (A) → y. דער RNN:

- 01: קען באהאנדלען sequential דאטא.

- 02: באטראכט דעם יעצטיגן input און אויך די פריער באקומענע inputs.
- 03: קען געדענקען פריערדיגע inputs צוליב זיין אינעווייניגסטן זכרון.

אנוועגדונגען פון RNN (4:13:30)

- Image captioning: RNN ווערט גענוצט צו באשרייבן א בילד דורכן אנאליזירן די אקטיוויטעטן וואס זענען דערין. דאס בילד ווייזט א הונט וואס שפרינגט אויף א גרינעם פעלד.
- Time series prediction: יעדער time series פראבלעם, ווי למשל פאראויסזאגן די פרייזן פון סטאקס אין א געוויסן חודש, קען מען לייזן מיט RNN.
- Natural Language Processing: text mining און sentiment analysis קען מען דורכפירן מיט RNN.

RNN: אנוועגדונגען, גראדיענט-פראבלעמען און) LSTMלויטן עקראן) (4:15:00)

אנוועגדונגען פון RNN (4:15:00)

אויפן עקראן ווערט געוויזן א סלייד מיטן טיטל "Applications of RNN". צום ערשט ווייזט ער א בילד פון צוזאמגעשטעלטע פאזל-שטיקלעך מיט די ווערטער NATURAL LANGUAGE PROCESSING, און דערונטער דער נאמען "Natural Language Processing". די באשרייבונג זאגט: Text mining און Sentiment analysis קען מען דורכפירן מיט RNN פאר Natural Language Processing.

דערנאך ((4:15:30 ווייזט דער סלייד די צווייטע אנוועגדונג, "Machine Translation", מיט א בילד פון א טעלעפאן ארום וועלכן עס פליען אומעטום פאנען פון פארשידענע לענדער. די באשרייבונג: וועז מ'גיט אריין אן אינפוט אין איין שפראך, קען מען ניצן RNN צו איבערזעצן דעם אינפוט אין אנדערע שפראכן אלס אויספיר. ביי 4:16:30 קומט צו א באמערקונג: דא רעדט א מענטש אויף ענגליש, און עס ווערט איבערגעזעצט אויף כינעזיש, איטאליעניש, פראנצויזיש, דייטש און שפאניש.

ווי ארבעט א RNN (4:17:00)

דער סלייד "How does a RNN work?" ווייזט א דיאגראם פון דריי רונדע קנופן איינס נאך דאס צווייטע, איינס פאר יעדן צייט-שריט: t-1, t און t+1. אין יעדן קנופ קומט אריין פון אונטן אן אינפוט x(t-1), x(t), x(t+1), דורך א פייל מיטן צייכן B. פון יעדן קנופ גייט ארויף אן אויספיר y(t-1), y(t), y(t+1) , דורך א פייל מיטן צייכן A. ביי יעדן קנופ שטייט זיין באהאלטענער צושטאנד h(t-1), h(t), h(t+1), און צווישן די קנופן לויפט א האריזאנטאלע פייל מיטן צייכן C , וואס פירט פון איין שריט צום קומענדיגן. אונטן איז זעבאר די דעפיניציע: input vector at time step t = x(t). דער עקראן בלייבט ביז 4:17:30.

סארטן פון Recurrent Neural Network (4:18:00)

דער סלייד "Types of Recurrent Neural Network" ווייזט צום ערשט דעם סארט "one to one": א קאלאנע פון דריי בלאקן (אינפוט, א מיטלסטער שיכט, און "Single output"). די באשרייבונג זאגט אז א one to one נעץ ווערט אנגערופן Vanilla Neural Network, און מ'ניצט עס פאר געווענליכע מאשין-לערנען פראבלעמען.

ביי 4:18:30 ווערט געוויזן "many to one": עטליכע אינפוטס אונטן, דריי מיטלסטע בלאקן פארבונדן איינס צום צווייטן, און איין "Single output" אויבן. די באשרייבונג: א many to one נעץ נעמט אריין א סעריע

אינפוטס. א ביישפיל: Sentiment analysis, וואו מ'קען קלאסיפיצירן א געגעבענעם זאץ צי ער דריקט אויס פאזיטיוועס אדער נעגאטיוועס געפילן.

Vanishing Gradient און Exploding Gradient (4:19:00)

דער סלייד "Vanishing Gradient Problem" ווייזט א גראף מיט אן X און Y אקס, מיט א גלייכער שיפע ליניע. דארט זענען אנגעצייכנט "Change in X" און "Change in Y", וואס צוזאמען באשטימען דעם שיפוע (slope). דער טעקסט זאגט: ביים טריינירן א RNN קען דער שיפוע זיין צו קליין אדער זייער גרויס, און דאס מאכט דאס טריינירן שווער. וועז דער שיפוע איז צו קליין, הייסט די פראבלעם Vanishing gradient. אונטן ווייזט זיך א קייט פון פיר קנופן s0, s1, s2, s3 מיט אינפוטס x0 ביז x3 און טעותים E0 ביז E3. רויטע פיילן גייען צוריק פון s3 ביז s0, און איבער דער קייט שטייט א גרויסע פייל: "Loss of Information through time".

ביי 4:19:30 קומט דער סלייד "Exploding Gradient Problem". דער גראף ווייזט א ליניע וואס ווערט שטארק און שנעל אלץ העכער. דער טעקסט: וועז דער שיפוע וואקסט עקספאנענציעל אנשטאט צו פארקלענערן זיך, הייסט די פראבלעם Exploding gradient. ווידער ווערט געווייזן די קייט s0–s3 מיט "Loss of Information through time", און אונטן "Backpropagate the error". נעבן א בילד פון פיילן אין א ציל-ברעט שטייט "Issues in Gradient Problem":

- Long training time (לאנגע צייט פון טריינירן)
- Poor performance (שוואכע ארבעט)
- Bad accuracy (שלעכטע געניקייט)

דער עקראן בלייבט ביז 4:20:00.

ערקלערונג פון דער גראדיענט-פראבלעם (4:20:30)

דער סלייד "Explaining Gradient Problem" ברענגט צוויי ביישפילן כדי צו פארשטיין וואס זאל זיין דאס קומענדיגע וואָרט אין דער סעריע:

- "The person who took my bike and…, _ a thief."
- "The students who got into Engineering with…, _ from Asia."

אונטן איז א קייט פון קנופן s1, s2, s3 ביז st, וואס הייבט זיך אן ביי s0, מיט אינפוטס x1…xt און אויספירן y1…yt.

ביי 4:21:00 ווערט אין ערשטן זאץ באצייכנט דאס וואָרט "person" און די ליידיגע שטעל. א קעסטל זאגט: כדי צו פארשטיין וואס זאל זיין דאס קומענדיגע וואָרט, מוז דער RNN געדענקען דעם פריערדיגן קאנטעקסט, צי דער סוביעקט איז געווען א יחיד-נאמען אדער א רבים-נאמען. רויטע פיילן גייען צוריק פון st צו s1 . ביי 4:21:30 ווערט באצייכנט דער צווייטער זאץ: "students" און די ענטפער "were" ("were from Asia…"). א באמערקונג זייטווייז זאגט אז ס'קען אמאל זיין שווער פאר דעם טעות צו backpropagate ביז צום אנהייב.

לייזונגען צו דער גראדיענט-פראבלעם (4:22:00)

דער סלייד "Solution to Gradient Problem" ווייזט צוויי האלב-רינגען, יעדער מיט דריי לייזונגען:

פאר Exploding Gradient:

- 1. Identity Initialization
- 2. Truncated Backpropagation
- 3. Gradient Clipping

פאר Vanishing Gradient:

- 1. Weight initialization
- 2. Choosing the right Activation Function (אויסקלייבן די ריכטיגע אקטיוואציע-פונקציע)
- 3. Long Short-Term Memory Networks (LSTMs), וואס איז באזונדער ארויסגעהויבן מיט א ראם.

דער עקראן בלייבט אויף דעם ביז 4:23:00.

לאנג-טערמינדיגע אפהענגיגקייטן (4:23:30)

דער סלייד "Long-Term Dependencies" שטעלט פאר: לאמיר זאגן מיר פרובירן פאראויסצוזאגן דאס לעצטע ווארט אין טעקסט: "I have been staying in Spain for the last 10 years…I can speak fluent Spanish" נעבן דעם א בילד מיטן ווארט "España" אין זאמד. דער טעקסט זאגט: דאס ווארט וואס מיר זאגן פאראויס וועט אפהענגען פון די פריערדיגע פאר וועריאבלען קאנטעקסט. אין א קעסטל אונטן: דא דארף מען דעם קאנטעקסט פון Spain כדי פאראויסצוזאגן דאס לעצטע ווארט, און עס איז מעגליך אז דער אפשטאנד צווישן דער רעלעוואנטער אינפארמאציע און דעם פונקט וואו מ'דארף זי… (דער רעשט פונעם זאץ איז נישט זעבאר).

Long Short-Term Memory Networks (4:24:00)

דער סלייד "Long Short-Term Memory Networks" זאגט אויבן: LSTMs זענען א ספעציעלער סארט Recurrent Neural Networks, וואס קענען אויסלערנען לאנג-טערמינדיגע אפהענגיגקייטן; געדענקען אינפארמאציע פאר לאנגע צייטן איז זייער נאטירליכע אויפפירונג. דערנאך: אלע recurrent נויראל-נעצן האבן די פארעם פון א קייט פון איבערגעחזרטע מאדולן; אין א סטאנדארט RNN האט דער מאדול א זייער פשוטע סטרוקטור, ווי למשל אין איינציגע tanh שיכט. דער דיאגראם וויזט דריי בלאקן A, וואס דער מיטלסטער האט אין זיך אין tanh, מיט אויספירן h(t-1), h(t), h(t+1).

ביי 4:24:30 טוישט זיך דער טעקסט: LSTMs האבן אויך א קייט-סטרוקטור, אבער דער איבערגעחזרטער מאדול האט אן אנדערע סטרוקטור; אנשטאט איין שיכט פון נויראל-נעץ זענען דא פיר שיכטן וואס ווירקן איינס אויפן צווייטן אויף א גאנץ ספעציעלן אופן. דער מיטלסטער בלאק וויזט דריי σ (sigmoid) שיכטן און איין tanh שיכט, מיט קנופן פון פארמערן (×) און צולייגן (+), און נאך א tanh. דער עקראן בלייבט ביז 4:25:30.

ביי 4:26:00 שטייט "3 step process of LSTMs": א ראם צווישן C(t-1) און C(t), צעטיילט אויף דריי שריט. דער ערשטער שריט: "Forget irrelevant parts of previous state" (פארגעסן די נישט-רעלעוואנטע טיילן פונעם פריערדיגן צושטאנד). ביי 4:26:30 דער צווייטער שריט: "Selectively update cell state values" (אויסוועליג דערהיינטיגן די ווערטן פונעם צעל-צושטאנד).

ווי ארבעט LSTM: שריט 1 (4:27:00)

דער סלייד "Working of LSTMs", Step-1: "Decides how much of the past it should remember" (באשליסט וויפיל פון דער פארגאנגענהייט צו געדענקען). דער טעקסט: דער ערשטער שריט אין LSTM איז צו באשליסן וועלכע אינפארמאציע ארויסצולאזן פון דער צעל אין יענעם צייט-שריט; דאס

באשליסט די sigmoid פונקציע. זי קוקט אויפן פריערדיגן צושטאנד (h(t-1 און אויפן איצטיגן אינפוט (x(t און רעכנט אויס די פונקציע. די פארמל:

$$f_t = \sigma(W_f \cdot [x_t, h_{(t-1)}] + b_f)$$

וואו f_t = forget gate. דער דיאגראם ווייזט (h(t-1 וואס גייט איריין אין א σ, און ארויס קומט f_t.

ביי 4:27:30 ווערט געגעבן א ביישפיל: מ'גיט אריין אין LSTM אינפוטס פון פריערדיגן און איצטיגן צייט-שריט:

- Previous Output (h(t-1)): "Alice is good in Physics. John on the other hand is good in Chemistry."
- Current Input (x(t)): "John plays football well. He told me yesterday over phone that he had served as the captain of his college football team."

ביי 4:28:00 קומט צו א שורה: "The next sentence talks about John, so the information..." (דער רעשט איז נישט זעבאר).

ווי ארבעט LSTM: שריט 2 (4:28:30)

Step-2: "Decides how much should this unit add to the current state" (באשליסט וויפיל די איינהייט זאל צולייגן צום איצטיגן צושטאנד). דער טעקסט: אין דער צווייטער שיכט זענען דא צוויי טיילן, איינס איז די sigmoid פונקציע און די אנדערע די tanh. די sigmoid פונקציע באשליסט וועלכע ווערטן דורכצולאזן (0 אדער 1); די tanh פונקציע גיט א וואג פאר די ווערטן וואס וועלן דורכגעלאזט, און באשליסט זייער מדרגה פון וויכטיגקייט (פון -1 ביז 1). די פארמלען:

$$i_t = \sigma(W_i \cdot [x_t, h_{(t-1)}] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [x_t, h_{(t-1)}] + b_C)$$

וואו i_t = input gate. דער דיאגראם ווייזט (h(t-1 וואס גייט איריין אין א σ (ארויס i_t) און אין א tanh (ארויס $\tilde{C}_t$), און די צוויי וועלן גענמערט. דער עקראן בלייבט אויף דעם ביז 4:29:30.

LSTM: פון טויערן ביז אקציע-פרייזן (לויטן עקראן) (4:30:00)

דער input gate און דער קראנטער אינפוט (4:30:00)

אויפן עקראן ווערט געווייזן א סלייד מיטן טיטל "Working of LSTMs". אויבן שטייט: "Consider the current input at x_t". אין א פונקטירטן קעסטל שטייט א ביישפיל-זאץ: "John plays football well. He told me yesterday over phone that he had served as the captain of his college football team." א פייל ווייזט פונעם זאץ צו א קעסטל "Current Input".

אין א וואלקן-בלאז אויפן סלייד שטייט: "This process of adding some new information can be done via the input gate."

אויף דער רעכטער זייט זעט מען א חלק פונעם LSTM-דיאגראם: די ליניע _{t-1}h קומט אריין, און דערפון גייען ארויף צוויי קעסטלעך, איינס מיט σ (סיגמאָיד) וואס גיט i_t, און איינס מיט tanh וואס גיט C̃_t. די צוויי ווערן צוזאמגעמולטיפליצירט ביים צייכן ×. דער סלייד בלייבט אויפן עקראן א האלבע מינוט.

דריטער שריט: דער output gate (4:31:00)

דער קומענדיגער סלייד, ווידער "Working of LSTMs", האט אן אויפשריפט "Step-3" מיטן טעקסט: "Decides what part of the current cell state makes it to the output".

אונטן ווערט עס ערקלערט: דער דריטער שריט באשליסט וואס וועט זיין דער אויסגאנג. צוערשט לויפט א סיגמאָיד-שיכט וואס באשליסט וועלכע טיילן פונעם cell state קומען אריין אינעם אויסגאנג. דערנאך לאזט מען דעם cell state דורך tanh, כדי די ווערטן זאלן זיין צווישן 1- און 1, און מען מולטיפליצירט עס מיטן אויסגאנג פונעם סיגמאָיד-טויער.

די פארמולעס אויפן סלייד:

- $o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$
- $h_t = o_t * \tanh(C_t)$

אינעם דיאגראם זעט מען: פון h_{t-1} גייט מען דורך σ צו o_t; דער C_t cell state גייט דורך tanh; ביידע ווערן געמולטיפליצירט (×), און דאס גיט h_t , וואס גייט ארויס אויף רעכטס און אויך ארויף אין א רויטן קרייז h_t.

ביישפיל: פאָרויסזאָגן דאָס קומענדיגע וואָרט (4:31:30)

דער סלייד שרייבט: "Let's consider this example to predicting the next word in the sentence:" און דער זאץ: "John played tremendously well against the opponent and won for his team. For his contributions, brave was awarded player of the match".

אין וואלקן-בלאז שטייט צוערשט: "There could be a lot of choices for the empty space". א האלבע מינוט שפעטער ((4:32:00 טוישט זיך דער טעקסט: "'John could be the best output after brave". דאס זעלבע דיאגראם פונעם output gate בלייבט אויף רעכטס.

א פּראַקטישער פאל: אקציע-פרייזן מיט LSTM (4:32:30)

א נייער סלייד מיטן טיטל "Use case implementation of LSTM". אויבן שטייט: "Let's predict the prices of stocks using LSTM network". אויף לינקס, אין א גרינעם אוואל: "Based on the stock price data between 2012-2016"; א בילד פון א דאטא-באזע מיט א פייל ווייזט צו א בילד פון א גראפיק מיט אויפשטייגנדיגע סלופעס, און דערונטער: "Predict the stock prices of 2017". דער סלייד בלייבט ביז 4:33:00.

דערנאך ((4:33:30 ווייזט דער זעלבער טיטל דריי נומערירטע שריט מיט קאוד:

- 1. Import the Libraries: מען אימפארטירט numpy אלס np, matplotlib.pyplot אלס plt, און pandas אלס pd.

- 2. Import the training dataset: מען ליינט איין דעם טעקע home/ubuntu/Downloads// Google_Stock_Price_Train.csv דורך pd.read_csv אין dataset_train, און training_set באקומט דורך iloc[:, 1:2].values איין קאלום פונעם טאבעל.
- 3. Feature Scaling: מען אימפארטירט MinMaxScaler פון sklearn.preprocessing און שאפט sc = MinMaxScaler(feature_range = (0, 1)). די וויטערדיגע שורה איז צוגעדעקט.

(4:34:00) Anaconda Navigator

אויפן עקראן ווערט געעפנט Anaconda Navigator, אויף דער בלאט "Environments". אין דער רשימה פון סביבות זעט מען (anaconda3, kerasypython36בערך), python35 און python36. אויף רעכטס שטייט א רשימה פון אינסטאלירטע פעקלעך מיט באשרייבונג און ווערסיע, למשל: bleach (Easy whitelist-based HTML-sanitizing tool, 2.1.3), certifi (A Python package for providing mozilla's ca bundle),) colorama (Cross-platform colored terminal text in python, 0.3.9), decorator (Better living through python with decorators, 4.3.0), html5lib, ipykernel (IPython Kernel for Jupyter functionality,) 5.2.3), jupyter_core) 2.6.0), jupyter_client) 2.10), jsonschema) jinja2 (A small but fast and easy to use stand-alone template engine written in pure python), 4.4.0), libsodium. דער עקראן בלייבט א מינוט.

Jupyter: די טעקעס פונעם פּראָיעקט (4:35:00)

אין Jupyter זעט מען דעם פאלדער Documents / DocsBusiness / Simplilearn_ / marketing_videos_ / RNN_Tutorial-Recurrent Neural Network / 2018_06-08. דארט ליגן: Google stock price prediction.ipynb) Running, 43.7 kB), RNN-Stock.ipynb) Running, 555 B),) Google_Stock_Price_Test.csv) 1.03 kB), Google_Stock_Price_Train.csv) 63.5 kB), עטלעכע Recording #53.mp4–49 טעקעס, RNN ppt.pdf און RNN ppt.pptx.

דער ערשטער קאוד-צעל און א סקיצע פונעם פּלאַן (4:35:30)

דער נאָוטבוק RNN-Stock ווערט געעפנט (Python 3). דער ערשטער צעל טראגט די באמערקונג "# Importing the libraries" מיט די דריי אימפארטן: numpy אלס np, matplotlib.pyplot אלס plt, pandas אלס pd. ביי 4:36:00 איז דער צעל שוין אויסגעפירט (In [1]) און א נייער ליידיגער צעל עפנט זיך.

איבערן עקראן ווערט מיט דער האנט געצייכנט א סקיצע: גרויסע געוווּנדענע קלאמערן, מיט די ווערטער "data Prep" ביי דער ערשטער גרופּע, א "1/2" אויף דער זייט, דערנאך "karas" (Keras) ביי דער מיטלסטער גרופּע, און ביי דער לעצטער א ווארט וואס הייבט זיך אן מיט "evalu..." (וואו עס איז אפגעשניטן). ביי 4:37:30 איז די סקיצע אפגעווישט.

דער CSV-טעקע פון Google (4:38:00)

אין Windows File Explorer זעט מען דעם זעלבן פאלדער מיט די טעקעס, צווישן זיי Google_Stock_Price_Test.csv און Google_Stock_Price_Train.csv. דער Train -טעקע ווערט געעפנט אין WordPad. די ערשטע שורה איז די קעפלעך: Date, Open, High, Low, Close, Volume. דערנאך קומען טעגלעכע שורות, למשל:

- 1/3/2012, 325.25, 332.83, 324.97, 663.59, "7,380,500"

- 1/4/2012, 331.27, 333.87, 329.08, 666.45, "5,749,400"
- 1/5/2012, 329.83, 330.75, 326.89, 657.21, "6,590,300"

און אזוי ווייטער דורך יאנואר און פעברואר 2012. ביי 4:40:00 ווערט דער טעקע נאכאמאל געוויזן, מיטן קורסאר ביי די קעפּלעך און די ערשטע דאטעס.

איינלייענען דעם טרייניר-סעט (4:39:00)

אין צווייטן צעל, "# Importing the training set":

pd.read_csv('Google_Stock_Price_Train.csv') = dataset_train

dataset_train.iloc[:, 1:2].values = training_set

דאָ ווערט דער טעקע גלייך פונעם לאקאלן פאלדער געלייענט, און iloc[:, 1:2] נעמט אלע שורות פון קאלום נומער 1, וואס איז דער Open-קאלום. ביי 4:40:30 איז דער צעל אויסגעפירט (In [2]), מיט 1:2 אונטערגעשטראכן.

Feature Scaling (4:41:00)

דער דריטער צעל, "# Feature Scaling":

- from sklearn.preprocessing import MinMaxScaler
- sc = MinMaxScaler(feature_range = (0, 1)) – שאפט א סקאלירער וואס ברענגט די ווערטן צווישן 0 און 1.
- training_set_scaled = sc.fit_transform(training_set) – פאסט אן דעם סקאלירער צו די דאטא און פארוואנדלט זיי.

ביי 4:41:30 ווערט מיט דער האנט געצייכנט א קליינע גראפיק: אויבן "100 Max", אונטן "5 Min", מיט א פייל אראפ. ביי 4:42:00 איז דאס ווארט "scaled" אין training_set_scaled אונטערגעשטראכן.

א דאטא-סטרוקטור מיט 60 צייט-שריט (4:42:30)

דער פערטער צעל, "# Creating a data structure with 60 timesteps and 1 output":

- מען שאפט צוויי ליידיגע רשימות X_train און y_train.
- א שלייף for i in range(60, 1258): ביי יעדן i לייגט מען צו X_train די 60 פריערדיגע ווערטן [training_set_scaled[i-60:i, 0], און צו y_train דעם ווערט fun טאג i גופא, training_set_scaled[i, 0].
- דערנאך ווערן ביידע פארוואנדלט אין np.array.
- אונטער "# Reshaping": X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1)).

ביי 4:43:00 ווערט "60 timesteps" אונטערגעשטראכן, מיט פיילן ביי range(60, ...) און ביי i-60 . ביי 4:43:30 ווערט די שורה פון X_train.append באצייכנט. ביי 4:44:30 ווערט צוגעצייכנט "0-59" ביי dem חלק i-60:i, א קרייזל ארום דעם i אין y_train, און א לאנגער שמאלער רעקטאנגל אויבן, ווי א בילד פון א רייע ווערטן.

דער דאטא-סטרוקטור מיט 60 טריט (4:45:00)

אויפן עקראן ווערט געוויזן א Jupyter notebook (Python 3), וואו די ערשטע פיר צעלן זענען שוין אויסגעשטעלט. אין צעל 1 ווערן אריינגעברענגט די ביבליאטעקן: numpy אלס np, matplotlib.pyplot אלס plt, און pandas אלס pd.

אין צעל 2 ווערט געלייענט דער טרייניר-דאטא-סעט פון דער טעקע Google_Stock_Price_Train.csv מיט pd.read_csv. דערנאך ווערט מיט dataset_train.iloc[:, 1:2].values ארויסגענומען בלויז איין קאלום. דאס הייסט training_set.

אין צעל 3 ("Feature Scaling") ווערט אריינגעברענגט MinMaxScaler פון sklearn.preprocessing. מען שאפט sc = MinMaxScaler(feature_range = (0, 1)), און training_set_scaled = sc.fit_transform(training_set) נארמאליזירט די ווערטן צווישן 0 און 1.

צעל 4 טראגט דעם קאמענטאר "Creating a data structure with 60 timesteps and 1 output". מען שאפט צוויי ליידיגע ליסטעס X_train און y_train. א לופ גייט איבער i in range(60, 1258). ביי יעדן טריט ווערט צו X_train צוגעלייגט training_set_scaled[i-60:i, 0], דאס זענען די פריערדיגע 60 ווערטן. צו y_train ווערט צוגעלייגט training_set_scaled[i, 0], דאס איז דער איינציגער ווערט וואס קומט גלייך נאך זיי. דערנאך ווערן ביידע פארוואנדלט אין np.array.

אויפן עקראן איז אויך אנגעצייכנט מיט דער האנט א שטרייף פון 0 ביז 59, און דערנעבן 60. דאס ווייזט אז די 60 ווערטן פון 0 ביז 59 דינען אלס אריינגאנג, און דער ווערט ביי 60 איז דער ציל. מען זעט אויך די צאל 1258, א פייל צו "0.54", און מארקירונגען אויף [i-60:i, 0] און [i, 0].

Reshaping (4:46:00)

אונטער "Reshaping" שטייט די שורה:

np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1)) = X_train

X_train.shape[0] און X_train.shape[1] זענען אונטערגעשטראכן. דערנעבן איז אנגעצייכנט א פירעק מיט "60" אויבן, וואס שטעלט פאר א 3-דימענסיאנעלן אריינגאנג. ביי 4:47:00 איז צעל 4 שוין אויסגעפירט (In [4]).

Keras ביבליאטעקן (4:47:30)

אין צעל 5 ("Importing the Keras libraries and packages") ווערן אריינגעברענגט Sequential פון keras.models, און Dense, LSTM און Dropout פון keras.layers. דער רעזולטאט ווייזט א FutureWarning פון h5py: "Conversion of the second argument of issubdtype from float to np.floating is deprecated". נאך דעם שטייט "Using TensorFlow backend".

בויען דעם RNN (4:49:00)

- צעל 6, "Initialising the RNN": regressor = Sequential().

- צעל 7, "Adding the first LSTM layer and some Dropout regularisation": regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1],)) אויפן עקראן זענען regressor.add(Dropout(0.2)). און דערנאך ((1), און Dropout(0.2) באצייכנט מיט פיילן. נאך א פייל ציט זיך פון דער ערשטער שיכט צו דער צווייטער.
- צעלן 8 און 9 לייגן צו א צווייטע און א דריטע LSTM-שיכט, יעדע מיט units = 50 און return_sequences = True, און יעדע מיט Dropout(0.2).
- צעל 10 לייגט צו א פערטע LSTM-שיכט: LSTM(units = 50), אן return_sequences , אויך מיט Dropout(0.2).
- צעל 11, "Adding the output layer": regressor.add(Dense(units = 1)).
- צעל 12, "Compiling the RNN": regressor.compile(optimizer = 'adam', loss = 'mean_squared_error').

דער עקראן בלייבט אויף די צעלן עטלעכע מינוט.

טריינירן דעם מאדעל (4:55:00)

צעל 13, "Fitting the RNN to the Training set":

regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)

דער רעזולטאט ווייזט יעדן Epoch מיט 1198/1198 שטיקלעך:

- Epoch 1/100: בערך 13 סעקונדעס, loss: 0.0444.
- Epoch 2/100: loss: 0.0059.
- ביי די לעצטע עפאָכס (93 ביז 100) ליגט דער loss צווישן 0.0014 און 0.0017, בערך 10 ms/step.
- דער לעצטער Epoch, 100/100, ענדיגט מיט loss: 0.0014.

דער אויסגאנג Out[13] ווייזט <keras.callbacks.History at 0x2dd92cc9048>.

Part 3: פאָרויסזאגן און ווייזן די רעזולטאטן (4:57:00)

צעל 14 טראגט דעם טיטל "Making the predictions and visualising the results - Part 3". אונטער "Getting the real stock price of 2017" ווערט געלייענט Google_Stock_Price_Test.csv אין dataset_test. דערנאך גיט dataset_test.iloc[:, 1:2].values = real_stock_price די אמת'ע פרייזן.

אויך ווערט געוויזן א סלייד, "Use case implementation of LSTM", מיט צוויי טריט:

- טריט 11, "Load the stock price test data for 2017": דער זעלבער קאוד, מיט דעם פולן וועג '/home/ubuntu/Downloads/Google_Stock_Price_Test.csv'.
- טריט 12, "Get the predicted stock price of 2017": דער קאוד וואס ווערט דא באשריבן ווייטער.

דער קאוד פאר דער פאָרויסזאגונג ("Getting the predicted stock price of 2017") גייט אזוי:

- פארבינדט דעם dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']), axis = 0) 'Open' קאלום פון ביידע טעקעס.

- inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values נעמט די לעצטע 60 טעג פון טריינירן צוואמען מיט די טעסט-טעג.
- inputs.reshape(-1,1) און דערנאך sc.transform(inputs) גרייטן צו די ווערטן.
- א לופ for i in range(60, 80) לייגט צו X_test.append(inputs[i-60:i, 0]).
- דערנאך X_test = np.array(X_test) און np.reshape צו (X_test.shape[0], X_test.shape[1], 1).
- predicted_stock_price = regressor.predict(X_test) מאכט די פאָרויסזאגונג.
- sc.inverse_transform(predicted_stock_price) קערט צוריק די פרייזן צו זייער ריכטיגער גרויס.

אויפן עקראן ווערט אנגעצייכנט ווי די 60 טעג פון טריינירן-דאטא ("Tr") שטייען פאר די טעסט-דאטא. אויך ווערט אונטערגעשטראכן dataset_train['Open'] און 1:2.

פון אקציע-פרייזן צו Keras און נויראל-נעצן (לויטן עקראן) (5:00:00)

דער סוף פונעם טריינירן און די טעסט-דאטא (5:00:00)

אויפן עקראן ווערט גענויזן א Jupyter notebook מיטן נאמען RNN-Stock.ipynb, אין א טעקע פון א "Recurrent Neural Network" טוטאריאל. אויבן זעט מען דעם סוף פונעם טריינירן: Epoch 95/100 ביז Epoch 100/100. יעדע עפאך גייט דורך 1198/1198 שריט, נעמט בערך 11 ביז 13 סעקונדעס (9 ביז 11 ms/step), און דער loss שטייט צווישן 0.0014 און 0.0017. ביי דער לעצטער עפאך איז דער loss 0.0014. דער אויסגאנג פון דער צעל איז א keras.callbacks.History אביעקט.

די קומענדיגע צעל (In [15]) הייסט "Making the predictions and visualising the results - Part 3". אונטער דער באמערקונג "Getting the real stock price of 2017" לייענט דער קאוד מיט pd.read_csv די טעקע Google_Stock_Price_Test.csv אריין אין dataset_test, און נעמט ארויס די צווייטע קאלאנע (:,1)iloc.values אלס real_stock_price.

אויסרעכענען די פארויסגעזאגטע פרייזן (5:00:00–5:01:30)

די צעל "Getting the predicted stock price of 2017" (שפעטער In [16]) טוט די פאלגנדע שריט:

- מיט pd.concat ווערן צוואמגעשטעלט די 'Open' קאלאנעס פון dataset_train און dataset_test (axis = 0) אין dataset_total.
- inputs באקומט פון dataset_total די לעצטע רייען, אנהייבנדיג פון len(dataset_total) - len(dataset_test) - 60, ד"ה די 60 טעג פאר דער טעסט-תקופה מיט דער טעסט-תקופה אליין. די שורה ווערט אויפן עקראן אנגעצייכנט.
- inputs ווערט איבערגעפורעמט מיט reshape(-1,1) און נארמאליזירט מיט sc.transform.
- א ליידיגע ליסטע X_test ווערט געשאפן; א לופ for i in range(60, 80) לייגט צו יעדעס מאל א שטיק פון 60 ווערטן, inputs[i-60:i, 0].
- X_test ווערט פארוואנדלט אין א np.array און איבערגעפורעמט מיט np.reshape צו דער פארעם (X_test.shape[0], X_test.shape[1], 1).

- regressor.predict(X_test) גיט predicted_stock_price, און sc.inverse_transform דרייט עס צוריק צו אמת'דיגע פרייז-ווערטן.

דער גראף: אמת'דיגער קעגן פארויסגעזאגטן פרייז (5:01:30–5:05:00)

די צעל "Visualising the results" (In [17]) צייכנט מיט plt.plot צוויי ליניעס: real_stock_price אין רויט מיטן לעיבל 'Real Google Stock Price', און predicted_stock_price אין בלוי מיטן לעיבל 'Predicted Google Stock Price'. דערנאך קומען plt.title('Google Stock Price Prediction'), plt.xlabel('Time'), plt.ylabel('Google Stock Price'), plt.legend() און plt.show(). אויפן עקראן ווערן די שורות מיט פיילן און אונטערשטרייכונגען אנגעצייכנט.

דער אויסגאנג איז א גראף מיטן טיטל "Google Stock Price Prediction". די Y-אקס גייט פון בערך 780 ביז 840; די X-אקס האט ציפערן ווי 12.5, 15.0, 17.5. ביידע ליניעס גייען ארויף פון ארום 790–780 ביז ארום 810–800, שטייען כמעט גלייך א שטיק צייט, דערנאך שפרינגען זיי ארויף. די אמת'דיגע ליניע דערגרייכט ארום 840–835 און פאלט שארף אראפ צו ארום 795, און די פארויסגעזאגטע ליניע גייט נאכדעם און דערגרייכט ארום 825 א ביסל שפעטער. אויפן עקראן ווערט ארומגערינגלט די לעגענדע, ארויסגעצייכנט די ענלעכע פארעם פון ביידע ליניעס, און מיט א שטריך אנגעצייכנט דער פונקט וואו די אמת'דיגע ליניע הייבט אן צו פאלן. דער גראף בלייבט אויפן עקראן עטליכע מינוט.

וואס איז Keras? (5:05:30)

א נייער סלייד, "What is Keras?", זאגט: Keras איז א high-level דיפ-לערנען API, געשריבן אין python, כדי לייכט צו בויען נויראל-נעצן. עס באנוצט דיפ-לערנען פריימווערקס ווי Tensorflow און Pytorch אלס backend, כדי די חשבונות זאלן גיין שנעלער. די דיאגראם צייגט Keras אלס "Frontend", און פיילן צו די backends: Microsoft Cognitive Toolkit, mxnet, plaidML, TensorFlow (די ארענדזש לאגא) און theano.

דער צווייטער סלייד מיטן זעלבן טיטל (5:06:00) זאגט: Keras ארבעט דורכן באנוצן קאמפליצירטע דיפ-לערנען פריימווערקס ווי tensorflow, pytorch, mlplaid אלס backend פאר שנעלע חשבונות, און גיט דערביי א פריינטליכע און לייכט-צו-לערנען frontend. א דיאגראם צייגט שיכטן: אויבן "Keras API Spec", אין מיטן "TF- Keras", "Theano-Keras" און "....", און אונטן "TensorFlow Workflow".

ווי Keras ארבעט (5:07:00)

דער סלייד "Working principle of Keras" זאגט אז Keras באנוצט computational graphs (חשבון-גראפן) כדי אויסצודריקן און אויסצורעכענען מאטעמאטישע אויסדרוקן. די פונקטן:

- 1. אויסדריקן קאמפליצירטע פראבלעמען אלס א צוזאמשטעל פון פשוט'ע מאטעמאטישע אפעראטארן.
- 2. ניצלעך צו רעכענען דעריוואטיוון דורך backpropagation.
- 3. גרינגער צו מאכן צעטיילטע (distributed) חשבונות.
- 4. ביים לייזן קאמפליצירטע פראבלעמען, באשטימט מען input און outputs און מען פארזיכערט אז אלע נאדס זענען פארבונדן.

די דיאגראם: די Variables)אדער Inputs) זענען X, Y און 2; Y קומט פון דעם Operator X^X .0.3 און Y גייען אריין אין א "%" נאד, Y און 2 אין א "X" (פארמערן) נאד, און ביידע פירן צו א "-" נאד אויבן, וואס איז דער Output / Dataflow.

Keras מאדעלן (5:08:30)

דער סלייד "Keras Model", פונקט 1: Sequential Model. א Sequential מאדעל איז א גלייכע רייע פון שיכטן, וואו יעדע פריערדיגע שיכט פירט אריין אין דער קומענדיגער; ניצלעך פאר פשוט'ע classifier אדער decoder מאדעלן. דער קאוד-ביישפיל: model = keras.Sequential מיט layers.Dense(1, "relu"activation= ,("layer1"name= ,"relu"layers.Dense(2, activation= ,("layer2"name= ") און ("layer3"Dense(3, name= ; דערנאך ווערט דער מאדעל אויסגעפרואווט אויף א טעסט-אינפוט x = (3, 3))tf.ones(. די דיאגראם צייגט 2) Input Layer נאדס), 4) Layer 1 נאדס), 3) Layer 2 נאדס) און 1) Output layer נאד).

פונקט Functional Model :(5:09:30) 2, א מאדעל מיט כמה אינפוטס און כמה אויטפוטס; א קאמפליצירטער מאדעל וואס צעצווייגט זיך אין צווי אדער מער צווייגן. דער קאוד: keras.Input(shape=(32, = img_inputs (32, 32, 3)) וואס ווערט געווענדט אויף inputs, נאך א Dense(64, activation= "relu"layers.Dense(64) אויטפוטס פון layers.Dense(10)(x), און keras.Model(inputs=inputs, "relu"activation= "mnist_model"outputs=outputs, name=). די דיאגראם צייגט צווי אויבערשטע נאדס וואס פירן צו צווי באזונדערע גרופעס (רויט און בלוי), וואס קומען צוזאמען אין איין נאד אונטן. דער סלייד בלייבט ביז 5:11:00.

וואס זענען נוירעל-נעצן? (5:11:30)

דער סלייד זאגט: נוירעל-נעצן זענען דיפ-לערנען אלגאריטמען געבויט נאכן מוסטער פונעם מענטשליכן מוח. זיי באנוצן אסאך נוירא נען, וואס זענען מאטעמאטישע אפעראציעס, כדי צעצוטיילן און לייזן קאמפליצירטע מאטעמאטישע פראבלעמען. די בילד צייגט א ביאלאגישן נוירא ן ("Neurons " → א רויטן נאד → א נעץ מיט 2) Input Layer נאדס), 3) Hidden Layer נאדס) און 1) Output Layer נאד).

צוגרייטן די דאטא (5:12:00)

א טיטל-עקראן: "Data pre-processing". דער סלייד "Data preprocessing" ((5:12:30 זאגט: מען וועט שאפן אן אייגענעם דאטא-ביישפיל מיט Keras. די דאטא באשטייט פון א קליניש עקספערימענט אויף 2100 פאציענטן פון עלטער 13 ביז 100: האלב אונטער 65 און האלב איבער 65. דער ציל: געפינען די מעגליכקייט אז א פאציענט וועט האבן side effects צוליב זיין עלטער. די בילדער צייגן "65 ;gt& 50%" (עלטערע לייט) און "65 ;lt& 50%" (א יונגער).

בויען נוירעל-נעצן מיט Keras (5:13:00)

א טיטל-עקראן: "Implementing Neural Networks with Keras". דערנאך ((5:13:30 ווערט געוויזן Anaconda Navigator, מיט די אפליקאציעס Datalore, IBM Watson Studio Cloud, JupyterLab, Jupyter Notebook, CMD.exe Prompt, Glueviz, Orange 3 און Powershell Prompt; דאס Notebook-קעסטל איז אנגעצייכנט.

אין (5:14:00) Jupyter ווערט אן "Untitled" notebook איבערגערופן, דורכן פענצטערל "Rename Notebook", צו "keras". דער נאמען ווערט שפעטער (5:14:30) Keras_Tutorial. די ערשטע צעל אימפארטירט:

```
import numpy as np

from random import randint

from sklearn.utils import shuffle
```

און אין דער פערטער שורה from sklearn.preprocessing import MinMaxScaler, אלע מכשירים צוצוגרייטן און צו נארמאליזירן די דאטא.

א Keras-נויראל-נעץ פון אנהייב (לויטן עקראן) (5:15:00)

אימפארטן און ליידיקע ליסטעס (5:15:00)

אויפן עקראן ווערט געוויזן א Jupyter נאטבוק מיטן נאמען Keras_Tutorial. אין דער ערשטער צעל (In [1]) ווערן אריינגעברענגט די נויטיגע ביבליאטעקן:

- NumPy אונטערן קורצן נאמען np;
- די פונקציע randint פון random, צו שאפן צופעליגע גאנצע צאלן;
- shuffle פון sklearn.utils, צו מישן די דאטא;
- MinMaxScaler פון sklearn.preprocessing, צו נארמאליזירן צאלן.

אין דער צווייטער צעל (In [2]) ווערן געשאפן צוויי ליידיקע ליסטעס: train_labels (די ענטפערס) און train_samples (דיי ביישפילן).

דאס ביישפיל: א מעדיצין-עקספערימענט (5:15:30)

א Markdown-צעל מיטן קעפל "Example Data" שילדערט די לאגע: אן עקספערימענטאלע מעדיצין איז אויסגעפרואווט געווארן אויף 2100 מענטשן פון 13 ביז 100 יאר אלט. האלב פון די באטייליגטע זענען אונטער 65. בערך 95% פון די אונטער 65 האבן נישט געהאט קיין זייט-עפעקטן, און בערך 95% פון די איבער 65 האבן יא געהאט זייט-עפעקטן.

שאפן די דאטא מיט שלייפן (5:16:00)

דער קאוד באשטייט פון צוויי שלייפן. די ערשטע שלייף (וואס ווערט געזען בלויז טיילווייז) שאפט די קליינע מינדערהייט:

- "די ~5% יונגערע וואס האבן יא געהאט זייט-עפעקטן": random_younger = randint(13,64), דער עלטער ווערט צוגעלייגט צו train_samples, און דער לעיבל 1 צו train_labels.
- "די ~5% עלטערע וואס האבן נישט געהאט זייט-עפעקטן": random_older = randint(65,100), מיט לעיבל 0.

די צווייטע שלייף, for i in range(1000), שאפט די מערהייט:

- "די ~95% יונגערע וואס האבן נישט געהאט זייט-עפעקטן": א צופעליגער עלטער צווישן 13 און 64, לעיבל 0.
- "די ~95% עלטערע וואס האבן יא געהאט זייט-עפעקטן": א צופעליגער עלטער צווישן 65 און 100, לעיבל 1.

אזוי באדייט 1 "זייט-עפעקטן" און 0 "קיין זייט-עפעקטן".

קאנטראלירן די דאטא (5:17:00)

מען שרייבט א צעל צו דרוקן די ביישפילן. ערשט קומט print(len(),train_samples[:5]) מיטן רעזולטאט [49, 85, 41, 68, 19]. נאכדעם ווערט עס פארריכט צו print(len(train_samples),train_samples[:5]), און דער רעזולטאט איז 2100 [49, 85, 41, 68, 19]. דאס מיינט 2100 ביישפילן, און די ערשטע פינף עלטערס.

אין צעל In [6] קומט צו א צווייטע שורה, וואס דרוקט די לענג און די ערשטע פינף לעיבלס. דער רעזולטאט איז 2100 [1, 0, 1, 0, 1].

פארוואנדלען, מישן און נארמאליזירן (5:18:00)

אין צעל In [7] ווערן ביידע ליסטעס פארוואנדלט אין NumPy-מאסיוון דורך np.array. נאכדעם ווערן זיי צוזאמען געמישט:

shuffle(train_labels, train_samples) = train_labels, train_samples

אין צעל In [8] און In [9] קומט די נארמאליזאציע:

MinMaxScaler(feature_range=(0,1)) = scaler

scaler.fit_transform(train_samples.reshape(-1,1)) = scaled_train_samples

די ערשטע שורה שאפט א סקאלירער וואס ברענגט אלע ווערטן אריין צווישן 0 און 1. די צווייטע שורה פארמט איבער די ביישפילן אין א זייל (reshape(-1,1)), און דאן פאסט זי צו און פארוואנדלט זיי. מען דרוקט scaled_train_samples[:5], און דער רעזולטאט איז א מאסיוו פון פינף ווערטן: 0.1954023, 0.20689655, 0.79310345, 1., 0.06896552.

שאפן אן קינסטלעכן נויראל-נעץ (5:21:00)

א נייע Markdown-צעל ווערט געשריבן: "Creating an Artificial Neural Network".

פאר א רגע ווערט געוויזן די Wikipedia-זייט וועגן TensorFlow, אינעם אפטייל History:

- DistBelief: אנגעהויבן אין 2011, האט Google Brain געבויט DistBelief אלס א פראפריעטארע מאשין-לערנען סיסטעם, באזירט אויף דיפ-לערנען נויראל-נעצן. Google האט באשטימט עטלעכע וויסנשאפטלער, צווישן זיי Jeff Dean, צו פארפשוטערן איר קאוד. אזוי איז געווארן TensorFlow. אין 2009 האט די גרופע, אנגעפירט פון Geoffrey Hinton, צוגעלייגט פארבעסערונגען, וואס האבן געבראכט צו א 25% רעדוקציע אין טעותים ביי רייד-דערקענונג.
- TensorFlow איז די צווייטע-דור סיסטעם פון Google Brain. ווערסיע 1.0.0 איז ארויס דעם 11טן פעברואר 2017. זי לויפט אויף CPUs און GPUs, אויף 64-ביט Linux, macOS, Windows, Android און

iOS. די חשבונות ווערן אויסגעדריקט אלס "dataflow graphs", און דער נאמען קומט פון די אפעראציעס אויף מערדימענסיאנאלע מאסיוון, וואס הייסן tensors.

• אין דעצעמבער 2017 איז איינגעפירט געווארן Kubeflow. אין מערץ 2018 איז ארויס TensorFlow.js 1.0. אין יאנואר 2019 האט Google געמאלדן TensorFlow 2.0, און אין סעפטעמבער 2019 איז עס אפיציעל ארויס. אין מאי 2019 איז געמאלדן געווארן TensorFlow Graphics.

• אין מאי 2016 האט Google געמאלדן די Tensor processing unit (TPU), א ספעציעלער ASIC-טשיפ פאר מאשין-לערנען.

צוריק אינעם נאטבוק, אין צעל In [10]/[11], ווערן אריינגעברענגט (מיט פיַלן וואס ווייזן אויף יעדער שורה):

• tensorflow אלס tf, און keras;

• Sequential פון tensorflow.keras.models;

• Activation און Dense פון tensorflow.keras.layers;

• Adam פון) tensorflow.keras.optimizersאונטערגעשטראכן);

• categorical_crossentropy פון tensorflow.keras.metrics.

בויען דעם מאדעל (5:24:30)

דער מאדעל ווערט געבויט אלס Sequential, מיט דריי Dense-שיכטן:

• די ערשטע: 'units=16, input_shape=(1,), activation='relu. דאס איז 16 נויראנען, און דער אריינגאנג איז איין צאל (דער עלטער).

• די צווייטע: 'units=32, activation='relu.

• די דריטע: 'units=2, activation='softmax. דאס זענען צוויי אויסגאנגען, איינס פאר יעדע קלאס.

פיַלן און שטריכן ווייזן אויף די אקטיוואציע-פונקציעס relu און softmax.

ביים ערשטן לויפן (In [12]) קומט א טעות: SyntaxError: unexpected EOF while parsing, ווייל עס פעלט די פארמאכנדיקע קלאמער. אין In [13] ווערט צוגעלייגט "])", און דער קאוד לויפט ריכטיג.

די צוזאמענפאסונג פונעם מאדעל (5:28:00)

model.summary() ווייזט דעם מאדעל "sequential_1":

• dense_3 (Dense): אויסגאנג (None, 16), מיט 32 פאראמעטערס;

• dense_4 (Dense): אויסגאנג (None, 32), מיט 544 פאראמעטערס;

• dense_5 (Dense): אויסגאנג (None, 2), מיט 66 פאראמעטערס.

אינגאנצן זענען דא 642 פאראמעטערס, אלע 642 טריינירבאר, און 0 נישט-טריינירבאר.

טריינירן דעם מאדעל (5:29:00)

א נייע Markdown-צעל: "Training the model". דערנאך הייבט מען אן שרייבן:

'model.compile(optimizer=Adam(learning_rate=0.0001), loss='sparse_categorical_crossentropy

דער אפטימיזירער איז Adam, מיט א לערן-גיכקייט פון 0.0001 (אונטערגעשטראכן און אנגעצייכנט מיט א שטריך). די פארלוסט-פונקציע איז sparse_categorical_crossentropy.

טריינירן און פרובירן א Keras מאדעל (לויטן עקראן) (5:30:00)

דער טריינירונג-שריט אין Keras_Tutorial (5:30:00)

אויפן עקראן ווערט געוויזן א Jupyter נאטבוק מיטן נאמען Keras_Tutorial. אויבן זעט מען דעם סוף פון דעם מאדעל-סומאריום: Total params: 642, Trainable params: 642, Non-trainable params: 0.

דערנאך קומט א קעפל "Training the model" מיט צוויי קאוד-צעלן:

- צעל In [16] רופט model.compile מיט דעם optimizer Adam און learning_rate=0.0001, מיט loss='sparse_categorical_crossentropy' און metrics=['accuracy']. דאס צוגרייט דעם מאדעל צום טריינירן.
- די נעקסטע צעל באשטימט די פאראמעטערס פאר טריינירן: validation_split = 0.1 (צען פראצענט פון דער דאטא ווערט אפגעשיידט פאר אפשאצן), batch_size=10, shuffle = True (דאטא ווערט גענמישט), epochs=30 און verbose=2.

די רעזולטאטן פון טריינירן (5:31:00)

נאכן לויפן די צעל In [17]) ווייזט דער אויסגאנג: "Train on 1890 samples, validate on 210 samples". יעדע עפאך ווייזט loss, accuracy, val_loss און val_accuracy:

- Epoch 1/30: loss 0.6574, accuracy 0.5344, val_loss 0.6432, val_accuracy 0.6190
- Epoch 2/30: loss 0.6382, accuracy 0.6069, val_loss 0.6202, val_accuracy 0.6905
- Epoch 3/30: loss 0.6133, accuracy 0.6762, val_loss 0.5921, val_accuracy 0.7429
- Epoch 4/30: loss 0.5887, accuracy 0.7275, val_loss 0.5653, val_accuracy 0.7810
- Epoch 5/30: loss 0.5633, accuracy 0.7730, val_loss 0.5379, val_accuracy 0.7952
- Epoch 6/30: loss 0.5373, accuracy 0.7974, val_loss 0.5102, val_accuracy 0.8238

ביים סוף:

- Epoch 28/30: loss 0.2765, accuracy 0.9280, val_loss 0.2486, val_accuracy 0.9429
- Epoch 29/30: loss 0.2744, accuracy 0.9307, val_loss 0.2471, val_accuracy 0.9429
- Epoch 30/30: loss 0.2725, accuracy 0.9317, val_loss 0.2448, val_accuracy 0.9429

מ'זעט אז דער loss פאלט כסדר און די accuracy שטייגט פון ארום 53% ביז ארום 93%. דער Out[17] ווייזט א tensorflow.python.keras.callbacks.History אביעקט.

בויען א טעסט-סעט און פרעדיקציעס (5:32:30)

א נייער קעפל: "Building a test set and predicting". דער קאוד הייבט אן מיט צוויי ליידיגע ליסטעס, test_labels = [] און test_samples = [].

ביי 5:33:00 זעט מען מער פון דעם קאוד (שורות .(27–14 אין א לופ ווערט א צופעליג "יונגערע" עלטער מיט randint גענומען (random_younger), צוגעלייגט צו test_samples, און דער לעיבל 0 צו test_labels. א קאמענטאר זאגט: "# The 95% of older individuals who did experience side effects "; דערנאך random_older = randint(65,100), צוגעלייגט צו test_samples, מיט לעיבל 1.

נאכן לופ:

- ביידע ליסטעס ווערן פארוואנדלט אין np.array.
- די צוויי ווערן צוזאמען געמישט מיט shuffle(test_labels, test_samples).
- scaled_test_samples = scaler.fit_transform(test_samples.reshape(-1,1)); אין דעם שפעטערדיגן בילד שטייט שוין scaler.transform. דאס סקאלירט די טעסט-ווערטן.

אין צעל In [18]:

```
model.predict(x=scaled_test_samples, batch_size=10, verbose=0) = predictions
```

אין צעל In [19] ווערט געדרוקט print(predictions[:5]), די ערשטע פינף פרעדיקציעס, יעדע מיט צוויי וואַרשיינליכקייטן:

- [0.02750092 ,0.9724991]
- [0.9268539 ,0.0731461]
- [0.15210202 ,0.84789807]
- [0.02831327 ,0.97168666]
- [0.02879517 ,0.9712048]

דערנאך א צעל מיטן קאמענטאר "#rounding up predcitions to to get 0 or 1":

```
np.argmax(predictions, axis=-1) = rounded_predictions
```

```
rounded_predictions[:5]
```

דער אויסגאנג Out[20]: array([0, 1, 0, 0, 0], dtype=int64). יעדע פרעדיקציע ווערט פארוואנדלט אין 0 אדער 1, לויט וועלכער וואַרשיינליכקייט איז גרעסער.

Confusion Matrix פאר אפשאצן די גענויקייט (5:35:00)

א קעפל: "Confusion Matrices for accuracy check". אין צעל In [21] ווערט אימפארטירט:

- from sklearn.metrics import confusion_matrix
- import itertools
- import matplotlib.pyplot as plt

עס באווייזט זיך א מעלדונג: "Matplotlib is building the font cache; this may take a moment."

אין צעל In [22]: cm = confusion_matrix(y_true=test_labels, y_pred=rounded_predictions). דאס פארגלייכט די אמת'דיגע לעיבלס מיט די פרעדיקציעס.

דערנאך ווערט דעפינירט א פונקציע plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix', cmap=plt.cm.Blues) איר דאקומענטאציע-טעקסט זאגט: "This function prints and plots the confusion matrix. Normalization can be applied by setting `normalize=True`." דער קאוד:

- plt.imshow(cm, interpolation='nearest', cmap=cmap) ווייזט די מאטריץ ווי א בילד; plt.title(title) שטעלט דעם טיטל, און plt.colorbar() לייגט צו א פארב-סקאלע.
- np.arange(len(classes)) = tick_marks און plt.xticks / plt.yticks שרייבן די נעמען פון די קלאסן אויף די אקסן (אויף דער x-אקס מיט rotation=45).
- אויב normalize: cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis] און עס ווערט געדרוקט "Normalized confusion matrix"; אנדערש: 'Confusion matrix, without normalization' . דערנאך print(cm).
- thresh = cm.max() / 2. דערנאך גייט א לופ מיט itertools.product איבער אלע צעלן פון דער מאטריץ און שרייבט מיט plt.text יעדע צאל אין צענטער. די פארב איז "white " ווען דער ווערט איז גרעסער פון thresh, אנדערש "black".
- plt.tight_layout(), plt.ylabel('True label'), plt.xlabel('Predicted label').

אין צעל In [24]: cm_plot_labels = ['no_side_effects','had_side_effects'] . דערנאך ווערט געדרוקט plot_confusion_matrix(cm=cm, classes=cm_plot_labels, title='Confusion Matrix').

דער רעזולטאט פון דער מאטריץ (5:37:30)

דער אויסגאנג דרוקט "Confusion matrix, without normalization" מיט [[195 15] [10 200]], און ווייזט א בלויע גראפיק מיטן טיטל "Confusion Matrix":

- no_side_effects: 195 ריכטיג, 15 פאלש קלאסיפיצירט.
- had_side_effects: 10 פאלש, 200 ריכטיג.

די פארב-סקאלע גייט פון 25 ביז 200. די גראפיק בלייבט אויפן עקראן ביז ארום 5:38:00.

איבערבליק: די סטרוקטור פון מאדעל (5:38:30)

מ'גייט צוריק צו model.summary()צעל In [9]). דער מאדעל "sequential" בויט זיך פון Dense שיכטן:

- Dense(units=16, input_shape=(1,), activation='relu'), אויסגאנג (None, 16), 32 פאראמעטערס.
- Dense(units=32, activation='relu'), אויסגאנג (None, 32), 544 פאראמעטערס.
- Dense(units=2, activation='softmax'), 66 פאראמעטערס.
- Non-trainable params: 0.

סליידס: נויראל-נעץ, טרייניר און Confusion Matrix (5:39:00)

דער סלייד "Implementing Neural Network with Keras" זאגט: נאכן שאפן די samples און labels דארף מען שאפן דעם Keras נויראל-נעץ מאדעל; מ'ארבעט מיט א Sequential מאדעל מיט דריי שיכטן:

- Input Layer: Nodes = 16, Relu activation function

- 32, Relu activation = Hidden Layer: Nodes •
- 2, Softmax activation = Output Layer: Nodes •

דערנעבן איז א דיאגראם פון Input Layer, Hidden Layer און Output Layer מיט פארבונדענע קנעפ.

דער סלייד "Training Neural Network with Keras" זאגט: טרייַנירן איז א פראצעס פון צוויי שריט. פריער טוט מען Compile דעם מאדעל, און דערנאך טרייַנירט מען אים אויפן טרייַנירונג-דאטא-סעט.

- Compiling : קאמפיילן פארוואנדלט דעם קאוד אין א פארעם וואס די מאשין פארשטייט; מ'וועט באניצן Adam.
- Training: טרייַנירן דעם מאדעל מיינט לאזן אים לערנען פון דער טרייַנירונג-דאטא; מ'וועט אים טרייַנירן אויף דעם train_sample סעט.

ביי 5:40:00 ווייזט דער נאטבוק נאכאמאל דעם טרייַנירונג-שריט (צעלן [10], In [11]) מיט אן אנדער לויף: Epoch 1/30: loss 0.6716, accuracy 0.5556, val_loss 0.6575, val_accuracy 0.6286.

דער סלייד "Creating a Confusion Matrix with Keras" זאגט: ווייל מ'טוט קלאסיפיקאציע אויף דער דאטא, דארף מען א confusion matrix צו קאנטראלירן די רעזולטאטן. זי צעטיילט די פארשידענע פאלשע און ריכטיגע קלאסיפיקאציעס כדי צו באקומען די accuracy.

- True Positives: די צאל ריכטיגע פרעדיקציעס פאר דעם פאזיטיוון לעיבל.
- False Positives: די צאל נעגאטיוע ווערטן וואס זענען קלאסיפיצירט געווארן ווי פאזיטיוו.
- True Negatives: די צאל ריכטיג קלאסיפיצירטע נעגאטיוע ווערטן.

א טאבעלע ווייזט Actual (Positive/Negative) קעגן Predicted, מיט "True Positive" און "False Positive" אין דער פאזיטיווער שורה.

ביי 5:41:00 ווייזט דער נאטבוק נאכאמאל די מאטריץ [[195 15] [10 200]].

א נייער פראיעקט: Face Mask Detection (5:41:30)

דער סלייד "Face Mask Detection with Keras" זאגט: מ'וועט בויען א Keras מאדעל וואס קאנטראלירט אין רעאלער צייט צי א מענטש טראגט א מאסקע אדער נישט. צוויי בילדלעך זענען באצייכנט "Mask" און "No Mask". דער סלייד בלייבט ביז 5:42:00.

ביי 5:42:30 ווערט געעפנט א פאלדער אין Windows Explorer מיטן וועג gt;& programming gt;& simplilearn gt;& 2021_02_21_keras_OpenCV gt;& object detection gt;& Face-Mask-Detection-master gt;& dataset gt;& without_mask. אין דעם פאלדער ליגן א סך בילדער פון פנימער אן מאסקעס, מיט נעמען ווי 0_0_aidai_0014.

ביי 5:43:00 ווערט געוויזן Anaconda Navigator מיט דער סביבה py38 . דארט זענען דא אפליקאציעס: Datalore, IBM Watson Studio Cloud, Notebook (Jupyter 6.2.0), CMD.exe Prompt, Glueviz, JupyterLab, Orange 3, Powershell Prompt, PyCharm Professional, Qt Console, RStudio, Spyder.

ביי 5:43:30 קומט א סלייד "Keras Model", נומער 2: Functional Model:

- א מאדעל מיט עטלעכע אריינגאנגען און עטלעכע אויסגאנגען (multi input, multi output).

• א קאמפליצירטער מאדעל וואס צעטיילט זיך אין צוויי אדער מער צווייגן.

א קאוד-ביישפיל הייבט אן מיט img_inputs = keras.Input(shape=(32, 32, 3)) און dense = layers.Dense(64, activation="relu"). דערנעבן איז א דיאגראם פון א נעץ וואס צעטיילט זיך אין צווייגן.

ביי 5:44:00 קומט מען צוריק צו Anaconda Navigator. ביי 5:44:30 ווערט אין Jupyter געעפנט א נייער נאטבוק (Untitled) אינעם פאלדער Face-Mask-Detection-master. אין דעם פענצטער "Rename Notebook" ווערט אריינגעשריבן דער נייער נאמען Train_Mask.

טרייניירן א מאדעל צו דערקענען מאסקעס (לויטן עקראן) (5:45:00)

דער נאטבוק Train_Mask_Detection און די אימפארטן (5:45:00)

אויפן עקראן ווערט געווויזן א Jupyter נאטבוק מיטן נאמען Train_Mask_Detection, מיט Python 3. אין דער אדרעס-שורה זעט מען אז דער נאטבוק ליגט אין א טעקע Face-Mask-Detection-master. דער ערשטער קאוד-צעל אנטהאלט א לאנגע רשימה פון אימפארטן, שורה ביי שורה:

• פון tensorflow.keras.preprocessing.image ווערט אימפארטירט ImageDataGenerator.

• פון tensorflow.keras.applications ווערט אימפארטירט MobileNetV2.

• פון tensorflow.keras.layers ווערן אימפארטירט די שיכטן AveragePooling2D, Dropout, Flatten, Dense און Input.

• פון tensorflow.keras.models ווערט אימפארטירט Model, און פון tensorflow.keras.optimizers ווערט אימפארטירט Adam.

• פון tensorflow.keras.applications.mobilenet_v2 ווערט אימפארטירט preprocess_input.

• פון tensorflow.keras.preprocessing.image ווערן אימפארטירט img_to_array און load_img.

דער עקראן בלייבט אויף דעם צעל עטליכע מינוט. ביסלעכווייז ווערן אויף אים געצייכנט פיילן און ליניעס: די ערשטע שורה (ImageDataGenerator) ווערט אונטערגעשטראכן, פיילן ווייזן אויף AveragePooling2D, Dropout און Flatten, א וועלנדיגע קלאמער נעמט ארום Dense, Input און Model, א ליניע ווערט געצויגן אונטער Adam, און נאך א קלאמער ביי preprocess_input און די שורות נאך אים.

די רעשט פון די אימפארטן (5:47:30)

נאכן אראפסקראלן זעט מען די ווייטערדיגע שורות פונעם זעלבן צעל:

• פון tensorflow.keras.utils ווערט אימפארטירט to_categorical.

• פון sklearn.preprocessing ווערט אימפארטירט LabelBinarizer; פון sklearn.model_selection ווערט אימפארטירט train_test_split; פון sklearn.metrics ווערט אימפארטירט classification_report. אויף די דריי שורות ווערט געצייכנט א קלאמער.

• פון imutils ווערט אימפארטירט paths) די שורה ווערט אונטערגעשטראכן).

• עס ווערן אימפארטירט matplotlib.pyplot אלס plt, numpy אלס np, און os.

די הויפט-פאראמעטערס (5:48:30)

א נייער צעל, מיט א קאמענטאר וואס זאגט אז מען שטעלט איין די אנהייב learning rate, די צאל epochs פון טרייניירן, און די batch size:

1e-4 = INIT_LR

2 = EPOCHS

32 = BS

דער צעל וועלכע וועט אויסגעפירט ווערן (In [2]). שפעטער וועלכע וועט צוגעלייגט ווערן א זעקסטע שורה, DIRECTORY = "dataset", און דער צעל וועלכע וועט ווידער אויסגעפירט ווערן (In [4]).

דער דאטא-סעט אין דער טעקע (5:50:00)

אויפן עקראן וועלכע וועט געפונען ווערן א Windows טעקע-פענצטער מיטן וועג Documents › Simplilearn › 2021_02_21_keras_openCV › keras › object detection › Face-Mask-Detection-master › dataset › without_mask . אין דער טעקע זעט מען א סך קליינע בילדער פון פנימער אן מאסקעס, מיט טעקע-נעמען ווי 0_0_aidai_0014, 0_0_anhu_0004, 0_0_baobeier_0062, 0_0_caizhuoyan_0013 א.א.וו. (די בילדער אליין וועלן דא נישט באשריבן.)

אייגלייענען די בילדער (5:51:00)

א נייער צעל בויט די דאטא:

- עס וועלן געשאפן צוויי ליידיגע ליסטעס: data = [] און labels = [].
- א שלייף גייט איבער יעדע category אין os.listdir(DIRECTORY) ; פאר יעדער וועלכע געבויט דער וועג מיט os.path.join(DIRECTORY, category).
- אן אינערלעכער שלייף גייט איבער יעדעס img אין os.listdir(path), בויט img_path, לאדנט דאס בילד מיט load_img(img_path, target_size=(224, 224)), פארוואנדלט עס אין א מאסיוו מיט img_to_array, און דורכפירט preprocess_input.
- דאס בילד וועלכע צוגעלייגט צו data, און די category צו labels.
- צום סוף: print(labels[:10]).

ביים אויספירן קומט ארויס א ראזעווע ווארענונג פון PIL (Image.py:962: UserWarning) אז Palette בילדער מיט דורכזיכטיגקייט אין bytes דארפן ווערן פארוואנדלט אין RGBA בילדער. נאכדעם וועלכע גענדרוקט א ליסטע פון צען מאל 'without_mask'.

one-hot קאדירונג פון די לייבלס (5:52:00)

דער ווייטערדיגער צעל, מיטן קאמענטאר "perform one-hot encoding on the labels":

- LabelBinarizer() = lb
- lb.fit_transform(labels) = labels, און דערנאך to_categorical(labels) = labels.
- data = np.array(data, dtype="float32") און labels = np.array(labels).

א רעזולטאט (Out[7]) ווייזט א מאסיוו פון צען שורות [1., 0.] מיט dtype=float32. נאכדעם ווערט אויסגעפירט (In [8]) labels[:10], און דער רעזולטאט ווייזט צען שורות [0., 1.] מיט dtype=float32.

צעטיילן אויף טרייניירן און טעסטן (5:53:30)

א צעל טיילט די דאטא מיט train_test_split(data, labels, test_size=0.20, stratify=labels, random_state=42) אויף פיר טיילן: trainX, testX, trainY, testY. ער ווערט אויסגעפירט אלס In [9]; דער טייל test_size=0.20 ווערט אונטערגעצייכנט.

דאטא-אויגמענטאציע (5:54:00)

א צעל מיטן קאמענטאר "construct the training image generator for data augmentation" שאפט aug = ImageDataGenerator מיט די פאראמעטערס:

- rotation_range=20
- zoom_range=0.15
- width_shift_range=0.2 און height_shift_range=0.2
- shear_range=0.15
- horizontal_flip=True
- fill_mode="nearest"

ער ווערט אויסגעפירט אלס In [10].

לאדנען דעם MobileNetV2 באזע-מאדעל (5:55:00)

דער קאמענטאר זאגט אז מען לאדנט דאס MobileNetV2 נעץ, אזוי אז די FC head שיכטן ווערן אוועקגעלאזט. דער קאוד:

baseModel = MobileNetV2(weights="imagenet", include_top=False,

(input_tensor=Input(shape=(224, 224, 3))

ביים אויספירן (In [12]) ווערט include_top=False אונטערגעצייכנט, און עס קומט ארויס א WARNING פון tensorflow: אז input_shape איז נישט באשטימט אדער נישט קוואדראטיש, אדער rows איז נישט צווישן [96, 128, 160, 192, 224]; די weights פאר input shape (224, 224) וועלן געלאדנט ווערן אלס די default.

בויען דעם "קאפ" פונעם מאדעל (5:56:00)

א צעל מיטן קאמענטאר "construct the head of the model that will be placed on top of the base model". יעדע שורה לייגט צו א שיכט אויף headModel:

- headModel = baseModel.output — מען הייבט אן פונעם אויסגאנג פונעם באזע-מאדעל.
- AveragePooling2D(pool_size=(7, 7))
- Flatten(name="flatten")
- Dense(128, activation="relu")

- Dropout(0.5)
- "softmax"Dense(2, activation=") — דער לעצטער שיכט מיט צוויי אויסגאנגען.

דער צעל ווערט אויסגעפירט (In [13], נאכדעם In [14]); עטליכע שורות ווערן אונטערגעצייכנט, אויך די שורה מיט Dropout.

צוזאמשטעלן און "פרירן" די שיכטן (5:58:00)

א צעל (In [16]) מיטן קאמענטאר אז מען שטעלט דעם head FC מאדעל אויפן באוע-מאדעל, און דאס וועט ווערן דער אמתער מאדעל וואס מען וועט טריינירן:

Model(inputs=baseModel.input, outputs=headModel) = model

א צווייטער צעל (In [17]) גייט איבער אלע שיכטן אין baseModel.layers און שטעלט layer.trainable = False; דער קאמענטאר זאגט אז מען "פרירט" זיי כדי זיי זאלן *נישט* ווערן דערהיינטיגט בשעתן ערשטן טריינירן.

קאמפילירן דעם מאדעל (5:58:30)

א צעל מיטן קאמענטאר "compile our model":

- "...INFO] compiling model]"print(")
- (EPOCHS / Adam(lr=INIT_LR, decay=INIT_LR = opt
- =("accuracy"optimizer=opt, metrics=[,"binary_crossentropy"model.compile(loss

נאכן אויספירן (In [18]) ווערט געדרוקט: [INFO] compiling model...

טריינירן דעם קאפ פונעם נעץ (5:59:30)

דער לעצטער צעל אין דעם אפשניט, מיטן קאמענטאר "train the head of the network":

- "...INFO] training head]"print(")
- BS, // aug.flow(trainX, trainY, batch_size=BS), steps_per_epoch=len(trainX) מיט model.fit(= H
  .epochs=EPOCHS ,און BS // validation_data=(testX, testY), validation_steps=len(testX)

דער צעל לויפט נאך (In [*]). אינעם אויסגאנג זעט מען: [INFO] training head..., דערנאך Epoch 1/2, און א פולער פארשריט-באלקן 50/50 – 80s, 2s/step – loss: 0.4640, און דערנאך דאס ווערט) accuracyדי צאל דערפון זעט מען נישט אויפן עקראן).

טריינירן און לויפן א מאסקע-דעטעקטאר (לויטן עקראן) (6:00:00)

טריינירן דעם "קאפ" פון נויראל-נעץ (6:00:00)

אויפן עקראן ווערט געווויזן א Jupyter נאטבוק מיטן נאמען Train_Mask_Detection. פריער אינעם אויסדרוק שטייט "[INFO] compiling model...", דאס הייסט אז דער מאדעל איז שוין צונויפגעשטעלט (compiled).

די קומענדיגע צעל האט דעם קאמענטאר "# train the head of the network". דער קאוד:

- דרוקט אויס "[INFO] training head]...".

- רופט H = model.fit(...) און לייגט די רעזולטאטן אריין אין H.
- די טרייניירונג-דאטא קומט פון aug.flow(trainX, trainY, batch_size=BS) , דאס הייסט בילדער פון trainX מיט די לעיבלס trainY, אין גרופעס (batches) פון BS.
- BS // steps_per_epoch=len(trainX) – וויפיל שריט אין יעדע עפאק.
- validation_data=(testX, testY) – די טעסט-דאטא צו קאנטראלירן דעם מאדעל.
- BS // validation_steps=len(testX).
- epochs=EPOCHS.

אין די ערשטע פרעימס זענען די שורות מיט aug.flow און validation_data אונטערגעשטראכן.

רעזולטאטן פון טרייניירן (6:01:00)

דער אויסדרוק וויזט צוויי עפאקס, יעדע מיט 50/50 שריט, בערך 80 סעקונדעס (2s/step):

- Epoch 1/2: loss: 0.4640, accuracy: 0.7823, val_loss: 0.1797, val_accuracy: 0.9504.
- Epoch 2/2: loss: 0.1748, accuracy: 0.9380, val_loss: 0.1107, val_accuracy: 0.9702.

דער loss פאלט אראפ און די accuracy שטייגט, ביי די טעסט-דאטא ביז כמעט 97%.

פּאָרויסזאגן אויף דער טעסט-דאטא (6:01:30)

א נייע צעל מיטן קאמענטאר "# make predictions on the testing set" דרוקט "[INFO evaluating network..." און רופט predIdxs = model.predict(testX, batch_size=BS). די קומענדיגע צעל הייבט אן מיטן קאמענטאר "# for each image in the testing set we need to find the index of the...".

דער classification_report (6:02:00)

דער קאוד דרוקט print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_)). די טאבעלע:

- with_mask: precision 0.97, recall 0.97, f1-score 0.97, support 222.
- without_mask: precision 0.97, recall 0.97, f1-score 0.97, support 181.
- accuracy: 0.97 אויף 403 בילדער.
- macro avg און weighted avg: אלעס 0.97, support 403.

דער עקראן בלייבט אויף דער טאבעלע בערך א מינוט.

אפהיטן דעם מאדעל (6:03:00)

צעל מיטן קאמענטאר "# serialize the model to disk": דרוקט "[INFO saving mask detector model..." און רופט model.save("mask_detector.model", save_format="h5"), אזוי וועט דער מאדעל געשפייכלערט אויף דער דיסק.

גראפיק: Training Loss and Accuracy (6:03:30)

א צעל מיט matplotlib קאוד: ["val_accuracy"plt.plot(np.arange(0, N), H.history[,("val_acc"label= ,("Training Loss and Accuracy"plt.title(,("# Epoch"plt.xlabel(,("Loss/Accuracy"plt.ylabel(="lower left"plt.legend(loc=") און)"plot.png"plt.savefig(.

די גראפיק ווייזט פיר ליניעס:) train_loss רויט) פאלט פון בערך 0.46 ביז בערך 0.17;) val_loss בלוי) נידעריג, פאלנדיג ביז בערך 0.11; train_acc שטייגט פון בערך 0.78 ביז בערך 0.94; val_acc בלייבט הויך, בערך 0.95–0.97.

א נייער נאטבוק: mask Detector (6:05:00)

נאך א ליידיגן בלעטער-טאב עפנט זיך א נייער נאטבוק "mask Detector". די ערשטע צעל ("# import the necessary packages") אימפארטירט:

- preprocess_input פון tensorflow.keras.applications.mobilenet_v2.
- img_to_array פון tensorflow.keras.preprocessing.image.
- load_model פון tensorflow.keras.models.
- VideoStream פון imutils.video.
- numpy as np, imutils, time, cv2, os.

די פונקציע detect_and_predict_mask (6:06:00)

די פונקציע באקומט frame, faceNet און maskNet:

- frame.shape[:2] = (h, w) – די הייך און ברייט פונעם בילד.
- cv2.dnn.blobFromImage(frame, 1.0, (224, 224), (104.0, 177.0, 123.0)) – מאכט א "blob" פונעם בילד.
- faceNet.setInput(blob) און faceNet.forward() = detections – לאזט דורך דעם blob דורכן נעץ צו געפינען פנים; print(detections.shape) דרוקט די פארעם.
- מען מאכט ליידיגע רשימות (faces, locs, predsפנים, זייערע ערטער, און די פאָרויסזאגן פונעם מאסקע-נעץ).
- א for-לופ איבער range(0, detections.shape[2]); detections[0, 0, i, 2] = confidence – די זיכערקייט (probability) פון יעדע דעטעקציע.
- א קאמענטאר זאגט מען פילטערט אויס שוואכע דעטעקציעס; * detections[0, 0, i, 3:7] = box np.array([w, h, w, h]) רעכנט די קעסטל.
- (startX, startY) = (max(0, startX), max(0, startY)) און (endX, endY) = (endX, endY) - (1, endY) – האלט די קעסטל אינעווייניג פונעם בילד.
- דאס פנים ווערט ארויסגעשניטן: frame[startY:endY, startX:endX] = face, איבערגעמאכט פון BGR צו RGB מיט cv2.cvtColor(face, cv2.COLOR_BGR2RGB), געמאכט 224x224 מיט cv2.resize , דערנאך img_to_array און preprocess_input.
- faces.append(face) און locs.append((startX, startY, endX, endY)).

- if len(faces) gt& 0: – נאר אויב מען האט געפונען כאטש איין פנים. כדי עס זאל גיין שנעלער, מאכט מען פאָרויסזאגן אויף אלע פנים אויף אייַנמאל: faces = np.array(faces, dtype="float32") און preds = maskNet.predict(faces, batch_size=32).
- return (locs, preds) – גיט צוריק די ערטער און די פאָרויסזאגן.

לאדן די מאדעלן (6:10:30)

צעל "load our serialized face detector model from disk #":

- prototxtPath = r"deploy.prototxt\face_detector"
- weightsPath = r"res10_300x300_ssd_iter_140000.caffemodel\face_detector"
- faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
- דערנאך "load the face mask detector model from disk #": maskNet = load_model("mask_detector.model").

דער ווידעא-לופּ (6:12:00)

די לעצטע צעל דרוקט "[INFO] starting video stream..." און הייבט אן vs = VideoStream(src=0).start() די צוויי שורות זענען דא צוויי מאל). דערנאך while True:

- frame = vs.read() און imutils.resize(frame, width=400) – מאקסימום 400 פיקסלען ברייט.
- (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet).
- for (box, pred) in zip(locs, preds): צעלייגט box = (startX, startY, endX, endY) און (mask, withoutMask) = pred.
- label = "Mask" if mask gt& withoutMask else "No Mask"; color = (0, 255, 0) (גרין) ביי "Mask", אנדערש (0, 0, 255) (רויט).
- דער לעיבל באקומט דעם פראצענט: "{:.2f}".format(label, max(mask, withoutMask) * 100).
- cv2.putText(...) מיט cv2.FONT_HERSHEY_SIMPLEX, 0.45 און cv2.rectangle(...) צייכנט דעם טעקסט און די קעסטל.
- cv2.imshow("Frame", frame) און key = cv2.waitKey(1) & 0xFF; if key == ord("q"): break – דריקן "q" ענדיגט דעם לופּ.
- צום סוף: cv2.destroyAllWindows() און vs.stop().

לויפן דעם דעטעקטאר (6:14:00)

דער אויסדרוק ווייזט צוויי מאל "[INFO] starting video stream..." און דערנאך א לאנגע רייע פון detections.shape, למשל (1, 1, 121, 7), (1, 1, 135, 7), (1, 1, 127, 7), (1, 1, 129, 7), (1, 1, 118, 7), (1, 1, 130, 7). א קליין "Frame" פענצטער עפנט זיך מיטן לעבעדיגן ווידעא פון דער קאמערע, מיט א רויטער קעסטל ארום דעם דעטעקטירטן פנים.

דער קאוד פונעם mask Detector (6:15:00)

אויפן עקראן ווערט געוויזן א Jupyter נאטבוק מיטן נאמען "mask Detector" (מיט "unsaved changes"), מיט Python 3 Kernel אלס. די זייט איינעם בלעטערער ווייזט אויף א טעקע "mask detector.ipynb" אין א פראיעקט "Face-Mask-Detection-master". מען זעט א שטיקל קאוד פון ליניע 33 ביז 42 ארום:

- א קאמענטאר זאגט אז מען פילטערט אויס שוואכע דעטעקציעס, דורכן זיכער מאכן אז די "confidence" איז גרעסער ווי די מינימום-confidence. דערנאך קומט די באדינגונג: if confidence > 0.5:
- אינעווייניג רעכנט מען אויס די (x, y)-קאארדינאטן פונעם "bounding box" (דער ראם ארום דעם אביעקט). מען נעמט די ווערטן פון detections[0, 0, i, 3:7] און מען פילט זיי מאל np.array([w, h, w, h]), כדי זיי צו פארגרעסערן לויט דער בריים און הייך פונעם בילד. דאן צעטיילט מען דעם רעזולטאט אויף פיר צאלן, startX, startY, endX, endY, און מען פארוואנדלט זיי אויף גאנצע צאלן מיט box.astype("int").
- א צווייטער קאמענטאר: מען דארף זיכער מאכן אז די ראמען בלייבן אינערהאלב די גרעניצן פונעם בילד. דערפאר שטעלט מען (startX, startY) אויף max(0, startX) און max(0, startY), אז זיי זאלן נישט זיין ווייניגער ווי 0. אזוי אויך שטעלט מען (endX, endY) אויף min(w - 1, endX) און min(h - 1, endY), אז זיי זאלן נישט אריבערגיין דעם ראנד.
- א דריטער קאמענטאר זאגט אז מען שניידט ארויס דעם "face ROI" (דער טייל פונעם בילד וואו דאס פנים געפינט זיך), און מען פארוואנדלט אים פון BGR אויף RGB. ווייטער אונטן זעט מען די ליניע: face = img_to_array(face), וואס פארוואנדלט דאס פנים-בילד אויף א מאסיוו פון צאלן.

דער סלייד "Face Mask Detection with Keras" (6:15:30)

דער סלייד זאגט אז מען וועט בויען א Keras מאדעל וואס זאל קאנטראלירן, אין "real time", צי א מענטש טראגט א מאסקע אדער נישט. מען זעט צוויי בילדער אין ראמען: אויף איינעם א פיגור מיט א מאסקע, באצייכנט "Mask", און אויף דעם אנדערן א פיגור אן א מאסקע, באצייכנט "No Mask".

אן איינפירונג: Computers and AI (6:16:00)

עס הייבט זיך אן א נייער אפטייל פון Simplilearn. אויפן עקראן שטייט די אונטערשריפט "Computers and AI".

"What are Transformers?" (6:17:00)

דער סלייד ווייזט ערשט א בילד פון א ראבאט אויף א טעכנאלאגישן הינטערגרונט. דערנאך (6:17:30 קומט א דיאגראם פונעם "Transformer": אינעווייניג געפינען זיך צוויי טיילן, "Encoder" און "Decoder", מיט א פייל פונעם Encoder צום Decoder. אינעם Encoder גייט אריין דער ענגלישער זאץ "I like science", און פונעם Decoder קומט ארויס דער דייטשער זאץ "Ich mag Wissenschaften".

ביי 6:18:00 ווערן צוגעלייגט צוויי ביישפילן. אויף לינקס זעט מען א טעלעפאן-עקראן פון א "History" ליסטע פון איבערזעצונגען צווישן יאפאניש און ענגליש, למשל "whatever", "service unavailable", "what do you recommend", "takeaway", "please pick up your trash". אויף רעכטס זעט מען א

וועטער-אפליקאציע מיט א זוך-פעלד "Delhi". זי ווייזט "Noida, IN", דעם טאג "Thursday 17 August", א טעמפעראטור פון °C35, "BROKEN CLOUDS", און א ווינט-גיכקייט פון 2.4m/s.

"How Transformers Work?" (6:18:30)

א דיאגראם מיטן נאמען "Sequence to Sequence Model – Neural Machine Translation" איז צעטיילט אויף צוויי טיילן. אין דעם ערשטן טייל, "Encoding Stage", שטייען דריי קעסטלעך "Encoder RNN". אין דעם צווייטן טייל, "Decoding Stage", זעט מען "Decoder RNN" קעסטלעך, ערשט איינס און דערנאך צוויי. אונטן ווערט געוויזן אן ביישפיל: דער INPUT איז "Je suis étudiant", ער גייט אריין אין "THE TRANSFORMER", און דער OUTPUT איז "I am a student". ביי 6:19:30 ווערט אויף רעכטס צוגעשריבן "Recurrent Neural Networks (RNNs)".

"Comparison" – RNN קעגן Transformer (6:20:00)

אויף לינקס שטייט "Recurrent Neural Networks (RNNs)", מיט א קעסטל "RNN Cell" וואס האט אין זיך די פונקציע F(Input, Hidden). אויף רעכטס זעט מען דעם זאץ "Sun Rises in the east". יעדעס וואלרט גייט אריין אין א קרייזל, די קרייזלעך זענען פארבונדן איינס צום צווייטן מיט פיילן, און פון יעדן קרייזל קומט ארויס א רעזולטאט, O1 ביז O5.

ביי 6:20:30 קומט א צווייטער סלייד, "English-French Translation". אויף איין זייט שטייט א "RNN (Encoder)", וואס באקומט נאר איין וואלרט, "dog", און גיט ארויס איין וועקטאר [0.32, 0.19, 0.91, 0.01]. אויף דער אנדערער זייט שטייט א "Transformer (Encoder)", וואס באקומט די וועלרטער "The red dog" און גיט ארויס דריי וועקטארן: [0.76, 0.23, 0.11, 0.21], [0.32, 0.88, 0.41, 0.14] און [0.32, 0.19, 0.91, 0.01]. א רויטע הערה זאגט: ביים Transformer מאדעל גיט מען אריין אלע וועלרטער פונעם זאץ צוגלייך. אויף רעכטס זעט מען א GPT-2 ביישפיל. דער Input איז "recite the first law $ A robot may", און דער Output איז "A robot may not", וואלרט נאך וואלרט.

"Applications of Transformers" (6:21:00)

- נומער 2, "Document Summarization": מען זעט א Google Docs דאקומענט מיטן נאמען "Draft: OpenAI press release doc", מיט א פענצטער וואס וועזט "AI is generating text…" און די קנעפלעך "Cancel" און "Insert". דערנעבן שטייט א ChatGPT סאמערי פון אן ארטיקל וועגן פארוואנדלען ווידעאס אויף MP3.
- נומער 4, "Game Playing": מען זעט א שאך-ברעט, און א סימולאציע פון א פיגור וואס גייט אויף א באל, באצייכנט "Ball Walking".

"Training Process" (6:22:00 – 6:23:00)

דער סלייד בלייבט אויפן עקראן אן ערך א מינוט, און ער וועזט צוויי שריט:

- 1. Semi-supervised Learning
- 2. Pre-training and Fine-tuning

(6:23:30) "What Is Multimodal AI"

דער סלייד שרייבט: "You might describe the sights you saw, the sounds you heard, and even your emotions". דאס הייסט, מען קען באשרייבן וואס מען האט געזען, וואס מען האט געהערט, און אפילו די געפילן. דערנאך ווערט צוגעלייגט א קעסטל "Multimodal AI". ביי 6:24:30 קומט א טיטל-בילד פונעם אפטייל "Multimodal AI" פון Simplilearn.

(6:25:00) "Generative AI Vs Multimodal AI"

דער סלייד פארגלייכט די צוויי סארטן AI:

Generative AI:

- ציל: עס שאפט נייע דאטא, ענליך צו דער דאטא אויף וועלכער עס איז טריינירט געווארן.
- סארט דאטא: עס ארבעט מערסטנס מיט איין סארט דאטא, למשל טעקסט (ווי שרייבן לידער) אדער בילדער (ווי שאפן רעאליסטישע פארטרעטן).
- ביישפילן: Chatbots, מאדעלן וואס שאפן טעקסט, און מכשירים צו רעדאגירן בילדער.
- מעלות: עס קען שאפן קרעאטיוו און נייע תוכן, אויטאמאטיזירן ארבעט וואס חזרט זיך איבער, און צופאסן דעם עקספיריענס צו יעדן באניצער.

Multimodal AI:

- ציל: עס פארשטייט און פראצעסירט אינפארמאציע פון עטליכע מקורות (טעקסט, רעדע, בילדער, ווידעאס).
- סארט דאטא: עס ארבעט מיט פארשידענע סארטן דאטא, און דערמיט פארשטייט עס די וועלט אויף א מער פולשטענדיגן אופן.
- ביישפילן: ווירטועלע געהילפן, סיסטעמען פאר מעדיצינישע דיאגנאזע, און אויטאנאמישע פארמיטלען.
- מעלות: עס גיט א פארשטאנד פון דער וועלט וואס איז מער ענליך צו א מענטש, און עס פארבעסערט די געניקייט.

(6:26:00) "Benefits Of Multimodal AI"

א בילד ווייזט א פראגראמירער ביים קאמפיוטער, מיט די צייכנס CSS, HTML און C++. דערנאך ווערן צוגעלייגט דריי מעלות: Reasoning, Problem-Solving און Generation , און ס'באווייזט זיך דאס לאגא פון Gemini.

(6:27:00) "Multimodal AI: Big Challenges"

אין צענטער שטייט "Multimodal AI", און ארום אים זעט מען פיר שווערigקייטן:

- Data overload – צופיל דאטא
- Data Alignment – צווזאמפאסן פארשידענע סארטן דאטא
- Meaning's mystery – דער סוד פון באדייט
- Data scarcity – א מאנגל אין דאטא

"How To Build An iOS App Using AI" (6:28:00)

עס הייבט זיך אן א נייער אפטייל. ער האט א טיטל-בילד מיט א טעלעפאן און אפליקאציע-אייקאנס (MESSAGES, MAILS, PHOTOS, PHONE), און דערנאך קומט א בלוי-פלאן בילד פון "Apps " אייקאנס. ביי 6:29:00 ווערט געוויזן דער סלייד "Introduction", מיט דער פראגע "How to Build iOS App with AI?" און מיטן זאץ: "Apple Xcode IDE is preferred to build as iOS application for apple app store". דאס הייסט, צו בויען אן iOS אפליקאציע פאר Apple'ס App Store באניצט מען בעסער Apple Xcode. דערנעבן שטייט דאס Xcode אייקאן. ביי 6:29:30 איז דער סלייד ליידיג, אין מיטן פון א איבערגאנג.

בויען אן אפליקאציע מיט) Boltלויטן עקראן) (6:30:00)

זוכן א טרעף-ארט אין New York (6:30:00)

אויפן עקראן ווערט געוויזן א Google זוך: "vc meeting location new york". די רעזולטאטן: Convene ("Locations | Event & Meeting Spaces in Our Portfolio", מיט ערטער ווי Convene 101 Greenwich, New York"101 Park Avenue, 117 West 46th Street, 237 Park, 30 Hudson Yards), LiquidSpace ("Hourly Spaces with Video Conference", געגנטן ווי Chelsea / Midtown West, Flatiron, NoMad, Midtown East, Lower Manhattan, Tribeca Soho, Hell's Kitchen, Murray Hill, DUMBO & New York conference rooms and hourly"Brooklyn Heights, Gramercy Park), Breather ("Access 503 meeting rooms in the New York area", "meeting spaces Meeting"Regus (און) Rooms in NYC").

א האלבע מינוט שפעטער ווערט אין א נייעם טאב אריינגעשריבן "ios app for iphone", מיט א רשימה פון Google פארשלאגן.

Bolt, Expo און Claude (6:31:00)

אויפן עקראן ווערט געעפנט bolt.new. די הויפט-זייט פרעגט: "What do you want to build?" — "Create stunning apps & websites by chatting with AI", מיט א פעלד "Type your idea and we'll bring it to life", און די מעגלעכקייט צו אימפארטירן פון Figma אדער GitHub.

דערנאך ווערט געוויזן דער Dashboard פון Expo (expo.dev), מיט "Create a project to – No projects", א קנעפל "get started with Expo", א בלאג-ארטיקל "How I prompted my way to a real App Store app using Bolt and Expo", און א נאטיץ "Expo support for React Native 0.80". אין דער זייט-רשימה זעט מען Account, Access tokens, Apple devices, Android & iOS credentials settings.

אום 6:32:00 ווערט געעפנט Claude (claude.ai, Free plan, מאדעל Claude Sonnet 4). דער פראמפט: "Hi Claude can you please give me a prompt for an ai to help me create an app to help me"….

דער ענטפער פון Claude (6:32:30)

דער שמועס הייסט "Professional Meeting Venue Finder App". דער סוף פונעם ענטפער ווייזט:

- Display average meeting duration data from other users
- Option to save favorite venues and create meeting itineraries

- Integration with calendar apps for scheduling

אונטער "Technical Specifications":

- Cross-platform compatibility (iOS/Android or web-based)
- Database for venue information, user reviews, and booking data
- API integrations for real-time data (hours, availability, reviews)
- Secure payment processing for reservations

דערנאך: "Please provide a complete development roadmap, suggest appropriate tech stack, and outline the MVP features to launch first." און צום סוף: "This prompt should give you a comprehensive starting point for building your meeting venue finder app."

שרייבן דעם פראמפט אין Bolt (6:33:00)

צוריק אין Bolt ווערט אריינגעשריבן: "Hi Bolt can you please give me a prompt for an ai to help me create an app to help me find best cafes or hotels for a professional meeting with a venture capitalist for my new project". דער טעקסט ווערט דערנאך פארראכט: "can you please me create an app to help me find best cafes or hotels for a professional meeting with a venture capitalist. the app should have", מיט א נומערירטע רשימה:

1. location
2. date
3. budget

און דערצו: "based on the inputs from user, you need to find the best location for the said date and the cost of the hotel or cafe should be under the budget."

אום 6:34:30 ווייזט זיך "Enhancing prompt…" — Bolt פארבעסערט אליין דעם פראמפט.

Bolt בויט די אפליקאציע (6:35:00)

אויפן עקראן קומט ארויף א פענצטער "Tell us a little about yourself" (מיט א פארשפרעכונג "You'll be rewarded 1M tokens upon completion", קנעפלעך "Not now" און "Let's do it!"). אונטער אים זעט מען א טייל פונעם פארבעסערטן פראמפט: Quick-booking option, Save favorite venues, Share venue details with meeting participants, Calendar integration for scheduling, Directions and estimated travel time, און: "The app should ensure all recommended venues are specified … budget while maintaining professional … suitable for venture capital meetings." דערנאך ווערט פארגרעסערט די אויבערשטע שורה מיט די קנעפלעך Integrations, Export און Deploy.

אום 6:36:00 הייסט דער פראיעקט "Professional Meeting Venue Finder App". די שריט:

- Create initial files
- npm install — Install dependencies
- Create src/types/venue.ts

- Create src/data/mockVenues.ts
- Create src/components/SearchForm.tsx
- Create src/components/VenueCard.tsx (נאך אין גאנג)

רעכטס איז דער Preview / Code טייל, מיט "Your preview will appear here".

די פארטיגע אפליקאציע (6:36:30)

Bolt שרייבט: "I've created a comprehensive professional meeting venue finder app designed specifically for business professionals and investor meetings…. די אפליקאציע אנטהאלט: search mit location, date, time און budget filtering; filtering לויט ambiance און amenities; venue information mit photos און ratings; quick booking, favorites system און responsive design.

דער Preview (פארט 5173) ווייזט "Find the Perfect Meeting Venue" — "Professional spaces for investor meetings, business discussions, and executive presentations". אונטער "Find Your Perfect Meeting Venue" זענען פעלדער: Location ("Enter city or area"), Date (dd-mm-yyyy), Time, Budget per Person ($100), א קנעפל Filters און "Search Venues". מען קלויבט אויס א צייט (16:00 און דאטום 20-07-2025; דערנאך ווערט אריינגעשטעלט "new york" און א בודזשעט פון $125.

דער נאמען פון דער אפליקאציע: "Professional Meeting Venues – VenueFind", מיט Favorites, My Bookings און Profile. ביים פראיעקט-נאמען שטייט "This project is private".

Deploy צו Netlify און Export (6:39:00)

Bolt שרייבט: "I'll help you deploy this application to Netlify", מיט די שריט Build application און Deploy to netlify. עס ווייזט זיך אויך א קליינער אנקעטע "How easy is Bolt to use?".

אום 6:39:30 ווערט געוויזן א Windows טעקע Downloads › project-bolt-sb1-fq4tnc5a, מיט א טעקע "project" (03-07-2025 15:03.). אום 6:40:30 ווערט זי געעפנט אין VS Code; די טעקע package-lock.json ווייזט: "name": "vite-react-typescript-starter", "version": "0.0.0"; dependencies: lucide-react^ 0.344.0, react-dom^ 18.3.1, ^18.3.1 devDependencies; צווישן זיי: @eslint/js, @types/react, autoprefixer, eslint, postcss, tailwindcss@ vitejs/plugin-react, typescript^ 3.4.1, vite^ 5.5.3, ^5.4.2. אנדערע טעקעס: eslint.config.js, index.html, package.json, postcss.config.js, tailwind.config.js, tsconfig.app.json.

אום 6:41:00 ווייזט Bolt: npx vite build, Deploy to netlify, Start application, און: npm run dev — "Your site has been successfully deployed! You can view it at: https://silly-cajeta-e9c844.netlify.app". אינעם פראמפט-פעלד ווערט אנגעהויבן: "can add or create an api".

צולייגן OpenAI (6:42:00)

אויף platform.openai.com/api-keys ווערט געמאכט א נייער API שליסל ("API key generated!"). דער פענצטער "Save your key" וואַרנט, מען וועט אים שוין נישט קענען זען נאכאמאל, און Permissions: "Read and write API resources". דער שליסל אליין איז פארוואשן. אין דער רשימה: VC Meet, MCPServer, Godmode, Alexis.

צוריק אין Bolt ווערן באשריבן די נייע טיילן:

- OpenAI Service: Chat completions API integration, Venue recommendation generation, Meeting preparation tips, Structured API calls with error handling
- AI Capabilities: Venue Recommendations, Meeting Tips, Budget Guidance, Etiquette Advice
- Security Considerations: API key validation with helpful error messages, Graceful fallback when API is unavailable, Rate limiting considerations, Environment variable configuration

ביישפיל-פראגעס: "How much should I budget for a professional meeting?", "What should I prepare for a hotel lounge meeting?". אינעם Preview ווייזט זיך: "AI Assistant Unavailable: Please configure your OpenAI API key to enable AI features."

א פראבע מיט Los Angeles (6:43:30)

אין דער פולער Preview ווערט אריינגעשטעלט "Location "los angeles", דאטום 17-07-2025, צייט 18:00, בודזשעט $200; דער רעזולטאט: "4 Venues Found".

אום 6:44:30 ווערט געוויזן דער Code טייל: (src/screens/ProfileScreen.jsאויך FavoritesScreen, HomeScreen, SearchScreen, און src/data מיט mockVenues). דער קאוד דעפינירט א רשימה menuItems, וואו יעדער אינהאלט האט icon, title, subtitle און onPress — למשל 'Edit Profile' ("Update your personal information"), 'My Bookings' ("View your upcoming and past bookings"), און 'Payment Methods'; onPress שרייבט דערווייל נאר א מעלדונג מיט console.log.

פון Bolt-אפליקאציע ביז) Reinforcement Learningלויטן עקראן) (6:45:00)

די "Professional Meeting Venue Finder App" אין Bolt (6:45:00)

אויפן עקראן ווערט געוויזן דער וועבזייטל bolt.new, מיט א פראיעקט וואס הייסט "Professional Meeting Venue Finder App". אויף דער לינקער זייט איז דער שמועס-טייל, און אויף דער רעכטער זייט זענען דא צוויי טאבס: Code און Preview. אויבן רעכטס זענען דא די קנעפלעך Integrations, Export און Deploy.

אין דער לינקער זייט זעט מען, אז Bolt פירט אויס א באפעל npm install, און דערנאך שאפט ער איינס נאכן אנדערן די פיילס פונעם פראיעקט:

- app.json
- babel.config.js
- App.js
- src/data/mockVenues.js
- src/screens/HomeScreen.js
- src/screens/SearchScreen.js
- src/screens/FavoritesScreen.js
- src/screens/ProfileScreen.js

•) src/screens/VenueDetailScreen.jsווערט נאך געשאפן ביים כאפן דעם בילד

אין דער Preview-זייט שטייט נאך: "Your preview will appear here". אונטן איז דא א פראמפט-פענסטער מיט די ווערטער "How can Bolt help you today?".

אנווייזונגען פאר א מאביל-ווערסיע (6:45:30)

א האלבע מינוט שפעטער ווייזט דער שמועס-טייל אן אויסצוג פון וואס Bolt האט געבויט. צווישן די אייגנשאפטן שטייען "Native linking for phone, email, and maps" און "Responsive design for all screen sizes". דערנאך קומט א קאפיטל "How to Use" מיט פיר טריט:

• מען אינסטאלירט Expo Go אויפן טעלעפאן פון App Store אדער Google Play
• מען לאזט לויפן דעם development server מיטן באפעל וואס איז אויבן אנגעוויזן
• מען סקענט דעם QR-קאוד וואס באווייזט זיך אין טערמינאל אדער אין בראוזער
• די אפליקאציע לאדט זיך גלייך אריין אין Expo Go אויפן טעלעפאן

דער טעקסט לייגט צו, אז די אפליקאציע האט די גאנצע פונקציאנאליטעט פון דער וועב-ווערסיע, נאר זי איז איבערגעמאכט ספעציעל פאר מאביל, מיט native interactions, gestures און UI patterns.

Bolt פרובירט פארריכטן פראבלעמען (6:46:00)

אין א קלענערן בילד זעט מען א קעסטל מיט דער מעלדונג "Bolt is trying to resolve the problems". דארט זענען דא א קנעפל "Show problems", די פראגע "Should we try to fix these problems?", און צוויי קנעפלעך: "Attempt fix" און "Ignore". מען זעט אויך די צאל "141K daily tokens remaining".

דער פראיעקט ווערט איבערגעשטעלט אויף Vite React (6:46:30)

דערנאך ווייזט דער שמועס-טייל א רשימה פון פארריכטונגען:

• Fixed dependency conflicts: מען האט אראפגענומען די פראבלעמאטישע React Native פעקלעך (expo-calendar, react-native-maps א.א.וו.), וואס האבן גורם געווען ETARGET טעותים.
• Updated React versions: React און React-DOM זענען געשטעלט געווארן אויף דער זעלבער ווערסיע (18.3.1), כדי אויסצולייזן די ERESOLVE peer dependency warnings.
• Cleaned installation: מען האט אויסגעליידיגט דעם npm cache און אראפגענומען די אלטע node_modules, כדי צו מאכן א פרישע אינסטאלאציע.

צום סוף שטייט, אז דער פראיעקט איז יעצט אויפגעשטעלט ווי א ריכטיגע Vite React וועב-אפליקאציע, מיט Tailwind CSS און Lucide React icons. דער צאל טאקענס איז אראפ צו 138K.

די אפליקאציע אין Preview (6:47:00)

אין Preview באווייזט זיך א מיטל-באניצערשאפט (port 5173) אין א פאָרם פון iPhone 16, ביי 40% גרייס. די צאל טאקענס איז אראפ צו 116K.

ביי 6:47:30 ווערט דער בילד פארגרעסערט ביז 50%. מען זעט א פארם מיטן קעפל "Find Your Perfect Meeting Venue" און א קנעפל Filters. די פעלדער זענען:

• Location, וואו עס ווערט אריינגעשריבן "new"

- Date •
- Time •
- Budget per Person, מיטן וועלט 100

ביי 6:48:00 וועלט געוויזן א קארטל פון א פלאץ:

- נאמען: The Capital Lounge, מיטן צייכן Hotel
- באוועלטונג: 4.8 (342)
- אדרעס: 123 Financial District, 0.8 km, 12 min
- Business: 4.9
- פרייז: $45–$85 per person
- באקוועמלעכקייטן: WiFi, Parking, Private, Meeting Room
- Available slots today: 09:00, 10:30, 14:00, און נאך איינס (+1 more)
- קאנטאקט: capitallounge.com@reservations

אויפן קארטל זענען אויך דא א הארץ-קנעפל (פאר פאוואריטן) און א קנעפל צום טיילן.

ביי 6:48:30 ווייזט די הויפט-בלאט פון דער אפליקאציע דעם נאמען VenueFind און דעם קעפל "Find the Perfect Meeting Venue", מיטן אונטערשריפט "Professional spaces for investor meetings, business discussions, and executive presentations". דער טאב Code וועלט באצייכנט.

פארוואס Reinforcement Learning? (6:49:00)

דערנאך וועלט דער עקראן איבערגעביטן צו א סעריע סליידס איבער Reinforcement Learning. דער ערשטער סלייד הייסט "Why Reinforcement Learning?". אויף אים שטייט: "Learning from a small subset of actions will not help expand the vast realm of solutions that may work for a particular problem". דאס הייסט, אז לערנען פון בלויז א קליינעם טייל פון מעגלעכע אקציעס העלפט נישט אויסצובריטערן די ברייטע וועלט פון לייזונגען וואס קענען ארבעטן פאר א גאוויסער פראבלעם. דער בילד ווייזט ביכער און א פייל וואס גייט צו א ראבאט.

ביי 6:50:00 קומט א ווייטערדיגער טעקסט: "This is going slow the growth that technology is capable of. Machines need to learn to perform actions by themselves and not just learn off humans". דאס הייסט, אז דאס פארשטעלט דעם וואוקס וואס די טעכנאלאגיע איז פעאיג צו דערגרייכן, און מאשינען דארפן לערנען אויספירן אקציעס אליין, נישט בלויז לערנען פון מענטשן. דער בילד ווייזט א ראבאט מיט א פראגע-צייכן, און א בארג מיט א ווייגיקן ראטן וועג ביז א פאן אויפן שפיץ.

וואס איז Reinforcement Learning? (6:50:30)

דער סלייד דעפינירט: Reinforcement learning איז א צווייג פון מאשין-לערנען, וואס טרייגירט א מאדעל צוריקצוגעבן די בעסטע לייזונג פאר א פראבלעם, דורכן אליין נעמען א רייע באשלוסן. אלס ביישפיל: א ראבאט וואס לערנט זיך צו גיין פון איין ארט צום אנדערן. דער בילד ווייזט א ראבאט ביי א פאלעם-בוים, און ווייטער אוועק א קאסטן מיט אוצרות.

א פארגלייך פון דריי סארטן לערנען (6:51:00)

דער סלייד "Supervised vs Unsupervised vs Reinforcement Learning" ווייזט א טאבעלע. דער עקראן בלייבט דערויף בערך דריי מינוט, ביז 6:53:30:

- Supervised Learning : די דאטא איז באצייכנט, מיט אנגעוויזענע אויסגאנג-ווערטן; מען ניצט עס פאר Regression און classification פראבלעמען; מען ניצט labeled data; עס איז דא External Supervision.
- Unsupervised Learning: די דאטא איז נישט באצייכנט, די אויסגאנגען זענען נישט אנגעוויזן, און די מאשין מאכט אליין איר פאָרויסזאָג; מען ניצט עס פאר Association און clustering פראבלעמען; מען ניצט unlabeled data; עס איז נישטא קיין השגחה (No supervision).
- Reinforcement Learning: די מאשין לערנט זיך פון איר סביבה דורך באלוינונגען און טעותים; מען ניצט עס פאר Reward based פראבלעמען; מען ניצט נישט קיין פאָרויס-באשטימטע דאטא; עס איז נישטא קיין השגחה (No supervision).

די לעצטע שורה פון דער טאבעלע איז צוגעדעקט; מען זעט נאר די ווערטער "discovering output".

וויכטיגע באגריפן אין Reinforcement Learning (6:54:00)

דער סלייד "Important Terms in Reinforcement Learning" ברענגט איינס נאכן אנדערן די הויפט-באגריפן, מיטן בילד פון ראבאט, פאלעם-בוים און אוצר-קאסטן:

- Agent: דער מאדעל וואס ווערט געטרייניירט דורך reinforcement learning.
- Action (6:54:30): אלע מעגלעכע טריט וואס דער מאדעל קען נעמען.
- Environment (6:55:00): די טרייניירונג-סיטואציע צו וועלכער דער מאדעל מוז זיך אפטימירן ווערט געפרופן זיין environment.
- Reward (6:55:30): כדי צו העלפן דעם מאדעל גיין אין דער ריכטיגער ריכטונג, ווערט ער באלוינט, דאס הייסט מען גיט אים פונקטן ווען ער טוט א גוטע אקציע. דער ראבאט באקומט דא א מטבע-צייכן.
- Policy (6:56:00): Policy באשטימט ווי אן אגענט וועט זיך פירן אין יעדער צייט; זי ארבעט ווי א מאפע צווישן Action און דעם יעצטיגן State.

א ביישפיל: טרייניירן א הונט (6:56:30)

דער סלייד "Reinforcement Learning Example" הייבט אן מיט: "Consider a dog that we must house train", דאס הייסט, מען שטעלט זיך פאר א הונט וואס מען דארף צוגעווייענען צו פירן זיך אין שטוב.

ביי 6:57:00 שטייט ווייטער: דער הונט וועט נאכגיין א policy כדי צו מאקסימירן זיין באלוינונג, און דערפאר וועט ער פאָלגן יעדן באפעל, און ער קען אפילו אליין אויסלערנען א נייע אקציע, ווי למשל צו בעטן. די בילדער ווייזן א הונט מיט א באל, א הונט וואס שפרינגט ארויף אויף א מענטש, און א הונט וואס זיצט.

Markov's Decision Process (6:58:00)

דער סלייד "Markov's Decision Process" דעפיניירט: דאס איז א Reinforcement Learning policy וואס ווערט געוונצט צו מאפירן דעם יעצטיגן state צו אן action, וואו דער אגענט ארבעט כסדר מיט דער סביבה, כדי צו שאפן נייע לייזונגען און באקומען באלוינונגען. דער דיאגראם ווייזט א קעסטל Agent, פון וועלכן עס

גייט ארויס א פייל באצייכנט Action. צוריק צום Agent גייען פיילן באצייכנט Reward און State. דער אונטערשטער טייל פונעם דיאגראם איז צוגעדעקט.

ביי 6:59:00 קומט אויפן עקראן אויף א קורצער צייט צוריק דער Policy-סלייד, און ביי 6:59:30 קערט מען זיך ווידער אום צום Markov's Decision Process-סלייד מיט דעם זעלבן דיאגראם.

א טיק-טאק-טא שפיל מיט) Reinforcement Learningלויטן עקראן) (7:00:00)

Markov's Decision Process: דער גראף פון A ביז D (7:00:00)

אויפן עקראן ווערט געוויזן א סלייד מיטן טיטל "Markov's Decision Process". אויבן שטייט א קעסטל מיט דער אויפגאבע: אינעם דיאגראם דארף מען געפינען דעם קירצסטן וועג צווישן נאוד A און נאוד D. יעדער וועג האט א "reward" (באלוינונג) וואס איז צוגעבונדן צו אים, און מען וויל אויסקלייבן דעם וועג מיט די גרעסטע באלוינונג.

אין א קעסטל אויף דער לינקער זייט ווערן די באגריפן דערקלערט:

- A, B, C, D זענען די נאודס (פונקטן).
- אריבערגיין פון איין נאוד צום אנדערן (למשל פון A צו B) הייסט אן "action".
- "Reward" איז דער פרייז (cost) אויף יעדן וועג, און "policy" איז יעדער וועג.

אויף דער רעכטער זייט ווערט געוויזן א דיאגראם מיט געלע קרייזן (A, B און ווייטער אונטן) פארבונדן מיט רויטע פונקטירטע ליניעס. אין דעם צווייטן בילד באקומען די ליניעס צאלן: A–B איז 15, א ליניע אראפ פון A איז 5, צוויי קרייצנדיקע ליניעס זענען 20- און 0, און די ליניע אראפ פון B איז 10. דער סלייד בלייבט אויפן עקראן ביז בערך 7:01:00.

דעמא: Tic-Tac-Toe קעגן א מאדעל (7:01:30)

א סלייד מיטן טיטל "Demo" שרייבט אז מען וועט ניצן Reinforcement Learning כדי צו בויען א Tic-Tac-Toe שפיל, און דער באניצער וועט שפילן קעגן דעם מאשין-לערנען מאדעל. אויפן בילד זעט מען א ראבאט, א שפיל-ברעט מיט X און O, און א געצייכנטן מענטש.

Anaconda Navigator (7:02:00)

אויפן עקראן ווערט געוויזן דער Anaconda Navigator מיט אן ענווייראנמענט וואס הייסט "main_py36". די אפליקאציעס וואס מען זעט זענען: JupyterLab, Jupyter Notebook, PyCharm, CMD.exe Prompt, Glueviz, Orange 3, Powershell Prompt, Qt Console, RStudio און Spyder.

דער נאטבוק tic_tac_toe און די אימפארטן (7:02:30)

עס עפנט זיך א Jupyter נאטבוק מיטן נאמען "tic_tac_toe". אין דער ערשטער צעל ווערן אימפארטירט צוויי ביבליאטעקן:

```
import numpy as np

import pickle
```

דער קלאס State (7:03:00)

אין דער צווייטער צעל הייבט זיך אן דער קלאס `State`. די פונקציע `init(self, p1, p2)` שטעלט אויף:

- `self.board = np.zeros((BOARD_ROWS, BOARD_COLS))` – א ברעט פול מיט נולן;
- `self.p1` און `self.p2` – די צוויי שפילער;
- `self.isEnd = False` און `self.boardHash = None`;
- דער קאמענטאר "init p1 plays first" און `self.playerSymbol = 1`.

דערנאך קומט די פונקציע `getHash` (מיטן קאמענטאר "get unique hash of current board state"). זי מאכט דעם ברעט פלאך מיט `reshape(BOARD_COLS*BOARD_ROWS)`, פארוואנדלט עס אין א טעקסט מיט `str` און גיט צוריק `self.boardHash`. אויף די דאזיקע שורות ווערן ארויפגעצייכנט פיילן.

די פונקציע winner (7:03:30) –(7:04:00)

די פונקציע `winner(self)` קוקט צי איינער האט געוואונען:

- "row": פאר יעדע שורה, אויב `sum(self.board[i, :]) == 3` ווערט `isEnd` אויף True און זי גיט צוריק 1. אויב די סומע איז 3- גיט זי צוריק 1-.
- "col": די זעלבע פרובע ווערט געמאכט אויף יעדן קאלום מיט `self.board[:, i]`.
- "diagonal": `diag_sum1` איז די סומע פון `self.board[i, i]`, און `diag_sum2` איז די סומע פון `self.board[i, BOARD_COLS-i-1]`. `diag_sum` איז דער מאקסימום פון ביידע. ביי 3 גיט זי צוריק 1, און ביי 3- גיט זי צוריק 1-.
- "tie": אויב `len(self.availablePositions()) == 0`, איז די שפיל געענדיגט און זי גיט צוריק 0.
- "not end": אנדערש ווערט `isEnd = False` און זי גיט צוריק `None`.

availablePositions, updateState, giveReward (7:04:30) –(7:07:00)

- `availablePositions`: גייט דורך אלע `i` און `j`, און אויב `self.board[i, j] == 0` לייגט זי צו `(i, j)` צו דער ליסטע `positions` (מיטן קאמענטאר "need to be tuple"). צום סוף גיט זי צוריק די ליסטע.
- `updateState(self, position)`: שטעלט `self.playerSymbol = self.board[position]`. דערנאך, מיטן קאמענטאר "switch to another player", ווערט דער סימבאל 1- אויב ער איז געווען 1, און אנדערש ווערט ער 1.
- `giveReward` (מיטן קאמענטאר "only when game ends"): `result = self.winner()`, און דערנאך "backpropagate reward". אויב result איז 1, באקומט p1 דעם `feedReward(1)` און p2 באקומט 0. אויב result איז 1-, באקומט p1 0 און p2 באקומט 1. אנדערש (גלייך) באקומט p1 0.1 און p2 באקומט 0.5.

דערנאך הייבט זיך אן "board reset" מיט `reset`. דער עקראן בלייבט אויף דעם טייל עטליכע מינוט, מיט פיילן אויף `updateState`, `giveReward` און די reward-ווערטן.

די פונקציע play – טרייניר (7:07:30–7:09:30)

`play(self, rounds=100)`: פאר יעדע רונדע, אויב `i%1000 == 0` דרוקט זי "{} Rounds". זי לויפט אזוי לאנג ווי `not self.isEnd`:

- "Player 1": זי נעמט די פרייע פלעצער, `p1.chooseAction(positions, self.board, self.playerSymbol)`, `updateState`, `board_hash = self.getHash()` און `p1.addState(board_hash)`.
- "check board status if it is end": אויב `win` איז נישט `None` (ענדיגט זיך מיט געווינען אדער גלייך), רופט זי `giveReward()`, `p1.reset()`, `p2.reset()`, `self.reset()` און `break`.
- `else`: "Player 2" טוט די זעלבע שריט מיט p2.

play2 – שפילן קעגן א מענטש, און showBoard (7:09:00–7:10:00)

`play2` (מיטן קאמענטאר "play with human"): Player 1 קלייבט אויס, `updateState`, `showBoard()`. אויב `win == 1` דרוקט זי `self.p1.name, "wins!"`, אנדערש "tie!". דערנאך `reset` און `break`. Player 2 רופט `p2.chooseAction(positions)`, און ביי `win == -1` דרוקט זי ווער עס האט געוואונען.

`showBoard` (מיטן קאמענטאר "p1: x p2: o") דרוקט שורות מיט ליניעס, און וועקסלט 1 אויף 'x', 1- אויף 'o' און 0 אויף א לייידיקן פלאץ, מיט " | " צווישן זיי.

דער קלאס Player (7:10:30–7:12:00)

`init(self, name, exp_rate=0.3)`:

- `name`;
- `states = []` (מיטן קאמענטאר "record all positions taken");
- `lr = 0.2`;
- `exp_rate`;
- `decay_gamma = 0.9`;
- `states_value = {}` (מיטן קאמענטאר "state -> value").

`getHash(board)` איז ווי פריער.

`chooseAction(positions, current_board, symbol)`: אויב `np.random.uniform(0, 1) <= self.exp_rate`, נעמט זי א צופעליקע אקציע מיט `np.random.choice`. אנדערש איז `value_max = -999`. פאר יעדן p מאכט זי א קאפיע פונעם ברעט, שטעלט אריין דעם סימבאל, רעכנט אויס דעם hash, און `value` איז 0 אויב עס איז נישט פאראן אין `states_value`, אנדערש דער געהיטענער ווערט. ביי `value >= value_max` ווערט p די אקציע. צום סוף `return action`.

`addState` (מיטן קאמענטאר "append a hash state") לייגט צו דעם state צו דער ליסטע.

(7:14:30) –feedReward, savePolicy (7:12:30

`feedReward(self, reward)` (מיטן קאמענטאר "at the end of game, backpropagate and update" states value"): זי גייט דורך `reversed(self.states)`. אויב א ווערט איז נאך נישטא, שטעלט זי אים אויף 0. דערנאך:

(self.states_value[st] - self.lr*(self.decay_gamma*reward =+ self.states_value[st]

self.states_value[st] = reward

`reset` מאכט ליידיג `states`. `savePolicy` עפנט א טעקע `'policy_' + str(self.name)` אין 'wb' מאדע, שרייבט אריין `states_value` מיט `pickle.dump` און מאכט זי צו. עס הייבט זיך אויך אן `loadPolicy(self, file)`.

אויף דער פארמולע ווערן אויפן עקראן ארויפגעשריבן מיט דער האנט פיילן און צאלן: "1 2 4", "+10", "100", "9 + 10", "9x10.", "2" און אונטערשטרייכונגען אונטער די טיילן פון דער פארמולע.

טיק-טאק-טאו און Q-לערנען (לויטן עקראן) (7:15:00)

דער סוף פונעם Player-קלאס: היטן און לאדן די פאליסי (7:15:00)

אויפן עקראן ווערט געוויזן א Jupyter Notebook מיטן נאמען (tic_tac_toeאין שפיץ שטייט "Last Checkpoint: 41 minutes ago (unsaved changes)", דער קערנעל איז Python 3). מען זעט דעם סוף פונעם קלאס פונעם קאמפיוטער-שפילער. אויבן ענדיקט זיך די פונקציע וואס קלייבט אויס א צוג: אויב value >=value_max, ווערט value_max דער נייער ווערט און action = p. דערנאך קומט א פארמאכטע (קאמענטירטע) print-שורה, און return action.

- addState(self, state), מיטן קאמענטאר "append a hash state" : לייגט צו דעם מצב צו דער ליסטע self.states.
- feedReward(self, reward), מיטן קאמענטאר "at the end of game, backpropagate and update states value": גייט איבער די מצבים פון הינטן אהין (reversed(self.states)). אויב א מצב האט נאך נישט קיין ווערט אין self.states_value, ווערט ער אנגעשטעלט אויף 0. דערנאך ווערט דער ווערט דערהייבט לויט דער פארמל:

(self.states_value[st] - self.lr*(self.decay_gamma*reward =+ self.states_value[st]

נאכדעם ווערט reward דער נייער ווערט פון דעם מצב, און אזוי גייט עס צוריק צום פריערדיקן מצב.

- reset(self): מאכט ליידיג די ליסטע self.states.
- savePolicy(self): עפנט א טעקע מיטן נאמען 'policy_' פלוס דעם שפילערס נאמען, אין מאוד 'wb', און שרייבט אריין self.states_value דורך pickle.dump.
- loadPolicy(self, file): עפנט די טעקע אין מאוד 'rb' און לאדנט אריין self.states_value דורך pickle.load.

דער HumanPlayer-קלאס (7:15:30)

אין א נייער צעל ווערט דעפינירט class HumanPlayer, א שפילער פאר א מענטש:

- init(self, name): היט אפ דעם נאמען.
- chooseAction(self, positions): אין א שלייף while True בעט עס דעם באניצער אריינצושרייבן א שורה ("Input your action row:") און א קאלום ("Input your action col:"), ביידע פארוואנדלט אין int. די צוויי ווערן צוזאמענגעשטעלט אין action = (row, col). אויב די פאזיציע איז צווישן די פרייע positions, ווערט זי צוריקגעגעבן.
- addState, feedReward און reset זענען ליידיג (pass), ווייל א מענטש דארף נישט לערנען.

די צעל ווערט אויסגעפירט (In [5]). דערנאך ווערט אנגעהויבן א צעל מיט p1 = Player("p1") און p2 = Player("p2").

טרייגירן די צוויי קאמפיוטער-שפילער (7:17:00)

די צעל In [6] שטעלט אן BOARD_ROWS = 3 און BOARD_COLS = 3, באשאפט צוויי שפילער p1 און p2, און א שפיל-מצב State(p1, p2) = st. דערנאך דרוקט זי "training..." און רופט st.play(5), אלזא בלויז 5 שפילן. דער רעזולטאט ווייזט "training..." און "Rounds 0".

אין In [8] ווערן ביידע פאליסיס אפגעהיט: p1.savePolicy() און p2.savePolicy().

דערנאך קומט א צעל וואס שטעלט אן p1 = Player("computer", exp_rate=0), לאדנט p1.loadPolicy("policy_p1"), און באשאפט p2 = HumanPlayer("human").

א שפיל קעגן דעם קאמפיוטער נאך ווייניק טרייגירן (7:18:00)

אויפן עקראן ווערט געוויזן דער רעזולטאט פון שפילן: א 3×3 ברעט געצייכנט מיט טעקסט-שורות, מיט x און o. דער באניצער ווערט געבעטן "Input your action row:" און "Input your action col:". למשל ווערט אריינגעשריבן row:0 און col:0, און o ווערט געשטעלט אין ווינקל. דערנאך שטעלט דער קאמפיוטער x, און עס ווערט ווידער געבעטן א שורה.

טרייגירן מיט 50000 שפילן (7:18:30)

די טרייגינג-צעל ווערט געטוישט צו st.play(50000) און ווידער אויסגעפירט (In [*], ד.ה. זי לויפט נאך). דער רעזולטאט ווייזט "training..." און דערנאך שורות "Rounds 0", "Rounds 1000", "Rounds 2000", "Rounds 3000", "Rounds 4000" א.א.וו. ביי 7:20:00 זעט מען דעם סוף פון דער ליסטע, ביז "Rounds 49000". דערנאך ווערן די פאליסיס ווידער אפגעהיט (In [11], שפעטער In [12]: p1.savePolicy(), p2.savePolicy()).

שפילן קעגן דעם טרייגירטן קאמפיוטער (7:20:00)

די שפיל-צעל ווערט אויסגעפירט. איר אינהאלט:

```
p1 = Player("computer", exp_rate=0); p1.loadPolicy("policy_p1")

p2 = Player("human"); st = State(p1, p2); st.play2()
```

דער קאמפיוטער הייבט אן מיט x אין מיטן פונעם ברעט. אין א שפעטערע ווערסיע זעט מען x אין דעם אונטערשטן רעכטן ווינקל, און דער באניצער שרייבט row:1. ביי 7:21:00 זעט מען ווייטער אין דעם שפיל: דער באניצער שרייבט row:2, col:0, און שפעטער row:1, col:0. דער ברעט פילט זיך אן מיט x און o, ביז ער

איז כמעט פול. דערנאך ווערט די צעל ווידער אויסגעפירט (In [13])), און דער עקראן גלייט ארויף צו דער ליסטע פון Rounds. ביי 7:22:00 ווערט דער נאטבוק "autosaved".

וואס איז ?Q-Learning (7:22:30)

עס ווערט געוויזן א סלייד מיטן טיטל "What is Q-Learning?". די דעפיניציע: Q-Learning איז א Reinforcement learning פאליסי וואס געפינט דעם בעסטן קומענדיקן צוג (action), געגעבן דעם איצטיקן מצב (state). עס קלייבט אויס דעם צוג צופעליג (at random) און צילט צו מאקסימירן די באלוינונג (reward). אונטן ווערט געוויזן א דיאגראם מיט א קעסטל "Agent", א פייל "Action" ארויס, און א פייל "Reward" צוריק. דער סלייד בלייבט אויפן עקראן ביז 7:23:00.

ביישפיל: פארשלאגן רעקלאמעס (7:23:30)

דער סלייד "What is Q-Learning? : Ad suggestion" ווייזט א בלעטערער-פענצטער מיט א זוך נאך "Television", א בילד פון א טעלעוויזיע, און לעבן דעם "Advertisement" מיט א טעלעוויזיע-טיש, פאפקארן, א טרינקעכץ און DVD. דער טעקסט: מיט Q-learning קען מען בויען א סיסטעם פאר רעקלאמע-פארשלאגן, וואס שלאגט פאר פראדוקטן פארבונדן מיט א פריערדיקן איינקויף. די באלוינונג איז ווען דער באניצער קליקט אויפן פארשלאג.

עטליכע וויכטיקע טערמינען (7:24:00)

א צווישן-עקראן מיטן טיטל "Some Important Terms" ווייזט א דיאגראם: Agent שיקט Action צום Environment, און דאס Environment גיט צוריק State און Reward.

דער ערשטער סלייד ((7:24:30 גיט דריי באגריפן, יעדער מיט א בילד פון א הונט:

- States: דער מצב S שטעלט פאר וואו דער אגענט געפינט זיך איצט אינעם Environment.
- Action: דער צוג A איז דער טריט וואס דער אגענט טוט ווען ער איז אין א געוויסן מצב.
- Rewards: פאר יעדן צוג באקומט דער אגענט א פאזיטיווע אדער נעגאטיווע באלוינונג.

דער צווייטער סלייד (:7:25:00

- Episodes: ווען דער אגענט קומט אן צו א סוף-מצב (terminating state) און קען מער נישט טון קיין נייעם צוג.
- Q-Values: ווערט געגוצט צו באשטימען ווי גוט א צוג A איז אין א געוויסן מצב S. מען שרייבט עס Q(A,S).
- Temporal Difference: א פארמל צו געפינען דעם Q-Value, דורכן נוצן דעם ווערט פונעם איצטיקן מצב און צוג, און פונעם פריערדיקן מצב און צוג.

Bellman's Equation (7:25:30)

דער סלייד "Bellman's Equation" רעכנט אויס "Factors Influencing Q-values":

- Current State and Action (S, A)
- Previous State and Action (S', A')
- Reward for Action, R
- Maximum expected future reward

די פארמל:

$$Q(S, A)] - Max\ Q'(S', A')\ \gamma + R(S, A)]\ \alpha + Q(S, A) = New\ Q(S, A)$$

פיילן צייכענען אן: Q(S, A) איז "Current Q Value", α איז "Learning Rate", R(S, A) איז "Reward". אונטער γ און Max Q'(S', A') זענען אויך דא פיילן, און אונטן איז צו זען דאס ווארט "Reward". דער סלייד בלייבט אויפן עקראן ביז 7:26:00.

די טריט פון Q-Learning (7:26:30)

נאך א צווישן-עקראן "Steps in Q-Learning" (מיטן זעלבן Agent–Environment דיאגראם) ווערט געוויזן א טאבעלע. אירע קאלומען זענען Fetching, Sitting, Running, און אירע שורות זענען Start, Idle, Wrong Action, Correct Action.

- Step 2 (7:27:00): "Choose an action and perform it. Update values in the table", דאס הייסט קלייב אויס א צוג, פיר אים אויס, און דערהייב די ווערטן אין דער טאבעלע. אין דער טאבעלע איז בלויז Start/Sitting = 1, אלע אנדערע זענען 0.
- Step 4 (7:27:30): "Continue the same until the table is filled or an episode ends", דאס הייסט גיי אזוי ווייטער ביז די טאבעלע איז פול אדער די עפיזאד ענדיקט זיך. די ווערטן זענען:

Start: 5, 7, 10

Idle: 2, 5, 3

Wrong Action: 2, 6, 1

Correct Action: 54, 34, 17

דער סלייד בלייבט אויפן עקראן ביז 7:29:00.

דעמא (7:29:30)

א סלייד מיטן טיטל "Demo" זאגט אן: אין דער דעמא וועט מען נוצן Q-learning צו געפינען דעם קורצסטן וועג צווישן צוויי געגעבענע פונקטן.

א Q-Learning ביישפיל אין Python (לויטן עקראן) (7:30:00)

Anaconda Navigator (7:30:00)

אויפן עקראן ווערט געוויזן Anaconda Navigator, מיט דער סביבה "Applications on" אויסגעקליבן. מען זעט די פארשידענע געצייג: JupyterLab, Jupyter Notebook, PyCharm, CMD.exe Prompt, Glueviz, Orange 3, Powershell Prompt, Qt Console, RStudio און Spyder. ביי JupyterLab שטייט אז עס איז א ברייטערע סביבה פאר אינטעראקטיוו קאמפיוטינג, געבויט אויף Jupyter Notebook. ביי Notebook שטייט אז עס איז א web-באזירטע אינטעראקטיוו נאטבוק-סביבה.

דער נאטבוק Q_learning: סטעיטס און אקציעס (7:30:30)

עס עפנט זיך א Jupyter נאטבוק מיטן נאמען Q_learning, מיט Python 3 . די ערשטע צעל אימפארטירט NumPy:

```
import numpy as np
```

די צווייטע צעל, מיטן קאמענטאר "Define the states", דעפינירט א dictionary מיטן נאמען location_to_state. דארט ווערן ניין ערטער, 'L1' ביז 'L9', פארבונדן מיט נומערן 0 ביז 8. 'L1' איז 0, 'L2' איז 1, און אזוי ווייטער ביז 'L9' איז 8.

די דריטע צעל, "Define the actions", דעפינירט א ליסטע:

```
actions = [0,1,2,3,4,5,6,7,8]
```

דער עקראן בלייבט אויף די צעלן בערך ביז 7:33:00.

די rewards מאטריץ (7:33:30)

די פערטע צעל, "Define the rewards", בויט א 9×9 מאטריץ מיט np.array. יעדע ריי שטייט פאר איין סטעיט. א 1 באדייט אז מען קען גיין צו יענעם סטעיט, און א 0 באדייט אז מען קען נישט. די ריין זענען:

- [0,1,0,0,0,0,0,0,0]
- [1,0,1,0,1,0,0,0,0]
- [0,1,0,0,0,1,0,0,0]
- [0,0,0,0,0,0,1,0,0]
- [0,1,0,0,0,0,0,1,0]
- [0,0,1,0,0,0,0,0,0]
- [0,0,0,1,0,0,0,1,0]
- [0,0,0,0,1,0,1,0,1]
- [0,0,0,0,0,0,0,1,0]

די פאַרקערטע מאפע און פאראמעטערס (7:34:30)

די צעל In [5] (שפעטער In [6]) בויט א פארקערטע dictionary:

```
state_to_location = dict((state,location) for location,state in location_to_state.items())
```

זי פירט יעדן נומער צוריק צום נאמען פונעם ארט. דערנאך קומט דער קאמענטאר "Initialize parameters", מיט gamma = 0.75 און דער באמערקונג "Discount factor".

דער קלאס: init (7:35:30)

דער קאוד דעפינירט א קלאס. אין דער פונקציע ("init", "Initialize alpha, gamma, states, actions, rewards, and Q-values") ווערן אפגעהיט self.location_to_state, self.alpha, self.gamma, self.actions, self.rewards און self.state_to_location. דערנאך:

len(location_to_state) = M

(C' = None, order = np.zeros((M,M), dtype = self.Q

דאס הייסט, די Q-טאבעלע הייבט זיך אן ווי א M×M מאטריץ פון נולן.

די פונקציע training (7:36:00)

דער קאמענטאר זאגט "Training the robot in the environment". די פונקציע ווערט דעפינירט אזוי:

:def training(self, start_location, end_location, iterations)

זי מאכט א קאפיע פון דער rewards מאטריץ מיט np.copy, וואס הייסט rewards_new . דערנאך געפינט זי ending_state דורך [self.location_to_state[end_location. זי שטעלט rewards_new[ending_state, [ending_state = 999, אזוי אז דער ציל-סטעיט באקומט א זייער הויכן reward.

אונטער "picking a random current state" לויפט א לופ for i in range(iterations). אין יעדן דורכלויף ווילט מען א צופעליגן current_state מיט np.random.randint(0,9) , און מען הייבט אן א ליידיגע ליסטע playable_actions = [].

די קאמענטארן זאגן אז מען גייט דורך דער rewards מאטריץ צו געפינען די סטעיטס וואס מען קען דירעקט דערגרייכן פונעם צופעליג אויסגעקליבענעם current_state , און מען שטעלט זיי אריין אין דער ליסטע playable_actions. דער קאוד לויפט for j in range(9). אויב rewards_new[current_state,j] >0, לייגט מען j צו מיט playable_actions.append(j).

אונטער "choosing a random next state" ווערט next_state אויסגעקליבן מיט np.random.choice(playable_actions). פון 7:37:00 ביז 7:39:00 ווערן אויפן עקראן געצייכנט פיילן וואס ווייזן אויף די שורות: דער for-לופ, randint, playable_actions, דער for j לופ, דער תנאי און append. ביי next_state איז אויך געצייכנט א גלאק-פארמיגע ליניע.

Bellman's Equation (7:40:00)

א סלייד מיטן טיטל "Bellman's Equation". אויף דער לינקער זייט שטייט "Factors Influencing Q-values":

- Current State and Action (S, A)
- Previous State and Action (S', A')
- Reward for Action, R
- Maximum expected future reward

די פארמל איז:

$$New\ Q(S,A) = Q(S,A) + \alpha [R(S,A) + \gamma\ Max\ Q'(S',A') - Q(S,A)]$$

די צייכנס אויבן ערקלערן: "Current Q Value" ביי Q(S,A), "Learning Rate" ביי α, און "Reward " ביי R(S,A).

Temporal Difference און דער וועג (7:40:30)

צוריק אינעם קאוד, אונטער "finding temporal difference", ווערט TD אויסגערעכנט ווי self.Q[next_state, מאל self.gamma פלוס rewards_new[current_state,next_state] np.argmax(self.Q[next_state,])[[, מינוס self.Q[current_state,next_state]. דערנאך ווערט די Q-טאבעלע דערהיינטיקט:

TD * self.alpha =+ self.Q[current_state,next_state]

(self.alpha איז אונטערגעשטראכן.)

נאכן לופ שטעלט מען route = [[start_location און start_location = next_location. אונטער "Get the route" רופט מען self.get_optimal_route(start_location, end_location, next_location, route, self.Q).

get_optimal_route (7:41:00)

דער קאמענטאר זאגט "Get the optimal route". די פונקציע get_optimal_route(self, start_location, end_location, next_location, route, Q) לויפט א while-לופ אזוי לאנג ווי next_location != end_location. אין יעדן דורכלויף:

- מען נעמט starting_state פון self.location_to_state[start_location].
- next_state = np.argmax(Q[starting_state,]) קלייבט אויס דעם סטעיט מיטן העכסטן Q-ווערט.
- next_location = (self.state_to_location[next_state]די שורה איז באצייכנט).
- מען לייגט next_location צו צו route.
- start_location ווערט next_location.

צום סוף: print(route).

לויפן דעם אגענט און רעזולטאט (7:42:00)

א נייע צעל:

QAgent(alpha, gamma, location_to_state, actions, rewards, state_to_location) = qagent

qagent.training('L9', 'L1', 1000)

נאכן לויפן (In [8]) ווערט אויסגעדרוקט דער וועג:

['L9', 'L8', 'L5', 'L2', 'L1']

דאס איז דער אפטימאלער וועג פון L9 ביז L1 נאך 1000 איטעראציעס. דער עקראן בלייבט אויף דעם רעזולטאט ביז 7:43:30.

אנהייב פון Deep Learning Tutorial (7:44:00)

א טיטל-בילד: "DEEP LEARNING TUTORIAL", מיט א צייכענונג פון א מוח.

דערנאך קומט א סלייד "Geolocation with Neural Network", מיט דער פראגע "How does a neural network recognize the location of an image?". דער דיאגראם ווייזט פיר שריט:

- "Images of millions of places", א זאמלונג בילדער פון באקאנטע ערטער.
- א פייל צו "Artificial Neural" (א נויראל-נעץ).
- א פייל צו "New image of a place", א בילד מיטן Eiffel Tower.
- א פייל צו "PARIS", מיטן צייכן "Identifies the place".

נויראל-נעצן, פערסעפטראן און לאגישע טויערן (לויטן עקראן) (7:45:00)

געאלאקאציע מיט א נויראל-נעץ (7:45:00)

אויפן עקראן ווערט געוויזן א סלייד מיטן טיטל "Geolocation with Neural Network", מיט דער פראגע: ווי אזוי דערקענט א נויראל-נעץ דעם ארט פון א בילד?

דער ערשטער דיאגראם ווייזט א קייט פון שריט: לינקס א זאמלונג בילדער פון באקאנטע ערטער אויף דער וועלט (Images of millions of places), א פייל צו א מאניטאר מיט א מוח פון ליכטיקע פונקטן (Artificial Neural), ווייטער א פייל צו א נייעם בילד פון אן ארט, דער אייפל-טורעם (New image of a place), און צום סוף דאס וואָרט PARIS, מיט דער באצייכענונג "Identifies the place".

א האלבע מינוט שפעטער ווערט דער דיאגראם פארברייטערט צו פינף שריט:

- Pixels of millions of images – די פיקסלען פון מיליאנען בילדער;
- Feed to Neural Network – מען גיט זיי אריין אינעם נויראל-נעץ;
- Recognizes landmarks, landscapes, architectural styles, road signs, etc – דער נעץ דערקענט באקאנטע געביידעס, לאנדשאפטן, ארכיטעקטור-סטילן, גאס-שילדן און אזוי ווייטער (געוויזן מיט א בילד פון דער ערד-קויל);
- New image of a place – א נייער בילד פון אן ארט;
- Identifies the place – ער אידענטיפיצירט דעם ארט: PARIS.

דער עקראן בלייבט אויף דעם סלייד ביז 7:46:00.

וואס וועט מען לערנען (7:46:30)

א סלייד "What's in it for you?" גיט א רשימה פון די טעמעס:

- What is Deep Learning? – וואס איז דיפ-לערנען?
- Why do we need Deep Learning? – פארוואס דארף מען דיפ-לערנען?
- What is a Neural Network? – וואס איז א נויראל-נעץ?
- What is a Perceptron? – וואס איז א פערסעפטראן?
- Implementing Logic Gates using Perceptron – אויסשטעלן לאגישע טויערן מיט א פערסעפטראן
- Types of Neural Networks – סארטן נויראל-נעצן
- Applications of Deep Learning – אנווענדונגען פון דיפ-לערנען
- Working of Neural Network – ווי אזוי א נויראל-נעץ ארבעט

- Use case implementation using TensorFlow – א פראקטישער ביישפיל מיט TensorFlow

די באלייכטונג גייט אראפ איבער דער רשימה. דערנאך ((7:47:30 קומט א בלויער טיטל-עקראן מיט א מוח-בילד: "What is Deep Learning?"

וואס איז דיפ-לערנען (7:48:00)

דער סלייד גיט די דעפיניציע: דיפ-לערנען איז א טייל פון מאשין-לערנען וואס האט נעצן וועלכע קענען לערנען פון דאטא וואס איז נישט סטרוקטורירט אדער נישט באצייכנט (unstructured or unlabeled), און ארבעט ענליך צום פונקציאנירן פון א מענטשלעכן מוח.

א דיאגראם פון דריי קרייזן איינס אין צווייטן וווייזט די פארבינדונג:

- Artificial Intelligence) דער גרעסטער קרייז) – די פעאיקייט פון א מאשין נאכצומאכן אינטעליגענט מענטשלעך פירונג.
- Machine Learning) אינעווייניג) – אן אנווענדונג פון AI וואס לאזט א סיסטעם אליין לערנען און זיך פארבעסערן פון דערפארונג.
- Deep Learning) אינעם צענטער) – אן אנווענדונג פון מאשין-לערנען וואס ניצט קאמפליצירטע אלגאריטמען און... (דער רעשט פונעם טעקסט איז פארדעקט).

דער סלייד בלייבט ביז 7:48:30.

פארוואס דארף מען דיפ-לערנען (7:49:00)

דער סלייד "Why do we need Deep Learning?" שטעלט אויס פיר סיבות, יעדע מיט אן ערקלערונג:

- Works with unstructured data – מאשין-לערנען ארבעט נאר מיט גרויסע סעטס סטרוקטורירטע דאטא, בשעת דיפ-לערנען קען ארבעטן סיי מיט סטרוקטורירטע סיי מיט נישט-סטרוקטורירטע דאטא.
- Handle complex operations – דיפ-לערנען אלגאריטמען קענען גרינג דורכפירן קאמפליצירטע אפעראציעס, וואס מאשין-לערנען אלגאריטמען קענען נישט.
- Feature Extraction – מאשין-לערנען אלגאריטמען ניצן באצייכנטע ביישפיל-דאטא צו ארויסציען מוסטערן, בשעת דיפ-לערנען נעמט אן גרויסע מאסן דאטא אלס אינפוט און אנאליזירט דעם אינפוט כדי ארויסצוציען... (דער סוף פארדעקט).
- Achieve best performance – די לייסטונג פון מאשין-לערנען אלגאריטמען פאלט וועג די מאס דאטא וואקסט, דעריבער, כדי צו האלטן די לייסטונג פונעם מאדעל, דארף מען... (דער סוף פארדעקט).

דער עקראן בלייבט דא ביז 7:49:30.

וואס איז א נויראל-נעץ (7:50:00)

דער סלייד דעפינירט: נויראל-נעצן זענען געבויט לויטן מוסטער פון ביאלאגישע נויראל-נעצן, וואס לאזן קאמפיוטערס לערנען און זיך באנעמען ווי מענטשן. עס האט פארבונדענע נוירענען מיט דענדריטן וואס נעמען אן אינפוטס, און לויט די אינפוטס פראדוצירט עס אן עלעקטרישן סיגנאל, ד.ה. אן אויטפוט דורכן אקסאן. דערביי ווערט געווויזן א בילד פון א ביאלאגישן נוירען.

ביאלאגישער קעגן קינסטלעכן נויראן (7:50:30)

דער סלייד "Biological Neurons Vs Artificial Neuron" גלייכט די טיילן:

- Dendrites ------- >Inputs – די דענדריטן פונעם ביאלאגישן נויראן שטימען מיט די אינפוטס (Input 1, Input 2…) פונעם קינסטלעכן נויראן.
- Synapse ------- >Weights – די סינאפס שטימט מיט די וואגן (w1, w2 … wn.

וואס איז א פערסעפטראן (7:51:30)

דעפיניציע אויפן סלייד: א פערסעפטראן איז דער גרונט-טייל פון א נויראל-נעץ; ער שטעלט פאר איין איינציגן נויראן פונעם מענטשלעכן מוח און ווערט געוונט פאר בינארע קלאסיפיקאטארן.

דער דיאגראם: אינפוטס (x1, x2און מער, באצייכנט מיט פונקטן) גייען מיט וואגן w1, w2 אריין אין א Σ – "Net Input Function", פון דארט צו אן "Activation Function" (א טרעפ-פארעם), און ארויס צום אויטפוט Y. פון אויטפוט גייט א פייל "ERROR" צוריק צו די וואגן. דער סלייד בלייבט ביז 7:52:30.

דער פערסעפטראן לערן-כלל (7:53:00)

דער סלייד "Perceptron Learning Rule" ווייזט די שריט:

- Step 1 – די אינפוטס x1…(Input 1 xn (Input n … ווערן געמערט מיט די וואגן w1…wn; זיי קומען צוזאמען אין דער (Net Input FunctionΣ), צוזאמען מיט א (Bias.דער טעקסט הייבט אן "Inputs are multiplied with the…", מיט א סומע-צייכן Σ ביז (n. דער עקראן בלייבט דא ביז 7:54:00.
- Step 3 (7:54:30) – דער פערסעפטראן באקומט אינפוט-סיגנאלן, און אויב די סומע פון די סיגנאלן איז גרעסער ווי א געוויסער שוועל (threshold), גיט ער ארויס א סיגנאל, אדער ער גיט נישט קיין אויטפוט. די פארמל: דער אויטפוט איז 0 אויב Σ(wi*xi (i=1..n) + b <threshold.
- Step 4 (7:55:00) – "The error in the output is…" – דער פעלער פונעם אויטפוט גייט צוריק (ERROR-פייל) צו די וואגן w1, w2, wn, וואס זענען באלייכטעט.
- ((7:55:30 – דער גאנצער דיאגראם ווערט ווידער געוויזן, מיטן טעקסט "Repeat steps 1 to 4 in order to…" – מען איבערחזרט שריט 1 ביז 4.

לאגישע טויערן מיט א פערסעפטראן (7:56:00)

דער סלייד "Implementing Logic Gates using Perceptron" דערציילט: אין 1943 האבן דער נוירא-וויסנשאפטלער Warren McCulloch און דער לאגיקער Walter Pitts פארעפנטלעכט א ארבעט וואס ווייזט ווי נויראנען קענען אויספירן די דריי לאגישע פונקציעס (AND, OR, XOR). די נויראנען וואס זיי האבן געוונט זענען געווען פשוטע שוועל-נויראנען, וואס זיי האבן אנגערופן פערסעפטראן.

דער קומענדיגער סלייד ((7:56:30 דעפינירט: א לאגישער טויער איז דער גרונט-בוי-שטיין פון א דיגיטאלן קרייז. רוב טויערן האבן 2 אינפוטס און 1 אויטפוט. אין יעדן מאמענט איז יעדער טערמינאל אין איינעם פון צוויי בינארע מצבים: נידעריג (0) אדער הויך (1), וואס ווערן פארגעשטעלט דורך פארשידענע וואלטאזש-מדרגות. די פאפולערע טויערן: AND, OR, NAND, NOR, NOT; א קליין בילד ווייזט אויך XOR און XNOR מיט זייערע סימבאלן.

AND טויער (7:57:00)

אויבן שטייט: מיט די לאגישע טויערן קענען נויראל-נעצן אליין לערנען, אן דארפן מאנועל שרייבן די לאגיק אין קאוד. די טאבעלע פון) ANDאויטפוט AB):

- 0 → 0 ,0
- 0 → 1 ,0
- 0 → 0 ,1
- 1 → 1 ,1

אויפן קומענדיגן סלייד (A :(7:57:30 און B זענען אינפוט-נוראנען, דער רויטער נוירען איז דער אויטפוט-נוירען, דער שוועל (θ) איז 1. אויב די סומע פון די געוואוגענע אינפוטס איז גרעסער ווי דער שוועל, איז דער אויטפוט 1, אנדערש 0. ביידע וואגן זענען 0.7. די טאבעלע w2*x2 + w1*x1:

- 0 = 0*0.7 + 0*0.7
- 0.7 = 1*0.7 + 0*0.7
- 0.7 = 0*0.7 + 1*0.7
- 1.4 = 1*0.7 + 1*0.7 – דאס גייט איבער דעם שוועל, דעריבער איז דער אויטפוט 1.

OR טויער (7:58:00)

די טאבעלע פון) ORאויטפוט 0,0 :(A+B → 0; 1 → 0,1; 1 → 1,0; 1 → 1,1.

מיט וואגן 1.2 יעדער און שוועל θ = :(7:58:30) 1

- 0 = 0*1.2 + 0*1.2
- 1.2 = 1*1.2 + 0*1.2
- 1.2 = 0*1.2 + 1*1.2
- 2.4 = 1*1.2 + 1*1.2

די לעצטע דריי גייען איבער דעם שוועל, דעריבער איז דער אויטפוט 1.

XOR טויער (7:59:00)

די טאבעלע פון) XORאויטפוט A⊕0,0 :(B → 0; 1 → 0,1; 1 → 1,0; 1 → 1,1.0

דער סלייד ביי 7:59:30 זאגט: א XOR טויער פארלאנגט א צווישנדיגע באהאלטענע שיכט (hidden layer) כדי צו דערגרייכן זיין לאגיק. דער דיאגראם ווייזט אינפוטס x1, x2, באהאלטענע נוירענען h3, h4, און אויטפוט O5, מיט וואגן w13, w14, w23, w24, w35, w45. אונטער "Implementation":

- X1 און X2 זענען אינפוטס, h3 און h4 זענען די באהאלטענע שיכטן; B3, B4 און B5 זענען די biases פאר h3, h4 און O5.
- (B3 – X2*w23 + sigmoid(X1*w13 = h3
- (B4 – X2*w24 + sigmoid(X2*w14 = h4
- (B5 – h4*w45 + sigmoid(h3*w35 = O5

נויראל-נעצן: XOR, סארטן און ווי זיי ארבעטן (לויטן עקראן) (8:00:00)

דער XOR-טויער מיט א באהאלטענער שיכט (8:00:00)

אויפן עקראן ווערט געוויזן א סלייד מיטן טיטל "Implementing XOR Gate". אין א בלויער קעסטל שטייט: "XOR gate requires an intermediate hidden layer to achieve the logic of a XOR gate", דאס הייסט, א XOR-טויער פאדערט א באהאלטענע שיכט (hidden layer) אין מיטן, כדי צו קענען דערגרייכן די לאגיק פון XOR.

אויף דער לינקער זייט ווערט געוויזן א דיאגראם פון א קליינעם נויראל-נעץ:

- צוויי אריינגאבן: x1 און x2.
- צוויי באהאלטענע נויראנען: h3 און h4.
- איין אויסגאב-נויראן: o5.
- פון x1 צו h3 איז די וואג 20, און פון x1 צו h4 איז זי -20.
- פון x2 צו h3 איז די וואג 20, און פון x2 צו h4 איז זי -20.
- פון h3 צו o5 איז די וואג 20, און פון h4 צו o5 אויך 20.
- ביי h3 שטייט b = -10 און θ = 1. ביי o5 שטייט b = -30 און θ = 1.

אויף דער רעכטער זייט זענען דא טאבעלעס מיט חשבונות דורך דער Sigmoid-פונקציע.

טאבעלע h3:

- 0 ≈ Sigmoid(20*0+20*0-10)
- 1 ≈ Sigmoid(20*1+20*1-10)
- 1 ≈ Sigmoid(20*0+20*1-10)
- 1 ≈ Sigmoid(20*1+20*0-10)

טאבעלע h4:

- 1 ≈ Sigmoid(-20*0-20*0+30)
- 0 ≈ Sigmoid(-20*1-20*1+30)
- 1 ≈ Sigmoid(-20*0-20*1+30)
- 1 ≈ Sigmoid(-20*1-20*0+30)

אונטער "Output" זעט מען צוויי שורות:

- 0 ≈ Sigmoid(20*0+20*1-30)
- 0 ≈ Sigmoid(20*1+20*0-30)

די איבעריקע שורות פון דער Output-טאבעלע זענען צוגעדעקט און מען קען זיי נישט ליינען. דער סלייד בלייבט אויפן עקראן ביז 8:00:30.

סארטן נויראל-נעצן (8:01:00)

עס קומט א טיטל-בילד "Types of Neural Network" מיט א ציור פון א מוח אויף א בלויען הינטערגרונט.

ביי 8:01:30 ווערט געוויזן דער סלייד "Types of Neural Networks". אויבן שטייט: "Neural Networks are mainly classified into 5 types", דאס הייסט, נויראל-נעצן ווערן הויפטזעכלעך איינגעטיילט אין פינף סארטן. יעדער סארט ווערט באגלייט מיט א ביישפיל פון א באניץ:

- Artificial Neural Network – ANN; באניץ:) Pattern Recognitionדערקענען מוסטערן).
- Convolution Neural Network – CNN; באניץ:) Image Processingבאארבעטן בילדער).
- Recursive Neural Network – RNN; באניץ:) Speech Recognitionדערקענען רייד).
- Deep Neural Network – DNN; באניץ:) Acoustic Modelingקלאנג-מאדעלירונג).
- Deep Belief Network – DBN; באניץ:) Cancer Detectionאויסגעפינען קענסער).

באנוצונגען פון דיפ-לערנען (8:02:00)

א סעריע סליידס מיטן טיטל "Applications of Deep Learning" ווייזט, יעדער מאל, א בילד מיט אן אונטערשריפט:

- "Playing Games": א ראבאט וואס קיקט א פוסבאל. דער סלייד בלייבט ארום א מינוט.
- "Autonomous Driving Cars": דאס אינעווייניגסטע פון א קאר וואס פארט אליין, מיט דעם אויפשריפט "Self Driving" (8:03:00.)
- "Building Robots": א ראבאט וואס ארבעט ביי א טיש (8:03:30.)
- "Medical Diagnosis": א פיגור מיט א סטעטאסקאפ (8:04:00, ביז 8:04:30.)

ווי א נויראל-נעץ ארבעט: דערקענען פארמען (8:05:00)

דער סלייד "Working of Neural Network" שטעלט פאר די פראגע: "Let's find out how an Artificial Neural Network can be used to identify different shapes like Squares, Circles and Triangles", דאס הייסט, לאמיר זען ווי אזוי א קינסטלעכער נויראל-נעץ קען דערקענען פארשידענע פארמען, ווי קוואדראטן, קרייזן און דרייעקן. א פיגור טראכט: "How to identify various shapes using a neural network?", און א פייל ווייזט צו אן עקראן מיט פארמען, וואס איז אנגעצייכנט "Artificial Neural Network".

ביי 8:05:30 שטייט "Lets consider the shape of a square", דאס הייסט, לאמיר נעמען די פארעם פון א קוואדראט. אין דער מיט ווערט געוויזן א נעץ פון 28 אויף 28 קעסטלעך, מיט א קוואדראט אין מיטן. אויף דער רעכטער זייט, אין א קעסטל "Strengths", שטייט: "28*28 pixels of the input image is taken as input i.e. 784 neurons", דאס הייסט, די 28×28 פיקסלען פונעם בילד ווערן געגומען אלס אריינגאב, אזוי אז מען האט 784 נויראנען.

ביי 8:06:30 קומט צו א צווייטער פונקט: "Each neuron holds a number called Activation that represents grayscale value of the corresponding pixel ranging from 0 to 1. 1 for white pixel and 0 for black pixel". דאס הייסט, יעדער נויראן האלט א צאל וואס הייסט Activation, און זי שטעלט

פאר דעם גרוי-ווערט פונעם פאסיגן פיקסל, פון 0 ביז 1: א ווייסער פיקסל איז 1, א שווארצער פיקסל איז 0. א ליניע פארבינדט איין פיקסל אינעם קוואדראט מיטן טעקסט.

די שיכטן פונעם נעץ און דער אויסגאב (8:07:00)

אויף א שווארצן הינטערגרונט ווערט געוויזן א נעץ מיט 784 אריינגאב-נויראנען, צוויי באהאלטענע שיכטן, און דריי אויסגאבן: CIRCLE, SQUARE, TRIANGLE. אין דעם ווינקל זעט מען א בילד פון 28×28 מיטן אונטערשריפט "784 Pixels".

- ווען דער אריינגאב איז א קוואדראט, ליכטיגן זיך אן געוויסע נויראנען אין גרין, און דער אויסגאב SQUARE איז ארומגערינגלט (8:07:00 ביז 8:07:30).
- ביי 8:08:00 איז דער אריינגאב א קרייז; די אנגעצונדענע נויראנען זענען בלוי, און CIRCLE איז ארומגערינגלט.
- ביי 8:08:30 ווערט געוויזן א גרויס בילד פון א דרייעק אויף א נעץ פון 28×28.
- ביי 8:09:00 ווערט דער דרייעק אריינגעגעבן אינעם נעץ, נאך אן קיין אנגעצונדענע נויראנען.

די צוויי טריט אין א נויראן (8:09:30)

דער סלייד ווייזט ווי אזוי די פיקסלען פון בילדער ("Pixels of images fed as input") קומען אריין ווי x1, x2 און ווייטער, יעדער מיט א וואג w1, w2 און אזוי ווייטער.

- Step 1: מען רעכנט אויס די סומע $\sum$ (פון i=1 ביז n) פון $w_i x_i + b$, דאס הייסט, יעדער אריינגאב ווערט געמערט מיט זיין וואג, אלעס ווערט צוזאמגערעכנט, און מען לייגט צו דעם b.
- Step 2: מען לאזט די סומע דורך א פונקציע σ, דאס הייסט, $\sigma(\sum w_i x_i + b)$.

דער רעזולטאט טיילט די פארמען אין גרופעס: Squares, Circles, Triangles. דער סלייד בלייבט אויפן עקראן ביז 8:11:30.

וואגן, אקטיוויזאציע-פונקציעס, און פארגלייכן (8:12:00)

- ביי 8:12:00 ווערט געוויזן א נעץ מיט וואג-קאלאנעס צווישן די שיכטן: W1, W2, W3; דערנאך W11, W21, W31; און W22, W23, W33. צום סוף קומט דער אויסגאב $\hat{Y}$. אונטן שטייט "Apply the weights", דאס הייסט, מען וועגדט אן די וואגן.
- ביי 8:12:30 ביז 8:13:00 באקומט יעדער נויראן א קליין גראפיק פון אן אקטיוויזאציע-פונקציע, און אונטן שטייט "Apply the activation functions".
- ביי 8:13:30 ווערט $\hat{Y}$ ("Predicted Output", דער פאראויסגעזאגטער אויסגאב) פארגליכן מיט Y ("Actual Output", דער אמת'ער אויסגאב), מיטן אויפשריפט "Compare it with actual output".
- ביי 8:14:00 שטייט: "Applying the cost function to minimize the difference between predicted and actual output using gradient descent algorithm", דאס הייסט, מען וועגדט אן די cost function כדי צו פארקלענערן דעם אונטערשייד צווישן דעם פאראויסגעזאגטן און דעם אמת'ן אויסגאב, מיט'ן gradient descent אלגאריטם.

די Cost Function (8:14:30)

דער סלייד "Cost Function" דעפינירט: "The Cost value is the difference between the neural nets predicted output and the actual output from a set of labelled training data. The least cost value is obtained by making adjustments to the weights and biases iteratively throughout the training process."

דאס הייסט, דער Cost-ווערט איז דער אונטערשייד צווישן דעם אויסגאב וואס דער נעץ זאגט פאראויס און דעם אמת'ן אויסגאב, לויט א סעט באצייכנטע טריינירונגס-דאטא. דעם קלענסטן Cost-ווערט באקומט מען דורך פארראכטן די וואגן און די biases, איין מאל נאכן אנדערן, במשך דעם גאנצן טריינירן.

אינעם דיאגראם גייען די פיילן צוריק, פון Y צו $\hat{Y}$ און דורך די וואג-קאלאנעס W צוריק צו די אריינגאבן.

גראדיענט-אראפגאנג, TensorFlow און MNIST) לויטן עקראן) (8:15:00)

Gradient Descent: דער בארג-משל (8:15:00)

אויפן עקראן ווערט געוויזן א סלייד מיטן טיטל "Gradient Descent". אין א בלויער קעסטל שטייט די דעפיניציע: Gradient Descent איז אן אפטימיזאציע-אלגאריטם צו געפינען דעם מינימום פון א פונקציע.

אויף דער לינקער זייט איז א בילד פון א גרינעם בארג מיט פעלזן, און א פיגור וואס קריכט ארויף. אויף דער רעכטער זייט זענען צוויי גרינע דרייעקן: אויף דעם ערשטן, מיט א שטילן שיפוע, גייט א מענטש אראפ, אנגעצייכנט "Slower". אויף דעם צווייטן, מיט א שטאציקן שיפוע, גייט ער אראפ, אנגעצייכנט "Faster". אונטן שטייט אין א קעסטל אז דער מענטש וועט דארפן מער צייט צו דערגרייכן דעם פוס פונעם בארג וועו דער שיפוע איז שטיל, און וועט אראפקומען שנעלער וועו דער שיפוע איז שטאציק (דער סוף פונעם זאץ איז פארדעקט). דער סלייד בלייבט אויפן עקראן ביז 8:16:00.

דער גראפיק פון Cost און Weight (8:16:30)

דער זעלבער טיטל און די זעלבע דעפיניציע בלייבן, אבער אונטן קומען צוויי גראפיקן. אויפן לינקן, מיט דער אקס C, ליגן געלע קרייצלעך אין א פארעם פון א "U"; דאס קרייצל צום נידעריקסטן פונקט איז ארומגערינגלט און אנגעצייכנט "Best". א פייל ווייזט צום רעכטן גראפיק: א קורוו אין פארעם פון א "U", מיט קיילעכלעך וואס גייען אראפ צום דנאָ. דאס אויבערשטע קיילעכל איז אנגעצייכנט "Initial Weight" (דער אנהייב-וואג), און ביי יעדן קיילעכל ווייזט א שטריך דעם שיפוע, אנגעצייכנט "Gradient". פיילן ווייזן דעם וועג פון ביידע זייטן צום מינימום. אין א קעסטל דערנעבן שטייט די פארמל פון דער Cost-פונקציע:

$$C = ½(Y - \hat{Y})^2$$

Stochastic Gradient Descent (8:17:30)

א נייער סלייד מיטן טיטל "Stochastic Gradient Descent". דער גראפיק ווייזט א קורוו מיט דער אקס C (ארויף) און W (אין דער בריט). די קורוו האט צוויי טאלן: א לאקאלן טאל, וואו קיילעכלעך בלייבן שטעקן, און א טיפערן טאל מיט א רויטן קיילעכל, אנגעצייכנט "Best". נעבן דעם זענען דריי קעסטלעך:

- עס פארלאנגט נישט אז די Cost-פונקציע זאל זיין קאנוועקס (convex).

- עס נעמט די ריי-ען (rows) איינס נאך דאס אנדערע, לאזט לויפן דעם נויראל-נעץ און דערנאך פארריכטעט עס די וואגן (weights).
- עס העלפט אויסמיידן דעם לאקאלן מינימום, ווייל עס טוט 1 איטעראציע אויף איין מאל.

פלאטפארמעס פאר דיפ-לערנען (8:18:00)

דער סלייד "Deep Learning Platforms" ווייזט א מוח אין דער מיט, און אויבן שטייט "4 main platforms". ארום דעם מוח זענען אנגעצייכנט: TensorFlow (מיט איר לאגא), Keras (מיט דער לאגא און די ווערטער "A deep learning library"), Torch, און DeepLearning4J (java).

הקדמה צו TensorFlow: וואס איז א טענזאר (8:18:30)

דער סלייד "Introduction to TensorFlow" האט דריי פונקטן:

- TensorFlow איז א געצייג פאר דיפ-לערנען, צו דעפינירן און לויפן חשבונות וואס באטרעפן טענזארן.
- א טענזאר איז א פאראלגעמיינערונג פון וועקטארן און מאטריצן צו אפשר העכערע דימענסיעס.
- דער "array of data" (די ריי דאטא) וואס ווערט דורכגעגעבן אין יעדער שיכט פון נאָדעס, הייסט א טענזאר.

אונטן זענען דריי בילדער: לינקס א זייל מיט די אותיות (a, d, k, m וועקטאר); אין דער מיט א קוביק פון קאלירטע קלעצלעך (א דריי-דימענסיאנעלער טענזאר); רעכטס א טאבעלע פון ציפערן (א מאטריץ), מיט די ריי-ען: 2 9 4 7; 1 4 3 5; 8 2 6 2; 4 3 6 8. דער סלייד בלייבט ביז 8:19:30.

ביי 8:20:00 ווערט אונטער די זעלבע פונקטן געוויזן א בילד פון א נויראל-נעץ: לינקס א טאבעלע מיט ציפערן (1.2, 3.5, 2.2; 4.7, 10, 4.5; 2.2, 7.8, 8.1; 3.6, 2.5, 4.5), אנגעצייכנט "Arrays of data with" (דער רעשט פארדעקט). די טאבעלע פירט אריין אין גרויע נאָדעס, פון זיי אין א שיכט בלויע נאָדעס, און צום סוף אין איין רויטן אויסגאנג-נאָד.

די ראנגען פון טענזארן (8:20:30)

דער סלייד "Tensor Ranks" ווייזט די TensorFlow לאגא אין דער מיט, מיט פיר ביישפילן:

- s: Tensor of Rank 0 = [107]
- v: Tensor of Rank 1 = [10,20,30]
- m: Tensor of Rank 2 = V = [1,2,3],[4,5,6]
- t: Tensor of Rank 3 = [[[1],[2],[3]],[[4],[5],[6]],[[7],[8],[9]]]

דער סלייד בלייבט ביז 8:21:00.

די מעלות פון TensorFlow (8:21:30)

נאכאמאל א סלייד "Introduction to TensorFlow", הפעם ווי א האלבע רעדל ארום דער TensorFlow לאגא, מיט פינף פונקטן:

- די מערסט פאפולערע ביבליאטעק אין דיפ-לערנען.
- אן אפן-קוואל (open source) ביבליאטעק, אנטוויקלט דורך Google.

- קען לויפן אי אויף א CPU אי אויף א GPU.
- קען שאפן דאטא-פלוס גראפן (data flow graphs) וואס האבן נאָדעס און קאנטן (edges).
- ווערט געוווצט פאר מאשין-לערנען אפליקאציעס, למשל נויראל-נעצן.

דער סלייד בלייבט אויפן עקראן ביז 8:22:30.

א פראקטישער ביישפיל: MNIST (8:23:00)

דער סלייד "Use case Implementation using TensorFlow" זאגט: לאמיר בויען א נויראל-נעץ צו דערקענען האנט-געשריבענע ציפערן מיט דער MNIST דאטאבאזע. לינקס איז א בילד פון א סך האנט-געשריבענע ציפערן פון 0 ביז 9, אנגעצייכנט "Hand written digits from MNIST Database". רעכטס דריי קעסטלעך:

- Modified National Institute of Standards and Technology Database) דער פולער נאמען פון MNIST).
- האט א זאמלונג פון 70,000 האנט-געשריבענע ציפערן.
- די לעיבלס (digit labels) באצייכענען יעדע ציפער פון 0 ביז 9.

דער סלייד בלייבט ביז 8:24:30.

ביי 8:25:00 ווייזט דער זעלבער טיטל א פראצעס אין פיר טריט: האנט-געשריבענע ציפערן פון MNIST; די פיקסלען פון די ציפערן ווערן אריינגעגעבן אין נעץ; א נייע בילד פון א ציפער (א "3"); דער נעץ דערקענט די ציפער — און א בלאָז-קעסטל זאגט "That's digit 3".

די Softmax פונקציע (8:25:30)

דער סלייד "Softmax Function" ווייזט א בילד פון א האנט-געשריבענער "9" אויף א שווארצן הינטערגרונט. א פייל פירט צו א ריי קעסטלעך — די פיקסלען — מיט דער אויפשריפט 28*28=784 pixels. יעדער פיקסל איז פארבונדן מיט שטריכן צו א ריי נאָדעס אונטן, אנגעצייכנט 0, 1, 2 ... 9 — איין נאָד פאר יעדער ציפער. דער סלייד בלייבט ביז 8:27:00.

שריט 1: אריינברענגען די MNIST דאטא (8:27:30)

דער סלייד "Use case Implementation using TensorFlow" ווייזט קאוד אונטערן קעפל "1. Import MNIST Data using TensorFlow":

```
import tensorflow as tf

from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

די ערשטע שורה ברענגט אריין TensorFlow אונטערן נאמען tf. די צווייטע ברענגט אריין input_data פון די MNIST ביישפילן. די דריטע לאדנט אראפ די דאטא-סעטס אין דעם פאלדער MNIST_data , מיט one_hot=True. דער אויסגאנג ווייזט:

- train-images-idx3-ubyte.gz — 9912422 bytes, אראפגעלאדנט און אויסגעפאקט.
- train-labels-idx1-ubyte.gz — 28881 bytes.

- 1648877 bytes — t10k-images-idx3-ubyte.gz.
- 4542 bytes — t10k-labels-idx1-ubyte.gz.

יעדע טעקע ווערט אויסגעפאקט אין /MNIST_data. אונטן הייבט זיך אן "2. Check the type of Dataset" מיט א)type... שורה, אבער דער רעשט איז פארדעקט.

ביי 8:29:00 ווערט נעבן דעם אויסגאנג געצייכנט א קלאמער, און ביי 8:29:30 ווערן אין איר אריינגעשריבן א ריי נולן און איין "1" — א בילד פון א one-hot וועקטאר — מיט א נאָטיץ נעבן דער שורה פון train-labels.

MNIST-האנטשריפט מיט) TensorFlowלויטן עקראן) (8:30:00)

סליידס: אימפארטירן די MNIST דאטא (8:30:00)

דער סלייד טראגט דעם טיטל "Use case Implementation using TensorFlow". ערשטער שריט: "Import MNIST Data using TensorFlow".

- import tensorflow as tf – מען אימפארטירט TensorFlow אונטער דעם נאמען tf.
- from tensorflow.examples.tutorials.mnist import input_data – מען אימפארטירט דעם input_data מאדול פון די TensorFlow טוטאריאלס.
- mnist = input_data.read_data_sets("MNIST_data/", one_hot=True) – מען לאדנט אראפ דעם דאטא-סעט אין א פאלדער MNIST_data, מיט one_hot=True)אונטערגעשטראכן).

דער אויסגאנג ווייזט אז פיר טעקעס ווערן אראפגעלאדנט און ארויסגעצויגן: train-images-idx3-ubyte.gz (9912422 bytes), train-labels-idx1-ubyte.gz (28881 bytes), t10k-images-idx3-ubyte.gz (1648877 bytes), און t10k-labels-idx1-ubyte.gz (4542 bytes). דערנעבן איז אנגעצייכנט מיט דער האנט א זייל פון נולן מיט איין 1, וואס באווייזט ווי אזוי א one-hot ליבל זעט אויס. דער צווייטער שריט, "Check the type of Dataset", הייבט זיך אן מיט type(...), אבער דער רעזולטאט איז פארדעקט.

סליידס: די בילדער און זייער צאל (8:30:30)

שריט 3, "Array of Training images": mnist.train.images גיט צוריק אן array וואס באשטייט כמעט נאר פון נולן (0.), מיט dtype=float32.

שריט 4, "Number of images for Training, Testing and Validation":

- mnist.train.num_examples → 55000
- mnist.test.num_examples → 10000
- דער דריטער צעל (validation) → 5000

די ווערטער train און test זענען אונטערגעשטראכן.

סלייד: אויסווייזן א בילד (8:31:30)

דער קאוד plt.imshow(mnist.train.images[1].reshape(28,28), cmap='gist_gray') פארמט איבער דאס בילד צו 28 אויף 28 פיקסלען און ווייזט עס אין גרויע פארבן. דער אויסגאנג איז א matplotlib.image.AxesImage, און עס ווערט געווייזן א ווייסע האנטגעשריבענע ציפער 3 אויף א שווארצן

הינטערגרונט, מיט אקסן פון 0 ביז 25. שריט 6, "Maximum and minimum value of the pixels in the image", ווייזט דעם רעזולטאט 1.0.

סלייד: בויען דעם מאדעל (8:32:00)

שריט 7, "Create the Model". דער עקראן בלייבט דא עטלעכע מינוט, מיט אונטערשטרייכונגען אויף placeholder און Variable:

- x = tf.placeholder(tf.float32, shape=[None,784]) – אן אריינגאנג פאר א באליביקע צאל בילדער, יעדעס מיט 784 פיקסלען.
- א קאמענטאר "10 because 0-9 possible numbers", און W = tf.Variable(tf.zeros([784,10])) – א מאטריץ פון וואגן, אנגעפילט מיט נולן.
- b = tf.Variable(tf.zeros([10])) – דער bias, צען נולן.
- "Create the Graph": y = tf.matmul(x,W) + b – מאטריץ-מולטיפליקאציע פון x מיט W, פלוס b.

אונטער "Loss and Optimizer":

- y_true = tf.placeholder(tf.float32,[None,10]) – די ריכטיגע ליבלס.
- "Cross Entropy": cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_true, logits=y)).

אונטן זעט מען דעם אנהייב פון א צעל מיט optimizer און א צעל מיט train, אבער זיי זענען פארדעקט.

אום 8:34:30 קערט מען זיך אויף א רגע צוריק צו שריט 5, "Visualizing the Data":

- import matplotlib.pyplot as plt און %matplotlib inline.
- mnist.train.images[1].shape → (784,).
- plt.imshow(mnist.train.images[1].reshape(28,28)) – דאס זעלבע בילד אן אן אנגעגעבענעם cmap, אין געל אויף א לילא הינטערגרונט.

דערנאך ווערט ווידער געוויזן שריט 7. אום 8:36:00 ווערט דארט אנגעצייכנט א ליניע מיט דער האנט.

סלייד: Session און אפשאצונג (8:36:30)

שריט 8, "Create the Session":

- init = tf.global_variables_initializer() – גרייט צו דעם אינישיאליזירן אלע וואריאבלען.
- with tf.Session() as sess: און sess.run(init) – מען עפנט א Session און מען אינישיאליזירט.
- א קאמענטאר: "Train the model for 1000 steps on the training set using built in batch feeder from mnist" (מיט "1000 steps" אונטערגעשטראכן).
- for step in range(1000): – א שלייף פון 1000 שריט. אין יעדן שריט נעמט batch_x, batch_y = mnist.train.next_batch(100) א פעקל פון 100 בילדער, און sess.run(train, feed_dict={x:batch_x, y_true:batch_y}) טריינירט דעם מאדעל אויף דעם פעקל.

שריט 9, "Evaluate the Trained model on Test data": אונטער דעם קאמענטאר "Test the Trained Model" שטייט matches = tf.equal(tf.argmax(y,1), tf.argmax(y_true,1)) – מען פארגלייכט די

פארויסגעזאגטע ציפער מיט דער ריכטיגער. די ווייטערדיגע שורות (acc, pr...) זענען פארדעקט. דער רעזולטאט איז 0.9203, דאס הייסט בערך 92% געוויקייט.

דער AWS לאבאראטאריע-עקראן (8:37:30)

אויפן עקראן ווערט געוויזן א simplilearn פלאטפארמע (מענוען: Manage Marketplace, Manage Services, Reports), אונטער "Deep learning team". עס ווערט געוויזן אן אינסטאנץ "aws TensorFlow 1.6.0" מיט: "InProgress - Start" ID 4856, Platform AMAZON, Status Created On 11-03-2018, Last Start Time 23-04-2018 11:49:47 AM UTC, Last Stop Time 23-04-2018 12:17:59 PM UTC. אויבן ווייזט זיך די מעלדונג "Updated latest status".

Jupyter: לאדן דעם דאטא-סעט (8:38:00)

מען עפנט א Jupyter נאטבוק "Deep Learning tutorial" אויף localhost:8888, מיט Python 3 אויף Ubuntu. דער טיטל איז "MNIST Handwriting recognition", און דערונטער שטייט "Tight integration with TensorFlow". דער טעקסט זאגט אז עס זענען דא 55K טריינירונג-בילדער, 10K טעסט-בילדער און 5K וואלידאציע-בילדער, און אז עס זענען דא עטלעכע איינגעבויטע פונקציעס צוצוקומען צו דער דאטא.

צעלן In [1] ביז In [3] פירן אויס די זעלבע דריי אימפארט-שורות ווי אויפן סלייד. דער אויסגאנג ווייזט "Extracting" פאר די פיר טעקעס.

Jupyter: אויספארשן די דאטא (8:39:00)

- דער array פון בילדער, מיט dtype=float32.
- In [6]: mnist.train.num_examples → 55000. די צעלן פאר test און validation זענען נאך נישט אויסגעפירט.
- In [9]: אימפארטירט matplotlib.pyplot א קאפיטל "Visualizing the Data".
- In [10]: mnist.train.images.shape → (55000, 784).
- In [11]: mnist.train.images[1].shape → (784,).
- In [12]: mnist.train.images[1] ווייזט די לאנגע רשימה ווערטן. מערסטנס זענען עס נולן, אבער צווישן זיי זענען דא ברוכצאלן צווישן 0 און 1, למשל 0.12156864, 0.5176471, 0.9960785, 0.9921569, 0.8352942 און 0.98823535.
- In [13]: plt.imshow(mnist.train.images[1234].reshape(28,28)) – א בילד אין בלויע טענער.
- In [15]: א צווייטער אינדעקס ווייזט א ציפער 8 אין רויט אויף א בלויען הינטערגרונט.
- א צעל אויסגעקאמענטירט: #(plt.imshow(...cmap='gist_gray').
- In [17]: mnist.train.images[1].max() → 1.0. נאך דעם קומט א צעל פאר min(), און צוויי אויסגעקאמענטירטע צעלן מיט reshape(784,1), איינע מיט aspect=0.02.

Jupyter: מאדעל און אפטימיזירער (8:42:30)

אונטער "Create the Model" פירט מען אויס In [19] ביז In [22]: x, W, b, און y = tf.matmul(x,W) + b, אזוי ווי אויפן סלייד. אונטער "Loss and Optimizer" קומט In [23] מיט y_true, נאך דעם cross_entropy, און דערנאך:

- optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.5) – א gradient descent אפטימיזירער מיט א לערן-ראטע פון 0.5.
- train = optimizer.minimize(cross_entropy) – דער טריינירונג-שריט מינימיזירט דעם cross entropy.

א ווארנונג-טעקסט רעקאמענדירט "See tf.nn.softmax_cross_entropy_with_logits_v2". די צעלן In [25] און In [26] ווערן אויסגעפירט. דערנאך הייבט זיך אן דער קאפיטל "Create Session", מיט init = tf.global_variables_initializer() און דעם אנהייב פון with tf.Session().

פון MNIST-טרייניר ביז אביעקט-דערקענונג אין ווידעא (לויטן עקראן) (8:45:00)

דער "Deep Learning tutorial" נאטבוק: אפטימייזער און סעסיע (8:45:00)

אויפן עקראן ווערט געוויזן א Jupyter נאטבוק מיטן נאמען "Deep Learning tutorial", וואס לויפט אויף localhost:8888 אינעם פאלדער Desktop/dl-tutorial. דער קערנעל איז Python 3.

אין צעל In [25] ווערט געשאפן אן אפטימייזער: tf.train.GradientDescentOptimizer מיט learning_rate=0.5. איבער דעם צעל שטייט א הערה וואס ווייזט אויף tf.nn.softmax_cross_entropy_with_logits_v2.

אין צעל In [26] שטייט:

optimizer.minimize(cross_entropy) = train

דאס הייסט, דער אפטימייזער זאל מינימיזירן דעם cross_entropy-פארלוסט, און דער רעזולטאט ווערט אפגעהיטן אין train.

דערנאך קומט א קעפל "Create Session". אין צעל In [27] ווערט געשאפן init = tf.global_variables_initializer(), וואס גרייט צו אלע וואריאבלען פונעם מאדעל צום אנהייבן.

דער טריינירונג-לופ (8:47:00)

דער עקראן בלייבט עטליכע מינוט אויף דעם הויפט-צעל, וואס הייבט זיך אן מיט with tf.Session() as sess: און לויפט sess.run(init). די קאמענטארן אין קאוד זאגן: "Train the model for 1000 steps on the training set" און "Using built in batch feeder from mnist for convenience".

- for step in range(1000): – א לופ פון 1000 שריט.
- batch_x , batch_y = mnist.train.next_batch(100) – יעדן שריט נעמט מען א פעקל פון 100 בילדער מיט זייערע לעיבלס פונעם MNIST טריינירונג-דאטא.
- sess.run(train,feed_dict={x:batch_x,y_true:batch_y}) – מען לויפט איין טריינירונג-שריט, און מען שטופט אריין די בילדער אין x און די ריכטיגע ענטפערס אין y_true.

אויספרובירן דעם מאדעל און דער רעזולטאט (8:51:30)

אונטער דעם קאמענטאר "# Test the Train Model" קומען נאך דריי שורות:

- matches = tf.equal(tf.argmax(y,1),tf.argmax(y_true,1)) – מען פארגלייכט דעם קלאס וואס דער מאדעל האט פאראויסגעזאגט (דער העכסטער ווערט אין y) מיטן ריכטיגן קלאס אין y_true.

- acc = (tf.reduce_mean(tf.cast(matches,tf.float32)) – מען פארוואנדלט די ריכטיג/פאלש רעזולטאטן אין צאלן און רעכנט אויס דעם דורכשניט, דאס הייסט די ענויקייט.
- print(sess.run(acc,feed_dict={x:mnist.test.images,y_true:mnist.test.labels})) – מען רעכנט אויס די ענויקייט אויף די טעסט-בילדער פון MNIST און מען דרוקט עס אויס.

נאכן לויפן ווייזט דער אויסגאנג 0.9177, דאס הייסט בערך 91.8% ענויקייט אויף די טעסט-דאטא.

א נייער נאטבוק: object_detection_in_video (8:54:30)

דערנאך ווערט געעפנט אן אנדערער Jupyter נאטבוק אין Firefox, "object_detection_in_video". ער געפינט זיך אינעם פאלדער tutorials/models-master/research/object_detection. אונטערן קעפל "Imports" ווערן אריינגעברענגט:

- numpy as np, os, sys, six.moves.urllib as urllib, tarfile, tensorflow as tf, zipfile, imageio.
- עס ווערט געדרוקט imageio.plugins.ffmpeg.download(), וואס לאדט אראפ דעם ffmpeg-צוגאב צום ארבעטן מיט ווידעא.
- matplotlib, Image פון io, pyplot as plt פון collections, StringIO פון datetime, defaultdict פון PIL.
- sys.path.append("..") מיטן קאמענטאר אז דאס איז נויטיג ווייל דער נאטבוק ליגט אינעם פאלדער object_detection. דערנאך ווערט אריינגעברענגט ops as utils_ops פון object_detection.utils.

צוגרייטן דעם מאדעל (8:55:00)

אין צעל In [7] ווערט visualization_utils as vis_util פון utils. דערנאך קומט דער קעפל "Model preparation", מיט א חלק "Variables". די ערקלערונג אויפן עקראן זאגט אז מען קען לאדן יעדן מאדעל וואס איז ארויסגעגעבן געווארן מיטן געצייג export_inference_graph.py . מען דארף נאר טוישן PATH_TO_CKPT אז עס זאל ווייזן אויף א נייע .pb פייל. עס שטייט אויך אז אויטאמאטיש ניצט מען דא אן "SSD with Mobilenet" מאדעל, און אז אינעם "detection model zoo" געפינט מען אנדערע מאדעלן מיט פארשידענע גיכקייטן און ענויקייטן.

דער קאוד שטעלט איין די פאלגנדע ווערטן:

- MODEL_NAME = 'ssd_mobilenet_v1_coco_2017_11_17'
- MODEL_FILE = MODEL_NAME + 'tar.gz.'
- DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'
- PATH_TO_CKPT = MODEL_NAME + 'frozen_inference_graph.pb/' – דער "פראזן" גראף, וואס דאס איז דער מאדעל וואס טוט טאקע דערקענען די אביעקטן.
- PATH_TO_LABELS = os.path.join('data','mscoco_label_map.pbtxt") – די ליסטע פון לעיבלס וואס מען שרייבט צו יעדן קעסטל.
- NUM_CLASSES = 90

דער טאבעלע פון COCO-טרייניירטע מאדעלן (8:56:00)

אויף GitHub, אינעם פייל tensorflow/models … detection_model_zoo.md, ווערט געוויזן א טאבעלע מיטן קעפל "COCO-trained models". די קאלאמען זענען: נאמען, גיכקייט (ms), COCO mAP (און אויסגאנג:

- 30, 21, Boxes – ssd_mobilenet_v1_coco
- 31, 22, Boxes – ssd_mobilenet_v2_coco
- 27, 22, Boxes – ssdlite_mobilenet_v2_coco
- 42, 24, Boxes – ssd_inception_v2_coco
- 58, 28, Boxes – faster_rcnn_inception_v2_coco
- 89, 30, Boxes – faster_rcnn_resnet50_coco
- 64, Boxes – faster_rcnn_resnet50_lowproposals_coco
- 92, 30, Boxes – rfcn_resnet101_coco
- 106, 32, Boxes – faster_rcnn_resnet101_coco
- 82, Boxes – faster_rcnn_resnet101_lowproposals_coco

אראפלאדן, לאדן דעם גראף און די לעיבלס (8:56:30)

אונטערן קעפל "Download Model" לאדט מען אראפ דעם מאדעל מיט urllib.request.URLopener() און opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE). דערנאך עפנט מען דעם ארכיוו מיט tarfile.open, מען גייט דורך tar_file.getmembers(), און יעדן פייל וואס הייסט 'frozen_inference_graph.pb' פאקט מען אויס אינעם איצטיגן פאלדער (os.getcwd()).

אונטערן קעפל "Load a (frozen) Tensorflow model into memory" שאפט מען detection_graph = tf.Graph() און א tf.GraphDef(). מען לייענט דעם פייל PATH_TO_CKPT מיט tf.gfile.GFile, מען פארט עס אויס מיט ParseFromString, און מען ברענגט אריין דעם גראף מיט tf.import_graph_def.

דער חלק "Loading label map" ערקלערט אז לעיבל-מאפעס פארבינדן נומערן מיט נעמען פון קאטעגאריעס. למשל, ווען די קאנוואלושאן-נעץ זאגט פאראויס 5, ווייסט מען אז דאס איז airplane. אין קאוד ווערן געשאפן label_map, categories און category_index מיט די פונקציעס load_labelmap, convert_label_map_to_categories(מיט max_num_classes=NUM_CLASSES און use_display_name=True) און create_category_index.

אונטערן קעפל "Helper code" ווערט דעפינירט די פונקציע load_image_into_numpy_array(image). זי נעמט די בריית און הייך פונעם בילד און פארוואנדלט דאס בילד אין א NumPy מאסיוו מיטן פארעם (im_height, im_width, 3) און טיפ np.uint8.

Detection און ווייזואליזאציע אויף ווידעא (8:58:00)

אונטערן קעפל "Detection" שטייט אין די קאמענטארן אז צוליב פשטות ניצט מען נאר 2 בילדער, image1.jpg און image2.jpg. עס ווערט באשטימט PATH_TO_TEST_IMAGES_DIR = 'test_images', און TEST_IMAGE_PATHS ווערט צוזאמגעשטעלט פון בילדער מיטן מוסטער 'jpg.{}beach' פאר i אין range(1, 3). די גרייס פון די אויסגאנג-בילדער (אין אינטשעס) איז IMAGE_SIZE = (12, 8).

דערנאך קומט די פונקציע run_inference_for_single_image(image, graph). זי נעמט די אפעראציעס פונעם גראף מיט get_operations() און זוכט די טענזארס 'num_detections', 'detection_boxes', 'detection_scores' ,'detection_classes' און 'detection_masks'.

אונטערן קעפל "Visualization of the results of a detection" שטייט א רוף צו vis_util.visualize_boxes_and_labels_on_image_array מיט די קעסטלעך, קלאסן, פונקטן, category_index, use_normalized_coordinates=True און line_thickness=8. דערנאך ווייזט מען דאס בילד מיט plt.figure און plt.imshow.

אינעם הויפט-צעל עפנט מען א tf.Session מיטן detection_graph, און מען נעמט מיט get_tensor_by_name די טענזארס 'image_tensor:0', 'detection_boxes:0', 'detection_scores:0', 'detection_classes:0' און 'num_detections:0'. דערנאך:

- 'input_video='traffic
- video_reader=imageio.get_reader('%s.mp4'%input_video) – לייענט דעם ווידעא traffic.mp4.
- video_writer=imageio.get_writer('%s_annotated.mp4'% input_video,fps=10) – שרייבט א נייעם באצייכנטן ווידעא מיט 10 בילדער א סעקונדע.
- מען נעמט די צייט מיט t0 = datetime.now(), שטעלט n_frames=0, און גייט דורך יעדן בילד פונעם ווידעא (image_np=frame, n_frames+=1). צום סוף ווערט יעדער בילד צוגעלייגט מיט video_writer.append_data(image_np).

דער ווידעא traffic.mp4 (8:59:30)

צום סוף ווערט געעפנט א ווידעא-שפילער מיט א פלעיליסט פון traffic.mp4. מען זעט א וועג מיט ווייסע פיילן און ליניעס, און דער שפילער שטייט אפגעשטעלט ביי 0:13 פון 0:13.

אביעקט-דערקענונג אין ווידעא און) CNNלויטן עקראן) (9:00:00)

דער קאוד פון object_detection_in_video (9:00:00)

אויפן עקראן ווערט געוויזן א Jupyter נאטיצבוך מיטן נאמען object_detection_in_video, אין א Firefox בלעטערער אויף Ubuntu. דער פאד איז localhost:8888/notebooks/Desktop/tutorials/models-master/ research/object_detection/object_detection_in_video.ipynb. אין די אנדערע טאבס זעט מען 04-TensorFlow-Regr… און models/detection… דער קערנעל איז Python 3.

דער קאוד הייבט אן מיט with detection_graph.as_default() :און דערנאך with tf.Session(graph=detection_graph) as sess:, דאס הייסט מען עפנט א TensorFlow סעסיע אויפן דערקענונג-גראף. נאכדעם נעמט מען ארויס פונעם גראף, דורך get_tensor_by_name, פינף טענזארס:

- image_tensor:0, דער אריינגאנג פאר דאס בילד
- detection_boxes:0, די קעסטלעך ארום די געפונענע זאכן
- detection_scores:0, די זיכערקייט-פונקטן
- detection_classes:0, די קלאסן (וואס פאר א זאך)

- num_detections:0, ווי פיל זאכן מען האט געפונען

דערנאך שטעלט מען input_video='traffic'. דער ליינער ווערט געמאכט מיט imageio.get_reader('%s.mp4'%input_video), און דער שרייבער מיט imageio.get_writer('%s_annotated.mp4'%input_video, fps=10). דאס הייסט, פון traffic.mp4 ווערט א נייע טעקע traffic_annotated.mp4, מיט 10 בילדער א סעקונדע.

א באמערקונג אינעם קאוד זאגט "loop through and process each frame". מען היט אפ די אנהייב-צייט מיט t0 = datetime.now() און שטעלט n_frames=0. דערנאך גייט א לופ for frame in video_reader:. ביי יעדן בילד שטעלט מען image_np=frame און מען לייגט צו 1 צו n_frames.

אין דער לופ זענען פאראן עטלעכע טריט:

- לויט דער באמערקונג "Expand dim since the expects images to have shape[1,None]" רופט מען np.expand_dims(image_np,axis=0), כדי צוצולייגן נאך א דימענסיע.
- "Actual detection": מען רופט sess.run מיט [detection_boxes, detection_scores, detection_classes, num_detections] און א feed_dict, און מען באקומט צוריק (boxes, scores, classes, num).
- "visualization": vis_util.visualize_boxes_and_labels_on_image_array צייכנט די קעסטלעך און די נעמען אויפן בילד. מען גיט איר np.squeeze(boxes), np.squeeze(classes).astype(…) א.א.וו.
- "video writer": video_writer.append_data(image_np) שרייבט דאס באצייכנטע בילד איריין אין דער נייער טעקע.

נאך דער לופ רעכנט מען אויס fps=n_frames/(datetime.now()-t0).total_seconds() און מען דרוקט "Frames processed: %s fps,Speed:%s s". צום סוף, אונטער "cleanup", מאכט מען צו דעם שרייבער מיט video_writer.close().

דער רעזולטאט: traffic_annotated.mp4 און cat.mp4 (9:01:30)

א ווידעא-שפילער ווייזט traffic_annotated.mp4, א ווידעא פון 0:31. מען זעט פון אויבן א גאס מיט פיילן אויפן וועג. ארום א ווייסן אויטא איז געצייכנט א גרין קעסטל מיטן צייכן "car" און א פראצענט. שפעטער אינעם ווידעא איז די גאס ליידיג.

דערנאך ווערט געוויזן דער אריגינעלער ווידעא cat.mp4 (0:20), מיט א געלער קאץ אויף א שטיינערדיקן וועג.

דער זעלבער קאוד מיט 'cat' (9:03:00)

אינעם נאטיצבוך איז input_video איצט געביטן צו 'cat'. דער איבעריקער קאוד איז דער זעלבער ווי פריער.

דער רעזולטאט: cat_annotated.mp4 (9:04:00)

דער ווידעא cat_annotated.mp4 (0:48) ווייזט א בלוי-גרין קעסטל ארום דעם קאץ. אין אנפאנג איז אויך דא א ווייס קעסטל ארום א זאך אינעם הינטערגרונט. שפעטער, ווען דער קאץ גייט אוועק, בלייבט דאס קעסטל נאך אלץ ארום אים.

דער אויסגאנג פונעם קאוד (9:05:00)

אונטער דער צעל באווייזט זיך א ווארענונג: WARNING:root:IMAGEIO FFMPEG_WRITER WARNING: input image is not divisible by macro_block_size=16, resizing from (1080, 192)…. דאס הייסט, די גרייס פונעם בילד ווערט געטוישט.

דערנאך שטייט: Frames processed: 481, Speed:0.904276 fps… מען האט אלזא פראצעסירט 481 בילדער, מיט א שנעלקייט פון בערך 0.9 בילדער א סעקונדע.

ווי אזוי בילד-דערקענונג ארבעט (9:05:30)

דער סלייד "How image recognition works?" פרעגט: "Do you know how Deep Learning recognizes the objects in an image? It does it using a Convolution Neural Network". דער ענטפער: דאס הייסט, דיפ-לערנען דערקענט די זאכן אין א בילד מיט א קאנוואלוציע-נויראל-נעץ.

דער ווייטערדיקער סלייד, "Let's see how CNN identifies the image of a bird", ווייזט א בילד פון א פויגל וואס איז צעטיילט אין א גריד מיט צאלן (12, 19, 21, 7, 40 …). די צאלן גייען אריין אין דעם נעץ. אויפן סלייד שטייט: "Input layer accepts the pixels of the image as input in the form of arrays". דער אריינגאנג-שיכט נעמט אן די פיקסלען פונעם בילד אלס מאסיוון (arrays).

דערנאך ווערט באצייכנט דער "Convolution Layer", מיט א "Matrix Filter" פון 3×3:

- 1 0 1
- 0 1 0
- 1 0 1

אויפן סלייד שטייט: "This layer uses a matrix filter and performs convolution operation to detect patterns in the image". דער שיכט ניצט א מאטריץ-פילטער און טוט א קאנוואלוציע, כדי צו געפינען מוסטערן אינעם בילד.

א צווייטער קעסטל זאגט אז ס'זענען דא עטלעכע באהאלטענע שיכטן (Hidden Layers), למשל Convolution, ReLU און Pooling, וואס ציען ארויס די אייגנשאפטן (feature extraction) פונעם בילד.

ביים "Pooling" שטייט: "Pooling layer also uses multiple filters to detect edges, corners, eyes, feathers, beak, etc". דער שיכט ניצט אויך עטלעכע פילטערס, כדי צו געפינען קאנטן,ווינקלען, אויגן, פעדערן, א שנאבל א.א.וו.

דער פלאן: What's in it for you? (9:07:30)

דער סלייד רעכנט אויס די טעמעס:

- Introduction to CNN
- What is Convolution neural network? (באצייכנט)
- How CNN recognizes images?
- Layers in convolution neural network
- Use case implementation using CNN

Built the first Convolution Neural Network called LeNet" Yann LeCun :"Introduction to CNN" in 1988". ער האט געבויט דעם ערשטן קאנוואלוציע-נעץ, LeNet, אין 1988.

וואס איז א ?CNN (9:08:30)

די דעפיניציע אויפן סלייד: "CNN is a feed forward neural network that is generally used to analyze visual images by processing data with grid like topology. A CNN is also known as a 'ConvNet'". דאס הייסט, א CNN איז א נויראל-נעץ וואס פירט די דאטא נאר פאראויס. מען ניצט אים געווענליך צו אנאליזירן בילדער, ווייל ער ארבעט מיט דאטא וואס איז אויסגעשטעלט ווי א גריד. מען רופט אים אויך "ConvNet".

דער דיאגראם ווייזט בילדער פון בלומען וואס גייען דורך דעם נעץ. ביים "Output Layer" קומען ארויס צוויי ענטפערס, "Orchid" און "Rose", מיטן אונטערשריפט "Identifies the flowers".

דער ווייטערדיקער סלייד זאגט: "Convolution operation forms the basis of any Convolution Neural Network". די קאנוואלוציע איז דער יסוד פון יעדן אוא נעץ. ווייטער שטייט: "In CNN, every image is represented in the form of arrays of pixel values". יעדעס בילד ווערט פארגעשטעלט אלס מאסיוון פון פיקסל-ווערטן.

אלס ביישפיל ווייזט מען די ציפער 8. ערשט ווערט זי געלייגט אויף א גריד, און דערנאך ווערט זי א טאבעלע פון 0 און 1:

- 0 0 1 1 0 0
- 0 1 0 0 1 0
- 0 0 1 1 0 0
- 0 1 0 0 1 0
- 0 0 1 1 0 0

א ביישפיל פון קאנוואלוציע (9:10:30)

דער סלייד זאגט: "Let's understand the convolution operation using 2 matrices a and b of 1 dimension". מען וועט פארשטיין די קאנוואלוציע מיט צוויי איין-דימענסיאנאלע מאטריצן.

אינעם רויטן קעסטל שטייט a = [5, 3, 7, 5, 9, 7] און b = [1, 2, 3]. אינעם רעכענונג-פענצטער שטייט אבער a = [5, 3, 2, 5, 9, 7], און די רעכענונג גייט מיט די צאלן פון דעם פענצטער. די אפעראציע איז a * b. יעדעס מאל טוט מען צוויי טריט: "Multiply the arrays element wise" (מען פארמערט עלעמענט ביי עלעמענט) און "Sum the product" (מען סומירט די רעזולטאטן):

- [5, 6, 6] → 17
- [3, 4, 15] → 22
- [2, 10, 27] → 39
- ... (ווייטער אזוי, וועט b רוקט זיך איבער a)

ווי אזוי CNN דערקענט בילדער (9:13:00)

"Consider the following 2 images": צוויי גרידס פון 2×2 מיט בלויע קעסטלעך. איינער איז "image for the symbol \" און דער אנדערער "image for the symbol /". אין מיטן שטייט א קאמפיוטער.

דער וויטערדיקער סלייד ווייזט א געצייכנט שמייכלענדיק פנים. ערשט ווערט עס אריבערגעפירט אויף א גריד מיט שווארצע קעסטלעך, און דערנאך אין א טאבעלע פון 0 און 1, למשל די שורות 0 1 0 0 0 1 0 און 1 1 0 0 1 0 0 1.

די שיכטן אין א CNN (9:14:00)

דער סלייד "Layers in Convolution Neural Network" ווייזט ארום א קרייז "CNN" פיר שיכטן:

- 1. Convolution Layer
- 2. ReLU Layer
- 3. Pooling Layer
- 4. Fully Connected Layer

דער עקראן בלייבט אויף דעם סלייד ביזן סוף פון דעם אפשניט.

די שיכטן פון א CNN און דער CIFAR-10 ביישפיל (לויטן עקראן) (9:15:00)

דער Convolution Layer (9:15:00)

אויפן עקראן ווערט געווויזן א סלייד מיטן טיטל "Convolution Layer". אויבן שטייט די דעפיניציע: א Convolution Layer האט א צאל פילטערס וואס פירן אדורך א convolution אפעראציע.

דערונטער שטייט אז יעדעס בילד ווערט באטראכט ווי א מאטריץ פון פיקסל-ווערטן. מען נעמט א 5×5 בילד וואס זיינע פיקסל-ווערטן זענען נאר 0 און 1.

אויף דער לינקער זייט שטייט די טאבעלע "Image pixels". די ערשטע פיר שורות וואס זעט זיך זענען:

- 0 0 1 1 1
- 0 1 1 1 0
- 1 1 1 0 0
- 0 1 1 0 0

אין מיטן שטייט א 3×3 "Filter" מיט די שורות 1 0 1, 0 1 0, 1 0 1. אויף דער רעכטער זייט שטייט דער רעזולטאט, "Convolved Feature", א 3×3 מאטריץ מיט די שורות 4 3 4, 2 4 3, 2 3 4.

די סלייד בלייבט אויפן עקראן פון 9:15:00 ביז 9:16:00. אום 9:16:30 ווערט געווויזן דער ערשטער שריט פונעם חשבון. די אויבערשטע לינקע 3×3 שטח פונעם בילד איז אנגעצייכנט אין בלוי. ביי יעדן פיקסל שטייט קליין 1× אדער 0×, לויט דעם ווערט פונעם פילטער אויף יענעם ארט. אין דער ערשטער צעל פונעם Convolved Feature ערשיינט דער רעזולטאט 4, צעצייכנט אין אראנזש.

דער ReLU Layer (9:17:00)

א נייע סלייד, "ReLU Layer". אויבן שטייט: נאכדעם וואס די feature maps זענען ארויסגעצויגן, איז דער קומענדיגער שריט זיי אריבערצופירן צו א ReLU שיכט.

לינקס ווערט געוויזן א גראפיק. אויף דער ווערטיקאלער אקס גייען די ווערטן פון 0 ביז 10. די ליניע ליגט פלאך אויף 0 אויף דער נעגאטיווער זייט, און פון 0 ווייטער הייבט זי זיך גלייך ארויף.

אום 9:17:30 ביז 9:18:00 ערשיינט רעכטס א דיאגראם, א קרייז מיטן ווארט "ReLU" אין מיטן, מיט פיר פונקטן ארום:

- Sets all negative pixels to 0, ער שטעלט אלע נעגאטיווע פיקסלען אויף 0
- Introduces non-linearity to the network, ער ברענגט אריין נישט-לינעארקייט אינעם נעץ
- Performs element wise, ער ארבעט עלעמענט ביי עלעמענט (דער רעשט פונעם זאץ איז פארדעקט)
- The output is a rectified feature map, דער רעזולטאט איז א "rectified feature map"

אום 9:18:30 ביז 9:19:00 ווייזט די סלייד א שווארצע שטח מיטן טיטל "Input Feature Map". דארט ליגט א גרויע feature map פון א בילד פון א פויגל אויף א צווייג, און אויבן רעכטס שטייט א קליינער ווייסער קעסטל.

דער Pooling Layer (9:19:30)

די סלייד "Pooling Layer" זאגט: די rectified feature map גייט יעצט אדורך א pooling שיכט. Pooling איז א down-sampling אפעראציע וואס פארקלענערט די דימענסיעס פון דער feature map.

לינקס שטייט א 4×4 "Rectified feature map", צעטיילט אין פיר קאלירטע 2×2 שטחים:

- רויט: 1 4 / 2 6
- גרין: 2 7 / 8 5
- בלוי: 3 4 / 1 2 (ארומגערינגלט)
- לילא: 0 7 / 3 1

א פייל מיטן טעקסט "max pooling with 2x2 filters and stride 2" פירט צו א 2×2 "Pooled feature map" מיט די ווערטן 6, 8, 4, 7. יעדער ווערט איז דער גרעסטער פון זיין קאלירטן שטח. די סלייד בלייבט פון 9:19:30 ביז 9:21:00.

אום 9:21:30 שטייט אויף א צווייטער סלייד: א pooling שיכט ניצט פארשידענע פילטערס צו אידענטיפיצירן פארשידענע טיילן פונעם בילד, ווי קאנטן, ווינקלען, דער קערפער, פעדערן, אויגן, דער שנאבל א.א.וו. מען זעט צוויי בילדער פונעם פויגל: לינקס א גרויע feature map, און א פייל צו דער רעכטער זייט, וואו די ליניעס זענען ווייס אויף א שווארצן הינטערגרונט.

Flattening (9:22:00)

די סלייד "Flattening" זאגט: Flattening איז דער פראצעס פון איבערמאכן אלע צוויי-דימענסיאנאלע מערכות (arrays) פון דער pooled feature map אין איין לאנגן, אנגייענדיגן, לינעארן וועקטאר. די 2×2

"Pooled feature map" מיט 6, 8, 4, 7 ווערט, דורך א פייל מיטן נאמען "Flattening", א ווערטיקאלע זייל פון 6, 8, 4, 7. די סלייד בלייבט ביז 9:23:00.

דער Fully Connected Layer (9:23:30)

די סלייד "Fully Connected Layer" זאגט: די אויסגעגלייכטע (flattened) מאטריץ פון דער pooling שיכט ווערט אריינגעגעבן אלס אינפוט צום Fully Connected Layer, כדי צו קלאסיפיצירן דאס בילד.

דער דיאגראם ווייזט א נוראל-נעץ מיט אן אינפוט-שיכט פון זיבן קנעפלעך, צוויי באהאלטענע שיכטן פון זיבן קנעפלעך יעדע, און אן אויספיר-שיכט פון דריי קנעפלעך. אלע קנעפלעך זענען פארבונדן מיט אלע קנעפלעך פון דער קומענדיגער שיכט. לינקס שטייט "Flattened Matrix". דער דיאגראם בלייבט ביז 9:24:30.

אום 9:25:00 ערשיינט לינקס א פארביג בילד פון א פויגל (א טוקאן) אויף א צווייג, מיטן טעקסט "Pixels from the flattened matrix fed as input". רעכטס, ביי דער אויספיר-שיכט, איז "Bird" אנגעצייכנט, מיטן טעקסט "Identifies the image".

פון 9:25:30 ביז 9:26:30 קומט א סלייד מיטן טעקסט: לאמיר זען דעם גאנצן פראצעס ווי א CNN דערקענט א פויגל. דאס בילד גייט דורך עטליכע גרופעס פון שיכטן, אנגעצייכנט "Max + ReLU + Convolution Pooling". אונטן שטייט "Feature Extraction in multiple hidden layers". דערנאך קומט דער "Fully Connected Layer" מיטן רעזולטאט "Bird", מיטן אונטערשריפט "Classification in the output layer".

א פראקטישער ביישפיל מיט CIFAR-10 (9:27:00)

די סלייד "Use case implementation using CNN" זאגט: מען וועט ניצן דעם CIFAR-10 דאטא-סעט (פונעם Canadian Institute For Advanced Research) צו קלאסיפיצירן בילדער אין 10 קאטעגאריעס. די קאטעגאריעס זענען:

- 01 airplane
- 02 automobile
- 03 bird
- 04 cat
- 05 deer
- 06 dog
- 07 frog
- 08 horse
- 09 ship
- 10 truck

אום 9:28:00 ווייזט א סלייד דעם שריט "4. Display images using matplotlib". די קאוד-צעלן:

- `import matplotlib.pyplot as plt` און `%matplotlib inline` ברענגען אריין די גראפיק-ביבליאטעק און שטעלן אז די בילדער זאלן זיך ווייזן אינעם notebook.
- `import numpy as np` ברענגט אריין NumPy.

- `X = b[1]data_batch1"data"]` נעמט די בילד-דאטא פונעם ערשטן batch.
- `X = X.reshape(10000, 3, 32, 32).transpose(0,2,3,1).astype("uint8")` פארמט איבער די דאטא אין 10000 בילדער פון 3 קאלירן און 32×32 פיקסלען. דערנאך בייט זי איבער די סדר פון די אקסן, און מאכט די ווערטן פון טיפ uint8.
- `X[0].max()` און `(X[0]/255).max()` געפינען דעם גרעסטן ווערט פונעם ערשטן בילד, און דעם גרעסטן ווערט נאכן צעטיילן אויף 255.
- `plt.imshow(X[0])` ווייזט דאס ערשטע בילד. דער רעזולטאט איז א matplotlib AxesImage, מיט א קליין פיקסלדיג בילד.

דער Jupyter notebook (9:28:30)

אויפן עקראן ווערט געוויזן א Jupyter notebook (Python 3) . דער טעקע-נאמען ענדיגט זיך מיט CNN_picture_Classification.ipynb.

אויבן זעט מען די לעצטע שורות פון א צעל וואס טיילט אויס די דאטא:

- `all_data[0] = batch_meta`
- `data_batch1` ביז `data_batch5`, יעדער פון `all_data[1]` ביז `all_data[5]`
- `all_data[6] = test_batch`

אין צעל In [6] ווערט אויסגעדרוקט `print(batch_meta)`, מיטן רעזולטאט:

- `num_cases_per_batch`: 10000
- `label_names`: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
- `num_vis`: 3072

דערנאך קומט א הערה: פארוואס שטייט א b פאר די סטרינגס? Bytes literals הייבן זיך שטענדיג אן מיט b אדער B. זיי שאפן אן אביעקט פונעם טיפ bytes אנשטאט str, זיי מעגן אנטהאלטן נאר ASCII אותיות, און bytes מיט א ווערט פון 128 אדער מער דארפן ווערן אויסגעדריקט מיט escapes. עס ווערט אנגעגעבן א לינק צו stackoverflow.com.

אין צעל In [8] גיט `data_batch1.keys()` צוריק די שליסלען `batch_label`, `labels`, `data`, `filenames`.

אין דער לעצטער צעל ווערט געשריבן דער קאוד פון דער סלייד: די import שורות, און אום 9:29:00 ווערן צוגעלייגט די שורות `X = b[1]data_batch1"data"]` און די reshape שורה.

אום 9:29:30 זענען אין דער reshape שורה די צוויי צאלן 32, 32 ארומגערינגלט מיט א האנט-געצייכנטן קרייז, און א פייל ווייזט אויף זיי.

CIFAR-10 בילדער: צוגרייטן די דאטא פאר א CNN לויטן עקראן) (9:30:00)

די באטשעס, די label_names און דער ערשטער reshape (9:30:00)

אויפן עקראן ווערט געוויזן א Jupyter notebook מיטן נאמען CNN_picture_Classification.ipynb , מיט א Python 3 קערנעל. אויבן אין דער צעל זעט מען די סוף-שורות פון פריערדיגן קאוד: data_batch3, data_batch4 און data_batch5 באקומען all_data[3], all_data[4] און all_data[5], און test_batch באקומט all_data[6].

אין צעל In [6] דרוקט מען אויס batch_meta. דער רעזולטאט איז א dictionary מיט דריי שליסלען:

- num_cases_per_batch: 10000.
- label_names: די צען קאטעגאריעס airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.
- num_vis: 3072.

דערנאך קומט א טעקסט-צעל. זי פרעגט פארוואס עס שטייט א b פארן string, און ענטפערט: bytes literals הייבן זיך שטענדיק אן מיט 'b' אדער 'B'. זיי שאפן אן אביעקט פון טיפ bytes אנשטאט str, זיי מעגן אנטהאלטן נאר ASCII אותיות, און א ביים מיט א ווערט פון 128 אדער העכער מוז מען שרייבן מיט escapes. אונטן שטייט א לינק צו stackoverflow אויף דער פראגע what-does-the-b-character-do-in-front-of-a-string-literal.

אין צעל In [8] רופט מען data_batch1.keys(). דער רעזולטאט ווייזט פיר שליסלען: batch_label, labels, data און filenames.

די נעקסטע צעל טוט אזוי:

- זי אימפארטירט matplotlib.pyplot אלס plt.
- זי שטעלט %matplotlib inline, כדי די בילדער זאלן זיך ווייזן אין notebook.
- זי אימפארטירט numpy אלס np.
- זי נעמט X = b[data_batch1"data"].
- דערנאך שרייבט זי:

X = X.reshape(10000, 3, 32, 32).transpose(0,2,3,1).astype("uint8")

דאס הייסט: מען שטעלט איבער די דאטא אין 10000 בילדער, יעדעס מיט 3 קאליר-קאנאלן אויף 32×32 פונקטן. נאכדעם בייט מען איבער די סדר פון די אקסן, אזוי אז דער קאנאל קומט צום סוף. צולעצט פארוואנדלט מען אלעס אין טיפ uint8.

אויפן עקראן זענען אנגעצייכנט פיילן און קרייזן ארום די צאלן 10000, 3, 32, 32 און ארום transpose(0,2,3,1). ביז 9:31:30 ווערט די צעל אויסגעפירט, און זי באקומט די נומער In [9].

קוקן אויף די בילדער (9:32:00)

אין צעל In [10] רופט מען X[0].max(), און דער רעזולטאט איז 255.

אין צעל In [11] ווייזט (plt.imshow(X[0] א פארשוואומען בילד פון 32×32 פונקטן. מען זעט דארט א גרינ-ברוינעם זשאבע-ענליכן געשטאלט. די אקסן גייען פון 0 ביז 30.

אין צעל In [15] ווייזט (plt.imshow(X[4] א בילד פון אן אויטא.

סלייד: "Use case implementation using CNN" (9:33:30)

דער סלייד טראגט דעם טיטל "5. Helper function to handle data". ער ווייזט דעם קלאס CifarHelper מיט די פונקציעס init און set_up_images. דער זעלבער קאוד ווערט באלד אריינגעשריבן אין notebook, און ער ווערט אונטן באשריבן.

די פונקציע one_hot_encode (9:34:00)

אין א נייער צעל ווערט דעפינירט one_hot_encode(vec, vals=10). דער docstring זאגט אז זי איז געמאכט צו one-hot encode די 10 מעגליכע labels. דער קאוד טוט די פאלגנדע שריט:

- n = len(vec): ער מעסט ווי־פל labels עס זענען דא.
- out = np.zeros((n, vals)): ער שאפט א מאטריץ פון נולן.
- out[range(n), vec] = 1: אין יעדער שורה שטעלט ער א 1 אין דעם קאלום וואס שטימט מיטן label.
- return out: ער גיט צוריק דעם רעזולטאט.

די עקראן בלייבט בייַ דער פונקציע ביז 9:35:00.

דער קלאס CifarHelper אין notebook (9:35:30)

אין init שטעלט מען אריין די אנהייב-ווערטן:

- self.i = 0.
- self.all_train_batches באקומט א ליסטע פון data_batch1 ביז data_batch5.
- self.test_batch באקומט [test_batch].
- self.training_images, self.training_labels, self.test_images און self.test_labels ווערן אלע None.

די פונקציע set_up_images טוט די פאלגנדע זאכן:

- זי דרוקט אויס "Setting Up Training Images and Labels".
- training_images ווערט np.vstack פון d[b"data"] פון אלע טרייניר־באטשעס, דאס הייסט אלע באטשעס ווערן צוזאמגעלייגט איינס אויפן אנדערן.
- train_len ווערט די לענג דערפון.
- training_images ווערט reshape(train_len,3,32,32).transpose(0,2,3,1)/255. די צעטיילונג דורך 255 איז אויפן עקראן באזונדער אנגעצייכנט.
- training_labels ווערט one_hot_encode פון np.hstack פון d[b"labels"], מיט 10 קאטעגאריעס.
- דאס זעלבע טוט מען פאר די טעסט-דאטא, נאכן אויסדרוק "Setting Up Test Images and Labels". דא ניצט מען test_len און test_labels.

אויפן עקראן זענען ארויסגעשטעלט די זאכן מיט פיילן, אונטערשטרייכונגען און קרייזן.

(9:39:30) next_batch

די פונקציע next_batch(self, batch_size) טוט אזוי:

- x נעמט training_images פון self.i ביז self.i+batch_size, מיט reshape(100,32,32,3).
- y נעמט די פאסיגע training_labels.
- self.i = (batch_size + self.i) % len(self.training_images), אזוי אז דער צייגער הייבט זיך ווידער אן פון אנהייב ווען ער קומט צום סוף.
- זי גיט צוריק x און y.

אויפן עקראן איז אנגעשריבן מיט דער האנט "10,000, 32, 32, 3" און "100". לעבן return איז אנגעצייכנט "label".

(9:42:00) אויספירן און TensorFlow

אין צעל In [17] שטייט א באמערקונג: "Before Your tf.Session run these two lines". דערנאך קומען די שורות:

- CifarHelper() = ch
- ch.set_up_images()
- א קאמענטירטע שורה: # ch.next_batch(100) = batch

דער רעזולטאט ווייזט די צוויי אויסגעדרוקטע שורות: "Setting Up Training Images and Labels" און "Setting Up Test Images and Labels".

אין צעל In [18] שטייט import tensorflow as tf. אונטער איר קומט א רויטע FutureWarning פון h5py. זי זאגט אז די קאנווערסיע פון float צו np.floating אינעם צווייטן ארגומענט פון issubdtype איז deprecated, און אז אין צוקונפט וועט עס באהאנדלט ווערן ווי np.float64 == np.dtype(float).type. דער פאד צו דער ענווייראנמענט איז keraspython36.

אין דער לעצטער צעל ווערן דעפינירט דריי placeholders:

- x = tf.placeholder(tf.float32, shape=[None,32,32,3])
- y_true = tf.placeholder(tf.float32, shape=[None,10])
- hold_prob = tf.placeholder(tf.float32)

דער None אין shape איז ארויסגעשטעלט.

בויען דעם CNN מיט TensorFlow הילפס-פונקציעס (לויטן עקראן) (9:45:00)

דער סלייד "Applying the helper functions" (9:45:00)

אויפן עקראן ווערט געוויזן א סלייד מיטן טיטל "Use case implementation using CNN", שריט 8: "Applying the helper functions". דער סלייד ווייזט זעקס Python פונקציעס, וועלכע ווערן גלייך דערנאך אויך געוויזן אין א Jupyter נאטבוק.

די הילפס-פונקציעס אינעם נאטבוק (9:45:30)

אינעם נאטבוק (Python 3, א פייל וואס הייסט CNN_picture_Classification.ipynb) ווערט געוויזן א צעל וואס הייבט זיך אן מיטן קאמענטאר "# MNIST". דער קאוד דעפינירט די זעלבע זעקס פונקציעס:

- init_weights(shape): שאפט צופעליגע אנהייב-ווערטן דורך tf.truncated_normal מיט stddev=0.1, און גיט זיי צוריק אלס tf.Variable. דאס זענען די וואגן (weights).
- init_bias(shape): שאפט ווערטן וואס זענען אלע 0.1 דורך tf.constant(0.1, shape=shape), און גיט זיי צוריק אלס tf.Variable. דאס איז דער bias.
- conv2d(x, W): רופט tf.nn.conv2d מיט strides=[1, 1, 1, 1] און 'padding='SAME.
- max_pool_2by2(x): רופט tf.nn.max_pool מיט ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1] און 'padding='SAME.
- convolutional_layer(input_x, shape): שאפט W דורך init_weights(shape) און b דורך init_bias([shape[3]]), און גיט צוריק (b + tf.nn.relu(conv2d(input_x, W)).
- normal_full_layer(input_layer, size): נעמט input_size פון input_layer.get_shape()[1], שאפט W מיט דער פארעם [[input_size, size] און b מיט [[size], און גיט צוריק (b + tf.matmul(input_layer, W).

צווישן 9:46:00 און 9:48:00 ווערן אויפן קאוד צוגעצייכנט פיילן: ביי init_weights און stddev=0.1 , ביי init_bias און 0.1, און ביי conv2d, x, W און strides. דערנעבן ווערט געצייכנט א קליין קעסטל מיט א גיטער, און די אויפשריפט "2d" מיט "32 x 32 x 3"; פון דער לעצטער "3" גייט א ליניע אראפ צו convolutional_layer, און א צווייטע ליניע פארבינדט conv2d מיט דעם רוף conv2d(input_x, W) אין convolutional_layer.

דער דיאגראם "Fully Connected Layer" (9:48:30)

דער סלייד מיטן טיטל "Fully Connected Layer" ווייזט דעם גאנצן גאנג פון א CNN:

- "Input Image": א טאבעלע פון 5x5 נולן און איינסן.
- א פייל "Convolution" פירט צו א הויפן בלעטער: "Convolution Layer". אונטן דערונטער ווערט געוויזן א גראפיק פון "ReLU": א פלאכע ליניע ביי נול וואס הייבט זיך דערנאך גלייך אויף.
- א פייל "Pooling" פירט צו א קלענערן הויפן: "Pooling Layer".
- א פייל "Flattening" פירט צו א נויראל-נעץ מיט קרייזלעך אין שורות: "Fully Connected Layer", וואס ענדיגט זיך מיט דריי רויטע קרייזלעך.

אין די ווייטערדיגע ראמען ווערן אויף דער אינפוט-טאבעלע צוגעצייכנט קעסטלעך פון 2x2 ווערטן, און ReLU ווערט אונטערגעשטראכן. ביי Pooling ווערט געשריבן "Max" צוזאמען מיט א קליין גיטער און די ציפערן 3 און 2.

צוריק צום קאוד: די פולע שיכט (9:50:00)

אינעם נאטבוק ווערט א פייל געצייכנט צו normal_full_layer, און input_size, size ווערט אונטערגעשטראכן. אין דער עק פון עקראן איז געצייכנט א קליינע אויפשריפט, נישט קלאר צו לייענען.

בויען די שיכטן (9:51:00)

א נייע צעל ווערט צוגעלייגט:

convolutional_layer(x,shape=[4,4,3,32]) = convo_1

max_pool_2by2(convo_1) = convo_1_pooling

די ערשטע שורה מאכט די ערשטע קאנוואלוציע-שיכט; די צווייטע שורה מאכט Max Pooling דערויף. די צעל ווערט דורכגעפירט (In [27], שפעטער In [28]). אויפן עקראן ווערן אין שאפע [4,4] און [3,32] אנגעצייכנט, מיט א געצייכנט קעסטל פון קליינע קוואדראטן. א פייל ווייזט ווי convo_1 גייט אריין אין max_pool_2by2.

ביי 9:52:30 קומט אן א צווייטע קאנוואלוציע-שיכט (In [29]):

convolutional_layer(convo_1_pooling,shape=[4,4,32,64]) = convo_2

max_pool_2by2(convo_2) = convo_2_pooling

אויספלאכן און פולע שיכטן (9:53:00)

- In [30]: א רעכענונג 8*8*64, און דער רעזולטאט Out[30] איז 4096.
- In [convo_2_flat :31] = tf.reshape(convo_2_pooling,[-1,8*8*64]). דאס פלאכט אויס דעם רעזולטאט פון דער צווייטער pooling.
- In [full_layer_one :32] = tf.nn.relu(normal_full_layer(convo_2_flat,1024)). דאס איז א פולע שיכט מיט 1024 איינסן, מיט ReLU.
- In [full_one_dropout :33] = tf.nn.dropout(full_layer_one,keep_prob=hold_prob).
- In [y_pred :34] = normal_full_layer(full_one_dropout,10). דאס איז די לעצטע שיכט מיט 10 רעזולטאטן.

דער סלייד פון שריטן 11–14 (9:55:30)

אויפן עקראן ווערט געוויזן א סלייד "Use case implementation using CNN" מיט פיר שריטן:

11. "Set output to y_pred" :y_pred = normal_full_layer(full_one_dropout,10). דער אויסגאנג איז <tf.Tensor 'add_9:0' shape=(?, 10) dtype=float32>.
12. "Apply the Loss function" :cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_true,logits=y_pred)).
13. "Create the optimizer" :optimizer = tf.train.AdamOptimizer(learning_rate=0.001) און train = optimizer.minimize(cross_entropy).
14. "Create a variable to initialize all the global tf variables" :init = tf.global_variables_initializer().

y_pred, Loss און Optimizer אינעם נאטבוק (9:56:00)

אינעם נאטבוק ווערן די ווערטן [4,4] אין convo_1 און convo_2 ארומגעקרייזלט. In [35] ווייזט y_pred, מיטן רעזולטאט <tf.Tensor' dtype=float32' shape=(?, 10)>.

ביי 9:57:00 ווערט געשריבן די Loss-צעל. labels=y_true און logits=y_pred ווערן אנגעצייכנט. די צעל In) [38]) האט די קאמענטארן "# Loss Function" און "# Create a cross_entropy Loss function", און דערנאך די cross_entropy שורה וואס ניצט tf.reduce_mean און softmax_cross_entropy_with_logits.

ביי 9:58:00 הייבט זיך אן א צעל מיט "#Optimizer" און "#Create the optimizer using an Adam Optimizer." דערביי ווערט אויפן עקראן געצייכנט א פייל ארויף און א קרומע ליניע.

ביי 9:59:30 ווערט די צעל In) [39]) פארענדיגט:

tf.train.AdamOptimizer(learning_rate=0.001) = optimizer

optimizer.minimize(cross_entropy) = train

דאס איז דער Adam אפטימייזער מיט א לערן-שנעלקייט פון 0.001, וואס פארקלענערט דעם cross_entropy פארלוסט.

טרייניר א CNN און א AI-קלאן מיט) HeyGenלויטן עקראן) (10:00:00)

דער סלייד: לויפן דעם מאדעל אין א Graph Session (10:00:00)

אויפן עקראן ווערט געוויזן א סלייד מיטן טיטל "Use case implementation using CNN", מיט דעם פונקט "15. Run the model by creating a Graph Session". אויף דער לינקער זייט שטייט דער קאוד, און אויף דער רעכטער זייט דער רעזולטאט פונעם טרייניר:

- Accuracy is: 0.0979 – Currently on step 0
- Accuracy is: 0.4065 – Currently on step 100
- Accuracy is: 0.4654 – Currently on step 200
- Accuracy is: 0.5065 – Currently on step 300
- Currently on step 400 – דער ווערט איז אפגעשניטן.

דער קאוד אין Jupyter Notebook (10:00:30)

דערנאך גייט דער עקראן איבער צו א Jupyter Notebook. אין דער אדרעס-שורה זעט מען א טעקע מיטן נאמען CNN_picture_Classification.ipynb, אין א פאלדער פון "CNN Convolutional Neural Network". דער קערנעל איז Python 3.

אויבן זעט מען נאך דעם סוף פון דער פריערדיגער צעל. דארט ווערט דער optimizer געשאפן מיט tf.train.AdamOptimizer מיט learning_rate=0.001, און train = optimizer.minimize(cross_entropy).

אין צעל In [40] שטייט דער קאמענטאר "Create a variable to intialize all the global tf variables", און דערנאך:

tf.global_variables_initializer() = init

די הויפט-צעל הייבט זיך אן מיט די קאמענטארן "Graph Session" און "Perform the training and test print outs in a Tf session and run your model!". דער קאוד טוט אזוי, שורה נאך שורה:

- with tf.Session() as sess: עפנט א TensorFlow סעסיע.
- sess.run(tf.global_variables_initializer()) גיט אנהייב-ווערטן פאר אלע געביטן (variables).
- for i in range(500): א לופ פון 500 שריט.
- batch = ch.next_batch(100) נעמט די קומענדיגע גרופע פון 100 בילדער.
- sess.run(train, feed_dict={x: batch[0], y_true: batch[1], hold_prob: 0.5}) טריינירט דעם מאדעל אויף דער גרופע. batch[0] איז די בילדער, batch[1] איז די ריכטיגע לעיבלס, און hold_prob איז 0.5.
- if i%100 == 0:, דאס שטייט "PRINT OUT A MESSAGE EVERY 100 STEPS" אונטער דעם קאמענטאר הייסט יעדע 100 שריט.
- print('{} Currently on step'.format(i)) און 'Accuracy is:' עס ווערט געדרוקט.
- matches = tf.equal(tf.argmax(y_pred, 1),tf.argmax(y_true,1)) שטייט "Test the Train Model" אונטער. דאס פארגלייכט די פאָרויסזאגונג פונעם מאדעל מיטן ריכטיגן ענטפער.
- acc = tf.reduce_mean(tf.cast(matches,tf.float32)) פארוואנדלט די פארגלייכן אין נומערן און רעכנט אויס דעם דורכשניט, דאס הייסט די אקיוראסי.
- print(sess.run(acc,feed_dict={x:ch.test_images,y_true:ch.test_labels,hold_prob:1.0})) רעכנט אויס די אקיוראסי אויף די טעסט-בילדער, מיט hold_prob 1.0. דערנאך קומט print('\n').

צייכענונגען אויפן קאוד (10:01:00– 10:02:30)

אין די קומענדיגע מינוטן ווערן אויפן קאוד געצייכנט פיילן און אונטערשטרייכן. די צייכענונגען ווייזן אויף with tf.Session, אויף sess.run(tf.global_variables_initializer()), און אויף range(500). לעבן range(500) איז אנגעשריבן מיט דער האנט "50,000".

דערנאך ווערן אונטערגעשטראכן next_batch(100), train, batch[0], batch[1] און hold_prob: 0.5. א שטריך צייכנט אפ דעם גאנצן בלאק וואס דרוקט אויס יעדע 100 שריט. פיילן ווייזן אויף דער matches-שורה, און אויף acc, tf.cast און tf.float32.

דער רעזולטאט פונעם לויפן (10:03:00– 10:06:00)

די צעל ווערט געלאפן, און עס קומען ארויס די רעזולטאטן:

- Accuracy 0.1028 – step 0
- Accuracy 0.3905 – step 100
- Accuracy 0.4365 – step 200
- step 300 – דער ווערט איז נישט לייענבאר.

אויפן רעזולטאט ווערט געצייכנט "Acc" און "Loss" מיט צוויי גראפיקן. דער גראפיק פאר Acc קריכט ארויף צו 1. דער גראפיק פאר Loss פאלט אראפ. אונטן ווערן אנגעשריבן די נומערן .1, .2 און .4 מיט א פייל. דער רעזולטאט פון step 0 ווערט ארומגערינגלט.

אין דער לעצטער צייכענונג שטייט ווידער "acc" און "LOSS", און ".5" איז ארומגערינגלט אונטער acc. דערנאך קערט זיך דער עקראן אום צום זעלבן סלייד פון פונקט .(10:06:30) 15

א נייער טייל: "How To Create An AI Clone" (10:07:00)

דער קומענדיגער טייל עפנט זיך מיט א בילד-טיטל "How To Create An AI Clone".

דער HeyGen בלאג (10:07:30–10:08:00)

אויפן עקראן ווערט געוויזן דער בלאג-בלאט heygen.com/blog/ai-avatar-clone. דער אינהאלט-צעטל אויף דער זייט רעכנט אויס די טיילן פונעם ארטיקל:

- What are AI Avatars and Digital Clones
- Benefits
- Applications in Personal and Business Settings
- Technologies and Platforms
- Steps to Create an AI Avatar
- Revolutionize Corporate Training

אונטער די מעלות שטייען צוויי פונקטן. Consistency: יעדער ארבעטער באקומט די זעלבע קוואליטעט פון לערנען. Cost-Effective: מען דארף נישט קיין נאכאנאנדיגע לערער און סעסיעס.

דער טייל Applications זאגט אז אין געשעפט קענען פירמעס געבן פערזענליכע טריינינגס אין א גרויסן מאס מיט AI אוואטארן, וואס פאסן זיך צו צום לערן-שטייגער פון יעדן. פאר פערזענליכער אנטוויקלונג קען מען לערנען אין אייגענעם טעמפא, און אוואטארן קענען דינען אלס "AI tutors in education".

די שריט צו שאפן אן אוואטאר זענען:

."Avatar" 1. Select

."Instant Avatar" 2. Choose

.3. Upload Your Footage

.4. Complete Identity Verification: א קורצער הסכם-ווידעא צו באשטעטיגן די אידענטיטעט

.5. Processing

נאכדעם האט מען א דיגיטאלן קלאן. צו שאפן א ווידעא מיטן אוואטאר זענען די שריט:

."Avatar Video" און "Create Video" 1. Select

.2. Choose Your Avatar

.3. Enter Your Script

די HeyGen פלענער (10:08:30–10:09:30)

אויף app.heygen.com ווייזט זיך ערשט א בלאט "Looks like your team's already using HeyGen". דערנאך קומט דער בלאט "Choose the perfect plan for you", מיט (Yearly (save up to 22%:

- Free – $0. 3 ווידעאס א חודש, ביז 3 מינוט, 3 Avatar IV, 720p, Standard processing ווידעאס א חודש.
- Creator – $24 א חודש (פריער $29), $288 א יאר. Unlimited ווידעאס, ביז 30 מינוט, 1080p, Fast processing, 5 מינוט Avatar IV א חודש.
- Team – $30 פאר א "seat" א חודש (פריער $39), מינימום $720, 2 seats א יאר. אנגעצייכנט אלס "Most Popular". ביז 30 מינוט, 5 מינוט Avatar IV, 4K, Faster processing.
- Enterprise – "Let's talk". קיין צייט-גרענעץ נישט, 5 מינוט Avatar IV, 4k, Fastest processing.

א ווידעא מיט Avatar IV (10:10:00–10:13:00)

אויפן היים-בלאט שטייט "Explore our next-gen Avatar IV" מיט א ביישפיל-ווידעא פון 0:47. אין דער זייטן-מעניו זעט מען Home, Projects, Templates, Avatars, AI Voice, Brand, Uploads און Integrations.

דאס פענצטער "Realistic talking video with Avatar IV" זאגט: "Turn a single photo and script into a high-quality avatar video". מען לייגט ארויף א בילד, און אין דעם Script-פעלד שטייט:

"Hey there I'm your digital assistant, here to show you how our amazing software is all powered by cutting-edge AI! – revolutionizing the way you edit photos, videos, and more"

דער סקריפט האט 178/210 אותיות, מאקסימום 15 סעקונדע. דאס קול איז "Voice 3 - Juno". אין דעם פעלד Custom motion (Beta) ווערט באשריבן אז דער אוואטאר האלט א בילד און קלאפט דערויף מיטן פינגער, דאס בילד פארוואנדלט זיך ווען דער AI פאררעכט די קאלירן און די ליכט, און ער מאכט א שריט פאראויס און ווייזט אראפ צום לינק.

מען שאלט איין "Enhance prompt" און "More expressive", קלייבט 720p, און דריקט "Generate video".

אין Projects שטייט דער ווידעא "Avatar IV Video" (Today at 4:30 PM) מיטן סטאטוס "In line for processing" און "Upgrade to skip the line". ביי Folders שטייט "You have no folders". דערנאך ווערט געוויזן דער פארטיגער ווידעא.

שאפן אן אוואטאר (10:13:30–10:14:30)

אין app.heygen.com/avatars ווערן געוויזן דריי מעגלעכקייטן:

- Photos + Hyper Realistic Avatar (0/1 Slots, Videos): מען לייגט ארויף א ווידעא.
- Custom Photo Avatar: פון עקזיסטירנדע בילדער.
- Generate: באשרייבן דעם אויסזען מיט א פראמפט.

מען קלייבט Generate, און עס עפנט זיך דער בלאט "Describe your avatar's look". די פעלדער זענען: "Name" neil", "Age" Young Adult", "Gender" Man", "Ethnicity" Unspecified". אין "Describe your avatar" שטייט א באשרייבונג פון איינעם וואס קען ערקלערן קאמפליצירטע פרטים אויף א פשוטן אופן.

צום סוף ווייזט זיך "Your avatar is almost ready!" מיט צוויי קנעפלעך: "Choose a voice for me" און "Add a voice". אויף דער זייט שטייט "Train Model" ("5min processing time") צו שאפן "Generate Looks".

HeyGen און AI-פראיעקטן פאר אנהייבער (לויטן עקראן) (10:15:00)

HeyGen: אוואטאר, ווידעא-רעדאקטאר און Labs (10:15:00)

אויפן עקראן וועלכן מען זעט HeyGen (app.heygen.com), אין א בלעטל פון דעם אוואטאר מיטן נאמען "neil". אין דער מיט שטייט א בילד פונעם אוואטאר מיטן צייכן "AI Photo". אויף דער לינקער זייט איז דער מעניו: Templates, Projects, Home, Upgrade, Create video, און אונטער "Assets": Avatars, AI Voice, Brand, Uploads, Integrations. אויבן רעכטס זענען דא די קנעפלעך "Avatar Voices" און "Generate Looks".

אין דער רעכטער זייל "Generate Looks" שטייט: "Generate Looks via prompts by training a Custom AI Model", דאס הייסט מען קען שאפן נייע אויסזעענישן פונעם אוואטאר דורך פראמפטס, נאכדעם וואס מען טריינירט א באזונדערן AI מאדעל. דארט איז אויך א לינק "See how it works", דער טיטל "Train AI Model", א קנעפל "Train Model", און די באמערקונג "~5min processing time", אז די ארבעט נעמט בערך פינף מינוט.

דער ווידעא-רעדאקטאר (10:15:30)

דערנאך עפנט זיך דער רעדאקטאר (create-v4) מיטן טיטל "Untitled Video". דער פארמאט איז 9:16, א שטייענדיקער ווידעא. אויבן שטייען די קנעפלעך Brand Kit, Feedback, Preview און Generate. אויף דער לינקער זייט איז די ערשטע סצענע, מיט דעם קול "Voice 1 - neil" און א פעלד "Type your script here, or upload audio", וואו מען שרייבט אריין דעם טעקסט אדער מען לאדנט ארויף א קול-פייל. אונטער דעם זענען די אפציעס Scene, Audio, Pause און Voice Director.

אויף דער רעכטער זייט איז א געצייג-ליניע: Avatars, Text, Media, Elements, Captions, Background, Layers. אין דער מיט שטייט דער אוואטאר. אונטן איז די צייט-ליניע מיט איין סצענע פון 0.0s, און א וואַרנונג "No script", אז קיין טעקסט איז נאך נישט אריינגעשריבן.

Brand Kit (10:16:00)

אין א קלענערן בילד וועלכן גיין בלעטל "Brand" (app.heygen.com/brand), אונטער "My brands" זעט מען "Demo HeyGen Brand Kit". דארט זענען לאגאס פון "momentum", קאליר-פלעקלעך מיט די קאודס #0165D0, #FEA0AB און #E4FD76, די שריפטן "Khula ExtraBold" און "Lithos Pro Medium", און א זאמלונג פון בילדער. עס איז אויך דא א קנעפל "New Brand Kit". אינעם מעניו זעט מען אויך Labs און Notifications.

(10:16:30) HeyGen Labs

דער בלעטל "HeyGen Labs" טראגט די אונטערשריפט "Explore more creative and useful AI generation capabilities by HeyGen". עס ווערן געוויזן פיר מעגליכקייטן:

- Video Podcast (New): שאפט א AI ווידעא-פאדקאסט מיט צוויי רעדנער פון א pdf אדער א Youtube לינק.
- Interactive Avatar: א AI שמועס וואס פילט זיך ווי מען רעדט מיט א לעבעדיקן מענטש.
- Instant Highlights (Beta): מאכט פון א לאנגן ווידעא עטליכע קורצע "highlight" שטיקלעך מיט איין קליק.
- URL to Video (Beta): מאכט פון א וועב-לינק א ווידעא.

אין דער לינקער זייל שטייט "Rendering your video 16%", וואס ווייזט אז דער ווידעא ווערט נאך פראדוצירט, און אונטן "3 videos left / 3".

די היים-בלאט און לעצטע ארבעטן (10:17:00)

אויף דער היים-בלאט (חשבון "Neelesh", Free) זענען דא קנעפלעך פאר: Translate Video, Create Your Avatar, Product Placement, Interactive Avatar, PPT/PDF to Video, Instant Highlights און Batch Mode. אונטער "Recent creations" ליגן דריי ווידעאס:

- "trainer" (Today at 5:06 PM), מיטן סטאטוס "In line for processing", דאס הייסט ער ווארט נאך אויף פראצעסירונג.
- "Untitled Video" (4:57 PM), מיטן סטאטוס "16% Ready".
- "Avatar IV Video" (4:41 PM), א פארטיקער ווידעא פון 11 סעקונדעס.

דער עקראן בלייבט אזוי ביז 10:17:30. ביי 10:18:00 קומט א איבערגאנג-בילד מיט א "AI" טשיפ ארומגערינגלט מיט דאלאר-צייכנס.

פראיעקט: Fake News Detection Using Machine Learning (10:19:30)

דאס איז א נייער טייל, וואס ברענגט פראיעקט-אידייען אויף סליידס. דער ערשטער סלייד הייסט "Fake News Detection Using Machine Learning", און ער שטעלט פאר א מאשין-לערנען פראיעקט וואס דערקענט פאלשע נייעס. ער האט די פאלגנדיקע שריט:

- Data Collection: דאטא-קוועלן זענען Kaggle, LIAR און FakeNewsNet.
- Preprocessing: מען גרייט צו דעם טעקסט מיט NLTK און spaCy.
- Feature Extraction: מען ציט ארויס די אייגנשאפטן פונעם טעקסט מיט NLTK און spaCy.
- Model Training: מען טריינירט א Logistic Regression, Naïve Bayes אדער RF מאדעל.
- Model Evaluation: מען מעסט דעם מאדעל לויט Accuracy, Recall און F1-Score.

(10:21:30) Image Recognition Using CNN

דער סלייד באשרייבט א פראיעקט פון בילדער-דערקענונג מיט א CNN:

- Dataset: מען נוצט CIFAR-10 און Kaggle.

- Preprocessing: מען גרייט צו די בילדער מיט OpenCV און PIL.
- Model Architecture: מען בויט דעם מאדעל מיט Keras און Tensorflow.
- Model Training: מען טיילט די דאטא אויף Training און Validation sets, און מען טריינירט דעם CNN.

(10:22:00) AI-Based Recipe Recommendation System

ביי 10:22:00 ווערט אויפן עקראן אנגעזאגט "3. AI-Based Recipe Recommendation System", א סיסטעם וואס רעקאמענדירט רעצעפטן. דער סלייד ((10:23:00–10:23:30 ווייזט:

- Dataset: מען נוצט Recipe1m און Kaggle.
- Preprocessing: מען נארמאליזירט און רייניקט די דאטא (Normalize & Clean data).
- Algorithm: CBF און Collaborative Filtering.
- Model Training: מען רעכנט די ענלעכקייט צווישן רעצעפטן מיט Cosine similarity.
- Interface & Tools: BeautifulSoup, pandas, scikit-learn און Flask.

(10:25:00) Chatbot with Sentiment Analysis

דער סלייד באשרייבט א טשאטבאט וואס דערקענט דעם געפיל פונעם באניצער:

- Chatbot Framework: Dialogflow אדער Rasa.
- Sentiment Analysis: מען אנאליזירט דאס געפיל מיט Vader אדער Bert.
- Conversational Flow: דער שמועס פירט זיך לויט דעם צי דאס געפיל איז Positive, Negative אדער Neutral.
- Integration: מען בויט אן אינטערפעיס. Deployment: מען שטעלט עס ארויף אויף א וועבזייטל אדער אן אפליקאציע.
- Tools: Dialogflow, Rasa, VADER, transformers, Flask.

ביי 10:26:30 ערשיינט אויפן עקראן די אונטערשריפט "Image generation, Video prediction".

(10:27:00) AI-Powered Image Colorization

דער סלייד באשרייבט א פראיעקט וואס פארבט אן שוואַרץ-ווייסע בילדער מיט AI:

- Dataset: מען נוצט קאלירטע בילדער פון Kaggle (Coloured images).
- Preprocessing: מען נארמאליזירט די פיקסלען פון די בילדער (Normalize image pixels).
- Model Architecture: U-net און GANs.
- Training & Evaluation: מען מעסט דעם רעזולטאט מיט MSE און PSNR.
- Deployment: מען שטעלט עס ארויף ווי א Web interface.

אויפן סלייד זעט מען אויך דעם TensorFlow לאגא.

(10:28:30) Object Detection Using YOLO

ביי 10:28:30 ערשיינט די אונטערשריפט "Preprocess image data, Format bounding box labels and Train a YOLO model". דערנאך קומט דער סלייד "Object Detection Using YOLO", א פראיעקט וואס גענפינט און דערקענט חפצים אין בילדער:

- Dataset: COCO (אזוי געשריבן), PASCAL VOC און Kaggle.
- Preprocessing: מען פארגרינגערט די בילדער צו א געוויסער גרייס (Resize Images) און מען נארמאליזירט די פיקסל-ווערטן (Normalize Pixel Values).
- Model Architecture: YOLO.
- Training & Evaluation: מען טריינירט מיט Darknet און מען מעסט מיט IoU.
- Deployment: א סיסטעם וואס דערקענט חפצים אין רעאלער צייט (Real-time object detection system).

AI-פעאיקייטן פאר פרילענסן און פרנסה (לויטן עקראן) (10:30:00)

סקיל 1: פראמפט-אינזשעניריע (10:31:00)

דער ערשטער פונקט אין דעם חלק איז א סלייד מיטן טיטל "Skill 1: Prompt Engineering", ד.ה. די ערשטע פעאיקייט אויף דער רשימה איז דאס שרייבן פראמפטס.

גלייך דערנאך ווערט אויפן עקראן געוויזן א בלעטערער אויף chatgpt.com. דער אויבנאויף פון ChatGPT איז ליידיג, מיט די ווערטער "Ready when you are." און א פעלד "Ask anything" מיט א "Tools" קנעפל, וואו מען קען אריינשרייבן א פראמפט.

וועלכע געצייג באנוצן פראמפט-אינזשעניערן? (10:32:00)

דער סלייד "Which tools do prompt engineers use?" ווייזט דריי געצייג:

- Jasper AI
- ChatGPT (by OpenAI)
- OpenAI Playground: א בילד פונעם Playground עקראן מיט א מעניו וואו מען קען אויסוויילן Chat, Assistants (Beta) אדער Complete (Legacy), און א פעלד "SYSTEM".

ווי אזוי צו פרילענסן אלס פראמפט-אינזשעניער (10:33:00)

דער סלייד "How to freelance as a prompt engineer?" ברענגט צוויי פונקטן:

- שרייבן קלארע AI פראמפטס כדי צו שאפן אינהאלט אדער אידייען.
- טריינירן טשעטבאטס או זיי זאלן בעסער ענטפערן קונים' פראגעס.

ביישפילן פון דער וועלט (10:33:30)

אונטערן טיטל "Real-world examples" ווערן געברענגט דריי ביישפילן:

• א סטארטאפ דינגט פראמפט-אינזשעניערן צו שאפן הונדערטער פראדוקט-באשרייבונגען יעדע וואך, און דאס שפארט הונדערטער שעה פון שרייבן מיט דער האנט.

• מארקעטינג-אגענטורן באנוצן פראמפט-אינזשעניערן צו שאפן פערזענלעכע אימעיל-קאמפיינס און סאשעל-מידיע מודעות אין מינוטן, נישט אין טעג.

• קונים-סערוויס באטס ווערן געטריינירט דורך פראמפט-אינזשעניערן צו ענטפערן שוועריקייטן און העפלעך.

ווי אזוי אנצוהייבן לערנען (10:34:30)

דער סלייד "How to start learning prompt engineering?" גיט פיר עצות:

• עפנט ChatGPT און פרובירט! פרעגט די זעלבע פראגע אויף פארשידענע אופנים און זעט וואס ארבעט בעסער.

• איבט זיך צו זאגן דעם AI ער זאל "act as", ד.ה. זיך פירן ווי פארשידענע מומחים אדער פערזאנען.

• לערנט זיך צו געבן קלארע אנווייזונגען וועגן דעם פארמאט וואס איר ווילט, למשל "Give me 5 bullet points" אדער "Summarize in 3 sentences".

• באנוצט פרייע מקורות און לעקציעס אויפן אינטערנעט. דער וועבזייטל פון OpenAI האט גוטע אנווייזונגען.

פרנסה: פראמפט-אינזשעניער (10:35:00)

א סלייד "Money Talk" ווייזט באצאלונג:

• אין אמעריקע: $60,000 ביז $120,000 א יאר.

• אין אינדיע: ₹6 לאך ביז ₹12 לאך א יאר.

סקיל 2: ווייכווארג-אנטוויקלונג מיט AI הילף (10:35:30)

דער סלייד "Skill 2: AI-Assisted Software Development" זאגט: מען דארף שוין נישט זיין קיין מומחה אין קאוד כדי צו בויען אפליקאציעס אדער ווייכווארג. דאנק AI און no-code tools קען יעדער איינער שאפן אפליקאציעס וואס לייזן אמת'ע פראבלעמען. עס ווערן געוויזן דריי געצייג: GitHub Copilot, bubble און Directual.

דערנאך ווערט אויפן עקראן געוויזן א שמועס אין ChatGPT 4o מיט "Step 2: Setup basic file structure". דארט שטייען די טעקעס:

• index.js (backend entry point)

• public/index.html (דער באניצער-אויבנאויף)

• public/style.css (אפציאנעל, פארן אויסזען)

• public/script.js (frontend JS פאר)

אונטער "In index.js, serve the frontend" ווערט געוויזן א קורצער JavaScript קאוד:

```
const express = require('express');

const app = express();
```

;3000 = const port

די ערשטע שורה לאדט די express ביבליאטעק, די צווייטע שאפט אן אפליקאציע-אביעקט, און די דריטע שטעלט דעם פארט 3000.

פרנסה: AI ווייכווארג-אנטוויקלער (10:37:00)

"Money Talk":

- אין אמעריקע: $100k ביז $200k א יאר.
- אין אינדיע: ₹12 לאך ביז ₹24 לאך א יאר.

סקיל 3: בילדער און ווידעא רעדאגירן מיט AI (10:37:30)

דער סלייד "Skill 3: AI-Powered Photo & Video Editing" ווייזט א פאר-און-נאך ביישפיל אין א טעלעפאן-אפליקאציע פון ווידעא-רעדאגירן. אין דער צווייטער בילד איז אן "AI Effects" מעניו מיט אפציעס ווי Hot, Scene, Border, Bling, Spiral, און צוגעלייגטע פיגורן. דער טעקסט זאגט: פאטאגראפן און שאפער נעמען אויף יעדן טאג אסאך מאטעריאל, אבער אלעס רעדאגירן נעמט אייביג. דא קומט AI איריין, כדי צו גיכער מאכן די ארבעט און עס גרינגער מאכן.

דער נעקסטער סלייד ווייזט Runway.ai, מיט אן עקראן "Generate Videos": שאפן ווידעאס פון טעקסט, בילדער אדער ווידעא-קליפס.

ווי הייבט מען אן פרילענסן? (10:38:30)

דער סלייד "How do you get started with freelancing?" ווייזט די פלאטפארמעס fiverr, upwork און freelancer. ער זאגט אז מען קען פאקן די סערוויסעס, למשל:

- "Get 50 enhanced photos delivered in 24 hours"
- "Viral-ready video clips edited with AI magic"

פרנסה: AI בילדער/ווידעא רעדאגירער (10:40:00)

"Money Talk": אין אמעריקע $60,000 ביז $170,000 א יאר.

סקיל 4: אנטוויקלען AI אגענטן (10:40:30)

דער סלייד "Skill 4: AI Agent Development" ווייזט אן עקראן פון ChatDev, "Communicative Agents for Software Development". אויף איין זייט איז א בילד פון א פירמע-אפיס מיט פיגורן, און אויף דער אנדערער זייט א שמועס צווישן די אגענטן. אויבן שטייט די אויפגאבע: שאפן א to-do list אפליקאציע מיט Alpine.js.

דער סלייד "How Does It Work?" מיטן זעלבן בילד ערקלערט:

- קודם מאכט מען א פלאן פון דער ארבעט וואס דער AI אגענט דארף טון: וואס פונקטליך דארף ער באהאנדלען?
- דערנאך טריינירט מען דעם אגענט מיט אמת'ע ביישפילן, ווי פריערדיגע טעלעפאן-רופן, טשעט-רעקארדס אדער אימעילס, כדי ער זאל זיך אויסלערנען ריכטיג צו טון די ארבעט.

פרנסה און אנהייב: AI אגענטן (10:41:30)

"Money Talk": אין אמעריקע $160,000 ביז $300,000 א יאר.

דער סלייד "How do you get Started?" ווייזט Dialogflow און RASA. א דיאגראם ווייזט RASA אין מיטן, פארבונדן מיט Website, Facebook Messenger, Slack, RocketChat, Telegram, Mattermost, Twilio, Custom Connectors און Microsoft Bot Framework.

א שנעלע אויפגאבע (10:42:30)

אונטערן טיטל "Quick challenge for you": טראכט וועגן א פשוט'ע ארבעט וואס חזר'ט זיך איבער, וואס איר אדער איינער וואס איר קענט טוט יעדן טאג, למשל ענטפערן די זעלבע פראגעס נאכאמאל און נאכאמאל. ווי אזוי וואלט איר אויסגעלערנט אן AI אגענט דאס צו באהאנדלען? שרייבט אראפ די טריט וואס איר וואלט אים געגעבן!

סקיל 5: שרייבן און קאפירייטינג מיט AI (10:43:00)

דער סלייד "Skill 5: AI-Powered Writing & Copywriting" זאגט: איר האט מסתמא געהערט דעם אויסדרוק "Content is king", און עס איז אמת! יעדער וועבזייטל, ביזנעס און בראנד דארף פרישן, אינטערעסאנטן אינהאלט כדי צו כאפן אויפמערקזאמקייט און וואקסן. די געצייג וואס ווערן געוויזן זענען Jasper, דער OpenAI לאגא און Writesonic.

דער סלייד "How to Start?" זאגט: דער סוד איז צו פאראייניגן אייער אייגענע שאפערישקייט מיט ענטוווורפן וואס AI שאפט.

"Money Talk": אין אמעריקע $60,000 ביז $120,000 א יאר.

סקיל 6: בויען און פירן AI טשעטבאטס (10:44:30)

דער סלייד "Skill 6: Build & Manage AI Chatbots" זאגט: די קלוגע געהילפן ארבעטן 24/7. זיי ענטפערן אפטע פראגעס (FAQs), שטעלן אפוינטמענטס, העלפן פארקויף-טימס און פירן קונים, אלעס אויטאמאטיש. ביישפילן פון פראגעס: "What are your opening hours?" און "Can I book an appointment?"

AI-שקילס און מאשין-לערנען אינטערוויו-פראגעס (לויטן עקראן) (10:45:00)

ווי אנצוהייבן מיט שמועס-באטס (10:45:00)

אויפן עקראן ווערט געוויזן א סלייד מיטן טיטל "How to Start?", מיט א בילד פון א ראבאט און א מענטש מיט שמועס-בלאזן. צוויי געצייג ווערן אויסגערעכנט:

- Botsonic: גוט צו בויען נאטירליכע באטס וואס פירן א שמועס.
- Chatbase: מען קען ארויפלאדן דאקומענטן און טריינירן דעם באט צו ענטפערן אויף געוויסע פראגעס.

"Money Talk": פארדינסט (10:45:30)

דער סלייד "Money Talk" ווייזט א פאן פון אמעריקע מיט די צאלן $90,000 ביז $160,000 א יאר.

Skill 7: AI-געטריבענע באצאלטע רעקלאמעס (10:46:00)

דער סלייד "Skill 7 : AI-Driven Paid Advertising Management" ברענגט א קאסטן מיטן טעקסט: וועט עס האט שוין פראבירט לויפן רעקלאמעס אויף Facebook, Google אדער Instagram, ווייסט אז עס איז נישט סתם קליקן אויף "boost". עס נעמט צייט, געלט און אסאך פראבירן ביז מען באקומט רעקלאמעס וואס ברענגען באמת קונים.

אין די קומענדיגע סעקונדעס קומען צו צוויי בילדער פון געצייג:

- Pencil: די וועבזייטל ווייזט דעם לאזונג "Make ads that work." און "Double your productivity and performance with Generative AI", מיט די קנעפלעך "Get Started Free" און "Book a Demo".
- adzooma: א בילד מיט די לאגאס פון Google Ads, facebook ads און Microsoft.

ווי וועט מען אן AI-רעקלאמע-קאנסולטאנט (10:47:00)

אונטערן טיטל "How to Start?" שטייט א בילד מיטן אונטערשריפט "AI Advertising Consultant", און דריי פונקטן:

- הייבט אן מיט ווערן אן עקספערט אין די AI-רעקלאמע-געצייג און פלאטפארמעס.
- לערנט זיך ווי זיי ארבעטן, און ווי אזוי אויפצושטעלן קאמפיינס וואס ברענגען געלט פאר קליינע געשעפטן.
- שטעלט אייך אוועק אלס AI-רעקלאמע-קאנסולטאנט וואס העלפט געשעפטן לויפן פראפיטירליכע קאמפיינס, און דערביי שפארן צייט און שניידן הוצאות.

דער ווייטערדיגער "Money Talk" סלייד ((10:47:30 ווייזט צווי פענער: אמעריקע, $80,000 ביז $150,000 א יאר; אינדיע, ₹8 lakh ביז ₹15 lakh א יאר.

Skill 8: CRM און פארקויף-קאנסולטינג מיט AI (10:48:00)

דער סלייד "Skill 8 : CRM & Sales Consulting with AI " ווערט געווזין אין א קלענערן פענצטער, מיט א בילד. דער טעקסט זאגט אז געשעפטן וועגן זיך צו Customer Relationship Management סיסטעמען (CRMs) וואס ווערן געטריבן דורך AI. די קלוגע סיסטעמען מאכן אויטאמאטיש לאנגווייליגע ארבעטן, ווי פירן לידס (lead management), נאכגיין פארקויפן (sales tracking), און אפילו געפינען נייע מעגליכע קונים. אזוי קענען די פארקויף-מאנשאפטן זיך קאנצענטרירן אויף פארמאכן געשעפטן.

דער "How to Start?" סלייד (10:48:30 ביז 10:49:00 זאגט: כדי אריינצוקומען אין דעם פעלד, דארף מען זיך ספעציאליזירן אין אריינשטעלן AI אין CRMs. דערנאך קומען דריי פונקטן:

- אויסרייניגן צעמישטע קונים-דאטא, כדי דער AI זאל ארבעטן מיט פינקטליכע אינפארמאציע.
- ניצן AI-געצייג צו געפינען און אפשאצן נייע לידס.
- אויטאמאטיזירן קאמפיינס פון זיך פארבינדן מיט קונים, ווי שיקן נאכפאלג-אימעילס אדער אויסשטעלן טעלעפאן-רופן.

אונטן שטייט א בילד: "AI" + "CRM".

Skill 9: פירן סאשעל-מעדיע בראנדס מיט AI (10:49:30)

דער סלייד "Skill 9: Manage Social Media Brands with AI" זאגט אז פירן סאשעל מעדיע פאר געשעפטן איז נישט גרינג. די געצייג קענען אויטאמאטיזירן דאס שאפן פון אינהאלט (automate content creation) און דאס אויסשטעלן א צייט-פלאן (scheduling). דאס הייסט, זיי העלפן שרייבן פאסטס און באשטימען ווען זיי זאלן אויטאמאטיש ארויסגיין. ביי 10:50:00 קומט צו א בילד פון א טעלעפאן מיט מעדיע-אייקאנס, און א סקרינשאט פונעם געצייג Ocoya.

דער "How to Start?" סלייד ((10:50:30 ברענגט:

- קודם, לערנט זיך ניצן AI-געצייג צו שאפן אינהאלט, און אפליקאציעס צו פלאנירן סאשעל-מעדיע פאסטס.
- הייבט אן מיט פירן עטליכע בראנדס אויפאמאל, און ניצט AI כדי די פאסטס זאלן בלייבן פריש און קאנסעקווענט.
- קוקט נאך אין די אנאליטיקס וועלכע פאסטס ארבעטן גוט.

אונטן ווערט געוויזן א רינג פון לאגאס (Twitter, Facebook, Vimeo, Google+, LinkedIn, YouTube, Skype, Pinterest) ארום א טשיפ.

Skill 10: AI-איבערזעצונג און SEO (10:51:00)

דער סלייד "Skill 10: AI Translation & SEO Services" זאגט: אסאך געשעפטן ווילן דערגרייכן קונים איבער דער וועלט. דערפאר דארפן זיי איבערזעצן זייערע וועבזייטלעך, פראדוקטן און אינהאלט אויף פארשידענע שפראכן. דאס בילד ווייזט א קאמפיוטער מיט א פארגרעסער-גלאז און "SEO".

דער "How to Start?" סלייד (10:51:30 ביז (10:52:00 ווייזט:

- איבערזעצן אינהאלט פאר אן אינטערנאציאנאלן עולם.
- פארבעסערן SEO כדי צו באקומען מער באזוכער און קונים.

אונטער "Tools to get started" ווערן געוויזן די לאגאס פון ChatGPT, Google Translate, אן אייקאן וואס זעט אויס ווי DeepL, און SURFER.

מאשין-לערנען אינטערוויו-פראגעס און ענטפערס (10:52:30)

א נייער חלק הייבט זיך אן מיטן טיטל "Machine Learning Interview Questions And Answers", מיט א בילד פון א ראבאט און א מוח-אייקאן. די קומענדיגע סליידס טראגן אלע דעם קאפ "Machine learning Interview Questions & Answers". ביי 10:53:00 ביז 10:53:30 איז דער עקראן ליידיג, א מעבר צווישן סליידס.

- פראגע 05: "What is Reinforcement Learning?" (10:54:00 ענטפער: Reinforcement Learning איז א סארט מאשין-לערנען, וואו אן אגענט לערנט זיך מאכן באשלוסן דורך טון אקציעס און באקומען באלוינונגען אדער שטראפן. דער ציל איז צו מאקסימיזירן די צוזאמגערעכנטע באלוינונג (cumulative reward).
- פראגע 07: "What is Overfitting?" (10:54:30 ענטפער: Overfitting טרעפט זיך ווען א מאשין-לערנען מאדעל ארבעט גוט אויף די טריינירונג-דאטא, אבער שלעכט אויף נייע דאטא וואס ער האט

נאך נישט געזען. דאס ווייזט אז דער מאדעל האט זיך אויסגעלערנט דעם "רַאש" (noise) און די פרטים פון דער טריינירונג-דאטא, אנשטאט די אמת'ע מוסטערן.

• פראגע 09: "What is a Confusion Matrix?" (10:55:00 דער ענטפער איז נאך נישט ארויפגעקומען אויפן בילד.

• פראגע 10: "What is Cross-Validation?" (10:55:30 ענטפער: Cross-validation איז א טעכניק צו אפשאצן ווי גוט די רעזולטאטן פון א סטאטיסטישער אנאליז וועלן גילטיג זיין אויף אן אומאפהענגיגן דאטא-סעט. מען צעטיילט די דאטא אויף טיילן (subsets), מען טריינירט דעם מאדעל אויף עטליכע טיילן, און מען פרואווט אים אויס אויף די איבעריגע.

• 10:56:00: א דיאגראם פון "relevant elements" און "retrieved elements". אין א קרייז ווערן געווייזן true positives) גרין) און (false positives רויט), און ארום אים false negatives און true negatives. אונטן ווערן געווייזן די פארמלען Precision און Recall: Precision איז true positives געטיילט דורך אלע ארויסגעבראכטע עלעמענטן, און Recall איז true positives געטיילט דורך אלע רעלעוואנטע עלעמענטן. ביי 10:56:30 איז דער עקראן ווידער ליידיג.

• פראגע 14: "What is Regularization?" (10:57:00 דער ענטפער ווערט נישט געזען.

• פראגע 15: "What is the Bias-Variance Tradeoff?" (10:57:30 ענטפער: דאס איז א יסודות'דיגע שאלה אין מאשין-לערנען. עס גייט אין באלאנסירן צווישן צוויי סארטן טעותים: דער טעות וואס קומט פון די הנחות פונעם מאדעל (bias), און דער טעות וואס קומט פון דער קאמפליצירטקייט פונעם מאדעל (variance). א גוטער מאדעל דארף האבן נידעריגן bias און נידעריגע variance . דער גראפיק ווייזט Model Complexity קעגן Training Error: די גייט אראפ, און די Validation Error פאלט קודם און שטייגט דערנאך. לינקס איז דער שטח "High Bias", און רעכטס "High Variance".

• פראגע 17: "What is Gradient Descent?" (10:58:00 ענטפער: Gradient Descent איז אן אפטימיזאציע-אלגאריטם וואס ווערט גענוצט צו מינימיזירן די cost function אין מאשין-לערנען מאדעלן. ער פאררעכנט די פאראמעטערס פונעם מאדעל שריט ביי שריט, אין דער ריכטונג ווו די cost function פאלט אראפ דאס שטארקסטע (steepest descent). דער גראפיק ווייזט א קרומע ליניע אין פארעם פון א "V" (C קעגן w). קוגלען ראלן אראפ צום "Global Minimum cost", און עס איז אנגעצייכנט א "Gradient".

• פראגע 18: "What is the Difference Between Bagging and Boosting?" (10:58:30 ענטפער: ביי Bagging (Bootstrap Aggregating) טריינירט מען עטליכע מאדעלן אויף פארשידענע טיילן פון דער דאטא, און מען נעמט דעם דורכשניט פון זייערע פאראויסזאגונגען. ביי Boosting טריינירט מען די מאדעלן איינס נאכן אנדערן, און יעדער נייער מאדעל פרובירט צו פאררעכטן די טעותים פון די פריערדיגע. דער דיאגראם ווייזט Bagging אלס "Parallel" (גלייכצייטיג) און Boosting אלס "Sequential" (איינס נאכן אנדערן).

• פראגע 19: "What is a Decision Tree?" (10:59:00 ענטפער: א Decision Tree איז א non-parametric supervised learning אלגאריטם פאר קלאסיפיקאציע און רעגרעסיע. ער צעטיילט די דאטא אויף טיילן לויט די ווערטן פון די אריינגאנג-אייגנשאפטן (input features), און דאס שאפט א סטרוקטור פון באשלוסן וואס זעט אויס ווי א בוים. דאס בילד ווייזט חיות וואס ווערן צעטיילט לויט דער פראגע "yellow? == Color": ביי True גייען זשיראפן און טיגערס, און ביי False עלעפאנטן און מאלפעס.

• פראגע 20: "What is Random Forest?" (10:59:30 ענטפער: Random Forest איז אן ensemble learning מעטאד וואס פאראייניגט עטליכע decision trees, כדי צו פארבעסערן די פינקטליכקייט און די פעסטקייט פונעם מאדעל. יעדער בוים ווערט געבויט מיט א צופעליגן טייל פון די features און דאטא-פונקטן, און דערנאך נעמט מען דעם דורכשניט פון זייערע פאראויסזאגונגען. דאס זעלבע בוים-בילד ווערט ווידער געוויזן.

ענדע פון די אינטערוויו-פראגעס (לויטן עקראן) (11:00:00)

א נייער סלייד הייבט זיך אן (11:00:00)

אויפן עקראן שטייט דער טיטל "Machine learning Interview Questions & Answers" מיט'ן Simplilearn צייכן. דער סלייד איז נאך ליידיג. מען זעט נאר א בלויען שטרייף אויף דער לינקער זייט, וואו עס קומט שפעטער די נומער פון דער פראגע.

פראגע 23: וואס איז א נויראל-נעץ? (11:00:30)

דער סלייד טראגט די נומער 23 מיט דער פראגע "What is a Neural Network?". אונטער "Solution" שטייט די דעפיניציע:

א נויראל-נעץ איז א סעריע פון אלגאריטמען וואס פרובירן צו דערקענען די פארבארגענע באציאונגען (underlying relationships) אין א סך דאטא. עס טוט דאס דורך א פראצעס וואס איז נאכגעמאכט ווי דער מענטשלעכער מוח ארבעט (brain operates). עס באשטייט פון שיכטן (layers) פון נאדעס (nodes) וואס זענען צונויפגעבונדן איינס מיט'ן אנדערן. די נאדעס הייסן "neurons".

רעכטס ווערט געוויזן א בילד פון א ביאלאגישן נוירון מיט דריי באצייכנטע טיילן:

• Dendrite: דער אריינגאנג (input) צום נוירון.

• Cell body: דא ווערט די אינפארמאציע פראצעסירט.

• Axon: דער ארויסגאנג (output) פונעם נוירון.

פראגע 25: וואס איז א CNN? (11:01:00)

דער סלייד פרעגט "What is Convolutional Neural Network (CNN)?". דער שריפטלעכער ענטפער ווערט נישט געוויזן אויף דעם פריים, נאר א דיאגראם פונעם גאנצן פראצעס פון לינקס קיין רעכטס:

• Input Image: א טאבעלע פון נולן און איינסן וואס שטעלט פאר א בילד.

• דערנאך קומט די שטאפל Convolution, וואס פירט צו א Convolution Layer. דאס איז א הויפן פון שיכטן.

• אונטער דעם Convolution Layer ווערט געוויזן א גראפיק פון ReLU.

• דערנאך קומט Pooling, וואס פירט צו א Pooling Layer.

• צום סוף קומט Flattening, וואס פירט צו א Fully Connected Layer. דאס איז א נעץ פון נאדעס, אלע פארבונדן איינס מיט'ן אנדערן.

פראגע 26: וואס איז א RNN? (11:01:30)

דער סלייד פרעגט "What is a Recurrent Neural Network (RNN)?". דער ענטפער לויטעט:

א Recurrent נויראל-נעץ איז א סארט נויראל-נעץ וואס איז געבויט פאר דאטא וואס קומט אין א ריי נאכאנאנד (sequential data). עס האט פארבינדונגען וואס שאפן ריכטונג-געבונדענע קרייזן (directed cycles). דאס דערלויבט אים צו האלטן א זכרון פון פריערדיגע אריינגאנגען, און צו פראצעסירן א ריי פון דאטא.

דער דיאגראם ווייזט לינקס א געווענליכן נעץ מיט דריי טיילן:

- Input Layer, מיט x נאדעס.
- צוויי Hidden Layers, מיט h נאדעס.
- Output Layer, מיט y נאדעס.

א פייל פירט צו א פארקירצטן בילד מיט'ן נאמען "Recurrent Neural Network". דארט גייט x ארויף צו h דורך B, h גייט ארויף צו y דורך A, און א שלייף C גייט פון h צוריק צו זיך אליין.

פראגע 27: Batch קעגן Stochastic Gradient Descent (11:02:00)

די פראגע לויטעט: "What is the Difference Between Batch Gradient Descent and Stochastic Gradient Descent?". דער ענטפער:

Batch Gradient Descent רעכנט אויס דעם גראדיענט פון דער cost function מיט'ן גאנצן טריינירונג-דאטא-סעט. Stochastic Gradient Descent (SGD) רעכנט אויס דעם גראדיענט מיט בלויז איין טריינירונג-ביישפיל אויף איין מאל. SGD איז שנעלער, אבער מער "noisier", דאס הייסט מער אומרואיג.

דער גראפיק ווייזט א קורוו אין פארם פון א U. אויף דער לינקער זייט פונעם שיפוע זענען פארהאן קלייניע בויגנס וואס ווייזן די טריט אראפ. די פיילן זענען באצייכנט "Error 1", "Error 2", און ביים אונטערשטן פונקט "Error Min".

פראגע 29: וואס איז Transfer Learning? (11:02:30)

דער סלייד פרעגט "What is Transfer Learning?". אויף דעם פריים זעט מען נאר א דיאגראם פון א קינסטלעכן נויראן, אן א שריפטלעכן ענטפער:

- אריינגאנגען Input 1, Input 2 ביז Input n, מיט די ווערטן x1, x2 ביז xn.
- די ווייטס w1, w2 ביז wn.
- Step 1 איז די סומע Σ פון $x_i \cdot w_i$ מיט א Bias. דאס איז באצייכנט "Transfer Function".
- Step 2 איז א גראפיק באצייכנט "Activation Function".
- צום סוף קומט דער ארויסגאנג Y, באצייכנט "Output".

פראגע 30: וואס איז א GAN? (11:03:00)

דער סלייד פרעגט "What is a Generative Adversarial Network (GAN)?". דער ענטפער:

א GAN איז א סארט דיפ-לערנען מאדעל וואס באשטייט פון צוויי נויראל-נעצן: א generator און א discriminator. ביידע ווערן טריינירט אין דער זעלבער צייט. דער generator שאפט פאלשע דאטא. דער discriminator פרובירט אונטערצושיידן צווישן עכטע און פאלשע דאטא. דאס פירט אז דער generator פראדוצירט אלץ מער רעאליסטישע דאטא.

דער דיאגראם ווייזט צוויי וועגן:

- פון א דאטאבאזע גייט א פייל מיט א געלט-שיין, באצייכנט "Real", צום Discriminator.
- פונעם Generator גייט א פייל מיט א געלט-שיין, באצייכנט "Fake", אויך צום Discriminator.

דער סוף פונעם ווידעא (11:03:30–11:05:03)

ביי 11:03:30 ווערט געווויזן א סיום-קליפ פון Simplilearn , אן קיין טעכנישן אינהאלט אויפן עקראן. פון 11:04:00 ביזן סוף בלייבט דער עקראן אינגאנצן שוואַרץ.

טייל Google Pixel 9a: 6 – א לאנג-טערמין רעצענזיע – Arran Brown

נאמען: Google Pixel 9a - Long Term, REAL WORLD Review!

קאנאל: Arran Brown

דאטום: 11 אויגוסט 2025

לענג: 16:05

לינק: https://youtu.be/bWW9YJsc_LQ



וועגן דעם ווידעא

וואס שטייט אין דער באשרייבונג

דער ווידעא, וואס איז ארויסגעקומען דעם 11טן אויגוסט 2025 אויפן קאנאל פון Arran Brown, איז א לאנג-טערמינ׳גע רעצענזיע פונעם טעלעפאן Google Pixel 9a, באזירט אויף טאג-טעגליכן באנוץ. דער ווידעא איז 16 מינוט און 5 סעקונדעס לאנג. לויט די קאפיטלען באהאנדלט ער דעם דיזיין און ווי שטארק דער טעלעפאן איז, די רעדנער, דעם עקראן, דעם פינגער-אפדרוק סענסאר, די קאמערעס, דעם אלגעמיינעם פערפארמענס און שפילן, ערנסטע באגס אינעם סאפטוועיר, די באטעריע און טשארדזשן, און צום סוף דעם פרייז און א מסקנא. שוין אינעם צווייטן קאפיטל דערציילט ער אז ער האט צעבראכן זיין Pixel 9a.

דער גרעסטער טייל פון דער באשרייבונג איז אן אנאנס (ad#) פון דער פירמע FlexiSpot, וואס איז דער ספאנסאר פונעם ווידעא. דער אנאנס רעדט פון א "9טן יובילייַ פארקויף" מיט גרויסע הנחות. ספעציפיש ווערט דערמאנט דער טיש FlexiSpot E7 Pro, וואס דער שרייבער ניצט אליין. לויט דער באשרייבונג גייט דער פארקויף פון 09:00 אינדערפרי (20 ,BST סטן אויגוסט, ביז 31סטן אויגוסט, און מען קען באקומען דעם

E7 Pro מיט ביז 50% הנחה. וויפיל פריער מען קויפט, אלץ מער שפארט מען. דערצו וועט דערמאנט א וועכנטליכע הנחה פון ביז 50% אויף א באשטעלונג יעדן דינסטאג, 10 אזייגער (BST). אויך האט איינער פון די קאפיטלען אינעם ווידעא דעם נאמען "Video Test: FlexiSpot".

דערנאך ברענגט די באשרייבונג לינקס צו די פראדוקטן וואס ווערן געוויזן אינעם ווידעא. דער שרייבער באמערקט אז דאס זענען Amazon "אפיליעיט" לינקס, דאס הייסט אז ער קען באקומען א קאמיסיע ווען מען קויפט דורך זיי. ער שרייבט אז דאס העלפט שטארק דעם קאנאל.

לינקס און רעסורסן וואס ווערן דערמאנט

- דער FlexiSpot 9טער יוביליי פארקויף (ספאנסאר): https://bit.ly/4m4uHlv
- דער טיש FlexiSpot E7 Pro וואס דער שרייבער ניצט: https://bit.ly/4oFMIYV
- דער טעלעפאן Google Pixel 9a (Amazon אפיליעיט לינק): https://geni.us/bkU2
- א UGreen Qi2 שטענדער פאר טשארדזשן אן א דראט (Amazon אפיליעיט לינק): https://geni.us/aKwmBN
- א שפיל-קאנטראלער EasySMX M15 Type C פאר טעלעפאנען (Amazon אפיליעיט לינק): https://geni.us/GhD18
- א רשימה פון דעם גאנצן עקוויפמענט וואס דער שרייבער ניצט צו שאפן זיינע YouTube ווידעאס: https://kit.co/arranpbrown/youtube-creator-gear
- א פרובע פון איין חודש בחינם ביי Epidemic Soundא סערוויס פאר מוזיק און קלאנגען פאר ווידעאס, וואס שטיצט אויך דעם קאנאל: https://www.epidemicsound.com/referral/ekgbvl/
- דער שרייבער שטעלט אויך לינקס צו זיינע חשבונות אויף Twitter און Instagram (arranpbrown).

די קאפיטלען פונעם ווידעא

- 0:00 הקדמה און הינטערגרונט
- 0:14 איך האב צעבראכן מיין Pixel 9a!
- 1:06 דיזיין און ווי שטארק דער טעלעפאן איז
- 3:08 די רעדנער (ספיקערס)
- 3:21 ווי שטארק דער עקראן איז
- 3:59 דער עקראן, דער פינגער-אפדרוק סענסאר און די ציטער-רעאקציע (Haptics)
- 5:12 די קאמערעס
- 6:26 ווידעא פרובע: FlexiSpot
- 8:01 ווידעא קוואליטעט און מעגלעכקייטן צו רעדאקטירן בילדער
- 8:33 אלגעמיינער פערפארמענס און שפילן
- 9:53 ערנסטע באגס אינעם סאפטווער
- 12:21 באטעריע און טשארדזשן
- 14:37 פרייז און מסקנא

די Pixel 9a נאך לאנגע באניצונג – ערשטער טייל (0:00)

הקדמה: ווי אזוי די טעלעפאן איז צעבראכן געווארן (0:00)

דער רעדנער, Arran Brown, זאגט אז ער האט באשטעלט די Pixel 9a אין פאראויס, דעם זעלבן טאג וואס Google האט זי אנאנסירט, בערך פיר חדשים צוריק. די לעצטע צוויי חדשים באניצט ער זיך מיט איר אלס זיין הויפט-טעלעפאן. אין דעם ווידעא וויל ער דערציילן זיין עכטע איבערלעבונג מיט דער טעלעפאן, סיי די גוטע זייטן און סיי די שוואכע.

גלייך ביַים אנהייב ווייזט ער אין וועלכן מצב זיין Pixel 9a געפינט זיך היינט. עטליכע טעג איידער ער האט געדארפט פילמען די צוגאב-בילדער (B-roll) פאר דעם ווידעא, איז ער ארויס מיטן בייסיקל און האט פארגעסן ריכטיג צו פארשליסן דעם האלטער פון טעלעפאן. די טעלעפאן איז ארויסגעפלויגן פונעם האלטער און מיט גרויס שנעלקייט געפאלן מיטן פנים אויפן וועג. דער סקרין איז אינגאנצן צעבראכן געווארן.

ער זאגט קלאר אז דאס איז הונדערט פראצענט געווען זיין אייגענע נארישקייט. א צעבראכענע טעלעפאן ווייזט נישט ריכטיג ווי די Pixel 9a זעט אויס, און זי זעט אויס שרעקליך פאר דער קאמערע. ער האט אויך געהאט א טערמין וועלכן דער ווידעא מוז ארויסקומען, דעריבער האט ער געמאכט, אין זיינע ווערטער, "א נאריש פינאנציעלע באשלוס" און געקויפט א נייע 9a. ער ווארנט דעם צוקוקער אז רוב צוגאב-בילדער אין ווידעא זענען פון דער נייער טעלעפאן.

ער האט דאס געוואלט דערציילן גלייך ביַים אנהייב, ווייל דער ערשטער טייל פונעם ווידעא רעדט טאקע וועגן אויסהאלטבארקייט און דיזיין. לויט אים איז דאס א שטיקל אירוניע.

דיזיין און אויסהאלטבארקייט (1:06)

דער רעדנער זאגט אז דער דיזיין פון דער Pixel 9a געפעלט אים.

- די הינטערשטע זייט איז געמאכט פון א מאטן פלאסטיק-קאמפאזיט. זי איז שיין, אבער זי האט נישט דאס חשובע געפיל פון די טייערע פלעגשיפ-טעלעפאנען, וואס האבן א גלאז הינטן.
- פון דער אנדערער זייט, אויב מען פלאנט צו טראגן די טעלעפאן אן א קעיס, איז דאס מסתמא א מעלה.
- אין דעם שווארצן קאליר "Obsidian" כאפט זי זייער שטארק פינגער-אפדרוקן, און מען קען זי שווער האלטן ריין.
- דעריבער זאגט ער אז אויב מען איז ווי ער, און עס שטערט אים וועז א מכשיר איז באדעקט מיט פינגער-אפדרוקן, איז בעסער צו נעמען איינע פון די דריי ליכטיגערע קאלירן: Iris, Peony אדער Porcelain.

ער באניצט זיך מיטן Spigen Nano Pop קעיס. דער דיזיין פונעם קעיס געפעלט אים. דער קעיס גיט דער טעלעפאן מאגנעטישע ווייערלעס-טשארדזשינג, און ער שיצט בכלל גענוג גוט. אבער וועז ער נעמט ארויס די טעלעפאן פונעם קעיס, בלייבן הינטן זעהבארע סימנים, בעיקר אויף דעם ארט וואו דער מאגנעט זיצט אינעם קעיס. די סימנים זענען שוין ווי עס זעט אויס פאר שטענדיג. ער האט פרובירט זיי אפצוורייניגן, אבער פון געוויסע ווינקלען זעט מען זיי נאך אלץ שוואך.

ער ערקלערט אז עס זענען דא צוויי מעגליכקייטן. איינע איז אז Google האט געוואלט שוואכן פלאסטיק פאר דער הינטערשטער זייט, און דאס וואלט געקענט פאסירן מיט יעדן אנדערן קעיס אויך. די צווייטע איז אז דאס איז א פראבלעם ספעציפיש מיט דעם Spigen קעיס. ער איז גאנץ זיכער אז אויף די אנדערע קאלירן וואלט מען

די סימנים נישט געזען. אבער ווייל ער קען נישט זיכער זאגן צי דער קעיס איז שולדיג אדער די טעלעפאן אליין, וועט ער נישט רעקאמענדירן דעם Nano Pop קעיס, און ער וועט נישט צולייגן קיין לינק, ווי ער פירט זיך געוועלין. ער בעט באניצער וואס האבן א Pixel 9a און האבן געזען ענליכע סימנים פון א קעיס, זיי זאלן אים שרייבן.

צוריק צום דיזיין: ער לויבט די פשטות. די הינטערשטע זייט איז ערפרישנד ריין און כמעט אינגאנצן פלאך, ווייל דער בלעטל וואס שטייט ארויס פאר די קאמערעס (camera bump) איז גאר קליין. דערפאר ליגט די טעלעפאן זייער סטאביל ווען מען לייגט זי מיטן פנים ארויף אויפן טיש. ער זאגט אז דאס ווייזט אז Google האט צוגעהערט צו באניצער ווי אים. זיי האבן געבעטן מען זאל מאכן די טעלעפאן גענוג דיק אז די קאמערעס זאלן אריינפאסן אן אן ארויסשטעקנדיגן בלעטל, און דערביי צולייגן מער באטעריע. דאס, זאגט ער, איז אויסגעצייכנט.

פונדעסטוועגן וואלט ער געוואלט אז די קאמערעס זאלן זיין אין דער מיט. דאס וואלט געווען מער סימעטריש און שענער פאר די אויגן. אבער ער גיט צו אז דאס איז א קליינע טענה וואס האט נישט צו טון מיט דער פראקטישער באניצונג.

די זייטן פון דער טעלעפאן זענען פון מאטן אלומיניום, און זיי פילן זיך טאקע חשוב. זיי זענען אויס און פילן זיך אויסגעצייכנט. די טעלעפאן האט אויך א IP68 שאץ פאר שיצונג קעגן שטויב און וואסער. דאס איז א פארבעסערונג לגבי דעם IP67 שאץ פון פאריג-יעריגער Pixel 8a.

די ספיקערס (3:08)

אויבן און אונטן אויף דער טעלעפאן זענען דא סטעריאָ ספיקערס. לויט דעם רעדנער קלינגען זיי גוט גענוג צו קוקן אינהאלט און הערן פאדקעסטס. אבער זיי זענען א ביסל ווייניגער קלאר און קלינגען אביסל ווייניגער ריין ווי די Pixel 9. ער זאגט אבער אז זיי קלינגען אויף זיכער בעסער ווי די ספיקערס פון דער Pixel 8a.

די אויסהאלטבארקייט פונעם סקרין (3:21)

דער פראנט פון דער טעלעפאן ווייזט נאך פשרות לגבי דער טייערערער שוועסטער, די Pixel 9. די 9a באניצט זיך מיטן עלטערן Gorilla Glass 3, אנשטאט Victus 2, און אויף זיין סקרין זענען דא באמערקבארע קראצן.

ער ערקלערט אז ער באניצט זיך בכיוון נישט מיט קיין שיץ-פלאסטיק אויפן סקרין (screen protector). אזוי קען ער טעסטן די אויסהאלטבארקייט פון די טעלעפאנען אויף א שטייגער וואס איז מער רעאליסטיש פאר זיינע צוקוקער. ער האט די טעלעפאן גארנישט געהיטן: זי איז געלעגן אין טאש אין דער זעלבער קעשענע מיט שליסלען און אנדערע זאכן, און זי איז שטארק געטרייסלט געווארן ווען זי איז געהאנגען אויף זיין עלעקטרישן בייסיקל (e-bike). דעריבער איז ער נישט איבערראשט אז זי האט קראצן און סימנים. אבער ער מיינט אז דאס גלאז דא קראצט זיך לייכטער ווי ביי רוב אנדערע טעלעפאנען וואס ער האט געטעסט. דערפאר רעקאמענדירט ער צו קויפן א שיץ-פלאסטיק פאר דעם סקרין, אויב מען וויל אז ער זאל בלייבן אינגאנצן אן קראצן.

סקרין, פינגער-סענסאר און ווירעציע (3:59)

אונטער דעם גלאז געפינט זיך א שיינער סקרין:

- גרייס: 6.3 אינטש.
- רעזאלוציע: 1080p.
- 120Hz, וואס בייט זיך אויטאמאטיש (דינאמיש) צווישן 60Hz און 120Hz.

• מאקסימום ליכטיגקייט: 2700 נייטס (nits). לויט אים זעט מען דעם סקרין אויסגעצייכנט אפילו אין די העלסטע מצבים.

עס איז דא אן אפטישער (optical) פינגער-אפדרוק סענסאר. ער איז נישט אזוי גוט ווי דער אולטראסאנישער סענסאר וואס געפינט זיך אין די איבעריגע טעלעפאנען פון דער 9 סעריע. פונדעסטוועגן מיינט דער רעדנער אז דער סענסאר פון דער 9a איז גענוג שנעל און גענוי, אזוי אז ער שטערט נישט ביי דער באניצונג.

נאך א פיזישער פרט וואס ווייזט אז די טעלעפאן איז ביליגער איז די ווי‌בראציע (haptics). זי פילט זיך בעסער ווי ביי דער 8a פון פאריג יאר, אבער זי איז נאך אלץ מער "זשומעדיג" און ווי‌ניגער פיין ווי ביי די פלעגשיפ 9 טעלעפאנען, וואו די ווי‌בראציע איז מער עדל און קאנטראלירט.

ארום דעם סקרין זענען דא גאנץ דיקע ראמען (bezels). ער ווייסט אז פאר טייל צוקוקער וועט דאס זיין א סיבה נישט צו קויפן. די ראמען מאכן די טעלעפאן א קליניגקייט העכער ווי די Pixel 9, אבער ער מיינט אז זי ליגט נאך אלץ גוט אין דער האנט. ווי ער האט געזאגט אין זיין רעצענזיע פון דער Pixel 8a בערך פאר א יאר, ווערן די ראמען גאר שנעל א טייל פון דעם סיסטעם, און מען באמערקט זיי נישט, אזוי אז זיי זענען נישט קיין פראבלעם. דערצו, נישט ווי ביי דער 8a, זענען די ראמען היי-יאר אינגאנצן סימעטריש, און דאס איז א שיינע פארבעסערונג.

אויבן אויפן סקרין, ענליך ווי ביי דער Pixel 9, האט די 9a א צו גרויסע לאך פאר דער קאמערע (hole punch cutout). דאס מאכט דעם דיזיין ווי‌ניגער ריין, למשל לגבי דער Samsung S25, וואו די לאך איז באדייטנד קלענער.

די קאמערעס (5:12)

דער רעדנער פאסט צוזאם אז אויסהאלטבארקייט און דיזיין זענען א געמיש פון גוטס און שלעכטס. אבער אין די קאמערעס, זאגט ער, ליכטיגט די 9a ארויס.

הינטן זענען דא:

• א נייע ברייטע (wide) קאמערע פון 48 מעגאפיקסעל.

• די זעלבע אולטרא-ברייטע (ultrawide) קאמערע פון 13 מעגאפיקסעל וואס איז געווען אין דער Pixel 8a.

דער סענסאר פון דער ברייטער קאמערע איז באדייטנד קלענער ווי דער אין דער Pixel 9. אבער דורך דער פשרה האט Google געקענט מאכן דעם בלעטל פאר די קאמערעס אזוי קליין, אן צו פארלירן אין קוואליטעט פון די בילדער.

• אין גוטע ליכט גיט די קאמערע אויסגעצייכנטע רעזולטאטן, מיט א גוטן באלאנס פון קאנטראסט, קאלירן און פרטים. דאס, זאגט ער, ערווארט ער שוין פון Pixel טעלעפאנען.

• אין שוואכע ליכט האלט זי די פרטים גאנץ גוט, אבער זי קען שווער "פארפרירן" זאכן וואס באווענגן זיך, און אפט ווערן זיי א ביסל פארשוווומען.

נאך א פארבעסערונג לגבי דער Pixel 8a איז דער מאקרא-מאדע (macro) פאר גאנץ נאענטע בילדער, וואס איז איצט אריינגעבויט אין דער הויפט-קאמערע. ביי רוב טעלעפאנען איז די פונקציע אין דער אולטרא-ברייטער קאמערע, אבער ער האט עס בעסער אין דער הויפט-קאמערע, ווייל דאן איז דער רעזולטאט נישט שטארק פארקרימט. ער זאגט אז ער איז ערליך זייער באאיינדרוקט פון די רעזולטאטן פונעם מאקרא-מאדע.

די אולטרא-ברייטע קאמערע גיט שיינע רעזולטאטן אין גוטע ליכט. אבער איר סענסאר איז נאך קלענער, און זי האט נישט קיין אפטישע בילד-סטאביליזאציע (OIS). דערפאר קומט אין שוואכע ליכט ארויס א זייער "רוישיג" און בלאטיג בילד. פונדעסטוועגן איז גוט צו האבן אן אולטרא-ברייטע קאמערע צו דער האנט, און זי איז גוט גענוג פאר רוב מצבים.

פארנט איז דא א 13 מעגאפיקסעל קאמערע, וואס איז אויסגעצייכנט פאר אמאל א סעלפי אויף א נסיעה, בתנאי אז ס'איז דא גענוג ליכט.

א טעסט פון ווידעא: דער FlexiSpot טייל (6:26)

כדי צו טעסטן די ווידעא-קוואליטעט, האט דער רעדנער געפילמט דעם ספאנסאר-טייל פונעם ווידעא מיט דער 9a. דער ספאנסאר איז FlexiSpot, א פירמע וואס פארקויפט טישן וואס מען קען אויפהייבן און אראפלאזן צו ארבעטן שטייענדיג. ער זאגט אז ער באניצט זיך שוין זייט 2020 מיט זייער E1 טיש, און איצט האבן זיי אים געשיקט זייער נייעם E7 Pro טיש. ער ציילט אויף דעם טישס מעלות: א ארדענונג פאר די קעבלס, א מאטאר וואס הייבט דעם טיש פון 63.5 ביז 128.5 צענטימעטער, קנעפלעך צו געדענקען פארשידענע הייכן, א פארמעג צו טראגן ביז 160 קילא, א USB C פארט אויף דעם קאנטראל, און א גראנטיע פון 10 יאר. ער דערמאנט אויך א הנחה און א לינק אין דער באשרייבונג. דאס איז פשוט א רעקלאמע, און מיר וועלן דא נישט אריינגיין אין מער פרטים.

ווידעא-קוואליטעט און בילד-רעדאגירונג (8:01)

ווי מען זעט פון דעם טייל, זאגט דער רעדנער, איז די ווידעא-קוואליטעט בכלל גאנץ גוט, בעיקר פון דער ברייטער קאמערע. אבער די אולטרא-ברייטע קאמערע האט שווערקייטן אין שוואכע ליכט, און אויך ביי פילמען פון דער האנט, ווייל זי האט נישט קיין אפטישע סטאביליזאציע. אין אלגעמיין איז ער גאנץ באאיינדרוקט פון די קאמערעס פון דער 9a.

לגבי רעדאגירן בילדער האט די 9a אלע זעלבע AI פונקציעס פאר רעדאגירונג וואס די Pixel 9 האט:

- Reimagine – לייגט צו נייע זאכן אין א בילד.
- AutoFrame – שאפט דאס בילד אין פארשידענע פראפארציעס (aspect ratios).
- Magic Eraser – מעקט אויס זאכן אדער מענטשן וואס שטערן אין א בילד.

אלגעמיינע לייסטונג און שפילן (8:33)

די לייסטונג פון דער טעלעפאן איז בכלל געווען גאנץ גוט. זי ארבעט מיט דעם זעלבן Tensor G4 טשיפ ווי די Pixel 9, אבער זי האט 8GB RAM אנשטאט די 12GB אין דער 9.

דער הויפט-אונטערשייד, זאגט ער, איז אז די טעלעפאן ארבעט מיט א קלענערע ווערסיע פון Googles AI מאדעל וואס לויפט אויף דער טעלעפאן אליין, Gemini Nano. דערפאר פעלן אייניגע פונקציעס וואס די אנדערע טעלעפאנען פון דער 9 סעריע האבן, למשל די Pixel Screenshots אפליקאציע, און Call Notes, וואס קען פאסן צוואם טעלעפאן-שמועסן. ער פערזענליך פעלט אים נישט די פונקציעס. אבער דער אונטערשייד אין RAM איז פונדעסטוועגן וויכטיג, ווייל מסתמא וועלן באזיצער פון דער Pixel 9a נישט באקומען צוקונפטיגע AI פונקציעס אויף דער טעלעפאן וואס דארפן דעם גרעסערן Gemini Nano מאדעל.

חוץ דעם, פילט זיך דאס שוויפן און סקראלן אין Pixel OS זייער גלאט, און דער סקרין מיט דער הויכער ריפרעש-ראטע (refresh rate) העלפט דערצו.

לגבי שפילן: דער Tensor G4 שטייט גאנץ ווייט הינטער טשיפס ווי Apples A18 אדער דער Snapdragon 8 Elite. פונדעסטוועגן איז ער געווען גאנץ באאיינדרוקט פון זיינע טעסטן מיט די שפילן Delta Force, Real Racing 3 און Wreckfest אויף דער 9a. ביי אלע דריי שפילן האט זי געהאלטן א פעסטע) 60fpsבילדער א סעקונדע), און מען האט טאקע גוט געקענט שפילן.

- ביי שפילן א לענגערע צייט ווערט די טעלעפאן גאנץ ווארעם, אבער זי איז קיינמאל נישט געווען צו הייס צו האלטן.
- ווייל זי ווערט ווארעם, גייט די באטעריע ביים שפילן באמערקבאר שנעלער אראפ.

לויט זיינע טעסטן מיינט ער אז די 9a איז גאנץ גוט פאר איינעם וואס שפילט אמאל לשם הנאה. אבער פאר איינעם וואס איז ערנסט מיט שפילן אויף טעלעפאן, זענען דא בעסערע טעלעפאנען.

גרויסע פראבלעמען אין דער ווייכווארג (9:53)

דא, זאגט דער רעדנער, האט ער געהאט גאר ערנסטע פראבלעמען. ער האט געזען די ערגסטע גליטשעס און באגס וואס ער האט אמאל געזען אויף א טעלעפאן.

- ווען ער האט ערשט ארויסגענומען די טעלעפאן פונעם קעסטל, האט ער געהאט א פראבלעם וואס אלע "Quick Settings" זענען געווען ליידיג. נאך א סך זוכן און פרובירן, אויך אויפן אינטערנעט, האט ער ענדליך געטראפן א תיקון דורך שפילן זיך מיט די סעטינגס פאר קאליר-קאנטראסט. ער מערקט אן אז א נייער Pixel באניצער וואס וואלט געהאט די פראבלעם ביים ערשטן מאל באניצן די טעלעפאן, וואלט אפשר נישט געווואוסט וואו אנצוהייבן צו פארריכטן דאס. דאס איז א שרעקליכע ערשטע איבערלעבונג מיט א טעלעפאן.
- אבער דאס איז נאך נישט דער סוף. די ערשטע סטאבילע אפדעיט צו Android 16 פאר דער Pixel 9a, אנהייב יוני, האט געברענגט א גאנצע ריי נייע פראבלעמען. דער ערשטער: יעדן אינדערפרי ווען ער איז אויפגעשטאנען, האט אן אויסנאם נישט געארבעט דאס אויטאמאטישע דרייען פונעם סקרין (auto rotate). עס האט נישט געהאלפן אויסצולעשן די פונקציע און ווידער אנצוצינדן. דערצו איז דער סקרין געווען זייער טונקל. אפילו מיטן ליכטיגקייט-שלייפער (slider) אויפן מאקסימום, האט מען דערגרייכט בלויז בערך 50 פראצענט פון דעם סקרינס אמתער ליכטיגקייט.

באגן, באטעריע און פרייז פון Pixel 9a (10:44)

די שווערע גליטשעס פון Android 16 און ווי זיי זענען פאריכט געווארן (10:44)

דער רעדנער פירט ווייטער זיין באריכט ווען די פראבלעמען וואס האבן זיך באוויזן נאך דעם ערשטן Android 16 אפדעיט. כדי צו פאריכטן דעם פראבלעם, זאגט ער, האט ער געמוזט ריסטארטן דעם טעלעפאן. נאר דאן האט די (auto rotateדאס אויטאמאטישע דרייען פונעם סקרין) ווידער געארבעט, און דער סקרין האט ווידער געקענט דערגרייכן זיין מאקסימום ליכטיגקייט. דאס האט אים שטארק געשטערט, ווייל ער האט דאס געמוזט טון יעדן איינציגן טאג אן אויסנאם.

אין דער זעלבער צייט איז געווען נאך א פראבלעם: ווען ער האט געדריקט דעם power-קנעפל כדי אויפצוכאפן דעם טעלעפאן, איז דורכגעגאנגען בערך 5 סעקונדעס ביז דער סקרין האט זיך אנגעצינדן. דאס זעלבע איז געשען ווען ער האט געפרואווט צושליסן דעם טעלעפאן. ער דערציילט אז ער איז דעמאלט געווען

אויף א רייזע אין Montenegro, און עטליכע מאל האט ער ממש פארפאסט א געלעגנהייט צו כאפן א בילד, ווייל דער טעלעפאן האט זיך פשוט נישט געוואלט אנצינדן.

דער נעקסטער אפדעיט פון Android 16 איז ערשט ארויסגעקומען בערך די צווייטע וואך פון July. דאס מיינט אז ער האט געמוזט לעבן מיט די גליטשעס פאר מער ווי א חודש. ער רופט זיי "deal breaker", דאס הייסט פראבלעמען וואס זענען גענוג ערנסט כדי מען זאל צוליב זיי נישט קויפן דעם טעלעפאן.

צום גליק, זאגט ער, האט דער לעצטער אפדעיט אינגאנצן פאריכט אלע גליטשעס וואס ער האט דערמאנט, און זינט דעמאלט האט ער נישט געהאט קיין נייע פראבלעמען. פונדעסטוועגן האלט ער אז עס איז זייער וויכטיג זיי אריינצונעמען אין דער רעצענזיע, ווייל זיי האבן שטארק אפגעווירקט אויף זיין איבערלעבונג במשך א היבשן טייל פון דער צייט וואס ער האט באנוצט דעם טעלעפאן. לויט אים האט דער ערשטער Android 16 ארויסגאב זיך געפילט ווי אן "alpha build", א פרייע פרי-ווערסיע וואס איז נאך נישט פארטיג, אנשטאט א סטאבילע ווערסיע.

די Pixel OS איבערלעבונג און Gemini (11:44)

דער רעדנער באמערקט אז עס איז מאדנע: זינט מען האט פאריכט די פראבלעמען, האט ער טאקע שטארק הנאה געהאט פון Pixel 9a. אן די גליטשעס איז די Pixel OS איבערלעבונג געווען אויסגעצייכנט, און כמעט די זעלבע ווי וואס ער האט געהאט אויף Pixel 9.

Gemini, זאגט ער, איז היינט א הויפט-טייל פון זיין טאג-טעגליכער ארבעט, און עס באקומט אלץ מער נוצליכע אייגנשאפטן. למשל, מען קען לייוו טיילן דאס וואס מען זעט דורך דער קאמערע און דערויף שטעלן פראגעס. ער באניצט זיך אפט מיט דעם כדי צו פארשטיין און פאריכטן פראבלעמען.

ווי אלע אנדערע Pixel טעלעפאנען וואס זענען לעצטנס ארויס, ווערט 9a באקומען 7 יאר OS- און זיכערהייט-אפדעיטס. דאס מיינט צוויי זאכן:

- מען ווערט נאך באקומען אפדעיטס ביז 2032.
- אויב עס וועלן זיך באווייזן נאך גליטשעס, ווערט Google זיי הלוואי שנעל פאריכטן אין דעם קומענדיגן אפדעיט.

באטעריע-לעבן (12:17)

דערנאך גייט דער רעדנער איבער צו באטעריע און לאדן. 9a האט די גרעסטע באטעריע פון אלע Pixel טעלעפאנען ביז היינט, 5100mAh, און מען זעט דאס אין דער באטעריע-ליסטונג. ער זאגט אז דאס באטעריע-לעבן האט זיך אויך פארבעסערט זינט דעם July אפדעיט.

אויף א נארמאלן טאג באקומט ער בדרך כלל בערך 5 ביז 6 שעה "screen on time", דאס הייסט שעהן וואס דער סקרין איז אנגעצונדן. דערביי בלייבט אים נאך 20 ביז 30 פראצענט באטעריע ביים סוף פון א טאג פון 15 שעה. די אפליקאציעס וואס פרעסן ביי אים די מערסטע באטעריע זענען Spotify און Google Maps. רוב מאל וואס ער באניצט Spotify איז דער סקרין גארנישט אנגעצונדן. דעריבער רעכנט ער אז ווער עס באניצט נישט Spotify, ווערט מעגליך באקומען א סך מער screen on time.

אין דער גרויסער בילד פון אלע טעלעפאנען וואס ער האט געטעסט, שטייט דער טעלעפאן הויך צווישן די בעסטע טעלעפאנען פון ענליכער גרייס. ער פירט אויס אז אויף טעג מיט שוואכער אדער מיטלמעסיגער באניצונג קען מען גאנץ באקווים באגרעניצן דאס לאדן צו 80 פראצענט, כדי די באטעריע זאל לענגער האלטן

במשך די יארן. אבער אויף טעג מיט שוועכער באניצונג, למשל ווען מען באניצט Google Maps אויף א לאנגער נסיעה, וועט מען לויט אים נאך אלץ דארפן א פול-געלאדענע באטעריע.

דער איינציגער טעלעפאן וואס ער האט באניצט און וואס שלאגט עס קלאר איז iPhone 16e. ער שטייט לויט אים אינגאנצן פאר זיך, הגם ער האט א היבש קלענערע באטעריע.

דער אלטער מאדעם און די נעץ-פארבינדונג (13:19)

דער רעדנער שטרייכט אונטער די וויכטיגסטע זאך צו וויסן וועגן 9a: עס באניצט דעם אלטן מאדעם וואס איז געווען אין דער Pixel 8 סעריע, און נישט דעם נייעם און פארבעסערטן מאדעם פון דער Pixel 9 סעריע. ער דערקלערט אז דער מאדעם איז דער טייל דורך וועלכן דער טעלעפאן פארבינדט זיך מיט דער מאביל-נעץ פאר אינטערנעט, רופן און טעקסטן.

ביי 9a פירט דער אלטער מאדעם צו א באמערקבאר שוואכערן באטעריע-לעבן ווען מען פארברענגט דעם טאג פארבונדן צו מאביל-דאטא, אין פארגלייך מיט ווען מען באניצט נאר wifi. אויסערדעם איז די נעץ-פארבינדונג פונעם מאדעם ווייט נישט אזוי גוט ווי דער נייער מאדעם אין Pixel 9.

ער דערציילט אז אפט האט דער טעלעפאן נישט קיין אינטערנעט אין ערטער וואו ער ווייסט אז Pixel 9 אדער iPhone 16e האבן יא פארבינדונג. וואס עס שטערט אים די מערסטע איז אז 9a ווייזט אפט פולע שטריכלעך (bars) פון 4G און 5G , און דאך קען עס נישט אראפלאדן א פשוטע וועב-בלאט. פאר אים איז דאס נישט קיין deal breaker, ווייל רוב מאל איז די פארבינדונג אין ארדענונג. אבער עס איז אוודאי איינער פון די הויפט-חסרונות פון 9a, ווייל זיין מאדעם איז אונטער דעם שטאנדארט פון דער רעשט אינדוסטריע. פאר א באניצער וואס פארלאזט זיך שטארק דערויף אז די נעץ-פארבינדונג פונעם טעלעפאן זאל זיין וואס מער פארלעסליך, קען דאס יא זיין א deal breaker.

לאדן-שנעלקייט (14:20)

די לאדן-שנעלקייט פון 9a איז גאנץ פאמעלעך: 23 וואט לאדן מיט א דראט, און זייער פאמעלעך 7.5 וואט לאדן אן א דראט (wireless). ער פארגלייכט עס דערביי צו א שלאף-חיה (sloth). דאס איז פאמעלעך אין פארגלייך מיט אסאך אנדערע טעלעפאנען אויפן מארק. פונדעסטוועגן, לויט זיין ערפארונג, מאכט עס נישט קיין גרויסן חילוק, ווייל ער דארף כמעט קיינמאל לאדן דעם טעלעפאן במשך דעם טאג, אזוי שטארק איז דאס באטעריע-לעבן.

פרייז און סוף-אורטייל (14:37)

צום לעצט רעדט ער וועגן פרייז. דער באזע-מאדעל פון 9a מיט 128GB קאסט £499 אין ענגלאנד אדער $499 אין אמעריקע. לויט אים איז דאס א גוטער פרייז, אויב דער טעלעפאן וועט ווייטער ארבעטן אזוי ווי ער ארבעט היינט, נאכדעם וואס די גליטשעס זענען פאריכט געווארן. דאס איז בפרט אזוי ווייל דער פרייז איז נישט געשטיגן אין פארגלייך מיט פאריג-יאריגן Pixel 8a. ביי פריערדיגע פארקויפן איז דער פרייז שוין אראפגעפאלן אונטער $430 אין אמעריקע, און צו £429 אין ענגלאנד בשעת דער אויפנאמע. דאס רופט ער אן אויסגעצייכנטן פרייז.

אבער מען דארף אויך באטראכטן Pixel 9, וואס קאסט בשעת דער אויפנאמע נאר £550 אין ענגלאנד. ביי אזא פרייז וואלט ער פערזענליך געזומען Pixel 9, צוליב די פיל בעסערע קאמערעס, דעם פארבעסערטן מאדעם, און בעסערע צוגרייטונג אויף דער צוקונפט מיט דעם עקסטערן RAM. דערצו איז דאס באטעריע-לעבן אויך

אויף יענעם טעלעפאן אויסגעצייכנט. אויב אבער Pixel 9 קאסט מער ווי 600$, וואלט ער גענומען 9a, ווייל עס גיט בערך 90 פראצענט פון דער זעלבער איבערלעבונג, און עס וועט באקומען די זעלבע מאס וווייכווארג-שטיצע.

דער רעדנער לייגט צו אז דאס איז פאר אים א מאדנע רעצענזיע. אין זיין היינטיגן צושטאנד איז 9a לויט אים אן אויסגעצייכנטער טעלעפאן פארן פרייז, און ער רעקאמענדירט אים שטארק. אבער נאך דעם ערשטן Android 16 אפדעיט וואלט ער אים נישט רעקאמענדירט פאר קיינעם. דעריבער לאזט ער איבער פאר דעם צוקוקער אליין צו באשליסן: צי עס לוינט זיך צו קויפן לויט דער רעצענזיע, אדער מען וויל בעסער קויפן א טעלעפאן פון אן אנדער פירמע וואס מען האלט פאר מער פארלעסליך. צום סוף דערמאנט ער כלאחר יד אז ער האט איבערגעלאזט א קויף-לינק, און פארשלאגט צו זען זיין לאנג-טערמיניגע רעצענזיע פון דעם טייערערן Pixel 9.

טייל 7: ווי אזוי צו באנוצן Veo 3.1 אומזיסט – Hyrexverse

נאמען: How to Use Veo 3.1 FREE Unlimited! Free AI Video Generator 2026

קאנאל: Hyrexverse

דאטום: 14 אקטאבער 2026

לענג: 7:03

לינק: https://youtu.be/Odmk15MPiUA



וועגן דעם ווידעא

וואס שטייט אין דער באשרייבונג

דער ווידעא, פון קאנאל Hyrexverse, ארויסגעגעבן דעם 14טן אקטאבער 2026 און 7:03 מינוט לאנג, פארשפרעכט צו ווייזן עטליכע פרייע AI-געצייג פאר מאכן ווידעאס, אן באצאלן פאר טייערע אבאנעמענטן. דער מחבר זאגט אז ער וועט אויך ווייזן א געצייג וואס לאזט באנוצן Veo 3 אומזיסט, און אז ער וועט ערקלערן דעם גאנצן פראצעס שריט ביי שריט.

ער בעט מען זאל קוקן ביזן סוף, ווייל דארט וועט ער ווייזן ווי מען באנוצט די געצייג, ווי מען שאפט AI-ווידעאס, און ווי מען וועכט אויס געווייגטלעכע פראבלעמען וואס קענען איבערראשן דאס שאפן פונעם ווידעא.

דער ווידעא האט נישט קיין צעטיילונג אין קאפיטלען אויף YouTube. אנשטאט א פלאן פונעם לימוד ברענגט די באשרייבונג א לאנגע רשימה פון זוך-ווערטער וואס ווערן אויסגערופן אלס "וואס איר וועט לערנען". צווישן זיי: ווי מען באנוצט Veo 3 אומזיסט; Veo 3 אומזיסט און אן א שיעור; Veo 3.1 אומזיסט; AI-ווידעא; אן אומבאגרעניצטער AI-ווידעא-מאכער; פון בילד צו ווידעא, אומזיסט; א "שטיק" (Trick) מיט Google Veo 3; דער בעסטער פרייער AI-ווידעא-מאכער פאר 2026; אן אפליקאציע פאר AI-ווידעאס; פון טעקסט צו ווידעא,

אומזיסט; און ווי מען מאכט AI-ווידעאס אן באצאלן. די באשרייבונג נעמט נישט אויס ביי נאמען קיין אנדער געצייג אחוץ Veo 3 און Veo 3.1. עס ווערן נישט געברענגט קיין צאלן אדער מעסטונגען, אחוץ די ווערטער "אומזיסט" און "אומבאגרענ׳יצט" (Unlimited) אינעם טיטל און אין דער רשימה.

די באשרייבונג אנטהאלט אויך צוויי ווארנונגען. די ערשטע איז א קלאל וועגן "Fair Use": דער ווידעא קען אנטהאלטן מאטעריאל מיט קאפירעכט, באנוצט פאר קאמענטארן, לערנען אדער קריטיק. די צווייטע זאגט אז דער ווידעא באזירט זיך אויף דעם מחברס פערזענלעכער איבערלעבונג און באאבאכטונגען. פרייע מעגלעכקייטן, באגרענ׳יצונגען אויף באנוץ און צי א געצייג איז בכלל צוטריטלעך קענען זיך ענדערן מיט דער צייט, און מען דארף שטענדיק נאכקוקן די איצטיגע תנאים פון יעדער AI-פלאטפארמע.

צום סוף חזרט די באשרייבונג איבער דעם ציל: ווער עס ווייל שאפן AI -ווידעאס אומזיסט און לערנען באנוצן Veo 3.1 אן א טייערן אבאנעמענט, זאל נישט פארפעלן דעם ווידעא. די באשרייבונג גייט נישט אן קיין באצאלטע קורסן אדער פראגראמען.

לינקס און רעסורסן וואס ווערן דערמאנט

- אן אימעיל-אדרעס פאר געשעפטלעכע אנפראגן: gmail.com@hyrexverseofficial
- אנדערע לינקס, נעמען פון וועבזייטלעך, אדער לינקס צו די AI-געצייג וואס ער ווייזט, ווערן נישט געברענגט אין דער באשרייבונג. עס ווערן אויך נישט אויסגערעכנט קיין קאנאלן אויף סאציאלע מעדיע.

דריי אומזיסטע AI ווידעא-געצייג, אריינגערעכנט Veo (0:00)

הקדמה: די פראבלעמען מיט AI ווידעא-געצייג (0:00)

דער רעדנער הייבט אן מיט א באמערקונג: היינט זענען פאראן אויפן מארק א סך AI-געצייג וואס שאפן ווידעאס, אבער ווען מען ווייל טאקע מאכן א ווידעא, טרעפט זיך כמעט שטענדיק א טעכנישער פעלער גלייך ווען די ווידעא ווערט געשאפן. טייל געצייג פאדערן גלייך געלט.

ער זאגט אז נאכן קוקן דעם ווידעא וועלן די אלע פראבלעמען געלייזט ווערן. ער וועט ווייזן עטלעכע AI ווידעא-געצייג מיט וועלכע מען קען שאפן ווידעאס אן א שיעור און אינגאנצן אומזיסט. אין איין טייל פונעם ווידעא וועט ער ווייזן א געצייג דורך וועלכן מען קען אומזיסט און אן א גרענץ ניצן Google'ס Veo. דערפאר בעט ער מען זאל נישט איבערשפרינגען קיין טייל.

דער ערשטער געצייג: אריינשרייבן זיך מיט Temp Mail (0:31)

דער ערשטער געצייג הייסט אין די אויטאמאטישע אונטערשריפטן "PWO AI", "Po AI" אדער "Pavo AI". דער פונקטליכער נאמען איז נישט קלאר, אבער ס'איז איין און די זעלבע וועבזייטל. מען דארף נישט האבן קיין קאמפיוטער. דער רעדנער ארבעט דעם גאנצן ווידעא אויפן טעלעפאן:

- מען עפנט דעם Chrome בראוזער אויפן טעלעפאן און מען זוכט דעם נאמען פונעם געצייג. דער רעדנער זאגט מען זאל זיך נישט זארגן וועגן ניצן עס אויפן טעלעפאן, ווייל עס ארבעט גוט.
- מען סקראלט א ביסל אראפ ביז מען זעט דאס ריכטיגע וועבזייטל, און מען דריקט דווקא אויף דעם. ער ווארנט מען זאל נישט דריקן אויף קיין אנדער וועבזייטל.

• עס עפנט זיך דער אינטערפעיס. אויב מען זעט אים נישט ווי ביי אים, דריקט מען אויף די דריי פינטעלעך אין Chrome און מען שאלט איין "desktop site". דאן קומט ארויס דער ריכטיגער אינטערפעיס.

• מען דריקט אויף "sign in", און דארט אויף "Don't have an account", ד.ה. "sign up".

עס פרעגט אן א בליצפאסט אדרעס. מען קען זיך אויך אריינשרייבן מיט א Google חשבון. אבער ער זאגט אז וועט עס וויל עס ניצן אומזיסט און אן א גרענעץ, זאל זיך אריינשרייבן מיט Temp Mail, א צייטווייליגער בליצפאסט. אזוי טוט מען דאס:

• מען דריקט אויפן פלוס-צייכן אין Chrome און מען עפנט א נייעם טאב.

• מען זוכט "Temp Mail" און מען דריקט אויפן ערשטן וועבזייטל.

• מען ווארט א רגע ביז עס ווייזט זיך א בליצפאסט אדרעס, און מען קאפירט אים.

• מען גייט צוריק צום וועבזייטל פונעם געצייג און מען לייגט אריין דעם אדרעס.

• מען שטעלט א פאסווארט ווי מען וויל, און מען דריקט צו שיקן דעם באשטעטיגונגס-קאוד.

• מען גייט צוריק צו Temp Mail און מען סקראלט אראפ. דארט ליגט א בריוו פונעם געצייג. מען עפנט אים, מען סקראלט אראפ ביזן קאוד, און מען קאפירט אים.

• מען לייגט אריין דעם קאוד אויפן וועבזייטל און מען דריקט "sign up".

ער חזר'ט נאכאמאל אז מען זאל זיך דא אריינשרייבן מיט Temp Mail.

שאפן א ווידעא פון א בילד: מאדעל, פארמאט, לענג און קוואליטעט (2:01)

דערנאך ווייזט ער ווי אזוי מען שאפט אומזיסט א ווידעא:

• מען דריקט אויפן פלוס-צייכן, דערנאך "local upload", און מען קלייבט פון דער גאלעריע דאס בילד וואס מען וויל פארוואנדלען אין א ווידעא.

• אין פענסטערל שרייבט מען וואס מען וויל זאל געשען אין ווידעא. ער האט פשוט געשריבן: "animate this image".

דא בעט ער מען זאל גוט אכטונג געבן אויף איין זאך, די ברירה פונעם מאדעל. אויב מען קלייבט דעם מאדעל וואס איז אנגעצייכנט אלס "free and unlimited", קען מען האבן פראבלעמען מיטן שאפן די ווידעא. דערפאר זאל מען קלייבן דעם מאדעל וואס הייסט אין די אונטערשריפטן "Augus Video 2.5". דער פונקטליכער נאמען איז נישט קלאר.

• דערנאך דריקט מען אויף "auto" און מען שטעלט דעם aspect ratio, ד.ה. די פארהעלטעניש פון ברייט צו הייך, אויף 16:9.

• דארט קלייבט מען אויך די לענג פון דער ווידעא. מען קען שאפן ביז א 12-סעקונדיגע ווידעא.

• מען קלייבט אויך די קוואליטעט, ביז 2K.

• ער אליין וויל א ווידעא פון 8 סעקונדעס, אלזא שטעלט ער 8 סעקונדעס, און די קוואליטעט לאזט ער אויף 720p.

ער גיט א עצה: אויב דאס בילד ווערט נישט ארויפגעלאדן, זאל מען עס אראפנעמען און נאכאמאל ארויפלאדן פון דער גאלעריע.

דער געצייג ווייזט אז די ווידעא וועט קאסטן 16 קרעדיטן. לויט דעם רעדנער קען מען מיט איין Temp Mail גרינג מאכן 20 ביז 25 ווידעאס א טאג. וועו מען ניצט עטליכע Temp Mail אדרעסן, קען מען שאפן ווידעאס אומזיסט און אן א גרענעץ.

- מען דריקט "send", און די ווידעא הייבט אן ווערן געשאפן. מען ווארט א ווייל ביז זי איז פארטיג.
- כדי אראפצולאדן, דריקט מען אויף דער אפציע וואס הייסט "SX". מען דריקט אויף דער ווידעא, דערנאך אויף די דריי פינטעלעך, און מען לאדט זי אראפ.

דער צווייטער געצייג: Dz AI (4:04)

דער צווייטער געצייג הייסט אין די אונטערשריפטן "Dz AI":

- מען עפנט Chrome, מען זוכט "DZ AI", און מען דריקט אויפן ערשטן וועבזייטל.
- עס עפנט זיך א אינטערפעיס, און מען דריקט אויפן "Create" קנעפל.
- מען קען זיך אריינשרייבן מיט א Gmail חשבון. אבער ווער עס וויל ניצן דעם געצייג אן א גרענעץ, זאל זיך אריינשרייבן מיט "10-minute mail", נאך א סארט צייטווייליגער בליצפאסט.

דא ווייזט ער ווי מען שרייבט זיך אריין מיט א Google חשבון:

- מען דריקט "Sign up for free", דערנאך "Continue with Google", און מען שרייבט זיך אריין.
- עס קומט ווידער דער אינטערפעיס. מען דריקט נאכאמאל "Create" און מען פארמאכט דאס בילד וואס ווייזט זיך דארט.
- מען דריקט "Upload" און מען קלייבט פון דער גאלעריע דאס בילד וואס מען ווייל פארוואנדלען אין א ווידעא.
- מען לייגט אריין דעם פראמפט.
- מען קען אויך קלייבן א מאדעל. ער זאגט אז דער מאדעל "Real Motion 3.5" (אזוי ווי ער הערט זיך אין די אונטערשריפטן) איז זייער גוט, און דערפאר ניצט ער אים.
- מען קען העכערן די קוואליטעט. ער קלייבט 540p.
- מען דריקט "Generate Video" און מען ווארט א ווייל ביז די ווידעא איז פארטיג.
- מען דריקט אויפן קנעפל צו אראפלאדן.

דער דריטער געצייג: Google Vids מיט Veo (5:36)

דער דריטער געצייג איז Google Vids. דורך אים, זאגט ער, קען מען ניצן Veo אומזיסט. אין די אונטערשריפטן הערט זיך עס ווי "B-roll", אבער ער מיינט Veo, ווי ער האט צוגעזאגט אין אנהייב. די טריט:

- מען עפנט Chrome, מען זוכט "Google Vids", און מען דריקט אויפן ערשטן וועבזייטל.
- מען דריקט "Sign in to Vids". ער ווארנט: כדי דאס צו ניצן אויפן טעלעפאן מוז מען איינשאלטן "desktop site", אנדערש קען ארויסקומען א פעלער.
- מען שרייבט זיך אריין מיט א Google חשבון.
- עס עפנט זיך דער אינטערפעיס, און מען דריקט "Create Video".
- עס קומט א נייער אינטערפעיס, און מען דריקט "Animate".

- מען דריקט "Add Image" און מען קלייבט פון דער גאלעריע דאס בילד וואס מען וויל "באלעבן".
- מען לייגט אריין דעם פראמפט.
- מען קען העכערן די רעזאלוציע ביז 1080p.
- מען קען אויך קלייבן דעם aspect ratio. ווייל ס'איז א לאנגע ווידעא, האבן זיי געקליבן landscape, ד.ה. א ליגנדיגע ברייטע בילד.
- מען דריקט "Send", און די ווידעא הייבט אן ווערן געשאפן. מען ווארט א ווייל.
- ווען די ווידעא איז פארטיג, האלט מען דעם פינגער לאנג אויף איר (long-press) און מען לאדט זי אראפ.

סוף (6:38)

דער רעדנער שליסט אז דאס זענען די געצייג מיט וועלכע מען קען שאפן ווידעאס אן א שיעור. ער בעט מען זאל אבאנירן דעם קאנאל, און ער זאגט צו אז אין קומענדיגן ווידעא וועט ער וויזן נאך א שטארקן געצייג.