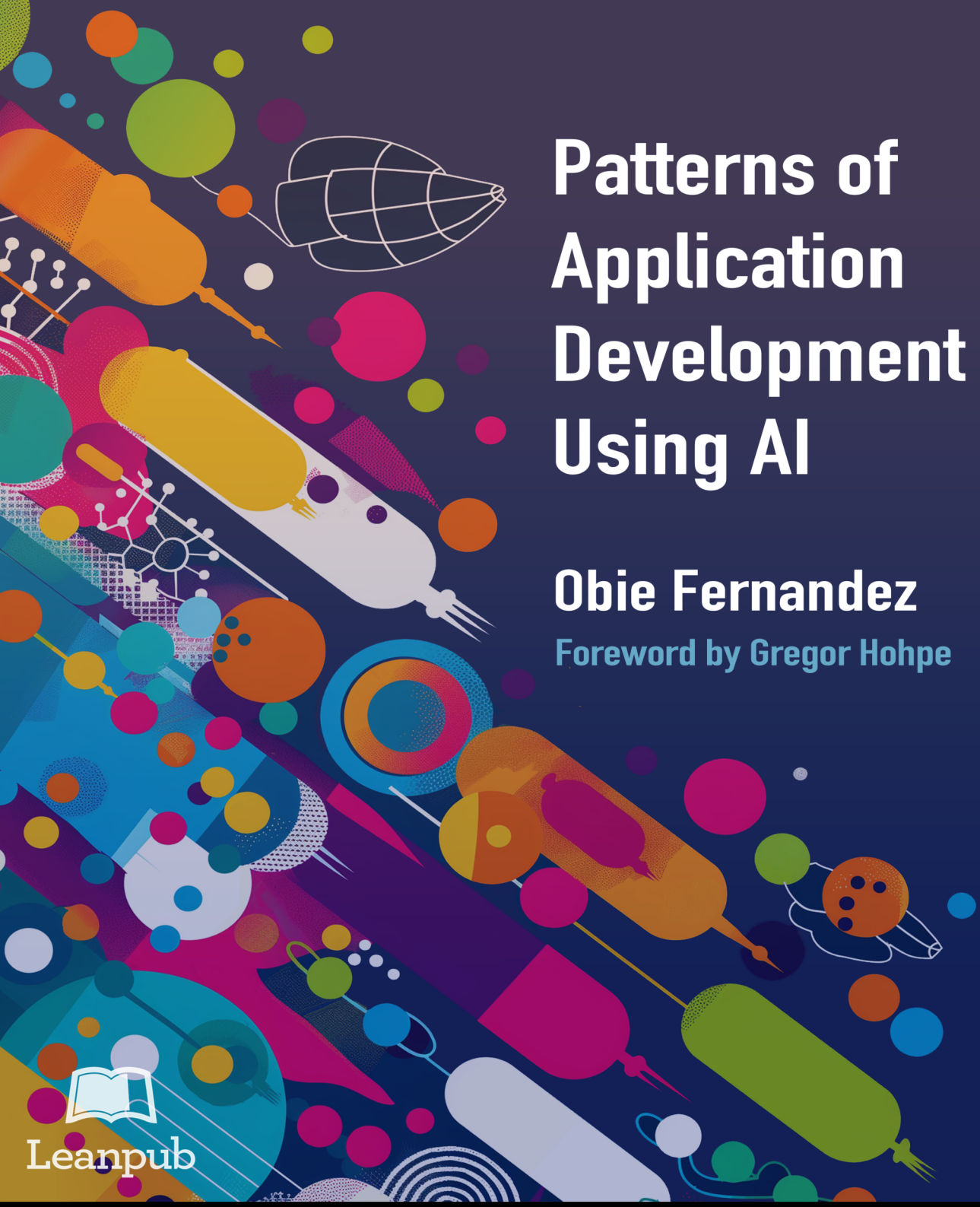


Patterns of Application Development Using AI

Obie Fernandez

Foreword by Gregor Hohpe

Leanpub

ਪੰਜਾਬੀ ਸੰਸਕਰਣ

ਏ.ਆਈ. ਦੀ ਵਰਤੋਂ ਨਾਲ ਐਪਲੀਕੇਸ਼ਨ ਵਿਕਾਸ ਦੇ ਨਮੂਨੇ (ਪੰਜਾਬੀ ਸੰਸਕਰਣ)

Obie Fernandez

ਇਹ ਕਿਤਾਬ
http://leanpub.com/patterns-of-application-development-using-ai-pa ’ਤੇ ਉਪਲਬਧ ਹੈ।

ਇਹ ਸੰਸਕਰਣ 2025-01-23 ਨੂੰ ਪ੍ਰਕਾਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਸੀ।

Leanpub

ਇਹ ਇੱਕ Leanpub ਕਿਤਾਬ ਹੈ। Leanpub ਲੇਖਕਾਂ ਅਤੇ ਪ੍ਰਕਾਸ਼ਕਾਂ ਨੂੰ Lean Publishing ਪ੍ਰਕਰਿਆ ਦੁਆਰਾ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ। Lean Publishing ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਸਾਦੇ ਸੰਦਾਂ ਅਤੇ ਵਾਰ-ਵਾਰ ਮੁੜਾਅ ਰਾਹੀਂ ਪਾਠਕਾਂ ਦੀ ਸੁਝਾਅ ਪ੍ਰਾਪਤ ਕਰਨਾ, ਉਸ ਨੂੰ ਬਦਲਣਾ ਜਸਿ ਨਾਲ ਤੁਹਾਨੂੰ ਸਹੀ ਕਿਤਾਬ ਮਲਿੇ ਅਤੇ ਜਦੋਂ ਇਹ ਮਲਿ ਜਾਵੇ, ਤਾਂ ਪਾਠਕਾਂ ਦੀ ਰੁਚੀ ਵਧਾਉਣੀ।

© 2025 Obie Fernandez

ਇਸ ਕਿਤਾਬ ਨੂੰ ਟਵੀਟ ਕਰੋ!

ਕਿਰਪਾ ਕਰਕੇ Obie Fernandez ਦੀ ਮਦਦ ਕਰੋ ਅਤੇ ਇਸ ਕਿਤਾਬ ਬਾਰੇ Twitter ਤੇ ਸ਼ਬਦ ਫੈਲਾਓ!

ਇਸ ਕਿਤਾਬ ਲਈ ਸੁਝਾਇਆ ਗਿਆ ਹੈਸ਼ਟੈਗ #poaduai ਹੈ.

ਇਸ ਲਿੰਕ 'ਤੇ ਕਲਿੱਕ ਕਰਕੇ ਜਾਣੋ ਕਿ ਹੋਰ ਲੋਕ ਇਸ ਕਿਤਾਬ ਬਾਰੇ ਕੀ ਕਹਿ ਰਹੇ ਹਨ, ਇਸ ਹੈਸ਼ਟੈਗ ਲਈ ਟਵਿੱਟਰ ਤੇ ਖੋਜ ਕਰਨ ਲਈ:

#poaduai

ਮੇਰੀ ਦਲੇਰ ਰਾਣੀ, ਮੇਰੀ ਪ੍ਰੇਰਣਾ, ਮੇਰੀ ਰੌਸ਼ਨੀ ਤੇ ਮੇਰਾ ਪਿਆਰ, Victoria ਨੂੰ

Also By Obie Fernandez

Patterns of Application Development Using AI

The Rails 8 Way

The Rails 7 Way

XML The Rails Way

Serverless

El Libro Principiante de Node

The Lean Enterprise

Contents

ਗਰੇਗਰ ਹੋਪ ਦੁਆਰਾ ਮੁੱਖਬੰਧ . . . . . **i**

ਮੁੱਖਬੰਦ . . . . . **ii**

ਕਤਿਾਬ ਬਾਰੇ iii

ਕੋਡ ਉਦਾਹਰਣਾਂ ਬਾਰੇ iii

ਮੈਂ ਕੀ ਕਵਰ ਨਹੀਂ ਕਰਦਾ iii

ਇਹ ਕਤਿਾਬ ਕਸਿ ਲਈ ਹੈ iii

ਇੱਕ ਸਾਂਝੀ ਸ਼ਬਦਾਵਲੀ ਦਾ ਨਰਿਮਾਣ iii

ਸ਼ਾਮਲ ਹੋਣਾ iii

ਧੰਨਵਾਦ iv

ਚੰਤਿਰਾਂ ਬਾਰੇ ਕੀ ਹੈ? iv

ਲੀਨ ਪਬਲਸ਼ਿਿੰਗ ਬਾਰੇ iv

ਲੇਖਕ ਬਾਰੇ v

ਜਾਣ-ਪਛਾਣ **1**

ਸਾਫਟਵੇਅਰ ਆਰਕੀਟੈਕਚਰ ਬਾਰੇ ਵਚਿਾਰ 2

ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ ਕੀ ਹੈ? 3

ਅਨੁਮਾਨ ਨੂੰ ਸਮਝਣਾ 5

ਪ੍ਰਦਰਸ਼ਨ ਬਾਰੇ ਸੋਚਣਾ 25

ਵੱਖ-ਵੱਖ ਐੱਲਐੱਲਐੱਮ ਮਾਡਲਾਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਨਾ 26

ਮਸ਼ਿਰਤਿ ਏ.ਆਈ. ਸਸਿਟਮ 27

ਭਾਗ 1: ਮੁੱਢਲੇ ਪਹੁੰਚ ਅਤੇ ਤਕਨੀਕਾਂ 35

ਰਾਹ ਨੂੰ ਤੰਗ ਕਰੋ 36

ਲੇਟੈਂਟ ਸਪੇਸ: ਅਕਲਪਨੀ ਤੌਰ 'ਤੇ ਵਸਿ਼ਾਲ 38

ਰਸਤਾ ਕਵਿੰ "ਸੀਮਤ" ਹੁੰਦਾ ਹੈ 42

ਰਾਅ ਬਨਾਮ ਇੰਸਟਰਕਟ-ਟਿਊਨਡ ਮਾਡਲ 45

ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ 51

ਪ੍ਰੌਮਪਟ ਡਸਿਟੀਲੇਸ਼ਨ 66

ਫਾਈਨ-ਟਿਊਨਿੰਗ ਬਾਰੇ ਕੀ? 73

ਰਟ੍ਰਿੀਵਲ ਔਗਮੈਂਟਡ ਜਨਰੇਸ਼ਨ (RAG) 74

ਰਟ੍ਰਿੀਵਲ ਔਗਮੈਂਟਡ ਜਨਰੇਸ਼ਨ ਕੀ ਹੈ? 74

RAG ਕਵਿੰ ਕੰਮ ਕਰਦਾ ਹੈ? 74

ਆਪਣੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ RAG ਦੀ ਵਰਤੋਂ ਕਿਉਂ ਕਰੀਏ? 74

ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ RAG ਨੂੰ ਲਾਗੂ ਕਰਨਾ 74

ਪ੍ਰਸਤਾਵ ਖੰਡੀਕਰਨ 75

ਰੈਗ ਦੀਆਂ ਅਸਲ-ਦੁਨੀਆ ਦੀਆਂ ਉਦਾਹਰਣਾਂ 75

ਬੁੱਧੀਮਾਨ ਕੁਐਰੀ ਔਪਟੀਮਾਈਜ਼ੇਸ਼ਨ (IQO) 76

ਮੁੜ-ਦਰਜਾਬੰਦੀ 76

RAG ਮੁਲਾਂਕਣ (RAGAs) 76

ਚੁਣੌਤੀਆਂ ਅਤੇ ਭਵਿੱਖ ਦਾ ਦ੍ਰਸਿ਼ਟੀਕੋਣ 78

ਕਾਮਿਆਂ ਦੀ ਭੀੜ 80

ਸੁਤੰਤਰ ਮੁੜ-ਵਰਤੋਂਯੋਗ ਕੰਪੋਨੈਂਟਸ ਵਜੋਂ ਏ.ਆਈ. ਕਾਮੇ 81

ਅਕਾਊਂਟ ਪ੍ਰਬੰਧਨ 83

ਈ-ਕਾਮਰਸ ਐਪਲੀਕੇਸ਼ਨਾਂ 84

ਸਹਿਤ ਸੰਭਾਲ ਐਪਲੀਕੇਸ਼ਨਾਂ 92

AI ਵਰਕਰ ਇੱਕ ਪ੍ਰੋਸੈਸ ਮੈਨੇਜਰ ਵਜੋਂ 95

ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਆਰਕੀਟੈਕਚਰ ਵਿੱਚ AI ਵਰਕਰਾਂ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨਾ 99

AI ਵਰਕਰਾਂ ਦੀ ਰਚਨਾਤਮਕਤਾ ਅਤੇ ਤਾਲਮੇਲ 102

ਪਰੰਪਰਾਗਤ NLP ਨੂੰ LLMs ਨਾਲ ਜੋੜਨਾ 110

ਟੂਲ ਵਰਤੋਂ 114
ਟੂਲ ਵਰਤੋਂ ਕੀ ਹੈ? 114
ਟੂਲ ਵਰਤੋਂ ਦੀ ਸੰਭਾਵਨਾ 116
ਟੂਲ ਵਰਤੋਂ ਵਰਕਫਲੋ 117
ਟੂਲ ਦੀ ਵਰਤੋਂ ਲਈ ਸਰਵੋਤਮ ਅਭਿਆਸ 131
ਟੂਲਜ਼ ਦੀ ਰਚਨਾ ਅਤੇ ਲੜੀਬੱਧਤਾ 135
ਭਵਿੱਖ ਦੀਆਂ ਦਿਸ਼ਾਵਾਂ 137

ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ 139
ReplyStream ਨੂੰ ਲਾਗੂ ਕਰਨਾ 140
“ਗੱਲਬਾਤ ਲੂਪ” 146
ਆਟੋ ਕੰਟੀਨਿਊਏਸ਼ਨ 148
ਸਿੱਟਾ 150

ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲਾ ਡਾਟਾ 152
ਵਵਿਹਾਰਕ ਕੇਸ ਸਟੱਡੀ: ਖਰਾਬ JSON ਨੂੰ ਠੀਕ ਕਰਨਾ 154
ਵਚਿਾਰ ਅਤੇ ਵਰਿੋਧੀ ਸੰਕੇਤ 159

ਪ੍ਰਸੰਗਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ 173
ਨੱਜਿੀਕਰਨ 174
ਉਤਪਾਦਕਤਾ 175
ਤੇਜ਼ ਦੁਹਰਾਓ ਅਤੇ ਪ੍ਰਯੋਗ 178
AI ਸੰਚਾਲਤਿ ਸਥਾਨੀਕਰਨ 180
ਯੂਜ਼ਰ ਟੈਸਟਿੰਗ ਅਤੇ ਫੀਡਬੈਕ ਦੀ ਮਹੱਤਤਾ 182

ਜਨਰੇਟਵਿ ਯੂਆਈ 183
ਯੂਜ਼ਰ ਇੰਟਰਫੇਸਾਂ ਲਈ ਕਾਪੀ ਤਆਿਰ ਕਰਨਾ 184
ਜਨਰੇਟਵਿ ਯੂਆਈ ਦੀ ਪਰਭਿਾਸ਼ਾ 193
ਉਦਾਹਰਨ 194

ਨਤੀਜਾ-ਕੇਂਦਰਤਿ ਡਿਜ਼ਾਇਨ ਵੱਲ ਤਬਦੀਲੀ 197
ਚੁਣੌਤੀਆਂ ਅਤੇ ਵਿਚਾਰ 198
ਭਵਿੱਖ ਦਾ ਨਜ਼ਰੀਆ ਅਤੇ ਮੌਕੇ 200

ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ 203
ਵਪਾਰਕ ਲੋੜ 204
ਮੁੱਖ ਲਾਭ 204
ਮੁੱਖ ਪੈਟਰਨ 205
ਅਪਵਾਦ ਸੰਭਾਲ ਅਤੇ ਰਿਕਵਰੀ 207
ਅਮਲੀ ਤੌਰ 'ਤੇ ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਯੋਜਨ ਨੂੰ ਲਾਗੂ ਕਰਨਾ 210
ਨਿਗਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ 224
ਸਕੇਲੇਬਲਿਟੀ ਅਤੇ ਪ੍ਰਦਰਸ਼ਨ ਵਿਚਾਰ 229
ਕਾਰਜ-ਪ੍ਰਵਾਹਾਂ ਦੀ ਟੈਸਟਿੰਗ ਅਤੇ ਪ੍ਰਮਾਣੀਕਰਨ 233

ਭਾਗ 2: ਪੈਟਰਨ 241

ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ 242
ਵਿਚਾਰਾਂ ਦੀ ਲੜੀ 243
ਮੋਡ ਸਵਿੱਚ 245
ਭੂਮਿਕਾ ਨਿਰਧਾਰਨ 246
ਪ੍ਰੌਮਪਟ ਔਬਜੈਕਟ 247
Prompt Template 248
ਸੰਰਚਤਿ ਇਨਪੁੱਟ-ਆਊਟਪੁੱਟ 249
ਪ੍ਰੌਮਪਟ ਚੇਨਿੰਗ 250
ਪ੍ਰੌਮਪਟ ਰੀਰਾਈਟਰ 251
ਰਸਿਪਾਂਸ ਫੈਸਿੰਗ 252
ਕਵੇਰੀ ਐਨਾਲਾਈਜ਼ਰ 253
ਕੁਐਰੀ ਰੀਰਾਈਟਰ 255
Ventriloquist 256

ਵੱਖਰੇ ਭਾਗ **257**
ਪ੍ਰੈਡੀਕੇਟ 258
ਏ.ਪੀ.ਆਈ. ਫਸਾਡ 259
ਨਤੀਜਾ ਵਆਿਖਆਿਕਾਰ 261
ਵਰਚੁਅਲ ਮਸ਼ੀਨ 262
ਸਪੈਸੀਫਕਿਸ਼ਨ ਅਤੇ ਟੈਸਟੰਗ 262

ਮਨੁੱਖੀ-ਸਮੂਲੀਅਤ ਪ੍ਰਣਾਲੀ (HITL) **264**
ਉੱਚ-ਪੱਧਰੀ ਪੈਟਰਨ 264
ਵਧਾਅ 265
ਫੀਡਬੈਕ ਲੂਪ 266
ਨਸ਼ਿਕਰਆਿ ਜਾਣਕਾਰੀ ਵਕਿੀਰਨ 267
ਸਹਯਿੋਗੀ ਫੈਸਲਾ ਲੈਣਾ (CDM) 269
ਲਗਾਤਾਰ ਸਖਿਣਾ 270
ਨੈਤਕਿ ਵਚਾਰ 270
ਤਕਨੀਕੀ ਤਰੱਕੀ ਅਤੇ ਭਵੱਖਿ ਦਾ ਦ੍ਰਸ਼ਟੀਕੋਣ 270

ਬੁੱਧੀਮਾਨ ਗਲਤੀ ਸੰਭਾਲ **272**
ਰਵਾਇਤੀ ਗਲਤੀ ਸੰਭਾਲ ਪਹੁੰਚਾਂ 272
ਸੰਦਰਭਕਿ ਗਲਤੀ ਨਦਿਾਨ 273
ਬੁੱਧੀਮਾਨ ਗਲਤੀ ਰਪਿਰਟੰਗ 274
ਭਵੱਖਿ-ਸੂਚਕ ਗਲਤੀ ਰੋਕਥਾਮ 275
ਸਮਾਰਟ ਗਲਤੀ ਰਕਿਵਰੀ 275
ਨੰਜੀ ਗਲਤੀ ਸੰਚਾਰ 276
ਅਨੁਕੂਲ ਗਲਤੀ ਨਪਿਟਾਰਾ ਵਰਕਫਲੋ 277

ਗੁਣਵੱਤਾ ਨਯਿੰਤਰਣ **278**
ਈਵੈਲ 279
ਸੁਰੱਖਆਿ ਉਪਾਅ 281
ਸੁਰੱਖਆਿ ਰੇਲਾਂ ਅਤੇ ਮੁਲਾਂਕਣ: ਇੱਕੋ ਸੱਕਿੇ ਦੇ ਦੋ ਪਾਸੇ 281

ਸ਼ਬਦਾਵਲੀ **283**
ਸ਼ਬਦਾਵਲੀ 283

Index **288**

ਗਰੇਗਰ ਹੋਪ ਦੁਆਰਾ ਮੁੱਖਬੰਧ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮੁੱਖਬੰਦ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਕਤਿਾਬ ਬਾਰੇ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਕੋਡ ਉਦਾਹਰਣਾਂ ਬਾਰੇ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮੈਂ ਕੀ ਕਵਰ ਨਹੀਂ ਕਰਦਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਤਿਾਬ ਕਸਿ ਲਈ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇੱਕ ਸਾਂਝੀ ਸ਼ਬਦਾਵਲੀ ਦਾ ਨਰਿਮਾਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸ਼ਾਮਲ ਹੋਣਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਧੰਨਵਾਦ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਚੱਤਿਰਾਂ ਬਾਰੇ ਕੀ ਹੈ?

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

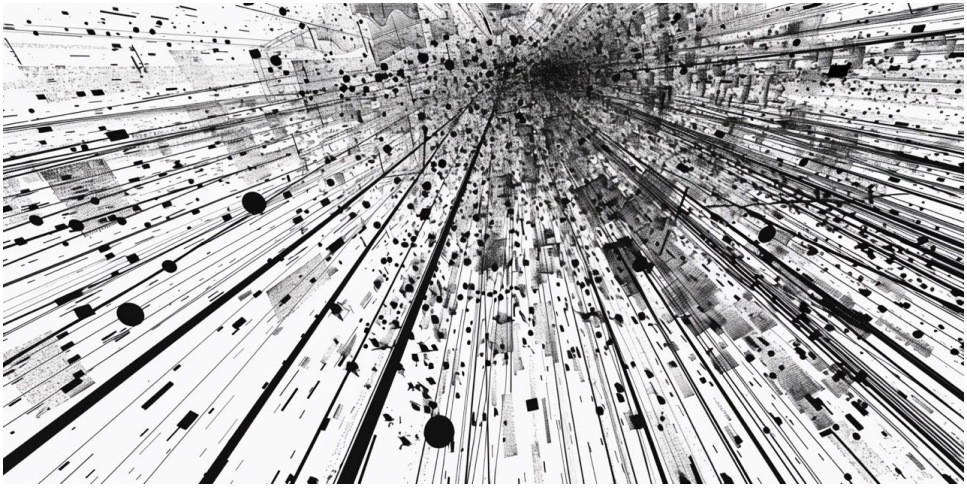
ਲੀਨ ਪਬਲਸ਼ਿੰਗ ਬਾਰੇ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਲੇਖਕ ਬਾਰੇ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਜਾਣ-ਪਛਾਣ



ਜੇ ਤੁਸੀਂ ਆਪਣੇ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪ੍ਰੋਜੈਕਟਾਂ ਵਿੱਚ AI ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ (LLMs) ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਲਈ ਉਤਸੁਕ ਹੋ, ਤਾਂ ਬੇਝਿਜਕ ਬਾਅਦ ਵਾਲੇ ਅਧਿਆਵਾਂ ਵਿੱਚ ਪੇਸ਼ ਕੀਤੇ ਪੈਟਰਨਾਂ ਅਤੇ ਕੋਡ ਉਦਾਹਰਣਾਂ ਵਿੱਚ ਡੁੱਬ ਜਾਓ। ਹਾਲਾਂਕਿ, ਇਨ੍ਹਾਂ ਪੈਟਰਨਾਂ ਦੀ ਸ਼ਕਤੀ ਅਤੇ ਸੰਭਾਵਨਾ ਨੂੰ ਪੂਰੀ ਤਰ੍ਹਾਂ ਸਮਝਣ ਲਈ, ਵਿਆਪਕ ਸੰਦਰਭ ਅਤੇ ਉਨ੍ਹਾਂ ਦੇ ਪ੍ਰਤੀਨਧਿਤਾ ਕਰਦੇ ਏਕੀਕ੍ਰਤਿ ਪਹੁੰਚ ਨੂੰ ਸਮਝਣ ਲਈ ਕੁਝ ਸਮਾਂ ਲੈਣਾ ਲਾਭਦਾਇਕ ਹੈ।

ਇਹ ਪੈਟਰਨ ਸਿਰਫ਼ ਵੱਖ-ਵੱਖ ਤਕਨੀਕਾਂ ਦਾ ਸੰਗ੍ਰਹਿ ਨਹੀਂ ਹਨ, ਬਲਕਿ ਤੁਹਾਡੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ AI ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਲਈ ਇੱਕ ਏਕੀਕ੍ਰਤਿ ਫਰੇਮਵਰਕ ਹਨ। ਮੈਂ Ruby on Rails ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹਾਂ, ਪਰ ਇਹ ਪੈਟਰਨ ਲਗਭਗ ਕਿਸੇ ਵੀ ਹੋਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਾਤਾਵਰਣ ਵਿੱਚ ਕੰਮ ਕਰਨੇ ਚਾਹੀਦੇ ਹਨ। ਇਹ ਡਾਟਾ ਪ੍ਰਬੰਧਨ ਅਤੇ ਕਾਰਗੁਜ਼ਾਰੀ ਅਨੁਕੂਲਨ ਤੋਂ ਲੈ ਕੇ ਯੂਜ਼ਰ ਅਨੁਭਵ ਅਤੇ ਸੁਰੱਖਿਆ ਤੱਕ, ਕਈ ਤਰ੍ਹਾਂ ਦੀਆਂ ਚੁਣੌਤੀਆਂ ਨੂੰ ਸੰਬੋਧਤਿ ਕਰਦੇ ਹਨ, ਜੋ AI ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਨਾਲ ਪਰੰਪਰਾਗਤ ਪ੍ਰੋਗਰਾਮਿੰਗ ਅਭਿਆਸਾਂ ਨੂੰ ਵਧਾਉਣ ਲਈ ਇੱਕ ਵਿਆਪਕ ਟੂਲਕਿੱਟ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

ਪੈਟਰਨਾਂ ਦੀ ਹਰ ਸ਼੍ਰੇਣੀ ਇੱਕ ਖਾਸ ਚੁਣੌਤੀ ਜਾਂ ਮੌਕੇ ਨੂੰ ਨਜਿੱਠਦੀ ਹੈ ਜੋ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ AI ਕੰਪੋਨੈਂਟਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨ ਵੇਲੇ ਪੈਦਾ ਹੁੰਦਾ ਹੈ। ਇਨ੍ਹਾਂ ਪੈਟਰਨਾਂ ਵਿਚਕਾਰ ਸੰਬੰਧਾਂ ਅਤੇ ਤਾਲਮੇਲ ਨੂੰ ਸਮਝ ਕੇ, ਤੁਸੀਂ AI ਨੂੰ

ਸਭ ਤੋਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਕੱਥੇ ਅਤੇ ਕਦਿੰ ਲਾਗੂ ਕਰਨਾ ਹੈ, ਬਾਰੇ ਸੂਚਤਿ ਫੈਸਲੇ ਲੈ ਸਕਦੇ ਹੋ।

ਪੈਟਰਨ ਕਦੇ ਵੀ ਨਰਿਧਾਰਤ ਹੱਲ ਨਹੀਂ ਹੁੰਦੇ ਅਤੇ ਇਨ੍ਹਾਂ ਨੂੰ ਅਜਹਿਾ ਨਹੀਂ ਮੰਨਆਿ ਜਾਣਾ ਚਾਹੀਦਾ। ਇਹ ਅਨੁਕੂਲ ਬਣਾਉਣ ਯੋਗ ਬਲਿਡੰਿਗ ਬਲਾਕ ਹਨ ਜੋ ਤੁਹਾਡੀ ਆਪਣੀ ਵਲਿੱਖਣ ਐਪਲੀਕੇਸ਼ਨ ਦੀਆਂ ਵਲਿੱਖਣ ਲੋੜਾਂ ਅਤੇ ਸੀਮਾਵਾਂ ਦੇ ਅਨੁਸਾਰ ਢਾਲੇ ਜਾਣੇ ਚਾਹੀਦੇ ਹਨ। ਇਨ੍ਹਾਂ ਪੈਟਰਨਾਂ ਦੀ ਸਫਲ ਵਰਤੋਂ (ਸਾਫਟਵੇਅਰ ਖੇਤਰ ਵੱਚਿ ਕਸਿੇ ਹੋਰ ਵਾਂਗ) ਸਮੱਸਆਿ ਖੇਤਰ, ਯੂਜ਼ਰ ਲੋੜਾਂ, ਅਤੇ ਤੁਹਾਡੇ ਪ੍ਰੋਜੈਕਟ ਦੀ ਸਮੁੱਚੀ ਤਕਨੀਕੀ ਆਰਕੀਟੈਕਚਰ ਦੀ ਡੂੰਘੀ ਸਮਝ ’ਤੇ ਨਰਿਭਰ ਕਰਦੀ ਹੈ।

ਸਾਫਟਵੇਅਰ ਆਰਕੀਟੈਕਚਰ ਬਾਰੇ ਵਚਿਾਰ

ਮੈਂ 1980 ਦੇ ਦਹਾਕੇ ਵੱਚਿ ਪ੍ਰੋਗਰਾਮਿੰਗ ਸ਼ੁਰੂ ਕੀਤੀ ਅਤੇ ਹੈਕਰ ਸੀਨ ਵੱਚਿ ਸ਼ਾਮਲ ਸੀ, ਅਤੇ ਪੇਸ਼ੇਵਰ ਸਾਫਟਵੇਅਰ ਡਵￍੈਲਪਰ ਬਣਨ ਤੋਂ ਬਾਅਦ ਵੀ ਮੈਂ ਕਦੇ ਆਪਣੀ ਹੈਕਰ ਮਾਨਸਕਿਤਾ ਨਹੀਂ ਗੁਆਈ। ਸ਼ੁਰੂ ਤੋਂ ਹੀ, ਮੇਰੇ ਕੋਲ ਹਮੇਸ਼ਾ ਇਸ ਬਾਰੇ ਇੱਕ ਸਹਿਤਮੰਦ ਸ਼ੱਕ ਸੀ ਕ ਿਆਪਣੇ ਹਾਥੀ ਦੰਦ ਦੇ ਬੁਰਜਾਂ ਵੱਚਿ ਬੈਠੇ ਸਾਫਟਵੇਅਰ ਆਰਕੀਟੈਕਟ ਅਸਲ ਵੱਚਿ ਕੀ ਯੋਗਦਾਨ ਪਾਉਂਦੇ ਸਨ।

ਕਾਰਨਾਂ ਵੱਚਿੋਂ ਇੱਕ ਜਸ਼ਿ ਕਰਕੇ ਮੈਂ ਨਜ਼ੀਿ ਤੌਰ ’ਤੇ AI ਤਕਨਾਲੋਜੀ ਦੀ ਇਸ ਸ਼ਕਤੀਸ਼ਾਲੀ ਨਵੀਂ ਲਹਰਿ ਦੁਆਰਾ ਲਆਿਂਦੇ ਗਏ ਬਦਲਾਵਾਂ ਬਾਰੇ ਬਹੁਤ ਉਤਸ਼ਾਹਤਿ ਹਾਂ, ਉਹ ਹੈ ਇਸਦਾ ਸਾਫਟਵੇਅਰ ਆਰਕੀਟੈਕਚਰ ਦੇ ਫੈਸਲਆਿਂ ’ਤੇ ਪ੍ਰਭਾਵ। ਇਹ ਇਸ ਗੱਲ ਦੀਆਂ ਪਰੰਪਰਾਗਤ ਧਾਰਨਾਵਾਂ ਨੂੰ ਚੁਣੌਤੀ ਦੇਂਦਾ ਹੈ ਕ ਿਸਾਡੇ ਸਾਫਟਵੇਅਰ ਪ੍ਰੋਜੈਕਟਾਂ ਨੂੰ ਡਜ਼ਾਿਈਨ ਅਤੇ ਲਾਗੂ ਕਰਨ ਦਾ “ਸਹੀ” ਤਰੀਕਾ ਕੀ ਹੈ। ਇਹ ਇਸ ਗੱਲ ਨੂੰ ਵੀ ਚੁਣੌਤੀ ਦੇਂਦਾ ਹੈ ਕ ਿਕੀ ਆਰਕੀਟੈਕਚਰ ਨੂੰ ਹਾਲੇ ਵੀ ਮੁੱਖ ਤੌਰ ’ਤੇ ਸਸਿਟਮ ਦੇ ਉਹਨਾਂ ਹ ੱਸਆਿਂ ਵਜ਼ੋਂ ਸੋਚਆਿ ਜਾ ਸਕਦਾ ਹੈ ਜਨ੍ਹਿਾਂ ਨੂੰ ਬਦਲਣਾ ਮੁਸ਼ਕਲ ਹੈ, ਕਉਂਕ ਿAI ਵਾਧਾ ਤੁਹਾਡੇ ਪ੍ਰੋਜੈਕਟ ਦੇ ਕਸਿ ਵੀ ਹ ੱਸ ਨੂੰ, ਕਸਿ ਵੀ ਸਮ ੇਂ ਬਦਲਣਾ ਪਹਲਿਾਂ ਨਾਲੋਂ ਵੀ ਆਸਾਨ ਬਣਾ ਰਹਿਾ ਹੈ।

ਸ਼ਾਇਦ ਅਸੀਂ ਸਾਫਟਵੇਅਰ ਇੰਜੀਨੀਅਰਿੰਗ ਦੇ “ਉੱਤਰ-ਆਧੁਨਕਿ” ਪਹੁੰਚ ਦੇ ਸਖਿਰਲੇ ਸਾਲਾਂ ਵੱਚਿ ਦਾਖਲ ਹੋ ਰਹੇ ਹਾਂ। ਇਸ ਸੰਦਰਭ ਵੱਚਿ, ਉੱਤਰ-ਆਧੁਨਕਿ ਪਰੰਪਰਾਗਤ ਪੈਰਾਡਾਈਮਾਂ ਤੋਂ ਇੱਕ ਮੌਲਕਿ ਬਦਲਾਅ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ, ਜੱਥੇ ਡਵￍੈਲਪਰ ਹਰ ਕੋਡ ਲਾਈਨ ਨੂੰ ਲਖਿਣ ਅਤੇ ਬਣਾਈ ਰੱਖਣ ਲਈ ਜ਼ੰਮਿੇਵਾਰ ਸਨ। ਇਸ ਦੀ ਬਜਾਏ, ਇਹ ਡੇਟਾ ਮੈਨੀਪੁਲੇਸ਼ਨ, ਜਟਲਿ ਐਲਗੋਰਦਿਮ, ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਲੌਜਕਿ ਦੇ ਪੂਰੇ ਹ ੱਸਆਿਂ ਨੂੰ ਤੀਜੀ-ਧਰਿ ਲਾਇਬ੍ਰੇਰੀਆਂ ਅਤੇ ਬਾਹਰੀ ਏਪੀਆਈਜ਼ ਨੂੰ ਸੌਂਪਣ ਦੇ ਵਚਿਾਰ ਨੂੰ ਅਪਣਾਉਂਦਾ ਹੈ। ਇਹ ਉੱਤਰ-ਆਧੁਨਕਿ ਬਦਲਾਅ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਮੁੱਢ ਤੋਂ ਬਣਾਉਣ ਦੀ ਰਵਾਇਤੀ ਸੋਚ ਤੋਂ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਵਚਿਲਨ ਦਰਸਾਉਂਦਾ ਹੈ, ਅਤੇ ਇਹ ਡਵￍੈਲਪਰਾਂ ਨੂੰ ਵਕਿਾਸ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਆਪਣੀ ਭੂਮਕਿਾ ਬਾਰੇ ਮੁੜ ਵਚਿਾਰ ਕਰਨ ਲਈ ਚੁਣੌਤੀ ਦੇਂਦਾ ਹੈ।

ਮੈਂ ਹਮੇਸ਼ਾ ਵਿਸ਼ਵਾਸ ਕੀਤਾ ਹੈ ਕਿ ਚੰਗੇ ਪ੍ਰੋਗਰਾਮਰ ਸਿਰਫ਼ ਉਹੀ ਕੋਡ ਲਿਖਦੇ ਹਨ ਜੋ ਬਿਲਕੁਲ ਜ਼ਰੂਰੀ ਹੈ, ਜੋ ਲੈਰੀ ਵਾਲ ਅਤੇ ਉਨ੍ਹਾਂ ਵਰਗੇ ਹੋਰ ਹੈਕਰ ਦਾਰਜਿਆਂ ਦੀਆਂ ਸਿੱਖਿਆਵਾਂ 'ਤੇ ਆਧਾਰਿਤ ਹੈ। ਲਿਖੇ ਜਾਣ ਵਾਲੇ ਕੋਡ ਦੀ ਮਾਤਰਾ ਨੂੰ ਘੱਟ ਕਰਕੇ, ਅਸੀਂ ਤੇਜ਼ੀ ਨਾਲ ਅੱਗੇ ਵੱਧ ਸਕਦੇ ਹਾਂ, ਬੱਗਾਂ ਲਈ ਸਤਹ ਖੇਤਰ ਨੂੰ ਘਟਾ ਸਕਦੇ ਹਾਂ, ਰੱਖ-ਰਖਾਅ ਨੂੰ ਸਰਲ ਬਣਾ ਸਕਦੇ ਹਾਂ, ਅਤੇ ਆਪਣੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦੀ ਸਮੁੱਚੀ ਭਰੋਸੇਯੋਗਤਾ ਨੂੰ ਸੁਧਾਰ ਸਕਦੇ ਹਾਂ। ਘੱਟ ਕੋਡ ਸਾਨੂੰ ਮੁੱਖ ਵਪਾਰਕ ਤਰਕ ਅਤੇ ਯੂਜ਼ਰ ਅਨੁਭਵ 'ਤੇ ਧਿਆਨ ਕੇਂਦਰਿਤ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ, ਜਦੋਂ ਕਿ ਹੋਰ ਕੰਮ ਨੂੰ ਹੋਰ ਸੇਵਾਵਾਂ ਨੂੰ ਸੌਂਪਦਾ ਹੈ।

ਹੁਣ ਜਦੋਂ AI-ਸੰਚਾਲਿਤ ਸਿਸਟਮ ਉਹਨਾਂ ਕਾਰਜਾਂ ਨੂੰ ਸੰਭਾਲ ਸਕਦੇ ਹਨ ਜੋ ਪਹਿਲਾਂ ਮਨੁੱਖੀ-ਲਿਖਿਤ ਕੋਡ ਦਾ ਵਿਸ਼ੇਸ਼ ਖੇਤਰ ਸਨ, ਸਾਨੂੰ ਹੋਰ ਵੀ ਵੱਧ ਉਤਪਾਦਕ ਅਤੇ ਚੁਸਤ ਹੋਣ ਦੇ ਯੋਗ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ, ਵਪਾਰਕ ਮੁੱਲ ਅਤੇ ਯੂਜ਼ਰ ਅਨੁਭਵ ਬਣਾਉਣ 'ਤੇ ਪਹਿਲਾਂ ਨਾਲੋਂ ਵੱਧ ਧਿਆਨ ਕੇਂਦਰਿਤ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ।

ਬੇਸ਼ੱਕ ਆਪਣੇ ਪ੍ਰੋਜੈਕਟ ਦੇ ਵੱਡੇ ਹੱਸਿਆਂ ਨੂੰ AI ਸਿਸਟਮਾਂ ਨੂੰ ਸੌਂਪਣ ਦੇ ਨੁਕਸਾਨ ਵੀ ਹਨ, ਜਵਿੇਂ ਕਿ ਨਿਯੰਤਰਣ ਦਾ ਸੰਭਾਵੀ ਨੁਕਸਾਨ, ਅਤੇ ਮਜ਼ਬੂਤ ਨਿਗਰਾਨੀ ਅਤੇ ਫੀਡਬੈਕ ਵਿਧੀਆਂ ਦੀ ਲੋੜ। ਇਸ ਲਈ ਇਸ ਨੂੰ ਹੁਨਰਾਂ ਅਤੇ ਗਿਆਨ ਦੇ ਇੱਕ ਨਵੇਂ ਸੈੱਟ ਦੀ ਲੋੜ ਹੈ, ਜਸਿ ਵਿੱਚ AI ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ ਇਸ ਬਾਰੇ ਘੱਟੋ-ਘੱਟ ਕੁਝ ਬੁਨਿਆਦੀ ਸਮਝ ਸ਼ਾਮਲ ਹੈ।

ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ ਕੀ ਹੈ?

ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ (LLMs) ਆਰਟੀਫੀਸ਼ੀਅਲ ਇੰਟੈਲੀਜੈਂਸ ਮਾਡਲ ਦੀ ਇੱਕ ਕਿਸਮ ਹਨ ਜਨ੍ਹਿਾਂ ਨੇ ਹਾਲ ਦੇ ਸਾਲਾਂ ਵਿੱਚ ਕਾਫ਼ੀ ਧਿਆਨ ਖਿੱਚਿਆ ਹੈ, 2020 ਵਿੱਚ OpenAI ਦੁਆਰਾ GPT-3 ਦੀ ਲਾਂਚ ਤੋਂ ਬਾਅਦ। LLMs ਨੂੰ ਮਨੁੱਖੀ ਭਾਸ਼ਾ ਨੂੰ ਸ਼ਾਨਦਾਰ ਸ਼ੁੱਧਤਾ ਅਤੇ ਪ੍ਰਵਾਹ ਨਾਲ ਪ੍ਰੋਸੈਸ ਕਰਨ, ਸਮਝਣ ਅਤੇ ਤਿਆਰ ਕਰਨ ਲਈ ਡਿਜ਼ਾਈਨ ਕੀਤਾ ਗਿਆ ਹੈ। ਇਸ ਭਾਗ ਵਿੱਚ, ਅਸੀਂ LLMs ਕਵਿੇਂ ਕੰਮ ਕਰਦੇ ਹਨ ਅਤੇ ਬੁੱਧੀਮਾਨ ਸਿਸਟਮ ਕੰਪੋਨੈਂਟਸ ਬਣਾਉਣ ਲਈ ਉਹ ਕਿਉਂ ਢੁਕਵੇਂ ਹਨ, 'ਤੇ ਇੱਕ ਸੰਖੇਪ ਨਜ਼ਰ ਮਾਰਾਂਗੇ।

ਆਪਣੇ ਮੂਲ ਵਿੱਚ, LLMs ਡੀਪ ਲਰਨਿੰਗ ਐਲਗੋਰਿਦਮ 'ਤੇ ਆਧਾਰਿਤ ਹਨ, ਖਾਸ ਤੌਰ 'ਤੇ ਤੰਤੂ ਨੈੱਟਵਰਕ। ਇਹ ਨੈੱਟਵਰਕ ਆਪਸ ਵਿੱਚ ਜੁੜੇ ਹੋਏ ਨੋਡਸ, ਜਾਂ ਨਿਊਰੋਨਸ ਦੇ ਬਣੇ ਹੁੰਦੇ ਹਨ, ਜੋ ਜਾਣਕਾਰੀ ਨੂੰ ਪ੍ਰੋਸੈਸ ਅਤੇ ਪ੍ਰਸਾਰਿਤ ਕਰਦੇ ਹਨ। LLMs ਲਈ ਚੋਣ ਦਾ ਢਾਂਚਾ ਅਕਸਰ ਟ੍ਰਾਂਸਫਾਰਮਰ ਮਾਡਲ ਹੁੰਦਾ ਹੈ, ਜੋ ਟੈਕਸਟ ਵਰਗੇ ਕ੍ਰਮਿਕ ਡੇਟਾ ਨੂੰ ਸੰਭਾਲਣ ਵਿੱਚ ਬਹੁਤ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਸਾਬਤ ਹੋਇਆ ਹੈ।

ਟ੍ਰਾਂਸਫਾਰਮਰ ਮਾਡਲ ਧਿਆਨ ਮੈਕੇਨਿਜ਼ਮ 'ਤੇ ਆਧਾਰਿਤ ਹਨ ਅਤੇ ਮੁੱਖ ਤੌਰ 'ਤੇ ਲੜੀਬੱਧ ਡਾਟਾ ਨਾਲ ਸੰਬੰਧਿਤ ਕੰਮਾਂ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ, ਜਵਿੇਂ ਕਿ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਪ੍ਰੋਸੈਸਿੰਗ। ਟ੍ਰਾਂਸਫਾਰਮਰ ਇਨਪੁੱਟ ਡਾਟਾ ਨੂੰ ਲੜੀਬੱਧ

ਤਰੀਕੇ ਦੀ ਬਜਾਏ ਇੱਕੋ ਵਾਰ ਪ੍ਰੋਸੈਸ ਕਰਦੇ ਹਨ, ਜੋ ਉਹਨਾਂ ਨੂੰ ਲੰਬੀ-ਦੂਰੀ ਦੇ ਸੰਬੰਧਾਂ ਨੂੰ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਸਮਝਣ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦਾ ਹੈ। ਉਹਨਾਂ ਵੱਚਿ ਧਿਆਨ ਮੈਕੇਨਜ਼ਿਮ ਦੀਆਂ ਪਰਤਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜੋ ਮਾਡਲ ਨੂੰ ਸੰਦਰਭ ਅਤੇ ਸੰਬੰਧਾਂ ਨੂੰ ਸਮਝਣ ਲਈ ਇਨਪੁੱਟ ਡਾਟਾ ਦੇ ਵੱਖ-ਵੱਖ ਹੰਸਿਆਂ 'ਤੇ ਧਿਆਨ ਕੇਂਦਰਤਿ ਕਰਨ ਵੱਚਿ ਮਦਦ ਕਰਦੀਆਂ ਹਨ।

ਐਲਐਲਐਮ ਲਈ ਸਖਿਲਾਈ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਮਾਡਲ ਨੂੰ ਵੱਡੀ ਮਾਤਰਾ ਵੱਚਿ ਲਖਿਤੀ ਡਾਟਾ ਦਾ ਸਾਹਮਣਾ ਕਰਨਾ ਸ਼ਾਮਲ ਹੁੰਦਾ ਹੈ, ਜਵਿੇਂ ਕਿ ਕਤਿਾਬਾਂ, ਲੇਖ, ਵੈਬਸਾਈਟਾਂ, ਅਤੇ ਕੋਡ ਰਪਿਜ਼ਟਰੀਆਂ। ਸਖਿਲਾਈ ਦੌਰਾਨ, ਮਾਡਲ ਲਖਿਤ ਵੱਚਿ ਪੈਟਰਨ, ਸੰਬੰਧ, ਅਤੇ ਢਾਂਚੇ ਨੂੰ ਪਛਾਣਨਾ ਸੱਖਿਦਾ ਹੈ। ਇਹ ਭਾਸ਼ਾ ਦੇ ਅੰਕੜਾ-ਵਧਿੀ ਗੁਣਾਂ ਨੂੰ ਕੈਪਚਰ ਕਰਦਾ ਹੈ, ਜਵਿੇਂ ਕਿ ਵਆਿਕਰਣ ਦੇ ਨਯਿਮ, ਸ਼ਬਦ ਸੰਬੰਧ, ਅਤੇ ਸੰਦਰਭਕ ਅਰਥ।

ਐਲਐਲਐਮ ਦੀ ਸਖਿਲਾਈ ਵੱਚਿ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਮੁੱਖ ਤਕਨੀਕਾਂ ਵੱਚੋਂ ਇੱਕ ਅਣਸੁਪਰਵਾਈਜ਼ਡ ਲਰਨਿੰਗ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਮਾਡਲ ਸਪੱਸ਼ਟ ਲੇਬਲਿੰਗ ਜਾਂ ਮਾਰਗਦਰਸ਼ਨ ਤੋਂ ਬਨਿਾਂ ਡਾਟਾ ਤੋਂ ਸੱਖਿਦਾ ਹੈ। ਇਹ ਸਖਿਲਾਈ ਡਾਟਾ ਵੱਚਿ ਸ਼ਬਦਾਂ ਅਤੇ ਵਾਕਾਂਸ਼ਾਂ ਦੀ ਸਹ-ਮੌਜੂਦਗੀ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਕੇ ਆਪਣੇ ਆਪ ਪੈਟਰਨ ਅਤੇ ਪ੍ਰਤੀਨਧਿਤਾ ਖੋਜਦਾ ਹੈ। ਇਹ ਐਲਐਲਐਮ ਨੂੰ ਭਾਸ਼ਾ ਅਤੇ ਇਸਦੀਆਂ ਬਾਰੀਕੀਆਂ ਦੀ ਡੂੰਘੀ ਸਮਝ ਵਕਿਸਤਿ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦਾ ਹੈ।

ਐਲਐਲਐਮ ਦਾ ਇੱਕ ਹੋਰ ਮਹੱਤਵਪੂਰਨ ਪਹਲਿੂ ਸੰਦਰਭ ਨੂੰ ਸੰਭਾਲਣ ਦੀ ਉਨ੍ਹਾਂ ਦੀ ਯੋਗਤਾ ਹੈ। ਲਖਿਤ ਦੇ ਟੁਕੜੇ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਦੇ ਸਮੇਂ, ਐਲਐਲਐਮ ਨਾ ਸਰਫ਼ ਵਅਿਕਤੀਗਤ ਸ਼ਬਦਾਂ ਨੂੰ, ਬਲਕਿ ਆਲੇ-ਦੁਆਲੇ ਦੇ ਸੰਦਰਭ ਨੂੰ ਵੀ ਧਿਆਨ ਵੱਚਿ ਰੱਖਦੇ ਹਨ। ਉਹ ਲਖਿਤ ਦੇ ਅਰਥ ਅਤੇ ਇਰਾਦੇ ਨੂੰ ਸਮਝਣ ਲਈ ਪਛਿਲੇ ਸ਼ਬਦਾਂ, ਵਾਕਾਂ, ਅਤੇ ਇੱਥੋਂ ਤੱਕ ਕਿ ਪੈਰ੍ਹਆਿਂ ਨੂੰ ਵੀ ਧਿਆਨ ਵੱਚਿ ਰੱਖਦੇ ਹਨ। ਇਹ ਸੰਦਰਭਕ ਸਮਝ ਐਲਐਲਐਮ ਨੂੰ ਸੁਸੰਗਤ ਅਤੇ ਢੁਕਵੇਂ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦੀ ਹੈ। ਕਸਿੇ ਦੱਤਿੇ ਗਏ ਐਲਐਲਐਮ ਮਾਡਲ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਦਾ ਮੁਲਾਂਕਣ ਕਰਨ ਦੇ ਮੁੱਖ ਤਰੀਕਆਿਂ ਵੱਚੋਂ ਇੱਕ ਉਹਨਾਂ ਦੇ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਲਈ ਵਚਿਾਰ ਕਰ ਸਕਣ ਵਾਲੇ ਸੰਦਰਭ ਦੇ ਆਕਾਰ 'ਤੇ ਵਚਿਾਰ ਕਰਨਾ ਹੈ।

ਸਖਿਲਾਈ ਤੋਂ ਬਾਅਦ, ਐਲਐਲਐਮ ਨੂੰ ਭਾਸ਼ਾ ਨਾਲ ਸੰਬੰਧਤਿ ਕਈ ਕੰਮਾਂ ਲਈ ਵਰਤਆਿ ਜਾ ਸਕਦਾ ਹੈ। ਉਹ ਮਨੁੱਖੀ-ਵਰਗੀ ਲਖਿਤ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ, ਸਵਾਲਾਂ ਦੇ ਜਵਾਬ ਦੇ ਸਕਦੇ ਹਨ, ਦਸਤਾਵੇਜ਼ਾਂ ਦਾ ਸਾਰ ਕੱਢ ਸਕਦੇ ਹਨ, ਭਾਸ਼ਾਵਾਂ ਦਾ ਅਨੁਵਾਦ ਕਰ ਸਕਦੇ ਹਨ, ਅਤੇ ਇੱਥੋਂ ਤੱਕ ਕਿ ਕੋਡ ਵੀ ਲਖਿ ਸਕਦੇ ਹਨ। ਐਲਐਲਐਮ ਦੀ ਬਹੁਮੁਖਤਾ ਉਹਨਾਂ ਨੂੰ ਬੁੱਧੀਮਾਨ ਸਸਿਟਮ ਕੰਪੋਨੈਂਟਸ ਬਣਾਉਣ ਲਈ ਮੁੱਲਵਾਨ ਬਣਾਉਂਦੀ ਹੈ ਜੋ ਉਪਭੋਗਤਾਵਾਂ ਨਾਲ ਗੱਲਬਾਤ ਕਰ ਸਕਦੇ ਹਨ, ਲਖਿਤ ਡਾਟਾ ਨੂੰ ਪ੍ਰੋਸੈਸ ਅਤੇ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰ ਸਕਦੇ ਹਨ, ਅਤੇ ਅਰਥਪੂਰਨ ਆਉਟਪੁੱਟ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ।

ਐਪਲੀਕੇਸ਼ਨ ਆਰਕੀਟੈਕਚਰ ਵੱਚਿ ਐਲਐਲਐਮ ਨੂੰ ਸ਼ਾਮਲ ਕਰਕੇ, ਤੁਸੀਂ ਅਜਹਿੇ ਏਆਈ ਕੰਪੋਨੈਂਟਸ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਉਪਭੋਗਤਾ ਇਨਪੁੱਟ ਨੂੰ ਸਮਝ ਅਤੇ ਪ੍ਰੋਸੈਸ ਕਰ ਸਕਦੇ ਹਨ, ਡਾਇਨਾਮਕਿ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰ ਸਕਦੇ

ਹਨ, ਅਤੇ ਬੁੱਧੀਮਾਨ ਸਫਿਾਰਸ਼ਾਂ ਜਾਂ ਕਾਰਵਾਈਆਂ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ। ਪਰ ਐੱਲਐੱਲਐੱਮ ਨਾਲ ਕੰਮ ਕਰਨ ਲਈ ਸਰੋਤਾਂ ਦੀਆਂ ਲੋੜਾਂ ਅਤੇ ਕਾਰਗੁਜ਼ਾਰੀ ਦੇ ਸਮਝੌਤਿਆਂ 'ਤੇ ਧਿਆਨਪੂਰਵਕ ਵਿਚਾਰ ਕਰਨ ਦੀ ਲੋੜ ਹੈ। ਐੱਲਐੱਲਐੱਮ ਕੰਪਿਊਟੇਸ਼ਨਲ ਤੌਰ 'ਤੇ ਗਹਨਿ ਹੁੰਦੇ ਹਨ ਅਤੇ ਚੱਲਣ ਲਈ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰੋਸੈਸਿੰਗ ਪਾਵਰ ਅਤੇ ਮੈਮੋਰੀ (ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਪੈਸੇ) ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ। ਸਾਡੇ ਵਿੱਚੋਂ ਜ਼ਿਆਦਾਤਰ ਨੂੰ ਸਾਡੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਐੱਲਐੱਲਐੱਮ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਦੇ ਲਾਗਤ ਪ੍ਰਭਾਵਾਂ ਦਾ ਮੁਲਾਂਕਣ ਕਰਨ ਅਤੇ ਉਸ ਅਨੁਸਾਰ ਕਾਰਵਾਈ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇਗੀ।

ਅਨੁਮਾਨ ਨੂੰ ਸਮਝਣਾ

ਅਨੁਮਾਨ ਉਹ ਪ੍ਰਕਰਿਆ ਹੈ ਜਸਿ ਦੁਆਰਾ ਇੱਕ ਮਾਡਲ ਨਵੇਂ, ਅਣਦੇਖੇ ਡਾਟਾ ਦੇ ਆਧਾਰ 'ਤੇ ਭਵਿੱਖਬਾਣੀਆਂ ਜਾਂ ਨਤੀਜੇ ਤਆਿਰ ਕਰਦਾ ਹੈ। ਇਹ ਉਹ ਪੜਾਅ ਹੈ ਜਿੱਥੇ ਸਖਿਲਾਈ ਪ੍ਰਾਪਤ ਮਾਡਲ ਨੂੰ ਉਪਭੋਗਤਾ ਦੇ ਇਨਪੁੱਟ ਦੇ ਜਵਾਬ ਵਿੱਚ ਫੈਸਲੇ ਲੈਣ ਜਾਂ ਟੈਕਸਟ, ਚੱਤਿਰ, ਜਾਂ ਹੋਰ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰਨ ਲਈ ਵਰਤਆਿ ਜਾਂਦਾ ਹੈ।

ਸਖਿਲਾਈ ਦੇ ਪੜਾਅ ਦੌਰਾਨ, ਇੱਕ AI ਮਾਡਲ ਆਪਣੀਆਂ ਭਵਿੱਖਬਾਣੀਆਂ ਵਿੱਚ ਗਲਤੀ ਨੂੰ ਘੱਟ ਕਰਨ ਲਈ ਆਪਣੇ ਮਾਪਦੰਡਾਂ ਨੂੰ ਵਵਿਸਥਤਿ ਕਰਕੇ ਇੱਕ ਵੱਡੇ ਡਾਟਾ-ਸਮੂਹ ਤੋਂ ਸਖਿਦਾ ਹੈ। ਇੱਕ ਵਾਰ ਸਖਿਲਾਈ ਪ੍ਰਾਪਤ ਕਰਨ ਤੋਂ ਬਾਅਦ, ਮਾਡਲ ਨਵੇਂ ਡਾਟਾ 'ਤੇ ਆਪਣੀ ਸਖਿੀ ਹੋਈ ਜਾਣਕਾਰੀ ਨੂੰ ਲਾਗੂ ਕਰ ਸਕਦਾ ਹੈ। ਅਨੁਮਾਨ ਉਹ ਤਰੀਕਾ ਹੈ ਜਸਿ ਨਾਲ ਮਾਡਲ ਆਪਣੇ ਸਖਿੇ ਹੋਏ ਪੈਟਰਨ ਅਤੇ ਗਆਿਨ ਨੂੰ ਨਤੀਜੇ ਤਆਿਰ ਕਰਨ ਲਈ ਵਰਤਦਾ ਹੈ।

LLMs ਲਈ, ਅਨੁਮਾਨ ਵਿੱਚ ਇੱਕ ਪ੍ਰੌਮਪਟ ਜਾਂ ਇਨਪੁੱਟ ਟੈਕਸਟ ਲੈਣਾ ਅਤੇ ਟੋਕਨਾਂ ਦੀ ਧਾਰਾ ਦੇ ਰੂਪ ਵਿੱਚ ਇੱਕ ਸੁਸੰਗਤ ਅਤੇ ਪ੍ਰਸੰਗਕਿ ਜਵਾਬ ਤਆਿਰ ਕਰਨਾ ਸ਼ਾਮਲ ਹੈ (ਜਸਿ ਬਾਰੇ ਅਸੀਂ ਜਲਦੀ ਹੀ ਗੱਲ ਕਰਾਂਗੇ)। ਇਹ ਕਈ ਹੋਰ ਕੰਮਾਂ ਦੇ ਨਾਲ-ਨਾਲ ਇੱਕ ਸਵਾਲ ਦਾ ਜਵਾਬ ਦੇਣਾ, ਇੱਕ ਵਾਕ ਨੂੰ ਪੂਰਾ ਕਰਨਾ, ਇੱਕ ਕਹਾਣੀ ਤਆਿਰ ਕਰਨਾ, ਜਾਂ ਟੈਕਸਟ ਦਾ ਅਨੁਵਾਦ ਕਰਨਾ ਹੋ ਸਕਦਾ ਹੈ।

ਤੁਹਾਡੇ ਅਤੇ ਮੇਰੇ ਸੋਚਣ ਦੇ ਤਰੀਕੇ ਦੇ ਉਲਟ, ਇੱਕ AI ਮਾਡਲ ਦੀ ਅਨੁਮਾਨ ਰਾਹੀਂ "ਸੋਚ" ਇੱਕ ਸਥਤਿੀ-ਰਹਤਿ ਕਾਰਜ ਵਿੱਚ ਹੁੰਦੀ ਹੈ। ਭਾਵ, ਇਸਦੀ ਸੋਚ ਇਸਦੀ ਉਤਪਾਦਨ ਪ੍ਰਕਰਿਆ ਤੱਕ ਸੀਮਤਿ ਹੈ। ਇਸਨੂੰ ਸੱਚਮੁੱਚ ਉੱਚੀ ਆਵਾਜ਼ ਵਿੱਚ ਸੋਚਣਾ ਪੈਂਦਾ ਹੈ, ਜਵਿੇਂ ਕ ਿਮੈਂ ਤੁਹਾਨੂੰ ਇੱਕ ਸਵਾਲ ਪੁੱਛਆਿ ਹੋਵੇ ਅਤੇ ਤੁਹਾਡੇ ਤੋਂ ਸਰਫ "ਚੇਤਨਾ ਦੀ ਧਾਰਾ" ਸ਼ੈਲੀ ਵਿੱਚ ਜਵਾਬ ਸਵੀਕਾਰ ਕੀਤਾ ਹੋਵੇ।

ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ ਕਈ ਆਕਾਰਾਂ ਅਤੇ ਕਿਸਮਾਂ ਵਿੱਚ ਆਉਂਦੇ ਹਨ

ਹਾਲਾਂਕਿ ਲਗਭਗ ਸਾਰੇ ਪ੍ਰਸਿੱਧ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ (LLMs) ਇੱਕੋ ਮੂਲ ਟ੍ਰਾਂਸਫਾਰਮਰ ਆਰਕੀਟੈਕਚਰ 'ਤੇ ਆਧਾਰਿਤ ਹਨ ਅਤੇ ਵੱਡੇ ਟੈਕਸਟ ਡਾਟਾ-ਸਮੂਹਾਂ 'ਤੇ ਸਿਖਲਾਈ ਪ੍ਰਾਪਤ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਉਹ ਵੱਖ-ਵੱਖ ਆਕਾਰਾਂ ਵਿੱਚ ਆਉਂਦੇ ਹਨ ਅਤੇ ਵੱਖ-ਵੱਖ ਉਦੇਸ਼ਾਂ ਲਈ ਸੂਖਮ-ਸਮਾਯੋਜਿਤ ਕੀਤੇ ਜਾਂਦੇ ਹਨ। ਇੱਕ LLM ਦਾ ਆਕਾਰ, ਜੋ ਇਸਦੇ ਤੰਤ੍ਰਿਕਾ ਨੈੱਟਵਰਕ ਵਿੱਚ ਮਾਪਦੰਡਾਂ ਦੀ ਸੰਖਿਆ ਦੁਆਰਾ ਮਾਪਿਆ ਜਾਂਦਾ ਹੈ, ਇਸਦੀਆਂ ਸਮਰੱਥਾਵਾਂ 'ਤੇ ਵੱਡਾ ਪ੍ਰਭਾਵ ਪਾਉਂਦਾ ਹੈ। ਵੱਧ ਮਾਪਦੰਡਾਂ ਵਾਲੇ ਵੱਡੇ ਮਾਡਲ, ਜਿਵੇਂ ਕਿ GPT-4, ਜਿਸ ਵਿੱਚ 1 ਤੋਂ 2 ਟ੍ਰਿਲੀਅਨ ਮਾਪਦੰਡ ਹੋਣ ਦੀ ਅਫਵਾਹ ਹੈ, ਆਮ ਤੌਰ 'ਤੇ ਛੋਟੇ ਮਾਡਲਾਂ ਨਾਲੋਂ ਵਧੇਰੇ ਗਿਆਨਵਾਨ ਅਤੇ ਸਮਰੱਥ ਹੁੰਦੇ ਹਨ। ਹਾਲਾਂਕਿ, ਵੱਡੇ ਮਾਡਲਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਬਹੁਤ ਜ਼ਿਆਦਾ ਕੰਪਿਊਟਿੰਗ ਸ਼ਕਤੀ ਦੀ ਵੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜੋ ਕਿ API ਕਾਲਾਂ ਰਾਹੀਂ ਉਹਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਵੇਲੇ ਵੱਧ ਖਰਚੇ ਵਿੱਚ ਤਬਦੀਲ ਹੁੰਦੀ ਹੈ।

LLMs ਨੂੰ ਵਧੇਰੇ ਵਿਵਹਾਰਕ ਅਤੇ ਵਿਸ਼ੇਸ਼ ਵਰਤੋਂ ਦੇ ਮਾਮਲਿਆਂ ਲਈ ਅਨੁਕੂਲ ਬਣਾਉਣ ਲਈ, ਬੇਸ ਮਾਡਲਾਂ ਨੂੰ ਅਕਸਰ ਵਧੇਰੇ ਲਕਸ਼ਿਤ ਡਾਟਾ-ਸਮੂਹਾਂ 'ਤੇ ਸੂਖਮ-ਸਮਾਯੋਜਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ, ਇੱਕ LLM ਨੂੰ ਸੰਵਾਦੀ AI ਲਈ ਵਿਸ਼ੇਸ਼ ਬਣਾਉਣ ਲਈ ਸੰਵਾਦ ਦੇ ਇੱਕ ਵੱਡੇ ਸੰਗ੍ਰਹਿ 'ਤੇ ਸਿਖਲਾਈ ਦਿੱਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਹੋਰਾਂ ਨੂੰ ਕੋਡ 'ਤੇ ਸਿਖਲਾਈ ਦਿੱਤੀ ਜਾਂਦੀ ਹੈ ਤਾਂ ਜੋ ਉਹਨਾਂ ਨੂੰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦਾ ਗਿਆਨ ਦਿੱਤਾ ਜਾ ਸਕੇ। ਇੱਥੋਂ ਤੱਕ ਕਿ ਕੁਝ ਮਾਡਲ ਹਨ ਜੋ ਉਪਭੋਗਤਾਵਾਂ ਨਾਲ ਰੋਲ-ਪਲੇਅ-ਸ਼ੈਲੀ ਦੀਆਂ ਗੱਲਬਾਤਾਂ ਲਈ ਵਿਸ਼ੇਸ਼ ਤੌਰ 'ਤੇ ਸਿਖਲਾਈ ਪ੍ਰਾਪਤ ਹਨ!

ਪੁਨਰ-ਪ੍ਰਾਪਤੀ ਬਨਾਮ ਉਤਪਾਦਕ ਮਾਡਲ

ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ (LLMs) ਦੀ ਦੁਨੀਆ ਵਿੱਚ, ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਦੇ ਦੋ ਮੁੱਖ ਤਰੀਕੇ ਹਨ: ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਆਧਾਰਿਤ ਮਾਡਲ ਅਤੇ ਉਤਪਾਦਕ ਮਾਡਲ। ਹਰ ਤਰੀਕੇ ਦੀਆਂ ਆਪਣੀਆਂ ਤਾਕਤਾਂ ਅਤੇ ਕਮਜ਼ੋਰੀਆਂ ਹਨ, ਅਤੇ ਇਹਨਾਂ ਵਿੱਚ ਅੰਤਰ ਨੂੰ ਸਮਝਣ ਨਾਲ ਤੁਹਾਨੂੰ ਆਪਣੀ ਖਾਸ ਵਰਤੋਂ ਲਈ ਸਹੀ ਮਾਡਲ ਚੁਣਨ ਵਿੱਚ ਮਦਦ ਮਿਲ ਸਕਦੀ ਹੈ।

ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਆਧਾਰਿਤ ਮਾਡਲ

ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਆਧਾਰਿਤ ਮਾਡਲ, ਜਿਨ੍ਹਾਂ ਨੂੰ ਜਾਣਕਾਰੀ ਪੁਨਰ-ਪ੍ਰਾਪਤੀ ਮਾਡਲ ਵੀ ਕਹਿਆ ਜਾਂਦਾ ਹੈ, ਪਹਿਲਾਂ ਤੋਂ ਮੌਜੂਦ ਟੈਕਸਟ ਦੇ ਵੱਡੇ ਡਾਟਾਬੇਸ ਵਿੱਚੋਂ ਖੋਜ ਕਰਕੇ ਅਤੇ ਇਨਪੁਟ ਕੀਤੇ ਸਵਾਲ ਦੇ ਆਧਾਰ 'ਤੇ ਸਭ ਤੋਂ ਢੁਕਵੇਂ ਹਿੱਸਿਆਂ ਦੀ ਚੋਣ ਕਰਕੇ ਜਵਾਬ ਤਿਆਰ ਕਰਦੇ ਹਨ। ਇਹ ਮਾਡਲ ਨਵਾਂ ਟੈਕਸਟ ਸ਼ੁਰੂ ਤੋਂ ਨਹੀਂ ਬਣਾਉਂਦੇ, ਸਗੋਂ ਇੱਕ ਸੁਸੰਗਤ ਜਵਾਬ ਬਣਾਉਣ ਲਈ ਡਾਟਾਬੇਸ ਵਿੱਚੋਂ ਹਿੱਸਿਆਂ ਨੂੰ ਜੋੜਦੇ ਹਨ।

ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਆਧਾਰਤਿ ਮਾਡਲਾਂ ਦਾ ਇੱਕ ਮੁੱਖ ਫਾਇਦਾ ਤੱਥਾਤਮਕ ਤੌਰ 'ਤੇ ਸਹੀ ਅਤੇ ਨਵੀਨਤਮ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਉਨ੍ਹਾਂ ਦੀ ਯੋਗਤਾ ਹੈ। ਕਿਉਂਕਿ ਇਹ ਸੰਭਾਲੇ ਹੋਏ ਟੈਕਸਟ ਦੇ ਡਾਟਾਬੇਸ 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹਨ, ਇਹ ਭਰੋਸੇਯੋਗ ਸਰੋਤਾਂ ਤੋਂ ਢੁਕਵੀਂ ਜਾਣਕਾਰੀ ਲੈ ਕੇ ਇਸਨੂੰ ਵਰਤੋਂਕਾਰ ਨੂੰ ਪੇਸ਼ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਉਨ੍ਹਾਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਬਹੁਤ ਢੁਕਵੇਂ ਹਨ ਜਨ੍ਹਿਾਂ ਨੂੰ ਸਟੀਕ, ਤੱਥਾਤਮਕ ਜਵਾਬਾਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜਵਿੇਂ ਕਿ ਸਵਾਲ-ਜਵਾਬ ਪ੍ਰਣਾਲੀਆਂ ਜਾਂ ਗਿਆਨ ਭੰਡਾਰ।

ਹਾਲਾਂਕਿ, ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਆਧਾਰਤਿ ਮਾਡਲਾਂ ਦੀਆਂ ਕੁਝ ਸੀਮਾਵਾਂ ਹਨ। ਇਹ ਸਰਿਫ ਉਨੇ ਹੀ ਵਧੀਆ ਹੁੰਦੇ ਹਨ ਜਿੰਨਾ ਉਹ ਡਾਟਾਬੇਸ ਹੁੰਦਾ ਹੈ ਜਸਿ ਵਿੱਚ ਉਹ ਖੋਜ ਕਰ ਰਹੇ ਹਨ, ਇਸ ਲਈ ਡਾਟਾਬੇਸ ਦੀ ਗੁਣਵੱਤਾ ਅਤੇ ਕਵਰੇਜ ਮਾਡਲ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਸਾਂਧਾ ਪ੍ਰਭਾਵਤਿ ਕਰਦੀ ਹੈ। ਇਸ ਤੋਂ ਇਲਾਵਾ, ਇਹ ਮਾਡਲ ਸੁਸੰਗਤ ਅਤੇ ਕੁਦਰਤੀ ਲੱਗਣ ਵਾਲੇ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਵਿੱਚ ਸੰਘਰਸ਼ ਕਰ ਸਕਦੇ ਹਨ, ਕਿਉਂਕਿ ਉਹ ਡਾਟਾਬੇਸ ਵਿੱਚ ਉਪਲਬਧ ਟੈਕਸਟ ਤੱਕ ਸੀਮਤਿ ਹੁੰਦੇ ਹਨ।

ਅਸੀਂ ਇਸ ਕਤਿਾਬ ਵਿੱਚ ਸ਼ੁੱਧ ਪੁਨਰ-ਪ੍ਰਾਪਤੀ ਮਾਡਲਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਕਵਰ ਨਹੀਂ ਕਰਦੇ।

ਉਤਪਾਦਕ ਮਾਡਲ

ਦੂਜੇ ਪਾਸੇ, ਉਤਪਾਦਕ ਮਾਡਲ ਸਖਿਲਾਈ ਦੌਰਾਨ ਸਾਂਖੇ ਪੈਟਰਨਾਂ ਅਤੇ ਸੰਬੰਧਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਸ਼ੁਰੂ ਤੋਂ ਨਵਾਂ ਟੈਕਸਟ ਬਣਾਉਂਦੇ ਹਨ। ਇਹ ਮਾਡਲ ਇਨਪੁਟ ਪ੍ਰੌਂਮਪਟ ਦੇ ਅਨੁਸਾਰ ਨਵੇਂ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਲਈ ਭਾਸ਼ਾ ਦੀ ਆਪਣੀ ਸਮਝ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ।

ਉਤਪਾਦਕ ਮਾਡਲਾਂ ਦੀ ਮੁੱਖ ਤਾਕਤ ਰਚਨਾਤਮਕ, ਸੁਸੰਗਤ, ਅਤੇ ਸੰਦਰਭ ਅਨੁਸਾਰ ਢੁਕਵਾਂ ਟੈਕਸਟ ਤਆਿਰ ਕਰਨ ਦੀ ਉਨ੍ਹਾਂ ਦੀ ਯੋਗਤਾ ਹੈ। ਇਹ ਖੁੱਲ੍ਹੀਆਂ ਗੱਲਬਾਤਾਂ ਵਿੱਚ ਹਸਿਾ ਲੈ ਸਕਦੇ ਹਨ, ਕਹਾਣੀਆਂ ਲਖਿ ਸਕਦੇ ਹਨ, ਅਤੇ ਇੱਥੋਂ ਤੱਕ ਕਿ ਕੋਡ ਵੀ ਲਖਿ ਸਕਦੇ ਹਨ। ਇਹ ਉਨ੍ਹਾਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਆਦਰਸ਼ ਹਨ ਜਨ੍ਹਿਾਂ ਨੂੰ ਵਧੇਰੇ ਖੁੱਲ੍ਹੀਆਂ ਅਤੇ ਗਤੀਸ਼ੀਲ ਅੰਤਰਕਰਿਆਵਾਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜਵਿੇਂ ਕਿ ਚੈਟਬੋਟ, ਸਮੱਗਰੀ ਨਰਿਮਾਣ, ਅਤੇ ਰਚਨਾਤਮਕ ਲੇਖਣ ਸਹਾਇਕ।

ਹਾਲਾਂਕਿ, ਉਤਪਾਦਕ ਮਾਡਲ ਕਈ ਵਾਰ ਅਸੰਗਤ ਜਾਂ ਤੱਥਾਤਮਕ ਤੌਰ 'ਤੇ ਗਲਤ ਜਾਣਕਾਰੀ ਪੈਦਾ ਕਰ ਸਕਦੇ ਹਨ, ਕਿਉਂਕਿ ਉਹ ਤੱਥਾਂ ਦੇ ਸੰਭਾਲੇ ਹੋਏ ਡਾਟਾਬੇਸ ਦੀ ਬਜਾਏ ਸਖਿਲਾਈ ਦੌਰਾਨ ਸਾਂਖੇ ਪੈਟਰਨਾਂ 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹਨ। ਉਹ ਪੱਖਪਾਤਾਂ ਅਤੇ ਭਰਮਾਂ ਦੇ ਵੀ ਵਧੇਰੇ ਅਧੀਨ ਹੋ ਸਕਦੇ ਹਨ, ਜੋ ਅਜਹਿਾ ਟੈਕਸਟ ਤਆਿਰ ਕਰਦੇ ਹਨ ਜੋ ਵਸ਼ਿਵਾਸਯੋਗ ਲੱਗਦਾ ਹੈ ਪਰ ਜ਼ਰੂਰੀ ਨਹੀਂ ਕਿ ਸੱਚ ਹੋਵੇ।

ਜਨਰੇਟਵਿ ਐੱਲਐੱਲਐੱਮ ਦੀਆਂ ਉਦਾਹਰਨਾਂ ਵਿੱਚ OpenAI ਦੀ GPT ਸੀਰੀਜ਼ (GPT-3, GPT-4) ਅਤੇ Anthropic ਦਾ Claude ਸ਼ਾਮਲ ਹਨ।

ਹਾਈਬ੍ਰਡਿ ਮਾਡਲ

ਕਈ ਵਪਾਰਕ ਤੌਰ ’ਤੇ ਉਪਲਬਧ ਐੱਲਐੱਲਐੱਮ ਇੱਕ ਹਾਈਬ੍ਰਡਿ ਮਾਡਲ ਵੱਿਚ ਰਟਿਰੀਵਲ ਅਤੇ ਜਨਰੇਟਵਿ ਦੋਵੇਂ ਪਹੁੰਚਾਂ ਨੂੰ ਜੋੜਦੇ ਹਨ। ਇਹ ਮਾਡਲ ਡਾਟਾਬੇਸ ਤੋਂ ਢੁਕਵੀਂ ਜਾਣਕਾਰੀ ਲੱਭਣ ਲਈ ਰਟਿਰੀਵਲ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ ਅਤੇ ਫਰਿ ਉਸ ਜਾਣਕਾਰੀ ਨੂੰ ਇੱਕ ਸੰਗਤ ਜਵਾਬ ਵੱਿਚ ਸੰਸਲੇਸ਼ਤਿ ਕਰਨ ਲਈ ਜਨਰੇਟਵਿ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ।

ਹਾਈਬ੍ਰਡਿ ਮਾਡਲ ਰਟਿਰੀਵਲ-ਆਧਾਰਤਿ ਮਾਡਲਾਂ ਦੀ ਤੱਥਾਤਮਕ ਸ਼ੁੱਧਤਾ ਨੂੰ ਜਨਰੇਟਵਿ ਮਾਡਲਾਂ ਦੀਆਂ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਜਨਰੇਸ਼ਨ ਸਮਰੱਥਾਵਾਂ ਨਾਲ ਜੋੜਨ ਦਾ ਟੀਚਾ ਰੱਖਦੇ ਹਨ। ਉਹ ਵਧੇਰੇ ਭਰੋਸੇਯੋਗ ਅਤੇ ਅੱਪ-ਟੂ-ਡੇਟ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ ਜਦੋਂ ਕ ਿਖੁੱਲ੍ਹੀਆਂ ਗੱਲਬਾਤਾਂ ਵੱਿਚ ਸ਼ਾਮਲ ਹੋਣ ਦੀ ਯੋਗਤਾ ਵੀ ਬਣਾਈ ਰੱਖਦੇ ਹਨ।

ਰਟਿਰੀਵਲ-ਆਧਾਰਤਿ ਅਤੇ ਜਨਰੇਟਵਿ ਮਾਡਲਾਂ ਵੱਿਚੋਂ ਚੋਣ ਕਰਦੇ ਸਮੇਂ, ਤੁਹਾਨੂੰ ਆਪਣੀ ਐਪਲੀਕੇਸ਼ਨ ਦੀਆਂ ਵਸ਼ੇਸ਼ ਲੋੜਾਂ ’ਤੇ ਵਚਿਾਰ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ। ਜੇਕਰ ਮੁੱਖ ਟੀਚਾ ਸਹੀ, ਤੱਥਾਤਮਕ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਨਾ ਹੈ, ਤਾਂ ਰਟਿਰੀਵਲ-ਆਧਾਰਤਿ ਮਾਡਲ ਸਭ ਤੋਂ ਵਧੀਆ ਚੋਣ ਹੋ ਸਕਦਾ ਹੈ। ਜੇਕਰ ਐਪਲੀਕੇਸ਼ਨ ਨੂੰ ਵਧੇਰੇ ਖੁੱਲ੍ਹੀਆਂ ਅਤੇ ਰਚਨਾਤਮਕ ਗੱਲਬਾਤਾਂ ਦੀ ਲੋੜ ਹੈ, ਤਾਂ ਜਨਰੇਟਵਿ ਮਾਡਲ ਵਧੇਰੇ ਢੁਕਵਾਂ ਹੋ ਸਕਦਾ ਹੈ। ਹਾਈਬ੍ਰਡਿ ਮਾਡਲ ਦੋਵਾਂ ਪਹੁੰਚਾਂ ਵਚਿਕਾਰ ਸੰਤੁਲਨ ਪੇਸ਼ ਕਰਦੇ ਹਨ ਅਤੇ ਉਨ੍ਹਾਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਚੰਗੀ ਚੋਣ ਹੋ ਸਕਦੇ ਹਨ ਜਨ੍ਹਿਾਂ ਨੂੰ ਤੱਥਾਤਮਕ ਸ਼ੁੱਧਤਾ ਅਤੇ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਜਨਰੇਸ਼ਨ ਦੋਵਾਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

ਅੰਤ ਵੱਿਚ, ਰਟਿਰੀਵਲ-ਆਧਾਰਤਿ ਅਤੇ ਜਨਰੇਟਵਿ ਮਾਡਲਾਂ ਵਚਿਕਾਰ ਚੋਣ ਵਸ਼ੇਸ਼ ਵਰਤੋਂ ਦੇ ਕੇਸ ਅਤੇ ਸ਼ੁੱਧਤਾ, ਰਚਨਾਤਮਕਤਾ, ਅਤੇ ਲਚਕਤਾ ਵਚਿਕਾਰ ਸਮਝੌਤਿਆਂ ’ਤੇ ਨਰਿਭਰ ਕਰਦੀ ਹੈ। ਹਰੇਕ ਪਹੁੰਚ ਦੀਆਂ ਤਾਕਤਾਂ ਅਤੇ ਸੀਮਾਵਾਂ ਨੂੰ ਸਮਝ ਕੇ, ਤੁਸੀਂ ਸੂਚਤਿ ਫੈਸਲੇ ਲੈ ਸਕਦੇ ਹੋ।

ਹਦਾਇਤ ਟਿਊਨਿੰਗ

ਹਦਾਇਤ ਟਿਊਨਿੰਗ ਗੱਲਬਾਤ ਸਖਿਲਾਈ ਦਾ ਇੱਕ ਉਪ-ਸਮੂਹ ਹੈ ਜੋ ਐਲਐਲਐਮ ਨੂੰ ਹਦਾਇਤਾਂ ਨੂੰ ਸਮਝਣ ਅਤੇ ਉਨ੍ਹਾਂ ਦੀ ਪਾਲਣਾ ਕਰਨ ਵੱਚਿ ਬਹਿਤਰ ਬਣਾਉਣ ਲਈ ਮਨੁੱਖ-ਲਖਿ ਪ੍ਰੌਂਪਟਸ ਅਤੇ ਜਵਾਬਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ। ਇੱਥੇ ਕੁਝ ਸਧਾਰਨ ਉਦਾਹਰਨਾਂ ਹਨ ਜੋ ਦਰਸਾਉਂਦੀਆਂ ਹਨ ਕਿ ਮਨੁੱਖ-ਲਖਿ ਪ੍ਰੌਂਪਟਸ ਅਤੇ ਜਵਾਬਾਂ ’ਤੇ ਹਦਾਇਤ ਟਿਊਨਿੰਗ ਐਲਐਲਐਮ ਨੂੰ ਹਦਾਇਤਾਂ ਨੂੰ ਸਮਝਣ ਅਤੇ ਉਨ੍ਹਾਂ ਦੀ ਪਾਲਣਾ ਕਰਨ ਵੱਚਿ ਕਵਿੇਂ ਬਹਿਤਰ ਬਣਾਉਂਦੀ ਹੈ:

ਉਦਾਹਰਨ 1: ਗਾਹਕ ਸਹਾਇਤਾ

ਹਦਾਇਤ: “ਰਫਿੰਡ ਲਈ ਪੁੱਛ ਰਹੇ ਗਾਹਕ ਨੂੰ ਜਵਾਬ ਦਓਿ।”

ਮਨੁੱਖ-ਲਖਿਆ ਪ੍ਰੌਂਪਟ: “ਮੈਂ ਆਪਣੀ ਹਾਲ ਹੀ ਦੀ ਖਰੀਦ ਲਈ ਰਫਿੰਡ ਦੀ ਬੇਨਤੀ ਕਰਨਾ ਚਾਹੁੰਦਾ/ਚਾਹੁੰਦੀ ਹਾਂ। ਉਤਪਾਦ ਮੇਰੀਆਂ ਉਮੀਦਾਂ ’ਤੇ ਖਰਾ ਨਹੀਂ ਉਤਰਿਆ।”

ਮਨੁੱਖ-ਲਖਿਆ ਜਵਾਬ: “ਮੈਨੂੰ ਸੁਣ ਕੇ ਦੁੱਖ ਹੋਇਆ ਕਿ ਉਤਪਾਦ ਤੁਹਾਡੀਆਂ ਉਮੀਦਾਂ ’ਤੇ ਖਰਾ ਨਹੀਂ ਉਤਰਿਆ। ਤੁਹਾਡੇ ਰਫਿੰਡ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਨ ਲਈ, ਕਰਿਪਾ ਕਰਕੇ ਆਪਣਾ ਆਰਡਰ ਨੰਬਰ ਅਤੇ ਵਾਪਸੀ ਦਾ ਕਾਰਨ ਪ੍ਰਦਾਨ ਕਰੋ। ਅਸੀਂ ਤੁਹਾਡੀ ਬੇਨਤੀ ਦੀ ਸਮੀਖਿਆ ਕਰਾਂਗੇ ਅਤੇ 2-3 ਕਾਰੋਬਾਰੀ ਦਨਿਾਂ ਦੇ ਅੰਦਰ ਤੁਹਾਡੇ ਨਾਲ ਸੰਪਰਕ ਕਰਾਂਗੇ।”

ਉਦਾਹਰਨ 2: ਵਦਿਅਕ ਸਹਾਇਤਾ

ਹਦਾਇਤ: “ਪ੍ਰਕਾਸ਼-ਸੰਸ਼ਲੇਸ਼ਣ ਦੀ ਧਾਰਨਾ ਨੂੰ 10 ਸਾਲ ਦੇ ਬੱਚੇ ਨੂੰ ਸਮਝਾਓ।”

ਮਨੁੱਖ-ਲਖਿਆ ਪ੍ਰੌਂਪਟ: “ਕੀ ਤੁਸੀਂ ਦੱਸ ਸਕਦੇ ਹੋ ਕਿ ਪੌਦੇ ਆਪਣਾ ਭੋਜਨ ਕਵਿੇਂ ਬਣਾਉਂਦੇ ਹਨ?”

ਮਨੁੱਖ-ਲਖਿਆ ਜਵਾਬ: “ਜਰੂਰ! ਪੌਦੇ ਪ੍ਰਕਾਸ਼-ਸੰਸਲੇਸ਼ਣ ਨਾਮਕ ਪ੍ਰਕਰਿਆ ਰਾਹੀਂ ਆਪਣਾ ਭੋਜਨ ਬਣਾਉਂਦੇ ਹਨ। ਉਹ ਸੂਰਜ ਦੀ ਰੋਸ਼ਨੀ, ਪਾਣੀ, ਅਤੇ ਕਾਰਬਨ ਡਾਈਆਕਸਾਈਡ ਦੀ ਵਰਤੋਂ ਭੋਜਨ ਅਤੇ ਆਕਸੀਜਨ ਬਣਾਉਣ ਲਈ ਕਰਦੇ ਹਨ। ਪੌਦੇ ਦੇ ਪੱਤਿਆਂ ਵੱਚਿ ਕਲੋਰੋਫਲਿ ਨਾਮ ਦਾ ਇੱਕ ਖਾਸ ਹੰਸਿਾ ਹੁੰਦਾ ਹੈ ਜੋ ਉਨ੍ਹਾਂ ਨੂੰ ਸੂਰਜ ਦੀ ਰੋਸ਼ਨੀ ਨੂੰ ਫੜਨ ਵੱਚਿ ਮਦਦ ਕਰਦਾ ਹੈ।”

ਉਦਾਹਰਣ 3: ਨੌਂਜੀ ਸਹਾਇਕ

> ਹਦਾਇਤ: "ਅਗਲੇ ਮੰਗਲਵਾਰ ਦੁਪਹਰਿ 3 ਵਜੇ ਲਈ ਇੱਕ ਮੀਟਿੰਗ ਨਿਰਧਾਰਤਿ ਕਰੋ।"
>
> ਮਨੁੱਖ-ਲਿਖਿਆ ਪ੍ਰੌਂਪਟ: "ਕੀ ਤੁਸੀ ਮੰਗਲਵਾਰ ਦੁਪਹਰਿ 3 ਵਜੇ ਲਈ ਇੱਕ ਮੀਟਿੰਗ ਨਿਰਧਾਰਤਿ ਕਰ ਸਕਦੇ ਹੋ?"
>
> ਮਨੁੱਖ-ਲਿਖਿਆ ਜਵਾਬ: "ਜਰੂਰ, ਮੈਂ ਤੁਹਾਡੇ ਲਈ ਅਗਲੇ ਮੰਗਲਵਾਰ ਦੁਪਹਰਿ 3 ਵਜੇ ਇੱਕ ਮੀਟਿੰਗ ਨਿਰਧਾਰਤਿ ਕਰ ਦਿੱਤੀ ਹੈ। ਕੀ ਤੁਹਾਨੂੰ ਕੁਝ ਹੋਰ ਚਾਹੀਦਾ ਹੈ?"

ਨਤੀਜੇ ਵਜੋਂ ਵੱਖ-ਵੱਖ ਆਕਾਰਾਂ ਅਤੇ ਵੱਖ-ਵੱਖ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਵਾਲੇ LLMs ਦਾ ਇੱਕ ਵਭਿੰਨਿ ਈਕੋਸਿਸਟਮ ਬਣਦਾ ਹੈ। 1-7 ਬਲਿੀਅਨ ਪੈਰਾਮੀਟਰ ਰੇਂਜ ਵਾਲੇ ਛੋਟੇ ਮਾਡਲ ਵਧੀਆ ਆਮ ਭਾਸ਼ਾ ਸਮਰੱਥਾਵਾਂ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ ਜਦੋਂ ਕਿ ਚਲਾਉਣ ਵਿੱਚ ਵਧੇਰੇ ਕੁਸ਼ਲ ਹੁੰਦੇ ਹਨ।

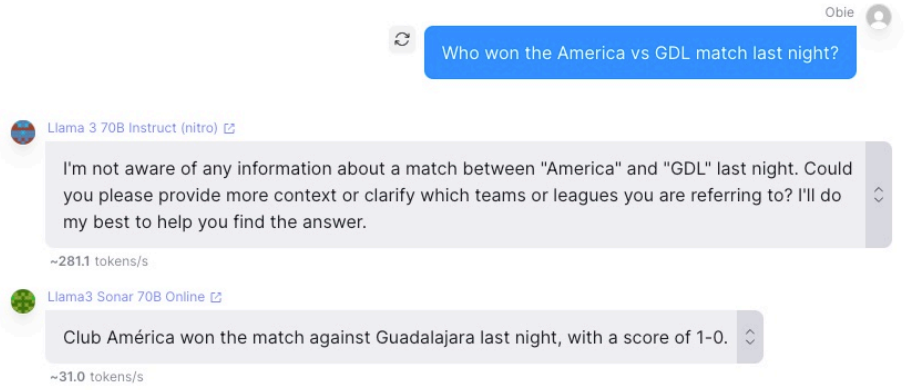
- Mistral 7B
- Llama 3 8B
- Gemma 7B

30-70 ਬਲਿੀਅਨ ਪੈਰਾਮੀਟਰ ਦੇ ਦਰਮਿਆਨੇ ਆਕਾਰ ਦੇ ਮਾਡਲ ਮਜ਼ਬੂਤ ਤਰਕ ਅਤੇ ਹਦਾਇਤਾਂ ਦੀ ਪਾਲਣਾ ਕਰਨ ਦੀਆਂ ਯੋਗਤਾਵਾਂ ਪੇਸ਼ ਕਰਦੇ ਹਨ।

- Llama 3 70B
- Qwen2 70B
- Mixtral 8x22B

ਕਸਿੇ ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ ਸ਼ਾਮਲ ਕਰਨ ਲਈ LLM ਦੀ ਚੋਣ ਕਰਦੇ ਸਮੇਂ, ਤੁਹਾਨੂੰ ਮਾਡਲ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਲਾਗਤ, ਦੇਰੀ, ਸੰਦਰਭ ਲੰਬਾਈ, ਅਤੇ ਸਮੱਗਰੀ ਫਲਿਟਰਿੰਗ ਵਰਗੇ ਵਹਿਾਰਕ ਕਾਰਕਾਂ ਨਾਲ ਸੰਤੁਲਤਿ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ। ਛੋਟੇ, ਹਦਾਇਤ-ਟਊਿਨ ਕੀਤੇ ਮਾਡਲ ਅਕਸਰ ਸਧਾਰਨ ਭਾਸ਼ਾ ਕਾਰਜਾਂ ਲਈ ਸਭ ਤੋਂ ਵਧੀਆ ਚੋਣ ਹੁੰਦੇ ਹਨ, ਜਦੋਂ ਕਿ ਜਟਲਿ ਤਰਕ ਜਾਂ ਵਸ਼ਿਲੇਸ਼ਣ ਲਈ ਸਭ ਤੋਂ ਵੱਡੇ ਮਾਡਲਾਂ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ। ਮਾਡਲ ਦਾ ਸਖਿਲਾਈ ਡੇਟਾ ਵੀ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਵਚਿਾਰ ਹੈ, ਕਉਿਂਕਿ ਇਹ ਮਾਡਲ ਦੀ ਗਆਿਨ ਕੱਟ-ਆਫ਼ ਤਾਰੀਖ ਨਰਿਧਾਰਤਿ ਕਰਦਾ ਹੈ।

Perplexity ਵਰਗੇ ਕੁਝ ਮਾਡਲ ਰੀਅਲਟਾਈਮ ਜਾਣਕਾਰੀ ਸਰੋਤਾਂ ਨਾਲ ਜੁੜੇ ਹੋਏ ਹਨ, ਇਸ ਲਈ ਉਨ੍ਹਾਂ ਦੀ ਅਸਲ ਵੱਚਿ ਕੋਈ ਕੱਟ-ਆਫ਼ ਤਾਰੀਖ ਨਹੀ ਹੈ। ਜਦੋਂ ਤੁਸੀ ਉਨ੍ਹਾਂ ਨੂੰ ਸਵਾਲ ਪੁੱਛਦੇ ਹੋ, ਤਾਂ ਉਹ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਲਈ ਸੁਤੰਤਰ ਤੌਰ 'ਤੇ ਵੈੱਬ ਖੋਜਾਂ ਕਰਨ ਅਤੇ ਮਨਮਰਜ਼ੀ ਦੇ ਵੈੱਬ ਪੰਨਆਿਂ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਦਾ ਫੈਸਲਾ ਕਰ ਸਕਦੇ ਹਨ।



ਚੱਤਰ 1. Llama3 ਆਨਲਾਈਨ ਪਹੁੰਚ ਦੇ ਨਾਲ ਅਤੇ ਬਨਿਾਂ

ਅੰਤ ਵੱਚਿ, ਕੋਈ ਵੀ ਇੱਕ-ਸਾਈਜ਼-ਫੱਟਿਸ-ਆਲ LLM ਨਹੀ ਹੈ। ਕਸਿੇ ਵੀ ਦੱਤਿੇ ਕੇਸ ਲਈ ਸਹੀ ਮਾਡਲ ਦੀ ਚੋਣ ਕਰਨ ਲਈ ਮਾਡਲ ਦੇ ਆਕਾਰ, ਆਰਕੀਟੈਕਚਰ, ਅਤੇ ਸਖਿਲਾਈ ਵੱਚਿ ਵੱਖ-ਵੱਖ ਤਬਦੀਲੀਆਂ ਨੂੰ ਸਮਝਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਵੱਖ-ਵੱਖ ਮਾਡਲਾਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਨਾ ਹੀ ਇਹ ਦੱਸਣ ਦਾ ਇੱਕੋ-ਇੱਕ ਵਹਿਾਰਕ ਤਰੀਕਾ ਹੈ ਕ ਿਕਹਿੜੇ ਮਾਡਲ ਦੱਤਿੇ ਕੰਮ ਲਈ ਸਭ ਤੋਂ ਵਧੀਆ ਪ੍ਰਦਰਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

ਟੋਕਨਾਈਜ਼ੇਸ਼ਨ: ਟੈਕਸਟ ਨੂੰ ਟੁਕੜਆਿਂ ਵੱਚਿ ਤੋੜਨਾ

ਇੱਕ ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ ਟੈਕਸਟ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਨ ਤੋਂ ਪਹਲਿਾਂ, ਉਸ ਟੈਕਸਟ ਨੂੰ ਟੋਕਨ ਕਹੇ ਜਾਣ ਵਾਲੇ ਛੋਟੇ ਹੱਸਆਿਂ ਵੱਚਿ ਵੰਡਆਿ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ। ਟੋਕਨ ਵਅਿਕਤੀਗਤ ਸ਼ਬਦ, ਸ਼ਬਦਾਂ ਦੇ ਹੱਸੇ, ਜਾਂ ਇੱਥੋ ਤੱਕ ਕ ਿਇਕੱਲੇ ਅੱਖਰ ਵੀ ਹੋ ਸਕਦੇ ਹਨ। ਟੈਕਸਟ ਨੂੰ ਟੋਕਨਾਂ ਵੱਚਿ ਵੰਡਣ ਦੀ ਪ੍ਰਕਰਿਆ ਨੂੰ ਟੋਕਨਾਈਜ਼ੇਸ਼ਨ ਕਹਿਾ ਜਾਂਦਾ ਹੈ, ਅਤੇ ਇਹ ਭਾਸ਼ਾ ਮਾਡਲ ਲਈ ਡੇਟਾ ਤਆਿਰ ਕਰਨ ਵੱਚਿ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਕਦਮ ਹੈ।

```
The process of splitting text into tokens is known as tokenization, and
 it's a crucial step in preparing data for a language model.
```

ਚਿੱਤਰ 2. ਇਸ ਵਾਕ ਵਿੱਚ 27 ਟੋਕਨ ਹਨ

ਵੱਖ-ਵੱਖ LLMs ਵੱਖ-ਵੱਖ ਟੋਕਨਾਈਜ਼ੇਸ਼ਨ ਰਣਨੀਤੀਆਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ, ਜੋ ਮਾਡਲ ਦੇ ਪ੍ਰਦਰਸ਼ਨ ਅਤੇ ਸਮਰੱਥਾਵਾਂ 'ਤੇ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦੀਆਂ ਹਨ। LLMs ਦੁਆਰਾ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਕੁਝ ਆਮ ਟੋਕਨਾਈਜ਼ਰ ਹਨ:

- **GPT (ਬਾਈਟ ਪੇਅਰ ਇਨਕੋਡਿੰਗ):** GPT ਟੋਕਨਾਈਜ਼ਰ ਬਾਈਟ ਪੇਅਰ ਇਨਕੋਡਿੰਗ (BPE) ਨਾਮਕ ਤਕਨੀਕ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ ਜੋ ਟੈਕਸਟ ਨੂੰ ਸਬ-ਵਰਡ ਯੂਨਟਿਾਂ ਵਿੱਚ ਤੋੜਦੀ ਹੈ। BPE ਟੈਕਸਟ ਕੋਰਪਸ ਵਿੱਚ ਸਭ ਤੋਂ ਵੱਧ ਵਾਰ ਆਉਣ ਵਾਲੇ ਬਾਈਟ ਜੋੜਿਆਂ ਨੂੰ ਲਗਾਤਾਰ ਮਲਿਾਉਂਦੀ ਹੈ, ਜੋ ਸਬ-ਵਰਡ ਟੋਕਨਾਂ ਦੀ ਸ਼ਬਦਾਵਲੀ ਬਣਾਉਂਦੀ ਹੈ। ਇਹ ਟੋਕਨਾਈਜ਼ਰ ਨੂੰ ਦੁਰਲੱਭ ਅਤੇ ਨਵੇਂ ਸ਼ਬਦਾਂ ਨੂੰ ਵਧੇਰੇ ਆਮ ਸਬ-ਵਰਡ ਟੁਕੜਿਆਂ ਵਿੱਚ ਤੋੜ ਕੇ ਸੰਭਾਲਣ ਦੀ ਆਗਿਆ ਦੇਂਦੀ ਹੈ। GPT ਟੋਕਨਾਈਜ਼ਰ GPT-3 ਅਤੇ GPT-4 ਵਰਗੇ ਮਾਡਲਾਂ ਦੁਆਰਾ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

- **Llama (SentencePiece):** Llama ਟੋਕਨਾਈਜ਼ਰ SentencePiece ਲਾਇਬ੍ਰੇਰੀ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ, ਜੋ ਇੱਕ ਬਨਿਾਂ ਨਗਿਰਾਨੀ ਵਾਲਾ ਟੈਕਸਟ ਟੋਕਨਾਈਜ਼ਰ ਅਤੇ ਡੀਟੋਕਨਾਈਜ਼ਰ ਹੈ। SentencePiece ਇਨਪੁੱਟ ਟੈਕਸਟ ਨੂੰ ਯੂਨੀਕੋਡ ਅੱਖਰਾਂ ਦੀ ਲੜੀ ਵਜੋਂ ਮੰਨਦਾ ਹੈ ਅਤੇ ਟ੍ਰੇਨਿੰਗ ਕੋਰਪਸ ਦੇ ਆਧਾਰ 'ਤੇ ਸਬ-ਵਰਡ ਸ਼ਬਦਾਵਲੀ ਸਖਿਦਾ ਹੈ। ਇਹ ਕਸਿੇ ਵੀ ਭਾਸ਼ਾ ਨੂੰ ਸੰਭਾਲ ਸਕਦਾ ਹੈ ਜੋ ਯੂਨੀਕੋਡ ਵਿੱਚ ਇਨਕੋਡ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ, ਜੋ ਇਸਨੂੰ ਬਹੁ-ਭਾਸ਼ਾਈ ਮਾਡਲਾਂ ਲਈ ਢੁਕਵਾਂ ਬਣਾਉਂਦੀ ਹੈ। Llama ਟੋਕਨਾਈਜ਼ਰ Meta ਦੇ Llama ਅਤੇ Alpaca ਵਰਗੇ ਮਾਡਲਾਂ ਦੁਆਰਾ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

- **ਸੈਨਟੈਸਪੀਸ (ਯੂਨੀਗ੍ਰਾਮ):** ਸੈਨਟੈਸਪੀਸ ਟੋਕਨਾਈਜ਼ਰ ਯੂਨੀਗ੍ਰਾਮ ਨਾਮਕ ਇੱਕ ਵੱਖਰਾ ਐਲਗੋਰਦਿਮ ਵੀ ਵਰਤ ਸਕਦੇ ਹਨ, ਜੋ ਉਪ-ਸ਼ਬਦ ਨਯਿਮਤੀਕਰਨ ਤਕਨੀਕ 'ਤੇ ਆਧਾਰਤਿ ਹੈ। ਯੂਨੀਗ੍ਰਾਮ ਟੋਕਨਾਈਜ਼ੇਸ਼ਨ ਇੱਕ ਯੂਨੀਗ੍ਰਾਮ ਭਾਸ਼ਾ ਮਾਡਲ ਦੇ ਆਧਾਰ 'ਤੇ ਸਭ ਤੋਂ ਵਧੀਆ ਉਪ-ਸ਼ਬਦ ਸ਼ਬਦਾਵਲੀ ਨਰਿਧਾਰਤ ਕਰਦਾ ਹੈ, ਜੋ ਵਅਿਕਤੀਗਤ ਉਪ-ਸ਼ਬਦ ਇਕਾਈਆਂ ਨੂੰ ਸੰਭਾਵਨਾਵਾਂ ਨਰਿਧਾਰਤ ਕਰਦਾ ਹੈ। ਇਹ ਪਹੁੰਚ BPE ਦੇ ਮੁਕਾਬਲੇ ਵਧੇਰੇ ਅਰਥਪੂਰਨ ਉਪ-ਸ਼ਬਦ ਪੈਦਾ ਕਰ ਸਕਦੀ ਹੈ। ਯੂਨੀਗ੍ਰਾਮ ਵਾਲਾ ਸੈਨਟੈਸਪੀਸ Google ਦੇ T5 ਅਤੇ BERT ਵਰਗੇ ਮਾਡਲਾਂ ਦੁਆਰਾ ਵਰਤਅਿਾ ਜਾਂਦਾ ਹੈ।

- **Google Gemini (ਬਹੁ-ਮਾਧਅਿਮੀ ਟੋਕਨਾਈਜ਼ੇਸ਼ਨ):** Google Gemini ਇੱਕ ਅਜਹਿੀ ਟੋਕਨਾਈਜ਼ੇਸ਼ਨ ਸਕੀਮ ਵਰਤਦਾ ਹੈ ਜੋ ਟੈਕਸਟ, ਚੱਤਿਰ, ਆਡੀਓ, ਵੀਡੀਓ, ਅਤੇ ਕੋਡ ਸਮੇਤ ਵੱਖ-ਵੱਖ ਡਾਟਾ ਕਸਿਮਾਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਤਆਿਰ ਕੀਤੀ ਗਈ ਹੈ। ਇਹ ਬਹੁ-ਮਾਧਅਿਮੀ ਸਮਰੱਥਾ Gemini ਨੂੰ ਵੱਖ-ਵੱਖ ਕਸਿਮਾਂ ਦੀ ਜਾਣਕਾਰੀ ਨੂੰ ਪ੍ਰੋਸੈਸ ਅਤੇ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਦੀ ਆਗਆਿ ਦੇਂਦੀ ਹੈ। ਖਾਸ ਤੌਰ 'ਤੇ, Google Gemini 1.5 Pro ਵੱਚਿ ਇੱਕ ਸੰਦਰਭ ਵੱਡਿੇ ਹੈ ਜੋ ਲੱਖਾਂ ਟੋਕਨਾਂ ਨੂੰ ਸੰਭਾਲ ਸਕਦੀ ਹੈ, ਜੋ ਪਛਿਲੇ ਮਾਡਲਾਂ ਨਾਲੋਂ ਬਹੁਤ ਵੱਡੀ ਹੈ। ਇਹ ਵਸ਼ਿਾਲ ਸੰਦਰਭ ਵੱਡਿੇ ਮਾਡਲ ਨੂੰ ਵੱਡੇ ਸੰਦਰਭ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਨ ਦੀ ਆਗਆਿ ਦੇਂਦੀ ਹੈ, ਜੋ ਸੰਭਾਵਤ ਤੌਰ 'ਤੇ ਵਧੇਰੇ ਸਟੀਕ ਜਵਾਬਾਂ ਵੱਲ ਲੈ ਜਾਂਦੀ ਹੈ। ਹਾਲਾਂਕ,ਿ ਇਹ ਨੋਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕ ਿGemini ਦੀ ਟੋਕਨਾਈਜ਼ੇਸ਼ਨ ਸਕੀਮ ਹੋਰ ਮਾਡਲਾਂ ਦੇ ਮੁਕਾਬਲੇ ਪ੍ਰਤੀ ਅੱਖਰ ਇੱਕ ਟੋਕਨ ਦੇ ਬਹੁਤ ਨੇੜੇ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਹੈ ਕ ਿਜੇਕਰ ਤੁਸੀਂ GPT ਵਰਗੇ ਮਾਡਲਾਂ ਦੀ ਵਰਤੋਂ ਦੇ ਆਦੀ ਹੋ, ਤਾਂ Gemini ਮਾਡਲਾਂ ਦੀ ਵਰਤੋਂ ਦੀ ਅਸਲ ਲਾਗਤ ਉਮੀਦ ਨਾਲੋਂ ਕਾਫ਼ੀ ਜ਼ਆਿਦਾ ਹੋ ਸਕਦੀ ਹੈ, ਕਉਿਂਕ ਿGoogle ਦੀ ਕੀਮਤ ਟੋਕਨਾਂ ਦੀ ਬਜਾਏ ਅੱਖਰਾਂ 'ਤੇ ਆਧਾਰਤਿ ਹੈ।

ਟੋਕਨਾਈਜ਼ਰ ਦੀ ਚੋਣ LLM ਦੇ ਕਈ ਪਹਲੂਿਆਂ ਨੂੰ ਪ੍ਰਭਾਵਤਿ ਕਰਦੀ ਹੈ, ਜਸਿ ਵੱਚਿ ਸ਼ਾਮਲ ਹਨ:

- **ਸ਼ਬਦਾਵਲੀ ਦਾ ਆਕਾਰ:** ਟੋਕਨਾਈਜ਼ਰ ਮਾਡਲ ਦੀ ਸ਼ਬਦਾਵਲੀ ਦਾ ਆਕਾਰ ਨਰਿਧਾਰਤ ਕਰਦਾ ਹੈ, ਜੋ ਇਸ ਦੁਆਰਾ ਪਛਾਣੇ ਜਾਣ ਵਾਲੇ ਵਲਿੱਖਣ ਟੋਕਨਾਂ ਦਾ ਸੈੱਟ ਹੈ। ਇੱਕ ਵੱਡੀ, ਵਧੇਰੇ ਸੂਖਮ ਸ਼ਬਦਾਵਲੀ ਮਾਡਲ ਨੂੰ ਵਆਿਪਕ ਸ਼ਬਦਾਂ ਅਤੇ ਵਾਕਾਂਸ਼ਾਂ ਨੂੰ ਸੰਭਾਲਣ ਅਤੇ ਇੱਥੋਂ ਤੱਕ ਕ ਿਬਹੁ-ਮਾਧਅਿਮੀ (ਟੈਕਸਟ ਤੋਂ ਇਲਾਵਾ ਹੋਰ ਚੀਜ਼ਾਂ ਨੂੰ ਸਮਝਣ ਅਤੇ ਤਆਿਰ ਕਰਨ ਦੇ ਯੋਗ) ਬਣਨ ਵੱਚਿ ਮਦਦ ਕਰ ਸਕਦੀ ਹੈ, ਪਰ ਇਹ ਮਾਡਲ ਦੀਆਂ ਮੈਮੋਰੀ ਲੋੜਾਂ ਅਤੇ ਕੰਪਊਿਟੇਸ਼ਨਲ ਜਟਲਿਤਾ ਨੂੰ ਵੀ ਵਧਾਉਂਦੀ ਹੈ।
- **ਦੁਰਲੱਭ ਅਤੇ ਅਣਜਾਣ ਸ਼ਬਦਾਂ ਨੂੰ ਸੰਭਾਲਣਾ:** BPE ਅਤੇ ਸੈਨਟੈਂਸਪੀਸ ਵਰਗੇ ਉਪ-ਸ਼ਬਦ ਇਕਾਈਆਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਵਾਲੇ ਟੋਕਨਾਈਜ਼ਰ ਦੁਰਲੱਭ ਅਤੇ ਅਣਜਾਣ ਸ਼ਬਦਾਂ ਨੂੰ ਵਧੇਰੇ ਆਮ ਉਪ-ਸ਼ਬਦ ਟੁਕੜਆਿਂ ਵੱਚਿ ਤੋੜ ਸਕਦੇ ਹਨ। ਇਹ ਮਾਡਲ ਨੂੰ ਉਹਨਾਂ ਸ਼ਬਦਾਂ ਦੇ ਅਰਥ ਬਾਰੇ ਸੰਖਆਿਤ ਅੰਦਾਜ਼ੇ ਲਗਾਉਣ ਦੀ ਆਗਆਿ ਦੇਂਦਾ ਹੈ ਜੋ ਇਸਨੇ ਪਹਲਿਾਂ ਨਹੀ ਦੇਖੇ, ਉਹਨਾਂ ਵੱਚਿ ਮੌਜੂਦ ਉਪ-ਸ਼ਬਦਾਂ ਦੇ ਆਧਾਰ 'ਤੇ।
- **ਬਹੁਭਾਸ਼ਾਈ ਸਹਾਇਤਾ:** ਸੈਨਟੈਂਸਪੀਸ ਵਰਗੇ ਟੋਕਨਾਈਜ਼ਰ, ਜੋ ਕਸਿੇ ਵੀ ਯੂਨੀਕੋਡ-ਏਨਕੋਡੇਬਲ ਭਾਸ਼ਾ ਨੂੰ ਸੰਭਾਲ ਸਕਦੇ ਹਨ, ਬਹੁਭਾਸ਼ਾਈ ਮਾਡਲਾਂ ਲਈ ਢੁਕਵੇਂ ਹਨ ਜਨ੍ਹਿਾਂ ਨੂੰ ਕਈ ਭਾਸ਼ਾਵਾਂ ਵੱਚਿ ਟੈਕਸਟ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

ਕਸਿ ਖਾਸ ਐਪਲੀਕੇਸ਼ਨ ਲਈ ਐੱਲਐੱਲਐੱਮ ਦੀ ਚੋਣ ਕਰਦੇ ਸਮੇਂ, ਇਹ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕ ਿਇਸ ਦੁਆਰਾ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਟੋਕਨਾਈਜ਼ਰ 'ਤੇ ਵਚਿਾਰ ਕੀਤਾ ਜਾਵੇ ਅਤੇ ਇਹ ਕੀਂਨੀ ਚੰਗੀ ਤਰ੍ਹਾਂ ਕਾਰਜ ਦੀਆਂ ਵਸ਼ਿੇਸ਼ ਭਾਸ਼ਾ ਪ੍ਰੋਸੈਸਿੰਗ ਲੋੜਾਂ ਨਾਲ ਮੇਲ ਖਾਂਦਾ ਹੈ। ਟੋਕਨਾਈਜ਼ਰ ਮਾਡਲ ਦੀ ਡੋਮੇਨ-ਵਸ਼ਿੇਸ਼ ਸ਼ਬਦਾਵਲੀ, ਦੁਰਲੱਭ ਸ਼ਬਦਾਂ, ਅਤੇ ਬਹੁ-ਭਾਸ਼ਾਈ ਟੈਕਸਟ ਨੂੰ ਸੰਭਾਲਣ ਦੀ ਯੋਗਤਾ 'ਤੇ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦਾ ਹੈ।

ਸੰਦਰਭ ਆਕਾਰ: ਭਾਸ਼ਾ ਮਾਡਲ ਅਨੁਮਾਨ ਦੌਰਾਨ ਕਿੰਨੀ ਜਾਣਕਾਰੀ ਵਰਤ ਸਕਦਾ ਹੈ?

ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੀ ਚਰਚਾ ਕਰਦੇ ਸਮੇਂ, ਸੰਦਰਭ ਆਕਾਰ ਉਸ ਟੈਕਸਟ ਦੀ ਮਾਤਰਾ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਜੋ ਇੱਕ ਮਾਡਲ ਆਪਣੇ ਜਵਾਬਾਂ ਦੀ ਪ੍ਰੋਸੈਸਿੰਗ ਜਾਂ ਜਨਰੇਸ਼ਨ ਦੌਰਾਨ ਵਿਚਾਰ ਕਰ ਸਕਦਾ ਹੈ। ਇਹ ਅਸਲ ਵਿੱਚ ਇਸ ਗੱਲ ਦਾ ਮਾਪ ਹੈ ਕਿ ਮਾਡਲ ਕਿੰਨੀ ਜਾਣਕਾਰੀ ਨੂੰ "ਯਾਦ" ਰੱਖ ਸਕਦਾ ਹੈ ਅਤੇ ਆਪਣੇ ਆਉਟਪੁੱਟ ਨੂੰ ਸੂਚਤਿ ਕਰਨ ਲਈ ਵਰਤ ਸਕਦਾ ਹੈ (ਟੋਕਨਾਂ ਵਿੱਚ ਪ੍ਰਗਟ ਕੀਤਾ ਗਿਆ)। ਭਾਸ਼ਾ ਮਾਡਲ ਦਾ ਸੰਦਰਭ ਆਕਾਰ ਇਸਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਅਤੇ ਇਹ ਜਹਿੜੇ ਕੰਮ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਕਰ ਸਕਦਾ ਹੈ, ਉਨ੍ਹਾਂ 'ਤੇ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦਾ ਹੈ।

ਸੰਦਰਭ ਆਕਾਰ ਕੀ ਹੈ?

ਤਕਨੀਕੀ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਸੰਦਰਭ ਆਕਾਰ ਉਨ੍ਹਾਂ ਟੋਕਨਾਂ (ਸ਼ਬਦਾਂ ਜਾਂ ਸ਼ਬਦ ਭਾਗਾਂ) ਦੀ ਸੰਖਿਆ ਦੁਆਰਾ ਨਿਰਧਾਰਤਿ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜੋ ਇੱਕ ਭਾਸ਼ਾ ਮਾਡਲ ਇੱਕ ਇਕੱਲੀ ਇਨਪੁੱਟ ਲੜੀ ਵਿੱਚ ਪ੍ਰੋਸੈੱਸ ਕਰ ਸਕਦਾ ਹੈ। ਇਸਨੂੰ ਅਕਸਰ ਮਾਡਲ ਦੀ "ਧਿਆਨ ਸੀਮਾ" ਜਾਂ "ਸੰਦਰਭ ਵਿੰਡੋ" ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਸੰਦਰਭ ਆਕਾਰ ਜਿੰਨਾ ਵੱਡਾ ਹੋਵੇਗਾ, ਮਾਡਲ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਜਾਂ ਕੋਈ ਕੰਮ ਕਰਨ ਵੇਲੇ ਓਨਾ ਹੀ ਵੱਧ ਟੈਕਸਟ 'ਤੇ ਵਿਚਾਰ ਕਰ ਸਕਦਾ ਹੈ।

ਵੱਖ-ਵੱਖ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੇ ਵੱਖ-ਵੱਖ ਸੰਦਰਭ ਆਕਾਰ ਹੁੰਦੇ ਹਨ, ਜੋ ਕੁਝ ਸੌ ਟੋਕਨਾਂ ਤੋਂ ਲੈ ਕੇ ਲੱਖਾਂ ਟੋਕਨਾਂ ਤੱਕ ਹੋ ਸਕਦੇ ਹਨ। ਹਵਾਲੇ ਲਈ, ਟੈਕਸਟ ਦੇ ਇੱਕ ਆਮ ਪੈਰੇ ਵਿੱਚ ਲਗਭਗ 100-150 ਟੋਕਨ ਹੋ ਸਕਦੇ ਹਨ, ਜਦੋਂ ਕਿ ਇੱਕ ਪੂਰੀ ਕਿਤਾਬ ਵਿੱਚ ਦਸ ਜਾਂ ਸੈਕੜੇ ਹਜ਼ਾਰ ਟੋਕਨ ਹੋ ਸਕਦੇ ਹਨ।

> ਟ੍ਰਾਂਸਫਾਰਮਰ-ਆਧਾਰਤਿ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ (ਐੱਲਐੱਲਐੱਮ) ਨੂੰ ਸੀਮਤਿ ਮੈਮੋਰੀ ਅਤੇ ਕੰਪਿਊਟੇਸ਼ਨ ਨਾਲ ਅਸੀਮਤਿ ਲੰਬੀਆਂ ਇਨਪੁੱਟਾਂ ਤੱਕ ਸਕੇਲ ਕਰਨ ਦੇ ਕੁਸ਼ਲ ਤਰੀਕਿਆਂ 'ਤੇ ਵੀ ਕੰਮ ਕੀਤਾ ਜਾ ਰਹਿਾ ਹੈ।

ਸੰਦਰਭ ਆਕਾਰ ਮਹੱਤਵਪੂਰਨ ਕਿਉਂ ਹੈ?

ਭਾਸ਼ਾ ਮਾਡਲ ਦਾ ਸੰਦਰਭ ਆਕਾਰ ਸਮਝਣ ਅਤੇ ਸੰਦਰਭ-ਅਨੁਸਾਰ ਢੁਕਵਾਂ, ਸੰਗਤ ਟੈਕਸਟ ਤਿਆਰ ਕਰਨ ਦੀ ਇਸਦੀ ਯੋਗਤਾ 'ਤੇ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਪਾਉਂਦਾ ਹੈ। ਇੱਥੇ ਕੁਝ ਮੁੱਖ ਕਾਰਨ ਹਨ ਕਿ ਸੰਦਰਭ ਆਕਾਰ ਕਿਉਂ ਮਹੱਤਵਪੂਰਨ ਹੈ:

1. **ਲੰਬੀ-ਫਾਰਮ ਸਮੱਗਰੀ ਨੂੰ ਸਮਝਣਾ:** ਵੱਡੇ ਸੰਦਰਭ ਆਕਾਰ ਵਾਲੇ ਮਾਡਲ ਲੰਬੇ ਟੈਕਸਟਾਂ, ਜਵਿੇਂ ਕ ਿਲੇਖਾਂ, ਰਪਿਰਟਾਂ, ਜਾਂ ਇੱਥੋਂ ਤੱਕ ਕ ਪੂਰੀਆਂ ਕਤਿਾਬਾਂ ਨੂੰ ਬਹਿਤਰ ਢੰਗ ਨਾਲ ਸਮਝ ਅਤੇ ਵਸਿ਼ਲੇਸ਼ਣ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਦਸਤਾਵੇਜ਼ ਸੰਖੇਪੀਕਰਨ, ਸਵਾਲ-ਜਵਾਬ, ਅਤੇ ਸਮੱਗਰੀ ਵਸਿ਼ਲੇਸ਼ਣ ਵਰਗੇ ਕੰਮਾਂ ਲਈ ਮਹੱਤਵਪੂਰਨ ਹੈ।

2. **ਸੰਗਤਤਾ ਬਣਾਈ ਰੱਖਣਾ:** ਇੱਕ ਵੱਡੀ ਸੰਦਰਭ ਵੰਡਿੋ ਮਾਡਲ ਨੂੰ ਲੰਬੇ ਆਉਟਪੁੱਟ ਵ ਿਚ ਸੰਗਤਤਾ ਅਤੇ ਇਕਸਾਰਤਾ ਬਣਾਈ ਰੱਖਣ ਦੀ ਇਜਾਜ਼ਤ ਦ ਿਂਦੀ ਹੈ। ਇਹ ਕਹਾਣੀ ਜਨਰੇਸ਼ਨ, ਸੰਵਾਦ ਸਸਿਟਮ, ਅਤੇ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਵਰਗੇ ਕੰਮਾਂ ਲਈ ਮਹੱਤਵਪੂਰਨ ਹੈ, ਜ ਿਥੇ ਇੱਕ ਸਥਰਿ ਕਥਾ ਜਾਂ ਵਸਿ਼ ਨੂੰ ਬਣਾਈ ਰੱਖਣਾ ਜ਼ਰੂਰੀ ਹੈ। ਇਹ ਢਾਂਚਾਗਤ ਡੇਟਾ ਨੂੰ ਤਆਿਰ ਕਰਨ ਜਾਂ ਬਦਲਣ ਲਈ ਐਲਐਲਐਮ ਦੀ ਵਰਤੋਂ ਕਰਨ ਵੇਲੇ ਵੀ ਬਲਿਕੁਲ ਜ਼ਰੂਰੀ ਹੈ।
3. **ਲੰਬੀ-ਦੂਰੀ ਦੀਆਂ ਨਰਿਭਰਤਾਵਾਂ ਨੂੰ ਕੈਪਚਰ ਕਰਨਾ:** ਕੁਝ ਭਾਸ਼ਾ ਕਾਰਜਾਂ ਲਈ ਟੈਕਸਟ ਵ ਿਚ ਦੂਰ-ਦੂਰ ਮੌਜੂਦ ਸ਼ਬਦਾਂ ਜਾਂ ਵਾਕਾਂਸ਼ਾਂ ਵਚਿਕਾਰ ਸਬੰਧਾਂ ਨੂੰ ਸਮਝਣ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਵੱਡੇ ਸੰਦਰਭ ਆਕਾਰ ਵਾਲੇ ਮਾਡਲ ਇਹਨਾਂ ਲੰਬੀ-ਦੂਰੀ ਦੀਆਂ ਨਰਿਭਰਤਾਵਾਂ ਨੂੰ ਕੈਪਚਰ ਕਰਨ ਲਈ ਬਹਿਤਰ ਤਰੀਕੇ ਨਾਲ ਤਆਿਰ ਹੁੰਦੇ ਹਨ, ਜੋ ਭਾਵਨਾ ਵਸਿ਼ਲੇਸ਼ਣ, ਅਨੁਵਾਦ, ਅਤੇ ਭਾਸ਼ਾ ਸਮਝ ਵਰਗੇ ਕਾਰਜਾਂ ਲਈ ਮਹੱਤਵਪੂਰਨ ਹੋ ਸਕਦੇ ਹਨ।
4. **ਗੁੰਝਲਦਾਰ ਹਦਾਇਤਾਂ ਨੂੰ ਸੰਭਾਲਣਾ:** ਉਹਨਾਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵ ਿਚ ਜ ਿਥੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੀ ਵਰਤੋਂ ਗੁੰਝਲਦਾਰ, ਬਹੁ-ਪੜਾਵੀ ਹਦਾਇਤਾਂ ਦੀ ਪਾਲਣਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਵੱਡਾ ਸੰਦਰਭ ਆਕਾਰ ਮਾਡਲ ਨੂੰ ਜਵਾਬ ਤਆਿਰ ਕਰਦੇ ਸਮੇਂ ਸਰਫ਼ ਹਾਲ ਹੀ ਦੇ ਕੁਝ ਸ਼ਬਦਾਂ ਦੀ ਬਜਾਏ ਹਦਾਇਤਾਂ ਦੇ ਪੂਰੇ ਸੈੱਟ ’ਤੇ ਵਚਿਾਰ ਕਰਨ ਦੀ ਆਗਆਿ ਦ ਿਂਦਾ ਹੈ।

ਵੱਖ-ਵੱਖ ਸੰਦਰਭ ਆਕਾਰਾਂ ਵਾਲੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੀਆਂ ਉਦਾਹਰਣਾਂ

ਇੱਥੇ ਵੱਖ-ਵੱਖ ਸੰਦਰਭ ਆਕਾਰਾਂ ਵਾਲੇ ਕੁਝ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ:

- OpenAI GPT-3.5 Turbo: 4,095 ਟੋਕਨ
- Mistral 7B Instruct: 32,768 ਟੋਕਨ
- Anthropic Claude v1: 100,000 ਟੋਕਨ
- OpenAI GPT-4 Turbo: 128,000 ਟੋਕਨ
- Anthropic Claude v2: 200,000 ਟੋਕਨ

- Google Gemini Pro 1.5: 2.8M ਟੋਕਨ

ਜਦਿੰ ਕਿ ਤੁਸੀ ਦੇਖ ਸਕਦੇ ਹੋ, ਇਹਨਾਂ ਮਾਡਲਾਂ ਵਿੱਚ ਸੰਦਰਭ ਆਕਾਰਾਂ ਦੀ ਵਿਆਪਕ ਸ਼੍ਰੇਣੀ ਹੈ, OpenAI GPT-3.5 Turbo ਮਾਡਲ ਲਈ ਲਗਭਗ 4,000 ਟੋਕਨਾਂ ਤੋਂ ਲੈ ਕੇ Anthropic Claude v2 ਮਾਡਲ ਲਈ 200,000 ਟੋਕਨ ਤੱਕ। ਕੁਝ ਮਾਡਲ, ਜਦਿੰ ਕਿ Google ਦਾ PaLM 2 ਅਤੇ OpenAI ਦਾ GPT-4, ਵੱਡੇ ਸੰਦਰਭ ਆਕਾਰਾਂ ਵਾਲੇ ਵੱਖ-ਵੱਖ ਰੂਪਾਂਤਰ ਪੇਸ਼ ਕਰਦੇ ਹਨ (ਜਦਿੰ ਕਿ "32k" ਵਰਜਨ), ਜੋ ਹੋਰ ਵੀ ਲੰਬੀਆਂ ਇਨਪੁੱਟ ਲੜੀਆਂ ਨੂੰ ਸੰਭਾਲ ਸਕਦੇ ਹਨ। ਅਤੇ ਇਸ ਸਮੇਂ (ਅਪ੍ਰੈਲ 2024) Google Gemini Pro ਲਗਭਗ 3 ਮਿਲੀਅਨ ਟੋਕਨਾਂ ਦਾ ਦਾਅਵਾ ਕਰ ਰਹਿਾ ਹੈ!

ਇਹ ਨੋਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਸੰਦਰਭ ਆਕਾਰ ਕਸਿੇ ਖਾਸ ਮਾਡਲ ਦੇ ਵਸਿ਼ਸ਼ ਲਾਗੂਕਰਨ ਅਤੇ ਵਰਜਨ 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹੋਏ ਵੱਖ-ਵੱਖ ਹੋ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ, ਮੂਲ OpenAI GPT-4 ਮਾਡਲ ਦਾ ਸੰਦਰਭ ਆਕਾਰ 8,191 ਟੋਕਨ ਹੈ, ਜਦੋਂ ਕਿ ਬਾਅਦ ਦੇ GPT-4 ਰੂਪਾਂਤਰ ਜਦਿੰ ਕਿ Turbo ਅਤੇ 4o ਦਾ ਬਹੁਤ ਵੱਡਾ ਸੰਦਰਭ ਆਕਾਰ 128,000 ਟੋਕਨ ਹੈ।

> Sam Altman ਨੇ ਮੌਜੂਦਾ ਸੰਦਰਭ ਸੀਮਾਵਾਂ ਦੀ ਤੁਲਨਾ 80 ਦੇ ਦਹਾਕੇ ਵਿੱਚ ਨਜ਼ਿੀ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਰਾਂ ਨੂੰ ਨਜਠਿਣ ਵਾਲੀ ਕਲਿੋਬਾਈਟਸ ਕਾਰਜਸ਼ੀਲ ਮੈਮੋਰੀ ਨਾਲ ਕੀਤੀ ਹੈ, ਅਤੇ ਕਹਿਾ ਹੈ ਕਿ ਨੇੜਲੇ ਭਵਖਿ ਵਿੱਚ ਅਸੀਂ "ਆਪਣਾ ਸਾਰਾ ਨਜ਼ਿੀ ਡਾਟਾ" ਇੱਕ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ ਦੇ ਸੰਦਰਭ ਵਿੱਚ ਫਟਿ ਕਰ ਸਕਾਂਗੇ।

ਸਹੀ ਸੰਦਰਭ ਆਕਾਰ ਦੀ ਚੋਣ ਕਰਨਾ

ਕਸਿੇ ਖਾਸ ਐਪਲੀਕੇਸ਼ਨ ਲਈ ਭਾਸ਼ਾ ਮਾਡਲ ਦੀ ਚੋਣ ਕਰਦੇ ਸਮੇਂ, ਕਾਰਜ ਦੀਆਂ ਸੰਦਰਭ ਆਕਾਰ ਲੋੜਾਂ 'ਤੇ ਵਚਿਾਰ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਛੋਟੇ, ਵੱਖਰੇ ਟੈਕਸਟ ਦੇ ਟੁਕੜਿਆਂ ਨਾਲ ਜੁੜੇ ਕੰਮਾਂ ਲਈ, ਜਦਿੰ ਕਿ ਭਾਵਨਾ ਵਸ਼ਿਲੇਸ਼ਣ ਜਾਂ ਸਧਾਰਨ ਸਵਾਲ-ਜਵਾਬ, ਇੱਕ ਛੋਟਾ ਸੰਦਰਭ ਆਕਾਰ ਕਾਫ਼ੀ ਹੋ ਸਕਦਾ ਹੈ। ਹਾਲਾਂਕਿ, ਲੰਬੇ, ਵਧੇਰੇ ਜਟਲਿ ਟੈਕਸਟਾਂ ਨੂੰ ਸਮਝਣ ਅਤੇ ਤਆਿਰ ਕਰਨ ਦੀ ਲੋੜ ਵਾਲੇ ਕੰਮਾਂ ਲਈ, ਵੱਡੇ ਸੰਦਰਭ ਆਕਾਰ ਦੀ ਜ਼ਰੂਰਤ ਹੋਵੇਗੀ।

ਇਹ ਨੋਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਵੱਡੇ ਸੰਦਰਭ ਆਕਾਰ ਅਕਸਰ ਵਧੇ ਹੋਏ ਕੰਪਿਊਟੇਸ਼ਨਲ ਖਰਚਿਆਂ ਅਤੇ ਹੌਲੀ ਪ੍ਰੋਸੈਸਿੰਗ ਸਮੇਂ ਨਾਲ ਆਉਂਦੇ ਹਨ, ਕਿਉਂਕਿ ਮਾਡਲ ਨੂੰ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਵੇਲੇ ਵਧੇਰੇ ਜਾਣਕਾਰੀ 'ਤੇ ਵਚਿਾਰ ਕਰਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਇਸ ਲਈ, ਆਪਣੀ ਐਪਲੀਕੇਸ਼ਨ ਲਈ ਭਾਸ਼ਾ ਮਾਡਲ ਚੁਣਦੇ ਸਮੇਂ ਤੁਹਾਨੂੰ ਸੰਦਰਭ ਆਕਾਰ ਅਤੇ ਪ੍ਰਦਰਸ਼ਨ ਵਚਿਕਾਰ ਸੰਤੁਲਨ ਬਣਾਉਣਾ ਚਾਹੀਦਾ ਹੈ।

> ਸਭ ਤੋਂ ਵੱਡੇ ਸੰਦਰਭ ਆਕਾਰ ਵਾਲੇ ਮਾਡਲ ਨੂੰ ਚੁਣ ਕੇ ਇਸ ਵੱਚਿ ਜੀਨੀ ਸੰਭਵ ਹੋ ਸਕੇ ਜਾਣਕਾਰੀ ਕਉਂ ਨਾ ਭਰੀ ਜਾਵੇ? ਖੈਰ, ਪ੍ਰਦਰਸ਼ਨ ਕਾਰਕਾਂ ਤੋਂ ਇਲਾਵਾ ਦੂਜਾ ਮੁੱਖ ਵਚਿਾਰ ਲਾਗਤ ਹੈ। ਮਾਰਚ 2024 ਵੱਚਿ Google Gemini Pro 1.5 ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਪੂਰੇ ਸੰਦਰਭ ਨਾਲ ਇੱਕ ਇਕੱਲਾ ਪ੍ਰੌਂਪਟ-ਜਵਾਬ ਚੱਕਰ ਤੁਹਾਨੂੰ ਲਗਭਗ $8 (USD) ਦਾ ਪਵੇਗਾ। ਜੇਕਰ ਤੁਹਾਡੇ ਕੋਲ ਕੋਈ ਅਜਹਿਾ ਕੇਸ ਹੈ ਜੋ ਇਸ ਖਰਚੇ ਨੂੰ ਜਾਇਜ਼ ਠਹਰਿਾਉਂਦਾ ਹੈ, ਤਾਂ ਬਹੁਤ ਵਧੀਆ! ਪਰ ਜ਼ਆਿਦਾਤਰ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ, ਇਹ ਕਈ ਗੁਣਾ ਜ਼ਆਿਦਾ ਮਹੰਗਿਾ ਹੈ।

ਘਾਹ ਦੇ ਢੇਰ ਵੱਚਿੋਂ ਸੂਈਆਂ ਲੱਭਣਾ

ਘਾਹ ਦੇ ਢੇਰ ਵੱਚਿੋਂ ਸੂਈ ਲੱਭਣ ਦੀ ਧਾਰਨਾ ਲੰਬੇ ਸਮੇਂ ਤੋਂ ਵੱਡੇ ਡੇਟਾਸੈੱਟਾਂ ਵੱਚਿ ਪੁਨਰਪ੍ਰਾਪਤੀ ਦੀਆਂ ਚੁਣੌਤੀਆਂ ਲਈ ਇੱਕ ਰੂਪਕ ਰਹੀ ਹੈ। LLMs ਦੇ ਖੇਤਰ ਵੱਚਿ, ਅਸੀਂ ਇਸ ਉਪਮਾ ਨੂੰ ਥੋੜ੍ਹਾ ਜਹਿਾ ਬਦਲਦੇ ਹਾਂ। ਕਲਪਨਾ ਕਰੋ ਕ ਿਅਸੀਂ ਸਰਫ਼ ਵਸ਼ਿਾਲ ਟੈਕਸਟ (ਜਵਿੇਂ ਕ ਿPaul Graham ਦੇ ਲੇਖਾਂ ਦਾ ਪੂਰਾ ਸੰਗ੍ਰਹ) ਵੱਚਿ ਦੱਬੇ ਇੱਕ ਤੱਥ ਦੀ ਭਾਲ ਨਹੀਂ ਕਰ ਰਹੇ, ਬਲਕ ਿਪੂਰੇ ਟੈਕਸਟ ਵੱਚਿ ਖੰਡ ਿਹੋਏ ਕਈ ਤੱਥਾਂ ਦੀ ਭਾਲ ਕਰ ਰਹੇ ਹਾਂ। ਇਹ ਸਥਤਿੀ ਇੱਕ ਘਾਹ ਦੇ ਢੇਰ ਦੀ ਬਜਾਏ ਇੱਕ ਫੈਲੇ ਹੋਏ ਖੇਤ ਵੱਚਿ ਕਈ ਸੂਈਆਂ ਲੱਭਣ ਵਰਗੀ ਹੈ। ਇੱਥੇ ਮੁੱਖ ਗੱਲ ਇਹ ਹੈ: ਸਾਨੂੰ ਨਾ ਸਰਫ਼ ਇਹਨਾਂ ਸੂਈਆਂ ਨੂੰ ਲੱਭਣਾ ਹੈ, ਬਲਕ ਿਉਹਨਾਂ ਨੂੰ ਇੱਕ ਸੁਸੰਗਤ ਧਾਗੇ ਵੱਚਿ ਵੀ ਪਰੋਣਾ ਹੈ।

ਜਦੋਂ ਲੰਬੇ ਸੰਦਰਭਾਂ ਵੱਚਿ ਦਰਜ ਕਈ ਤੱਥਾਂ ਨੂੰ ਪੁਨਰਪ੍ਰਾਪਤ ਕਰਨ ਅਤੇ ਉਨ੍ਹਾਂ ਬਾਰੇ ਤਰਕ ਕਰਨ ਦਾ ਕੰਮ ਦੰਤਿਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ LLMs ਨੂੰ ਦੋਹਰੀ ਚੁਣੌਤੀ ਦਾ ਸਾਹਮਣਾ ਕਰਨਾ ਪੈਂਦਾ ਹੈ। ਪਹਲਿਾਂ, ਪੁਨਰਪ੍ਰਾਪਤੀ ਸ਼ੁੱਧਤਾ ਦਾ ਸੰਧਿਾ ਮੁੱਦਾ ਹੈ - ਇਹ ਤੱਥਾਂ ਦੀ ਸੰਖਆ ਵਧਣ ਨਾਲ ਕੁਦਰਤੀ ਤੌਰ 'ਤੇ ਘੱਟ ਜਾਂਦੀ ਹੈ। ਇਹ ਉਮੀਦ ਕੀਤੀ ਜਾਂਦੀ ਹੈ; ਆਖਰਿਕਾਰ, ਫੈਲੇ ਹੋਏ ਟੈਕਸਟ ਵੱਚਿ ਕਈ ਵੇਰਵਆਿਂ ਦਾ ਧਆਿਨ ਰੱਖਣਾ ਸਭ ਤੋਂ ਉੱਨਤ ਮਾਡਲਾਂ ਨੂੰ ਵੀ ਪ੍ਰਭਾਵਤਿ ਕਰਦਾ ਹੈ।

ਦੂਜਾ, ਅਤੇ ਸ਼ਾਇਦ ਵਧੇਰੇ ਮਹੱਤਵਪੂਰਨ, ਇਨ੍ਹਾਂ ਤੱਥਾਂ ਨਾਲ ਤਰਕ ਕਰਨ ਦੀ ਚੁਣੌਤੀ ਹੈ। ਤੱਥਾਂ ਨੂੰ ਚੁਣਨਾ ਇੱਕ ਗੱਲ ਹੈ; ਉਨ੍ਹਾਂ ਨੂੰ ਇੱਕ ਸੁਸੰਗਤ ਕਹਾਣੀ ਜਾਂ ਜਵਾਬ ਵੱਚਿ ਸੰਸਲੇਸ਼ਣ ਕਰਨਾ ਬਲਿਕੁਲ ਵੱਖਰੀ ਗੱਲ ਹੈ। ਇੱਥੇ ਅਸਲ ਪਰੀਖਆਿ ਆਉਂਦੀ ਹੈ। ਤਰਕ ਕਾਰਜਾਂ ਵੱਚਿ LLMs ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਸਧਾਰਨ ਪੁਨਰ-ਪ੍ਰਾਪਤੀ ਕਾਰਜਾਂ ਨਾਲੋਂ ਵਧੇਰੇ ਘੱਟਦੀ ਹੈ। ਇਹ ਗਰਿਾਵਟ ਸਰਫ਼ ਮਾਤਰਾ ਬਾਰੇ ਨਹੀਂ ਹੈ; ਇਹ ਸੰਦਰਭ, ਪ੍ਰਸੰਗਕਿਤਾ, ਅਤੇ ਅਨੁਮਾਨ ਦਾ ਜਟਲਿ ਨਾਚ ਹੈ।

ਅਜਹिा ਕਉਂ ਹੁੰਦਾ ਹੈ? ਮਨੁੱਖੀ ਬੋਧ ਵੱਚਿ ਯਾਦਦਾਸ਼ਤ ਅਤੇ ਧਆਿਨ ਦੀ ਗਤੀਸ਼ੀਲਤਾ 'ਤੇ ਵਚਿਾਰ ਕਰੋ, ਜੋ ਕੁਝ ਹੱਦ ਤੱਕ LLMs ਵੱਚਿ ਪ੍ਰਤੀਬੰਬਿਤ ਹੁੰਦੀ ਹੈ। ਵੱਡੀ ਮਾਤਰਾ ਵੱਚਿ ਜਾਣਕਾਰੀ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਦੇ ਸਮੇਂ, LLMs,

ਮਨੁੱਖਾਂ ਵਾਂਗ, ਨਵੀਂ ਜਾਣਕਾਰੀ ਨੂੰ ਗ੍ਰਹਣਿ ਕਰਦੇ ਹੋਏ ਪਹਲਿਾਂ ਦੇ ਵੇਰਵਆਿਂ ਨੂੰ ਭੁੱਲ ਸਕਦੇ ਹਨ। ਇਹ ਖਾਸ ਤੌਰ 'ਤੇ ਉਨ੍ਹਾਂ ਮਾਡਲਾਂ ਵਚਿ ਸੱਚ ਹੈ ਜੋ ਸਵੈਚਲਤਿ ਤੌਰ 'ਤੇ ਟੈਕਸਟ ਦੇ ਪਹਲਿੇ ਹਸਿਆਂ ਨੂੰ ਤਰਜੀਹ ਦੇਣ ਜਾਂ ਦੁਬਾਰਾ ਦੇਖਣ ਲਈ ਵਸਿ਼ੇਸ਼ ਤੌਰ 'ਤੇ ਡਜਿ਼ਾਈਨ ਨਹੀਂ ਕੀਤੇ ਗਏ ਹਨ।

ਇਸ ਤੋਂ ਇਲਾਵਾ, ਇੱਕ LLM ਦੀ ਇਨ੍ਹਾਂ ਪ੍ਰਾਪਤ ਕੀਤੇ ਤੱਥਾਂ ਨੂੰ ਇੱਕ ਸੁਸੰਗਤ ਜਵਾਬ ਵਚਿ ਬੁਣਨ ਦੀ ਯੋਗਤਾ ਕਹਾਣੀ ਨਰਿਮਾਣ ਵਰਗੀ ਹੈ। ਇਸ ਲਈ ਸਰਿਫ਼ ਜਾਣਕਾਰੀ ਦੀ ਪੁਨਰ-ਪ੍ਰਾਪਤੀ ਹੀ ਨਹੀਂ, ਬਲਕਿ ਡੂੰਘੀ ਸਮਝ ਅਤੇ ਸੰਦਰਭਕ ਸਥਾਪਨਾ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜੋ ਮੌਜੂਦਾ AI ਲਈ ਇੱਕ ਵੱਡੀ ਚੁਣੌਤੀ ਬਣੀ ਹੋਈ ਹੈ।

ਤਾਂ, ਇਸਦਾ ਸਾਡੇ ਲਈ ਇਨ੍ਹਾਂ ਤਕਨਾਲੋਜੀਆਂ ਦੇ ਡਵੈਲਪਰਾਂ ਅਤੇ ਏਕੀਕਰਣਕਰਤਾਵਾਂ ਵਜੋਂ ਕੀ ਮਤਲਬ ਹੈ? ਸਾਨੂੰ ਉਨ੍ਹਾਂ ਸਸਿਟਮਾਂ ਨੂੰ ਡਜਿ਼ਾਈਨ ਕਰਦੇ ਸਮੇਂ ਇਨ੍ਹਾਂ ਸੀਮਾਵਾਂ ਤੋਂ ਬਹੁਤ ਜਾਗਰੂਕ ਹੋਣ ਦੀ ਲੋੜ ਹੈ ਜੋ ਜਟਲਿ, ਲੰਬੇ-ਰੂਪ ਦੇ ਕਾਰਜਾਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ LLMs 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹਨ। ਇਹ ਸਮਝਣਾ ਕਿ ਕੁਝ ਹਾਲਾਤਾਂ ਵਚਿ ਕਾਰਗੁਜ਼ਾਰੀ ਘੱਟ ਸਕਦੀ ਹੈ, ਸਾਨੂੰ ਯਥਾਰਥਵਾਦੀ ਉਮੀਦਾਂ ਸਥਾਪਤਿ ਕਰਨ ਅਤੇ ਬੇਹਤਰ ਬੈਕਅੱਪ ਵਧਿੀਆਂ ਜਾਂ ਪੂਰਕ ਰਣਨੀਤੀਆਂ ਨੂੰ ਤਆਿਰ ਕਰਨ ਵਚਿ ਮਦਦ ਕਰਦਾ ਹੈ।

ਮੋਡੈਲਟਿੀਆਂ: ਟੈਕਸਟ ਤੋਂ ਪਰੇ

ਜਦੋਂ ਕਿ ਅੱਜ ਦੇ ਜ਼ਆਿਦਾਤਰ ਭਾਸ਼ਾ ਮਾਡਲ ਟੈਕਸਟ ਦੀ ਪ੍ਰੋਸੈਸਿੰਗ ਅਤੇ ਜਨਰੇਸ਼ਨ 'ਤੇ ਕੇਂਦਰਤਿ ਹਨ, ਬਹੁ-ਮਾਧਅਿਮ ਮਾਡਲਾਂ ਵੱਲ ਇੱਕ ਵਧਦਾ ਰੁਝਾਨ ਹੈ ਜੋ ਕੁਦਰਤੀ ਤੌਰ 'ਤੇ ਕਈ ਕਸਿਮਾਂ ਦੇ ਡੇਟਾ, ਜਵਿੇਂ ਕਿ ਚੱਤਿਰ, ਆਡੀਓ, ਅਤੇ ਵੀਡੀਓ ਨੂੰ ਇਨਪੁੱਟ ਅਤੇ ਆਉਟਪੁੱਟ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਬਹੁ-ਮਾਧਅਿਮ ਮਾਡਲ AI-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਨਵੀਆਂ ਸੰਭਾਵਨਾਵਾਂ ਖੋਲ੍ਹਦੇ ਹਨ ਜੋ ਵੱਖ-ਵੱਖ ਮੋਡੈਲਟਿੀਆਂ ਵਚਿ ਸਮੱਗਰੀ ਨੂੰ ਸਮਝ ਅਤੇ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ।

ਮੋਡੈਲਟਿੀਆਂ ਕੀ ਹਨ?

ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੇ ਸੰਦਰਭ ਵਚਿ, ਮੋਡੈਲਟਿੀਆਂ ਵੱਖ-ਵੱਖ ਕਸਿਮਾਂ ਦੇ ਡੇਟਾ ਨੂੰ ਦਰਸਾਉਂਦੀਆਂ ਹਨ ਜੋ ਇੱਕ ਮਾਡਲ ਪ੍ਰੋਸੈਸ ਅਤੇ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ। ਸਭ ਤੋਂ ਆਮ ਮੋਡੈਲਟਿੀ ਟੈਕਸਟ ਹੈ, ਜਸਿ ਵਚਿ ਕਤਿਾਬਾਂ, ਲੇਖ, ਵੈੱਬਸਾਈਟਾਂ, ਅਤੇ ਸੋਸ਼ਲ ਮੀਡੀਆ ਪੋਸਟਾਂ ਵਰਗੇ ਵੱਖ-ਵੱਖ ਰੂਪਾਂ ਵਚਿ ਲਖਿਤੀ ਭਾਸ਼ਾ ਸ਼ਾਮਲ ਹੈ। ਹਾਲਾਂਕਿ, ਕਈ ਹੋਰ ਮੋਡੈਲਟਿੀਆਂ ਹਨ ਜੋ ਵਧਦੇ ਤੌਰ 'ਤੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਵਚਿ ਸ਼ਾਮਲ ਕੀਤੀਆਂ ਜਾ ਰਹੀਆਂ ਹਨ:

- **ਚੱਤਿਰ:** ਦ੍ਰਸਿ਼ਟੀਗਤ ਡੇਟਾ ਜਵਿੇਂ ਫੋਟੋਆਂ, ਚੱਤਿਰ, ਅਤੇ ਡਾਇਗਰਾਮ।
- **ਆਡੀਓ:** ਧੁਨੀ ਡੇਟਾ ਜਵਿੇਂ ਭਾਸ਼ਣ, ਸੰਗੀਤ, ਅਤੇ ਵਾਤਾਵਰਣਕ ਧੁਨੀਆਂ।

- **ਵੀਡੀਓ:** ਚੱਲਦਾ ਦ੍ਰਸ਼ਿਟੀਗਤ ਡੇਟਾ, ਅਕਸਰ ਆਡੀਓ ਨਾਲ, ਜਵਿੇਂ ਵੀਡੀਓ ਕਲਪਿਾਂ ਅਤੇ ਫਲਿਮਾਂ।

ਹਰ ਮਾਧਅਿਮ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਲਈ ਵੱਖਰੀਆਂ ਚੁਣੌਤੀਆਂ ਅਤੇ ਮੌਕੇ ਪੇਸ਼ ਕਰਦਾ ਹੈ। ਉਦਾਹਰਣ ਵਜੋਂ, ਚੱਤਿਰਾਂ ਲਈ ਮਾਡਲ ਨੂੰ ਦ੍ਰਸ਼ਿਟੀਗਤ ਧਾਰਨਾਵਾਂ ਅਤੇ ਸੰਬੰਧਾਂ ਨੂੰ ਸਮਝਣ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜਦੋਂ ਕਆਿਡੀਓ ਲਈ ਮਾਡਲ ਨੂੰ ਭਾਸ਼ਣ ਅਤੇ ਹੋਰ ਧੁਨੀਆਂ ਨੂੰ ਪ੍ਰੋਸੈਸ ਅਤੇ ਤਆਿਰ ਕਰਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

ਬਹੁ-ਮਾਧਅਿਮੀ ਭਾਸ਼ਾ ਮਾਡਲ

ਬਹੁ-ਮਾਧਅਿਮੀ ਭਾਸ਼ਾ ਮਾਡਲ ਇੱਕ ਸਿੰਗਲ ਮਾਡਲ ਵਚਿ ਕਈ ਮਾਧਅਿਮਾਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਤਆਿਰ ਕੀਤੇ ਗਏ ਹਨ। ਇਹਨਾਂ ਮਾਡਲਾਂ ਵਚਿ ਆਮ ਤੌਰ 'ਤੇ ਵਸ਼ਿੇਸ਼ ਭਾਗ ਜਾਂ ਪਰਤਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜੋ ਵੱਖ-ਵੱਖ ਮਾਧਅਿਮਾਂ ਵਚਿ ਇਨਪੁੱਟ ਨੂੰ ਸਮਝ ਅਤੇ ਆਉਟਪੁੱਟ ਡਾਟਾ ਤਆਿਰ ਕਰ ਸਕਦੀਆਂ ਹਨ। ਬਹੁ-ਮਾਧਅਿਮੀ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੇ ਕੁਝ ਪ੍ਰਮੁੱਖ ਉਦਾਹਰਣਾਂ ਵਚਿ ਸ਼ਾਮਲ ਹਨ:

- **OpenAI ਦਾ GPT-4o:** GPT-4o ਇੱਕ ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ ਹੈ ਜੋ ਟੈਕਸਟ ਦੇ ਨਾਲ-ਨਾਲ ਭਾਸ਼ਣ ਆਡੀਓ ਨੂੰ ਕੁਦਰਤੀ ਤੌਰ 'ਤੇ ਸਮਝਦਾ ਅਤੇ ਪ੍ਰੋਸੈਸ ਕਰਦਾ ਹੈ। ਇਹ ਸਮਰੱਥਾ GPT-4o ਨੂੰ ਬੋਲੀ ਗਈ ਭਾਸ਼ਾ ਨੂੰ ਟ੍ਰਾਂਸਕ੍ਰਾਈਬ ਕਰਨ, ਆਡੀਓ ਇਨਪੁੱਟ ਤੋਂ ਟੈਕਸਟ ਤਆਿਰ ਕਰਨ, ਅਤੇ ਬੋਲੇ ਗਏ ਸਵਾਲਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਜਵਾਬ ਦੇਣ ਵਰਗੇ ਕੰਮ ਕਰਨ ਦੀ ਯੋਗਤਾ ਦੇਂਦੀ ਹੈ।
- **OpenAI ਦਾ GPT-4 ਦ੍ਰਸ਼ਿਟੀਗਤ ਇਨਪੁੱਟ ਨਾਲ:** GPT-4 ਇੱਕ ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ ਹੈ ਜੋ ਟੈਕਸਟ ਅਤੇ ਚੱਤਿਰਾਂ ਦੋਵਾਂ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰ ਸਕਦਾ ਹੈ। ਜਦੋਂ ਇੱਕ ਚੱਤਿਰ ਇਨਪੁੱਟ ਵਜੋਂ ਦੇਂਤਿਾ ਜਾਂਦਾ ਹੈ, GPT-4 ਚੱਤਿਰ ਦੀ ਸਮੱਗਰੀ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰ ਸਕਦਾ ਹੈ ਅਤੇ ਦ੍ਰਸ਼ਿਟੀਗਤ ਜਾਣਕਾਰੀ ਦਾ ਵਰਣਨ ਕਰਦਾ ਜਾਂ ਉਸ ਦਾ ਜਵਾਬ ਦੇਂਦਿਾ ਟੈਕਸਟ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ।
- **Google ਦਾ Gemini:** Gemini ਇੱਕ ਬਹੁ-ਮਾਧਅਿਮੀ ਮਾਡਲ ਹੈ ਜੋ ਟੈਕਸਟ, ਚੱਤਿਰ, ਅਤੇ ਵੀਡੀਓ ਨੂੰ ਸੰਭਾਲ ਸਕਦਾ ਹੈ। ਇਹ ਇੱਕ ਏਕੀਕ੍ਰਤਿ ਆਰਕੀਟੈਕਚਰ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ ਜੋ ਕਰਾਸ-ਮਾਧਅਿਮ ਸਮਝ ਅਤੇ ਉਤਪਾਦਨ ਦੀ ਆਗਆਿ ਦੇਂਦਿਾ ਹੈ, ਜਸਿ ਨਾਲ ਚੱਤਿਰ ਵਰਣਨ, ਵੀਡੀਓ ਸੰਖੇਪੀਕਰਨ, ਅਤੇ ਦ੍ਰਸ਼ਿਟੀਗਤ ਸਵਾਲ-ਜਵਾਬ ਵਰਗੇ ਕੰਮ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ।

- **DALL-E ਅਤੇ Stable Diffusion:** ਭਾਵੇਂ ਇਹ ਪਰੰਪਰਾਗਤ ਅਰਥਾਂ ਵਚਿ ਭਾਸ਼ਾ ਮਾਡਲ ਨਹੀਂ ਹਨ, ਇਹ ਮਾਡਲ ਟੈਕਸਟ ਵੇਰਵਆਿਂ ਤੋਂ ਚੱਤਿਰ ਤਆਿਰ ਕਰਕੇ ਬਹੁ-ਮਾਧਅਿਮੀ AI ਦੀ ਸ਼ਕਤੀ ਨੂੰ ਦਰਸਾਉਂਦੇ ਹਨ। ਇਹ ਵੱਖ-ਵੱਖ ਮਾਧਅਿਮਾਂ ਵਚਿਕਾਰ ਅਨੁਵਾਦ ਕਰ ਸਕਣ ਵਾਲੇ ਮਾਡਲਾਂ ਦੀ ਸੰਭਾਵਨਾ ਨੂੰ ਪ੍ਰਦਰਸ਼ਤਿ ਕਰਦੇ ਹਨ।

ਬਹੁ-ਮਾਧਅਿਮੀ ਮਾਡਲਾਂ ਦੇ ਲਾਭ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨਾਂ

ਬਹੁ-ਮਾਧਅਿਮੀ ਭਾਸ਼ਾ ਮਾਡਲ ਕਈ ਲਾਭ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ ਅਤੇ ਵਅਿਾਪਕ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਸੰਭਵ ਬਣਾਉਂਦੇ ਹਨ, ਜਸਿ ਵੱਚਿ ਸ਼ਾਮਲ ਹਨ:

- **ਵਧੀ ਹੋਈ ਸਮਝ:** ਕਈ ਮਾਧਅਿਮਾਂ ਤੋਂ ਜਾਣਕਾਰੀ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਕੇ, ਇਹ ਮਾਡਲ ਦੁਨੀਆ ਦੀ ਵਧੇਰੇ ਵਅਿਾਪਕ ਸਮਝ ਪ੍ਰਾਪਤ ਕਰ ਸਕਦੇ ਹਨ, ਜਵਿੇਂ ਕ ਿਮਨੁੱਖ ਵੱਖ-ਵੱਖ ਸੰਵੇਦੀ ਇਨਪੁੱਟ ਤੋਂ ਸੱਖਿਦੇ ਹਨ।
- **ਕਰਾਸ-ਮਾਧਅਿਮ ਉਤਪਾਦਨ:** ਬਹੁ-ਮਾਧਅਿਮੀ ਮਾਡਲ ਇੱਕ ਮਾਧਅਿਮ ਤੋਂ ਇਨਪੁੱਟ ਦੇ ਆਧਾਰ 'ਤੇ ਦੂਜੇ ਮਾਧਅਿਮ ਵੱਚਿ ਸਮੱਗਰੀ ਤਅਿਾਰ ਕਰ ਸਕਦੇ ਹਨ, ਜਵਿੇਂ ਕ ਿਟੈਕਸਟ ਵੇਰਵੇ ਤੋਂ ਚੱਤਿਰ ਬਣਾਉਣਾ ਜਾਂ ਲਖਿਤੀ ਲੇਖ ਤੋਂ ਵੀਡੀਓ ਸੰਖੇਪ ਤਅਿਾਰ ਕਰਨਾ।
- **ਪਹੁੰਚਯੋਗਤਾ:** ਬਹੁ-ਮਾਧਅਿਮੀ ਮਾਡਲ ਮਾਧਅਿਮਾਂ ਵਚਿਕਾਰ ਅਨੁਵਾਦ ਕਰਕੇ ਜਾਣਕਾਰੀ ਨੂੰ ਵਧੇਰੇ ਪਹੁੰਚਯੋਗ ਬਣਾ ਸਕਦੇ ਹਨ, ਜਵਿੇਂ ਕ ਿਦ੍ਰਸਿ਼ਟੀਹੀਣ ਉਪਭੋਗਤਾਵਾਂ ਲਈ ਚੱਤਿਰਾਂ ਦੇ ਟੈਕਸਟ ਵੇਰਵੇ ਤਅਿਾਰ ਕਰਨਾ ਜਾਂ ਲਖਿਤੀ ਸਮੱਗਰੀ ਦੇ ਆਡੀਓ ਵਰਜਨ ਬਣਾਉਣਾ।
- **ਰਚਨਾਤਮਕ ਐਪਲੀਕੇਸ਼ਨਾਂ:** ਬਹੁ-ਮਾਧਅਿਮ ਮਾਡਲਾਂ ਨੂੰ ਕਲਾ, ਸੰਗੀਤ, ਜਾਂ ਵੀਡੀਓ ਬਣਾਉਣ ਵਰਗੇ ਰਚਨਾਤਮਕ ਕੰਮਾਂ ਲਈ ਵਰਤਅਿਾ ਜਾ ਸਕਦਾ ਹੈ, ਜੋ ਕ ਿਟੈਕਸਟ ਪ੍ਰੌਂਮਪਟਸ 'ਤੇ ਆਧਾਰਤਿ ਹਨ, ਜੋ ਕਲਾਕਾਰਾਂ ਅਤੇ ਸਮੱਗਰੀ ਨਰਿਮਾਤਾਵਾਂ ਲਈ ਨਵੇਂ ਮੌਕੇ ਖੋਲ੍ਹਦੇ ਹਨ।

ਜਵਿੇਂ-ਜਵਿੇਂ ਬਹੁ-ਮਾਧਅਿਮ ਭਾਸ਼ਾ ਮਾਡਲ ਅੱਗੇ ਵਧਦੇ ਹਨ, ਇਹ ਸੰਭਾਵਤ ਤੌਰ 'ਤੇ AI-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦੇ ਵਕਿਾਸ ਵੱਚਿ ਵਧੇਰੇ ਮਹੱਤਵਪੂਰਨ ਭੂਮਕਿਾ ਨਭਿਾਉਣਗੇ ਜੋ ਕਈ ਮਾਧਅਿਮਾਂ ਵੱਚਿ ਸਮੱਗਰੀ ਨੂੰ ਸਮਝ ਅਤੇ ਤਅਿਾਰ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਮਨੁੱਖਾਂ ਅਤੇ AI ਸਸਿਟਮਾਂ ਵਚਿਕਾਰ ਵਧੇਰੇ ਕੁਦਰਤੀ ਅਤੇ ਸਹਜਿ ਗੱਲਬਾਤ ਨੂੰ ਸਮਰੱਥ ਬਣਾਏਗਾ, ਨਾਲ ਹੀ ਰਚਨਾਤਮਕ ਪ੍ਰਗਟਾਵੇ ਅਤੇ ਗਅਿਾਨ ਪ੍ਰਸਾਰ ਲਈ ਨਵੇਂ ਮੌਕੇ ਖੋਲ੍ਹੇਗਾ।

ਪ੍ਰਦਾਤਾ ਈਕੋਸਸਿਟਮ

ਐਪਲੀਕੇਸ਼ਨਾਂ ਵੱਚਿ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ (LLMs) ਨੂੰ ਸ਼ਾਮਲ ਕਰਨ ਦੀ ਗੱਲ ਆਉਂਦੀ ਹੈ, ਤਾਂ ਤੁਹਾਡੇ ਕੋਲ ਚੁਣਨ ਲਈ ਵਕਿਲਪਾਂ ਦੀ ਵਧਦੀ ਰੇਂਜ ਹੈ। ਹਰ ਪ੍ਰਮੁੱਖ LLM ਪ੍ਰਦਾਤਾ, ਜਵਿੇਂ ਕ ਿOpenAI, Anthropic, Google, ਅਤੇ Cohere, ਆਪਣੇ ਮਾਡਲਾਂ, APIs, ਅਤੇ ਟੂਲਾਂ ਦਾ ਈਕੋਸਸਿਟਮ ਪੇਸ਼ ਕਰਦਾ ਹੈ। ਸਹੀ ਪ੍ਰਦਾਤਾ ਚੁਣਨ ਵੱਚਿ ਕਈ ਕਾਰਕਾਂ 'ਤੇ ਵਚਿਾਰ ਕਰਨਾ ਸ਼ਾਮਲ ਹੈ, ਜਸਿ ਵੱਚਿ ਕੀਮਤ, ਪ੍ਰਦਰਸ਼ਨ, ਸਮੱਗਰੀ ਫਲਿਟਰੰਗ, ਡੇਟਾ ਗੋਪਨੀਯਤਾ, ਅਤੇ ਕਸਟਮਾਈਜ਼ੇਸ਼ਨ ਵਕਿਲਪ ਸ਼ਾਮਲ ਹਨ।

OpenAI

OpenAI LLMs ਦੇ ਸਭ ਤੋਂ ਜਾਣੇ-ਪਛਾਣੇ ਪ੍ਰਦਾਤਾਵਾਂ ਵਿੱਚੋਂ ਇੱਕ ਹੈ, ਜਸਿਦੀ GPT ਸੀਰੀਜ਼ (GPT-3, GPT-4) ਵੱਖ-ਵੱਖ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਵਆਿਪਕ ਤੌਰ 'ਤੇ ਵਰਤੀ ਜਾਂਦੀ ਹੈ। OpenAI ਇੱਕ ਵਰਤੋਂਕਾਰ-ਅਨੁਕੂਲ API ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਜੋ ਤੁਹਾਨੂੰ ਆਸਾਨੀ ਨਾਲ ਉਹਨਾਂ ਦੇ ਮਾਡਲਾਂ ਨੂੰ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਦੀ ਆਗਆਿ ਦੰਦਿਾ ਹੈ। ਉਹ ਐਂਟਰੀ-ਲੈਵਲ Ada ਮਾਡਲ ਤੋਂ ਲੈ ਕੇ ਸ਼ਕਤੀਸ਼ਾਲੀ Davinci ਮਾਡਲ ਤੱਕ, ਵੱਖ-ਵੱਖ ਸਮਰੱਥਾਵਾਂ ਅਤੇ ਕੀਮਤ ਬੰਦ੍ਹਿਆਂ ਵਾਲੇ ਮਾਡਲਾਂ ਦੀ ਇੱਕ ਰੇਂਜ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

OpenAI ਦੇ ਈਕੋਸਸਿਟਮ ਵਿੱਚ OpenAI Playground ਵਰਗੇ ਟੂਲ ਵੀ ਸ਼ਾਮਲ ਹਨ, ਜੋ ਤੁਹਾਨੂੰ ਪ੍ਰੌਮਪਟਸ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਨ ਅਤੇ ਖਾਸ ਵਰਤੋਂ ਦੇ ਮਾਮਲਆਿਂ ਲਈ ਮਾਡਲਾਂ ਨੂੰ ਫਾਈਨ-ਟਊਿਨ ਕਰਨ ਦੀ ਆਗਆਿ ਦੰਦਿੇ ਹਨ। ਉਹ ਅਣਉਚਤਿ ਜਾਂ ਨੁਕਸਾਨਦੇਹ ਸਮੱਗਰੀ ਦੀ ਉਤਪਤੀ ਨੂੰ ਰੋਕਣ ਵਿੱਚ ਮਦਦ ਲਈ ਸਮੱਗਰੀ ਫਲਿਟਰਿੰਗ ਵਕਿਲਪ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

OpenAI ਦੇ ਮਾਡਲਾਂ ਨੂੰ ਸੰਧਿੇ ਵਰਤਣ ਲਈ, ਮੈਂ Alex Rudall ਦੀ ruby-openai ਲਾਇਬ੍ਰੇਰੀ 'ਤੇ ਨਰਿਭਰ ਕਰਦਾ ਹਾਂ।

Anthropic

Anthropic LLM ਖੇਤਰ ਵਿੱਚ ਇੱਕ ਹੋਰ ਪ੍ਰਮੁੱਖ ਖਡਿਾਰੀ ਹੈ, ਜਸਿਦੇ Claude ਮਾਡਲ ਮਜ਼ਬੂਤ ਪ੍ਰਦਰਸ਼ਨ ਅਤੇ ਨੈਤਕਿ ਵਚਿਾਰਾਂ ਲਈ ਹਰਮਨਪਆਿਰੇ ਹੋ ਰਹੇ ਹਨ। Anthropic ਸੁਰੱਖਅਿਤ ਅਤੇ ਜ਼ੰਮਿੇਵਾਰ AI ਸਸਿਟਮਾਂ ਦੇ ਵਕਿਾਸ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰਦਾ ਹੈ, ਸਮੱਗਰੀ ਫਲਿਟਰਿੰਗ ਅਤੇ ਨੁਕਸਾਨਦੇਹ ਆਉਟਪੁੱਟ ਤੋਂ ਬਚਣ 'ਤੇ ਜੋਰ ਦੰਦਿਾ ਹੈ।

Anthropic ਦੇ ਈਕੋਸਸਿਟਮ ਵਿੱਚ Claude API ਸ਼ਾਮਲ ਹੈ, ਜੋ ਤੁਹਾਨੂੰ ਮਾਡਲ ਨੂੰ ਆਪਣੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਦੀ ਆਗਆਿ ਦੰਦਿਾ ਹੈ, ਨਾਲ ਹੀ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਅਤੇ ਫਾਈਨ-ਟਊਿਨਿੰਗ ਲਈ ਟੂਲ ਵੀ। ਉਹ Claude Instant ਮਾਡਲ ਵੀ ਪੇਸ਼ ਕਰਦੇ ਹਨ, ਜੋ ਵਧੇਰੇ ਅਪ-ਟੂ-ਡੇਟ ਅਤੇ ਤੱਥਾਤਮਕ ਜਵਾਬਾਂ ਲਈ ਵੈੱਬ ਖੋਜ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਦਾ ਹੈ।

Anthropic ਦੇ ਮਾਡਲਾਂ ਨੂੰ ਸੰਧਿੇ ਵਰਤਦੇ ਸਮੇਂ, ਮੈਂ Alex Rudall ਦੀ anthrophic ਲਾਇਬ੍ਰੇਰੀ 'ਤੇ ਨਰਿਭਰ ਕਰਦਾ ਹਾਂ।

Google

Google ਨੇ ਕਈ ਸ਼ਕਤੀਸ਼ਾਲੀ LLMs ਵਿਕਸਿਤ ਕੀਤੇ ਹਨ, ਜਿਨ੍ਹਾਂ ਵਿੱਚ Gemini, BERT, T5, ਅਤੇ PaLM ਸ਼ਾਮਲ ਹਨ। ਇਹ ਮਾਡਲ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਪ੍ਰੋਸੈਸਿੰਗ ਦੇ ਵੱਖ-ਵੱਖ ਕੰਮਾਂ 'ਤੇ ਆਪਣੀ ਮਜ਼ਬੂਤ ਕਾਰਗੁਜ਼ਾਰੀ ਲਈ ਜਾਣੇ ਜਾਂਦੇ ਹਨ। Google ਦੇ ਈਕੋਸਿਸਟਮ ਵਿੱਚ TensorFlow ਅਤੇ Keras ਲਾਇਬ੍ਰੇਰੀਆਂ ਸ਼ਾਮਲ ਹਨ, ਜੋ ਮਸ਼ੀਨੀ ਸਿੱਖਿਆ ਮਾਡਲਾਂ ਨੂੰ ਬਣਾਉਣ ਅਤੇ ਸਿਖਲਾਈ ਦੇਣ ਲਈ ਟੂਲਜ਼ ਅਤੇ ਫਰੇਮਵਰਕ ਪ੍ਰਦਾਨ ਕਰਦੀਆਂ ਹਨ।

Google ਇੱਕ Cloud AI Platform ਵੀ ਪੇਸ਼ ਕਰਦਾ ਹੈ, ਜੋ ਤੁਹਾਨੂੰ ਕਲਾਉਡ ਵਿੱਚ ਉਨ੍ਹਾਂ ਦੇ ਮਾਡਲਾਂ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਤੈਨਾਤ ਅਤੇ ਸਕੇਲ ਕਰਨ ਦੀ ਸਹੂਲਤ ਦਿੰਦਾ ਹੈ। ਉਹ ਭਾਵਨਾ ਵਿਸ਼ਲੇਸ਼ਣ, ਇਕਾਈ ਪਛਾਣ, ਅਤੇ ਅਨੁਵਾਦ ਵਰਗੇ ਕੰਮਾਂ ਲਈ ਪਹਿਲਾਂ-ਸਿਖਲਾਈ ਪ੍ਰਾਪਤ ਮਾਡਲਾਂ ਅਤੇ APIs ਦੀ ਇੱਕ ਲੜੀ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

Meta

Meta, ਜੋ ਪਹਿਲਾਂ Facebook ਦੇ ਨਾਮ ਨਾਲ ਜਾਣਿਆ ਜਾਂਦਾ ਸੀ, ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੇ ਵਿਕਾਸ ਵਿੱਚ ਡੂੰਘਾਈ ਨਾਲ ਨਿਵੇਸ਼ ਕਰ ਰਿਹਾ ਹੈ, ਜਸਿ ਨੂੰ LLaMA ਅਤੇ OPT ਵਰਗੇ ਮਾਡਲਾਂ ਦੀ ਰਿਲੀਜ਼ ਨਾਲ ਉਜਾਗਰ ਕੀਤਾ ਗਿਆ ਹੈ। ਇਹ ਮਾਡਲ ਵੱਖ-ਵੱਖ ਭਾਸ਼ਾਈ ਕੰਮਾਂ ਵਿੱਚ ਆਪਣੀ ਮਜ਼ਬੂਤ ਕਾਰਗੁਜ਼ਾਰੀ ਲਈ ਉੱਭਰਦੇ ਹਨ ਅਤੇ ਮੁੱਖ ਤੌਰ 'ਤੇ ਓਪਨ-ਸੋਰਸ ਚੈਨਲਾਂ ਰਾਹੀਂ ਉਪਲਬਧ ਕਰਵਾਏ ਜਾਂਦੇ ਹਨ, ਜੋ ਖੋਜ ਅਤੇ ਕਮਿਊਨਿਟੀ ਸਹਿਯੋਗ ਪ੍ਰਤੀ Meta ਦੀ ਵਚਨਬੱਧਤਾ ਦੀ ਸਹਾਇਤਾ ਕਰਦੇ ਹਨ।

Meta ਦਾ ਈਕੋਸਿਸਟਮ ਮੁੱਖ ਤੌਰ 'ਤੇ PyTorch ਦੇ ਆਲੇ-ਦੁਆਲੇ ਬਣਿਆ ਹੈ, ਜੋ ਇੱਕ ਓਪਨ-ਸੋਰਸ ਮਸ਼ੀਨੀ ਸਿੱਖਿਆ ਲਾਇਬ੍ਰੇਰੀ ਹੈ ਜੋ ਆਪਣੀਆਂ ਡਾਇਨਾਮਿਕ ਕੰਪਿਊਟੇਸ਼ਨਲ ਸਮਰੱਥਾਵਾਂ ਅਤੇ ਲਚਕਤਾ ਲਈ ਪਸੰਦ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਜੋ ਨਵੀਨਤਾਕਾਰੀ AI ਖੋਜ ਅਤੇ ਵਿਕਾਸ ਨੂੰ ਸੌਖਾ ਬਣਾਉਂਦੀ ਹੈ।

ਆਪਣੀਆਂ ਤਕਨੀਕੀ ਪੇਸ਼ਕਸ਼ਾਂ ਤੋਂ ਇਲਾਵਾ, Meta ਨੈਤਿਕ AI ਵਿਕਾਸ 'ਤੇ ਜ਼ੋਰ ਦਿੰਦਾ ਹੈ। ਉਹ ਮਜ਼ਬੂਤ ਸਮੱਗਰੀ ਫਿਲਟਰਿੰਗ ਲਾਗੂ ਕਰਦੇ ਹਨ ਅਤੇ ਪੱਖਪਾਤਾਂ ਨੂੰ ਘਟਾਉਣ 'ਤੇ ਧਿਆਨ ਕੇਂਦਰਿਤ ਕਰਦੇ ਹਨ, ਜੋ AI ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਸੁਰੱਖਿਆ ਅਤੇ ਜ਼ਿੰਮੇਵਾਰੀ ਦੇ ਉਨ੍ਹਾਂ ਦੇ ਵਿਆਪਕ ਟੀਚਿਆਂ ਨਾਲ ਮੇਲ ਖਾਂਦਾ ਹੈ।

Cohere

Cohere LLM ਖੇਤਰ ਵਿੱਚ ਇੱਕ ਨਵਾਂ ਪ੍ਰਵੇਸ਼ ਕਰਨ ਵਾਲਾ ਹੈ, ਜੋ ਪ੍ਰਤੀਯੋਗੀਆਂ ਦੇ ਮੁਕਾਬਲੇ LLMs ਨੂੰ ਵਧੇਰੇ ਪਹੁੰਚਯੋਗ ਅਤੇ ਵਰਤਣ ਵਿੱਚ ਆਸਾਨ ਬਣਾਉਣ 'ਤੇ ਧਿਆਨ ਕੇਂਦਰਿਤ ਕਰ ਰਿਹਾ ਹੈ। ਉਨ੍ਹਾਂ ਦੇ ਈਕੋਸਿਸਟਮ

ਵੱਚਿ Cohere API ਸ਼ਾਮਲ ਹੈ, ਜੋ ਟੈਕਸਟ ਜਨਰੇਸ਼ਨ, ਵਰਗੀਕਰਨ, ਅਤੇ ਸੰਖੇਪ ਵਰਗੇ ਕੰਮਾਂ ਲਈ ਪਹਲਿਾਂ-ਸਖਿਲਾਈ ਪ੍ਰਾਪਤ ਮਾਡਲਾਂ ਦੀ ਇੱਕ ਲੜੀ ਤੱਕ ਪਹੁੰਚ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।

Cohere ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰੰਗਿ, ਫਾਈਨ-ਟਊਿਨੰਗਿ, ਅਤੇ ਸਮੱਗਰੀ ਫਲਿਟਰੰਗਿ ਲਈ ਟੂਲਜ਼ ਵੀ ਪੇਸ਼ ਕਰਦਾ ਹੈ। ਉਹ ਐਨਕ੍ਰਪਿਟਡ ਡਾਟਾ ਸਟੋਰੇਜ ਅਤੇ ਐਕਸੈਸ ਕੰਟਰੋਲ ਵਰਗੀਆਂ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਨਾਲ ਡਾਟਾ ਗੋਪਨੀਯਤਾ ਅਤੇ ਸੁਰੱਖਿਆ ’ਤੇ ਜ਼ੋਰ ਦੇਂਦੇ ਹਨ।

Ollama

Ollama ਇੱਕ ਸਵੈ-ਹੋਸਟਡ ਪਲੇਟਫਾਰਮ ਹੈ ਜੋ ਉਪਭੋਗਤਾਵਾਂ ਨੂੰ ਆਪਣੀਆਂ ਮਸ਼ੀਨਾਂ ’ਤੇ ਸਥਾਨਕ ਤੌਰ ’ਤੇ ਵੱਖ-ਵੱਖ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ (LLMs) ਦਾ ਪ੍ਰਬੰਧਨ ਅਤੇ ਤੈਨਾਤੀ ਕਰਨ ਦੀ ਆਗਿਆ ਦੇਂਦਾ ਹੈ, ਜੋ ਉਨ੍ਹਾਂ ਨੂੰ ਬਾਹਰੀ ਕਲਾਉਡ ਸੇਵਾਵਾਂ ’ਤੇ ਨਰਿਭਰ ਕੀਤੇ ਬਨਿਾਂ ਆਪਣੇ AI ਮਾਡਲਾਂ ’ਤੇ ਪੂਰਾ ਨਯਿੰਤਰਣ ਦੇਂਦਾ ਹੈ। ਇਹ ਸੈੱਟਅੱਪ ਉਨ੍ਹਾਂ ਲੋਕਾਂ ਲਈ ਆਦਰਸ਼ ਹੈ ਜੋ ਡਾਟਾ ਗੋਪਨੀਯਤਾ ਨੂੰ ਤਰਜੀਹ ਦੇਂਦੇ ਹਨ ਅਤੇ ਆਪਣੇ AI ਓਪਰੇਸ਼ਨਾਂ ਨੂੰ ਇਨ-ਹਾਊਸ ਸੰਭਾਲਣਾ ਚਾਹੁੰਦੇ ਹਨ।

ਪਲੇਟਫਾਰਮ ਕਈ ਤਰ੍ਹਾਂ ਦੇ ਮਾਡਲਾਂ ਦਾ ਸਮਰਥਨ ਕਰਦਾ ਹੈ, ਜਸਿ ਵੱਚਿ Llama, Phi, Gemma, ਅਤੇ Mistral ਦੇ ਵੱਖ-ਵੱਖ ਵਰਜਨ ਸ਼ਾਮਲ ਹਨ, ਜੋ ਆਕਾਰ ਅਤੇ ਕੰਪਊਿਟੇਸ਼ਨਲ ਲੋੜਾਂ ਵੱਚਿ ਵੱਖ-ਵੱਖ ਹੁੰਦੇ ਹਨ। Ollama ਇਹਨਾਂ ਮਾਡਲਾਂ ਨੂੰ `ollama run <model_name>` ਵਰਗੇ ਸਧਾਰਨ ਕਮਾਂਡਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਕਮਾਂਡ ਲਾਈਨ ਤੋਂ ਸੱਧਿਾ ਡਾਊਨਲੋਡ ਅਤੇ ਚਲਾਉਣਾ ਆਸਾਨ ਬਣਾਉਂਦਾ ਹੈ, ਅਤੇ ਇਹ macOS, Linux, ਅਤੇ Windows ਸਮੇਤ ਵੱਖ-ਵੱਖ ਓਪਰੇਟੰਗਿ ਸਸਿਟਮਾਂ ’ਤੇ ਕੰਮ ਕਰਨ ਲਈ ਤਆਿਰ ਕੀਤਾ ਗਆਿ ਹੈ।

ਡਵਿੈਲਪਰਾਂ ਲਈ ਜੋ ਰਮਿੋਟ API ਦੀ ਵਰਤੋਂ ਕੀਤੇ ਬਨਿਾਂ ਆਪਣੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵੱਚਿ ਓਪਨ-ਸੋਰਸ ਮਾਡਲਾਂ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਨ, Ollama ਕੰਟੇਨਰ ਪ੍ਰਬੰਧਨ ਟੂਲਾਂ ਵਾਂਗ ਮਾਡਲ ਲਾਈਫਸਾਈਕਲ ਦੇ ਪ੍ਰਬੰਧਨ ਲਈ CLI ਪੇਸ਼ ਕਰਦਾ ਹੈ। ਇਹ ਕਸਟਮ ਕੌਨਫਗਿਰੇਸ਼ਨਾਂ ਅਤੇ ਪ੍ਰੌਂਪਟਸ ਦਾ ਵੀ ਸਮਰਥਨ ਕਰਦਾ ਹੈ, ਜੋ ਖਾਸ ਲੋੜਾਂ ਜਾਂ ਵਰਤੋਂ ਦੇ ਕੇਸਾਂ ਲਈ ਮਾਡਲਾਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣ ਲਈ ਉੱਚ ਪੱਧਰ ਦੀ ਕਸਟਮਾਈਜ਼ੇਸ਼ਨ ਦੀ ਆਗਿਆ ਦੇਂਦਾ ਹੈ।

Ollama ਖਾਸ ਤੌਰ ’ਤੇ ਤਕਨੀਕੀ-ਜਾਣਕਾਰ ਯੂਜ਼ਰਾਂ ਅਤੇ ਡਵਿੈਲਪਰਾਂ ਲਈ ਢੁਕਵਾਂ ਹੈ ਕਉਂਕਿ ਇਸਦਾ ਕਮਾਂਡ-ਲਾਈਨ ਇੰਟਰਫੇਸ ਅਤੇ AI ਮਾਡਲਾਂ ਦੇ ਪ੍ਰਬੰਧਨ ਅਤੇ ਡਪਿਲੌਇਮੈਂਟ ਵੱਚਿ ਲਚਕਤਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਇਹ ਇਸਨੂੰ ਉਹਨਾਂ ਕਾਰੋਬਾਰਾਂ ਅਤੇ ਵਅਿਕਤੀਆਂ ਲਈ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਟੂਲ ਬਣਾਉਂਦਾ ਹੈ ਜਨ੍ਹਿਾਂ ਨੂੰ ਸੁਰੱਖਿਆ ਅਤੇ ਨਯਿੰਤਰਣ ਨਾਲ ਸਮਝੌਤਾ ਕੀਤੇ ਬਨਿਾਂ ਮਜ਼ਬੂਤ AI ਸਮਰੱਥਾਵਾਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

ਬਹੁ-ਮਾਡਲ ਪਲੇਟਫਾਰਮ

ਇਸ ਤੋਂ ਇਲਾਵਾ, ਕੁਝ ਪ੍ਰਦਾਤਾ ਹਨ ਜੋ ਵੱਖ-ਵੱਖ ਓਪਨ-ਸੋਰਸ ਮਾਡਲਾਂ ਦੀ ਹੋਸਟਿੰਗ ਕਰਦੇ ਹਨ, ਜਵਿੇਂ ਕਿ Together.ai ਅਤੇ Groq। ਇਹ ਪਲੇਟਫਾਰਮ ਲਚਕਤਾ ਅਤੇ ਕਸਟਮਾਈਜ਼ੇਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ, ਜੋ ਤੁਹਾਨੂੰ ਆਪਣੀਆਂ ਖਾਸ ਲੋੜਾਂ ਦੇ ਅਨੁਸਾਰ ਓਪਨ-ਸੋਰਸ ਮਾਡਲਾਂ ਨੂੰ ਚਲਾਉਣ ਅਤੇ, ਕੁਝ ਮਾਮਲਿਆਂ ਵਿੱਚ, ਫਾਈਨ-ਟਿਊਨ ਵੀ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦੇ ਹਨ। ਉਦਾਹਰਣ ਲਈ, Together.ai ਓਪਨ-ਸੋਰਸ LLMs ਦੀ ਇੱਕ ਰੇਂਜ ਤੱਕ ਪਹੁੰਚ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ, ਜੋ ਯੂਜ਼ਰਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਮਾਡਲਾਂ ਅਤੇ ਕੌਨਫਗਿਰੇਸ਼ਨਾਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ। Groq ਅਲਟਰਾ ਹਾਈ-ਪਰਫਾਰਮੈਂਸ ਕੰਪਲੀਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰਨ ’ਤੇ ਕੇਂਦਰਤਿ ਹੈ ਜੋ ਇਸ ਕਤਿਾਬ ਦੇ ਲਖਿਣ ਦੇ ਸਮੇਂ ਲਗਭਗ ਜਾਦੂਈ ਲੱਗਦਾ ਹੈ

LLM ਪ੍ਰਦਾਤਾ ਦੀ ਚੋਣ ਕਰਨਾ

LLM ਪ੍ਰਦਾਤਾ ਦੀ ਚੋਣ ਕਰਦੇ ਸਮੇਂ, ਤੁਹਾਨੂੰ ਇਹਨਾਂ ਕਾਰਕਾਂ ’ਤੇ ਵਚਿਾਰ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ:

- **ਕੀਮਤ:** ਵੱਖ-ਵੱਖ ਪ੍ਰਦਾਤਾ ਵੱਖ-ਵੱਖ ਕੀਮਤ ਮਾਡਲ ਪੇਸ਼ ਕਰਦੇ ਹਨ, ਜੋ ਪੇ-ਪਰ-ਯੂਜ਼ ਤੋਂ ਲੈ ਕੇ ਸਬਸਕ੍ਰਪਿਸ਼ਨ-ਆਧਾਰਤਿ ਯੋਜਨਾਵਾਂ ਤੱਕ ਹੁੰਦੇ ਹਨ। ਪ੍ਰਦਾਤਾ ਦੀ ਚੋਣ ਕਰਦੇ ਸਮੇਂ ਅਨੁਮਾਨਤਿ ਵਰਤੋਂ ਅਤੇ ਬਜਟ ’ਤੇ ਵਚਿਾਰ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ।
- **ਪ੍ਰਦਰਸ਼ਨ:** LLMs ਦਾ ਪ੍ਰਦਰਸ਼ਨ ਪ੍ਰਦਾਤਾਵਾਂ ਵਚਿਕਾਰ ਕਾਫ਼ੀ ਵੱਖਰਾ ਹੋ ਸਕਦਾ ਹੈ, ਇਸ ਲਈ ਫੈਸਲਾ ਲੈਣ ਤੋਂ ਪਹਲਿਾਂ ਖਾਸ ਵਰਤੋਂ ਦੇ ਕੇਸਾਂ ’ਤੇ ਮਾਡਲਾਂ ਦਾ ਬੈਂਚਮਾਰਕ ਅਤੇ ਟੈਸਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ।
- **ਸਮੱਗਰੀ ਫਲਿਟਰਿੰਗ:** ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਆਧਾਰ ’ਤੇ, ਸਮੱਗਰੀ ਫਲਿਟਰਿੰਗ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਵਚਿਾਰ ਹੋ ਸਕਦਾ ਹੈ। ਕੁਝ ਪ੍ਰਦਾਤਾ ਦੂਜਿਆਂ ਨਾਲੋਂ ਵਧੇਰੇ ਮਜ਼ਬੂਤ ਸਮੱਗਰੀ ਫਲਿਟਰਿੰਗ ਵਕਿਲਪ ਪੇਸ਼ ਕਰਦੇ ਹਨ।
- **ਡਾਟਾ ਗੋਪਨੀਯਤਾ:** ਜੇਕਰ ਐਪਲੀਕੇਸ਼ਨ ਸੰਵੇਦਨਸ਼ੀਲ ਯੂਜ਼ਰ ਡਾਟਾ ਨੂੰ ਸੰਭਾਲਦੀ ਹੈ, ਤਾਂ ਮਜ਼ਬੂਤ ਡਾਟਾ ਗੋਪਨੀਯਤਾ ਅਤੇ ਸੁਰੱਖਿਆ ਅਭਿਆਸਾਂ ਵਾਲੇ ਪ੍ਰਦਾਤਾ ਦੀ ਚੋਣ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ।
- **ਅਨੁਕੂਲਤਾ:** ਕੁਝ ਪ੍ਰਦਾਤਾ ਖਾਸ ਵਰਤੋਂ ਦੇ ਕੇਸਾਂ ਲਈ ਮਾਡਲਾਂ ਨੂੰ ਫਾਈਨ-ਟਿਊਨ ਅਤੇ ਅਨੁਕੂਲ ਬਣਾਉਣ ਦੇ ਮਾਮਲੇ ਵਿੱਚ ਵਧੇਰੇ ਲਚਕਤਾ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

ਅੰਤ ਵਿੱਚ, ਐੱਲਐੱਲਐੱਮ ਪ੍ਰਦਾਤਾ ਦੀ ਚੋਣ ਐਪਲੀਕੇਸ਼ਨ ਦੀਆਂ ਵਸ਼ਿੇਸ਼ ਲੋੜਾਂ ਅਤੇ ਸੀਮਾਵਾਂ ’ਤੇ ਨਰਿਭਰ ਕਰਦੀ ਹੈ। ਵਕਿਲਪਾਂ ਦਾ ਧਿਆਨ ਨਾਲ ਮੁਲਾਂਕਣ ਕਰਕੇ ਅਤੇ ਕੀਮਤ, ਪ੍ਰਦਰਸ਼ਨ, ਅਤੇ ਡੇਟਾ ਗੋਪਨੀਯਤਾ ਵਰਗੇ ਕਾਰਕਾਂ

’ਤੇ ਵਿਚਾਰ ਕਰਕੇ, ਤੁਸੀਂ ਉਹ ਪ੍ਰਦਾਤਾ ਚੁਣ ਸਕਦੇ ਹੋ ਜੋ ਤੁਹਾਡੀਆਂ ਲੋੜਾਂ ਨੂੰ ਸਭ ਤੋਂ ਵਧੀਆ ਢੰਗ ਨਾਲ ਪੂਰਾ ਕਰਦਾ ਹੈ।

ਇਹ ਨੋਟ ਕਰਨਾ ਵੀ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਐੱਲਐੱਲਐੱਮ ਦਾ ਖੇਤਰ ਲਗਾਤਾਰ ਵਿਕਸਿਤ ਹੋ ਰਿਹਾ ਹੈ, ਜੱਥੇ ਨਵੇਂ ਪ੍ਰਦਾਤਾ ਅਤੇ ਮਾਡਲ ਨਿਯਮਿਤ ਤੌਰ ’ਤੇ ਸਾਹਮਣੇ ਆ ਰਹੇ ਹਨ। ਤੁਹਾਨੂੰ ਨਵੀਨਤਮ ਵਿਕਾਸ ਤੋਂ ਜਾਣੂ ਰਹਣਿਾ ਚਾਹੀਦਾ ਹੈ ਅਤੇ ਨਵੇਂ ਵਿਕਲਪਾਂ ਦੀ ਪੜਚੋਲ ਕਰਨ ਲਈ ਖੁੱਲ੍ਹੇ ਰਹਣਿਾ ਚਾਹੀਦਾ ਹੈ ਜਦੋਂ ਉਹ ਉਪਲਬਧ ਹੋਣ।

OpenRouter

ਇਸ ਕਤਿਾਬ ਵੱਚਿ ਮੈਂ ਪੂਰੀ ਤਰ੍ਹਾਂ OpenRouter ’ਤੇ ਆਪਣੇ ਪਸੰਦੀਦਾ API ਪ੍ਰਦਾਤਾ ਵਜੋਂ ਨਰਿਭਰ ਕਰਾਂਗਾ। ਕਾਰਨ ਸਧਾਰਨ ਹੈ: ਇਹ ਸਾਰੇ ਸਭ ਤੋਂ ਪ੍ਰਸਧਿ ਵਪਾਰਕ ਅਤੇ ਓਪਨ-ਸੋਰਸ ਮਾਡਲਾਂ ਲਈ ਇੱਕ ਵਨ-ਸਟਾਪ ਸ਼ਾਪ ਹੈ। ਜੇਕਰ ਤੁਸੀਂ ਕੁਝ AI ਕੋਡਿੰਗ ਨਾਲ ਆਪਣੇ ਹੱਥ ਰੰਦੇ ਕਰਨ ਲਈ ਉਤਸੁਕ ਹੋ, ਤਾਂ ਸ਼ੁਰੂ ਕਰਨ ਲਈ ਸਭ ਤੋਂ ਵਧੀਆ ਥਾਵਾਂ ਵੱਚਿੋਂ ਇੱਕ ਮੇਰੀ OpenRouter Ruby Library ਹੈ।

ਪ੍ਰਦਰਸ਼ਨ ਬਾਰੇ ਸੋਚਣਾ

ਐਪਲੀਕੇਸ਼ਨਾਂ ਵੱਚਿ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਦੇ ਸਮੇਂ, ਪ੍ਰਦਰਸ਼ਨ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਵਚਿਾਰ ਹੈ। ਭਾਸ਼ਾ ਮਾਡਲ ਦੇ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਇਸਦੀ ਦੇਰੀ (ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਵੱਚਿ ਲੱਗਣ ਵਾਲਾ ਸਮਾਂ) ਅਤੇ ਥਰੂਪੁੱਟ (ਪ੍ਰਤੀ ਸਮਾਂ ਇਕਾਈ ਵੱਚਿ ਸੰਭਾਲੇ ਜਾ ਸਕਣ ਵਾਲੀਆਂ ਬੇਨਤੀਆਂ ਦੀ ਸੰਖਆਿ) ਦੇ ਰੂਪ ਵੱਚਿ ਮਾਪਆਿ ਜਾ ਸਕਦਾ ਹੈ।

ਪਹਲਿੇ ਟੋਕਨ ਤੱਕ ਦਾ ਸਮਾਂ (TTFT) ਇੱਕ ਹੋਰ ਜ਼ਰੂਰੀ ਪ੍ਰਦਰਸ਼ਨ ਮੈਟ੍ਰਕਿ ਹੈ, ਜੋ ਖਾਸ ਤੌਰ ’ਤੇ ਚੈਟਬੋਟਾਂ ਅਤੇ ਇੰਟਰਐਕਟਵਿ, ਰੀਅਲ-ਟਾਈਮ ਜਵਾਬਾਂ ਦੀ ਲੋੜ ਵਾਲੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਢੁਕਵਾਂ ਹੈ। TTFT ਉਪਭੋਗਤਾ ਦੀ ਬੇਨਤੀ ਪ੍ਰਾਪਤ ਹੋਣ ਦੇ ਪਲ ਤੋਂ ਜਵਾਬ ਦੇ ਪਹਲਿੇ ਸ਼ਬਦ (ਜਾਂ ਟੋਕਨ) ਦੇ ਤਆਿਰ ਹੋਣ ਤੱਕ ਦੀ ਦੇਰੀ ਨੂੰ ਮਾਪਦਾ ਹੈ। ਇਹ ਮੈਟ੍ਰਕਿ ਇੱਕ ਨਰਿਵਘਿਨ ਅਤੇ ਆਕਰਸ਼ਕ ਉਪਭੋਗਤਾ ਅਨੁਭਵ ਨੂੰ ਬਣਾਈ ਰੱਖਣ ਲਈ ਮਹੱਤਵਪੂਰਨ ਹੈ, ਕਿਉਂਕਿ ਦੇਰੀ ਵਾਲੇ ਜਵਾਬ ਉਪਭੋਗਤਾ ਦੀ ਨਰਿਾਸ਼ਾ ਅਤੇ ਅਸੰਲਗਨਤਾ ਦਾ ਕਾਰਨ ਬਣ ਸਕਦੇ ਹਨ।

ਇਹ ਪ੍ਰਦਰਸ਼ਨ ਮੈਟ੍ਰਕਿਸ ਉਪਭੋਗਤਾ ਅਨੁਭਵ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਸਕੇਲੇਬਲਿਟੀ ’ਤੇ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦੇ ਹਨ।

ਕਈ ਕਾਰਕ ਹਨ ਜੋ ਭਾਸ਼ਾ ਮਾਡਲ ਦੇ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਪ੍ਰਭਾਵਤਿ ਕਰ ਸਕਦੇ ਹਨ, ਜਸਿ ਵੱਚਿ ਸ਼ਾਮਲ ਹਨ:

ਪੈਰਾਮੀਟਰ ਗਣਿਤੀ: ਵੱਧ ਪੈਰਾਮੀਟਰਾਂ ਵਾਲੇ ਵੱਡੇ ਮਾਡਲਾਂ ਨੂੰ ਆਮ ਤੌਰ 'ਤੇ ਵੱਧ ਕੰਪਊਟੇਸ਼ਨਲ ਸਰੋਤਾਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ ਅਤੇ ਛੋਟੇ ਮਾਡਲਾਂ ਦੇ ਮੁਕਾਬਲੇ ਵੱਧ ਦੇਰੀ ਅਤੇ ਘੱਟ ਥਰੂਪੁੱਟ ਹੋ ਸਕਦੀ ਹੈ।

ਹਾਰਡਵੇਅਰ: ਭਾਸ਼ਾ ਮਾਡਲ ਦਾ ਪ੍ਰਦਰਸ਼ਨ ਇਸ ਨੂੰ ਚਲਾਉਣ ਵਾਲੇ ਹਾਰਡਵੇਅਰ ਦੇ ਆਧਾਰ 'ਤੇ ਕਾਫ਼ੀ ਵੱਖਰਾ ਹੋ ਸਕਦਾ ਹੈ। ਕਲਾਉਡ ਪ੍ਰਦਾਤਾ ਮਸ਼ੀਨ ਲਰਨਿੰਗ ਵਰਕਲੋਡ ਲਈ ਅਨੁਕੂਲਤਿ GPU ਅਤੇ TPU ਇੰਸਟੈਂਸ ਪੇਸ਼ ਕਰਦੇ ਹਨ, ਜੋ ਮਾਡਲ ਇਨਫਰੈਂਸ ਨੂੰ ਕਾਫ਼ੀ ਤੇਜ਼ ਕਰ ਸਕਦੇ ਹਨ।

OpenRouter ਦੀਆਂ ਵਧੀਆ ਗੱਲਾਂ ਵਿੱਚੋਂ ਇੱਕ ਇਹ ਹੈ ਕਿ ਇਸ ਦੁਆਰਾ ਪੇਸ਼ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਬਹੁਤ ਸਾਰੇ ਮਾਡਲਾਂ ਲਈ, ਤੁਹਾਨੂੰ ਵੱਖ-ਵੱਖ ਪ੍ਰਦਰਸ਼ਨ ਪ੍ਰੋਫਾਈਲ ਅਤੇ ਕੀਮਤਾਂ ਵਾਲੇ ਕਲਾਉਡ ਪ੍ਰਦਾਤਾਵਾਂ ਦੀ ਚੋਣ ਮਿਲਦੀ ਹੈ।

ਕੁਆਂਟੀਕਰਨ: ਕੁਆਂਟੀਕਰਨ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਘੱਟ-ਸੁੱਧਤਾ ਵਾਲੇ ਡੇਟਾ ਪ੍ਰਕਾਰਾਂ ਨਾਲ ਭਾਰ ਅਤੇ ਸਰਗਰਮੀਆਂ ਨੂੰ ਦਰਸਾ ਕੇ ਮਾਡਲ ਦੀ ਮੈਮੋਰੀ ਫੁੱਟਪ੍ਰਿੰਟ ਅਤੇ ਕੰਪਊਟੇਸ਼ਨਲ ਲੋੜਾਂ ਨੂੰ ਘਟਾਉਣ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਇਹ ਗੁਣਵੱਤਾ ਨੂੰ ਮਹੱਤਵਪੂਰਨ ਤੌਰ 'ਤੇ ਘਟਾਏ ਬਿਨਾਂ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਬਹਿਤਰ ਬਣਾ ਸਕਦਾ ਹੈ। ਇੱਕ ਐਪਲੀਕੇਸ਼ਨ ਡਵੈਲਪਰ ਵਜੋਂ, ਤੁਸੀਂ ਸ਼ਾਇਦ ਵੱਖ-ਵੱਖ ਕੁਆਂਟੀਕਰਨ ਪੱਧਰਾਂ 'ਤੇ ਆਪਣੇ ਖੁਦ ਦੇ ਮਾਡਲਾਂ ਦੀ ਸਖਿਲਾਈ ਵਿੱਚ ਸ਼ਾਮਲ ਨਹੀਂ ਹੋਵੋਗੇ, ਪਰ ਸ਼ਬਦਾਵਲੀ ਨਾਲ ਜਾਣੂ ਹੋਣਾ ਚੰਗਾ ਹੈ।

ਬੈਚਿੰਗ: ਬੈਚਾਂ ਵਿੱਚ ਇੱਕੋ ਸਮੇਂ ਕਈ ਬੇਨਤੀਆਂ ਦੀ ਪ੍ਰਕਰਿਆ ਕਰਨ ਨਾਲ ਮਾਡਲ ਲੋਡਿੰਗ ਅਤੇ ਡੇਟਾ ਟ੍ਰਾਂਸਫਰ ਦੇ ਓਵਰਹੈੱਡ ਨੂੰ ਘਟਾ ਕੇ ਥਰੂਪੁੱਟ ਨੂੰ ਬਹਿਤਰ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

ਕੈਸ਼ਿੰਗ: ਅਕਸਰ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਪ੍ਰੌਂਪਟਸ ਜਾਂ ਇਨਪੁੱਟ ਸੀਕੁਐਂਸਾਂ ਦੇ ਨਤੀਜਿਆਂ ਦੀ ਕੈਸ਼ਿੰਗ ਅਨੁਮਾਨ ਬੇਨਤੀਆਂ ਦੀ ਗਣਿਤੀ ਨੂੰ ਘਟਾ ਸਕਦੀ ਹੈ ਅਤੇ ਸਮੁੱਚੇ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਬਹਿਤਰ ਬਣਾ ਸਕਦੀ ਹੈ।

ਪ੍ਰੋਡਕਸ਼ਨ ਐਪਲੀਕੇਸ਼ਨ ਲਈ ਭਾਸ਼ਾ ਮਾਡਲ ਦੀ ਚੋਣ ਕਰਦੇ ਸਮੇਂ, ਪ੍ਰਤੀਨਧਿੀ ਵਰਕਲੋਡ ਅਤੇ ਹਾਰਡਵੇਅਰ ਕੌਨਫਗਿਰੇਸ਼ਨਾਂ 'ਤੇ ਇਸਦੇ ਪ੍ਰਦਰਸ਼ਨ ਦਾ ਬੈਂਚਮਾਰਕ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਇਹ ਸੰਭਾਵੀ ਬੋਤਲਨੈੱਕਾਂ ਦੀ ਪਛਾਣ ਕਰਨ ਅਤੇ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰ ਸਕਦਾ ਹੈ ਕਿ ਮਾਡਲ ਲੋੜੀਂਦੇ ਪ੍ਰਦਰਸ਼ਨ ਟੀਚਿਆਂ ਨੂੰ ਪੂਰਾ ਕਰ ਸਕਦਾ ਹੈ।

ਮਾਡਲ ਪ੍ਰਦਰਸ਼ਨ ਅਤੇ ਹੋਰ ਕਾਰਕਾਂ ਜਵਿੇਂ ਕਿ ਕੀਮਤ, ਲਚਕਤਾ, ਅਤੇ ਏਕੀਕਰਨ ਦੀ ਸੌਖ ਵਚਿਕਾਰ ਟ੍ਰੇਡ-ਔਫ 'ਤੇ ਵਚਿਾਰ ਕਰਨਾ ਵੀ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਉਦਾਹਰਨ ਲਈ, ਘੱਟ ਦੇਰੀ ਵਾਲੇ ਛੋਟੇ, ਘੱਟ ਖਰਚੇ ਵਾਲੇ ਮਾਡਲ ਦੀ ਵਰਤੋਂ ਉਨ੍ਹਾਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਬਹਿਤਰ ਹੋ ਸਕਦੀ ਹੈ ਜਨ੍ਹਿਾਂ ਨੂੰ ਰੀਅਲ-ਟਾਈਮ ਜਵਾਬਾਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜਦੋਂ ਕਿ ਵੱਡਾ, ਵਧੇਰੇ ਸ਼ਕਤੀਸ਼ਾਲੀ ਮਾਡਲ ਬੈਚ ਪ੍ਰੋਸੈਸਿੰਗ ਜਾਂ ਗੁੰਝਲਦਾਰ ਤਰਕ ਕਾਰਜਾਂ ਲਈ ਵਧੇਰੇ ਢੁਕਵਾਂ ਹੋ ਸਕਦਾ ਹੈ।

ਵੱਖ-ਵੱਖ ਐੱਲਐੱਲਐੱਮ ਮਾਡਲਾਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਨਾ

ਐੱਲਐੱਲਐੱਮ ਦੀ ਚੋਣ ਕਰਨਾ ਸ਼ਾਇਦ ਹੀ ਕਦੇ ਇੱਕ ਸਥਾਈ ਫੈਸਲਾ ਹੁੰਦਾ ਹੈ। ਜਦੋਂ ਕਿ ਨਵੇਂ ਅਤੇ ਬਹਿਤਰ ਮਾਡਲ ਨਿਯਮਤਿ ਤੌਰ 'ਤੇ ਜਾਰੀ ਕੀਤੇ ਜਾਂਦੇ ਹਨ, ਇੱਕ ਮੋਡੂਲਰ ਤਰੀਕੇ ਨਾਲ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣਾ ਚੰਗਾ ਹੈ ਜੋ ਸਮੇਂ ਦੇ ਨਾਲ ਵੱਖ-ਵੱਖ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨੂੰ ਸਵੈਪ ਕਰਨ ਦੀ ਆਗਿਆ ਦੇਂਦਾ ਹੈ। ਪ੍ਰੌਂਪਟਸ ਅਤੇ ਡੇਟਾਸੈੱਟ ਅਕਸਰ ਮਾਮੂਲੀ ਤਬਦੀਲੀਆਂ ਨਾਲ ਮਾਡਲਾਂ ਵੱਚਿ ਮੁੜ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਇਹ ਤੁਹਾਨੂੰ ਆਪਣੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਪੂਰੀ ਤਰ੍ਹਾਂ ਮੁੜ ਡਿਜ਼ਾਈਨ ਕੀਤੇ ਬਨਿਾਂ ਭਾਸ਼ਾ ਮਾਡਲਾਂਗਿ ਵੱਚਿ ਨਵੀਨਤਮ ਤਰੱਕੀ ਦਾ ਫਾਇਦਾ ਲੈਣ ਦੀ ਆਗਿਆ ਦੇਂਦਾ ਹੈ।

> ਵੱਖ-ਵੱਖ ਮਾਡਲ ਚੋਣਾਂ ਵਚਿਕਾਰ ਆਸਾਨੀ ਨਾਲ ਸਵੈਪ ਕਰਨ ਦੀ ਯੋਗਤਾ ਇੱਕ ਹੋਰ ਕਾਰਨ ਹੈ ਕਿ ਮੈਂ OpenRouter ਨੂੰ ਪਸੰਦ ਕਰਦਾ/ਕਰਦੀ ਹਾਂ।

ਨਵੇਂ ਭਾਸ਼ਾ ਮਾਡਲ 'ਤੇ ਅੱਪਗ੍ਰੇਡ ਕਰਦੇ ਸਮੇਂ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਇਸਦੇ ਪ੍ਰਦਰਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ ਗੁਣਵੱਤਾ ਦੀ ਪੂਰੀ ਤਰ੍ਹਾਂ ਜਾਂਚ ਅਤੇ ਪ੍ਰਮਾਣਕਿਤਾ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਇਹ ਐਪਲੀਕੇਸ਼ਨ ਦੀਆਂ ਲੋੜਾਂ ਨੂੰ ਪੂਰਾ ਕਰਦਾ ਹੈ। ਇਸ ਵੱਚਿ ਡੋਮੇਨ-ਵਸ਼ਿਸ਼ ਡੇਟਾ 'ਤੇ ਮਾਡਲ ਨੂੰ ਮੁੜ ਸਖਿਲਾਈ ਜਾਂ ਫਾਈਨ-ਟਊਿਨਂਗਿ ਕਰਨਾ, ਅਤੇ ਨਾਲ ਹੀ ਕਸਿੇ ਵੀ ਡਾਊਨਸਟ੍ਰੀਮ ਕੰਪੋਨੈਂਟਸ ਨੂੰ ਅੱਪਡੇਟ ਕਰਨਾ ਸ਼ਾਮਲ ਹੋ ਸਕਦਾ ਹੈ ਜੋ ਮਾਡਲ ਦੇ ਆਉਟਪੁੱਟ 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹਨ।

ਪ੍ਰਦਰਸ਼ਨ ਅਤੇ ਮੋਡੂਲੈਰਟਿੀ ਨੂੰ ਧਆਿਨ ਵੱਚਿ ਰੱਖਦੇ ਹੋਏ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਡਿਜ਼ਾਈਨ ਕਰਕੇ, ਤੁਸੀਂ ਸਕੇਲੇਬਲ, ਕੁਸ਼ਲ, ਅਤੇ ਭਵਖਿ-ਪਰੂਫ ਸਸਿਟਮ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਭਾਸ਼ਾ ਮਾਡਲਾਂਗਿ ਤਕਨਾਲੋਜੀ ਦੇ ਤੇਜ਼ੀ ਨਾਲ ਵਕਿਸਤ ਹੋ ਰਹੇ ਖੇਤਰ ਨਾਲ ਅਨੁਕੂਲ ਹੋ ਸਕਦੇ ਹਨ।

ਮਸ਼ਿਰਤਿ ਏ.ਆਈ. ਸਸਿਟਮ

ਆਪਣੀ ਜਾਣ-ਪਛਾਣ ਨੂੰ ਖਤਮ ਕਰਨ ਤੋਂ ਪਹਲਿਾਂ, ਇਹ ਦੱਸਣਾ ਜ਼ਰੂਰੀ ਹੈ ਕਿ 2023 ਤੋਂ ਪਹਲਿਾਂ ਅਤੇ ChatGPT ਦੁਆਰਾ ਜਨਰੇਟਵਿ ਏ.ਆਈ. ਵੱਚਿ ਦਲਿਚਸਪੀ ਦੇ ਵਸਿਫੋਟ ਤੋਂ ਪਹਲਿਾਂ, ਰਵਾਇਤੀ ਏ.ਆਈ. ਪਹੁੰਚਾਂ ਆਮ ਤੌਰ 'ਤੇ ਇਕੱਲੇ, ਬੰਦ ਮਾਡਲਾਂ ਦੇ ਏਕੀਕਰਨ 'ਤੇ ਨਰਿਭਰ ਕਰਦੀਆਂ ਸਨ। ਇਸਦੇ ਉਲਟ, ਮਸ਼ਿਰਤਿ ਏ.ਆਈ. ਸਸਿਟਮ ਬੁੱਧੀਮਾਨ ਵਵਿਹਾਰ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਇੱਕ ਦੂਜੇ ਨਾਲ ਕੰਮ ਕਰਨ ਵਾਲੇ ਆਪਸ ਵੱਚਿ ਜੁੜੇ ਹੱਸਿਆਂ ਦੀਆਂ ਗੁੰਝਲਦਾਰ ਪਾਈਪਲਾਈਨਾਂ ਦਾ ਲਾਭ ਲੈਂਦੇ ਹਨ।

ਆਪਣੇ ਮੂਲ ਵੱਚਿ, ਮਸਿ਼ਰਤਿ ਏ.ਆਈ. ਸਸਿਟਮ ਕਈ ਮੋਡੀਊਲਾਂ ਤੋ ਬਣੇ ਹੁੰਦੇ ਹਨ, ਜਨ੍ਹਿਾਂ ਵੱਚਿੋਂ ਹਰੇਕ ਖਾਸ ਕੰਮ ਜਾਂ ਕਾਰਜ ਕਰਨ ਲਈ ਤਆਿਰ ਕੀਤਾ ਗਆਿ ਹੈ। ਇਹਨਾਂ ਮੋਡੀਊਲਾਂ ਵੱਚਿ ਜਨਰੇਟਰ, ਰਟਿਰੀਵਰ, ਰੈਂਕਰ, ਕਲਾਸੀਫਾਇਰ, ਅਤੇ ਹੋਰ ਵੱਖ-ਵੱਖ ਵਸ਼ਿੇਸ਼ ਘਟਕ ਸ਼ਾਮਲ ਹੋ ਸਕਦੇ ਹਨ। ਸਮੁੱਚੇ ਸਸਿਟਮ ਨੂੰ ਛੋਟੀਆਂ, ਕੇਂਦਰਤਿ ਇਕਾਈਆਂ ਵੱਚਿ ਵੰਡ ਕੇ, ਡਵੈਲਿਪਰ ਵਧੇਰੇ ਲਚਕਦਾਰ, ਸਕੇਲੇਬਲ, ਅਤੇ ਰੱਖ-ਰਖਾਅ ਯੋਗ ਏ.ਆਈ. ਆਰਕੀਟੈਕਚਰ ਬਣਾ ਸਕਦੇ ਹਨ।

ਮਸਿ਼ਰਤਿ ਏ.ਆਈ. ਸਸਿਟਮਾਂ ਦਾ ਇੱਕ ਮੁੱਖ ਫਾਇਦਾ ਵੱਖ-ਵੱਖ ਏ.ਆਈ. ਤਕਨੀਕਾਂ ਅਤੇ ਮਾਡਲਾਂ ਦੀਆਂ ਤਾਕਤਾਂ ਨੂੰ ਜੋੜਨ ਦੀ ਉਨ੍ਹਾਂ ਦੀ ਯੋਗਤਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ, ਇੱਕ ਸਸਿਟਮ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਦੀ ਸਮਝ ਅਤੇ ਉਤਪਾਦਨ ਲਈ ਇੱਕ ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ (LLM) ਵਰਤ ਸਕਦਾ ਹੈ, ਜਦੋਂ ਕਿ ਜਾਣਕਾਰੀ ਪ੍ਰਾਪਤੀ ਜਾਂ ਨਯਿਮ-ਆਧਾਰਤਿ ਫੈਸਲਾ ਲੈਣ ਲਈ ਇੱਕ ਵੱਖਰਾ ਮਾਡਲ ਵਰਤਦਾ ਹੈ। ਇਹ ਮੋਡੀਊਲਰ ਪਹੁੰਚ ਤੁਹਾਨੂੰ ਇੱਕ-ਆਕਾਰ-ਸਭ-ਲਈ-ਫਟ੿ ਹੱਲ ’ਤੇ ਨਰਿਭਰ ਕਰਨ ਦੀ ਬਜਾਏ ਹਰ ਖਾਸ ਕੰਮ ਲਈ ਸਭ ਤੋਂ ਵਧੀਆ ਟੂਲ ਅਤੇ ਤਕਨੀਕਾਂ ਦੀ ਚੋਣ ਕਰਨ ਦੀ ਆਗਆਿ ਦ੿ਂਦੀ ਹੈ।

ਹਾਲਾਂਕਿ, ਮਸਿ਼ਰਤਿ ਏ.ਆਈ. ਸਸਿਟਮ ਬਣਾਉਣਾ ਵੀ ਵਲਿੱਖਣ ਚੁਣੌਤੀਆਂ ਪੇਸ਼ ਕਰਦਾ ਹੈ। ਖਾਸ ਤੌਰ ’ਤੇ, ਸਸਿਟਮ ਦੇ ਵਵਿਹਾਰ ਦੀ ਸਮੁੱਚੀ ਸੰਗਤਤਾ ਅਤੇ ਇਕਸਾਰਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਮਜ਼ਬੂਤ ਟੈਸਟਿੰਗ, ਨਗਿਰਾਨੀ, ਅਤੇ ਪ੍ਰਸ਼ਾਸਨ ਵਧਿੀਆਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

GPT-4 ਵਰਗੇ ਸ਼ਕਤੀਸ਼ਾਲੀ LLMs ਦੇ ਆਉਣ ਨਾਲ ਸਾਨੂੰ ਪਹਲਿਾਂ ਨਾਲੋਂ ਕਤਿੇ ਵੱਧ ਆਸਾਨੀ ਨਾਲ ਮਸਿ਼ਰਤਿ ਏ.ਆਈ. ਸਸਿਟਮਾਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਨ ਦੀ ਆਗਆਿ ਮਲਿਦੀ ਹੈ, ਕਉਿਂਕਿ ਇਹ ਉੱਨਤ ਮਾਡਲ ਆਪਣੀਆਂ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਸਮਝ ਸਮਰੱਥਾਵਾਂ ਦੇ ਨਾਲ-ਨਾਲ ਇੱਕ ਮਸਿ਼ਰਤਿ ਸਸਿਟਮ ਦੇ ਅੰਦਰ ਕਈ ਭੂਮਕਿਾਵਾਂ ਨੂੰ ਸੰਭਾਲਣ ਦੇ ਯੋਗ ਹਨ, ਜਵਿੇਂ ਕਿ ਵਰਗੀਕਰਨ, ਰੈਂਕਿੰਗ, ਅਤੇ ਉਤਪਾਦਨ। ਇਹ ਬਹੁਮੁਖਤਾ ਡਵੈਲਿਪਰਾਂ ਨੂੰ ਮਸਿ਼ਰਤਿ ਏ.ਆਈ. ਆਰਕੀਟੈਕਚਰ ’ਤੇ ਤੇਜ਼ੀ ਨਾਲ ਪ੍ਰੋਟੋਟਾਈਪ ਅਤੇ ਇਟਰੇਟ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦੀ ਹੈ, ਜੋ ਬੁੱਧੀਮਾਨ ਐਪਲੀਕੇਸ਼ਨ ਵਕਿਾਸ ਲਈ ਨਵੀਆਂ ਸੰਭਾਵਨਾਵਾਂ ਖੋਲ੍ਹਦੀ ਹੈ।

ਮਸਿ਼ਰਤਿ ਏ.ਆਈ. ਸਸਿਟਮਾਂ ਲਈ ਡਪਿਲਾਏਮੈਂਟ ਪੈਟਰਨ

ਮਸਿ਼ਰਤਿ ਏ.ਆਈ. ਸਸਿਟਮਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਪੈਟਰਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਡਪਿਲੌਏ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਜਨ੍ਹਿਾਂ ਵੱਚਿੋਂ ਹਰੇਕ ਖਾਸ ਲੋੜਾਂ ਅਤੇ ਵਰਤੋਂ ਦੇ ਕੇਸਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਤਆਿਰ ਕੀਤਾ ਗਆਿ ਹੈ। ਆਓ ਚਾਰ ਆਮ ਡਪਿਲਾਏਮੈਂਟ ਪੈਟਰਨਾਂ ਦੀ ਪੜਚੋਲ ਕਰੀਏ: ਸਵਾਲ ਅਤੇ ਜਵਾਬ, ਬਹੁ-ਏਜੰਟ/ਏਜੰਟਕਿ ਸਮੱਸਆਿ ਹੱਲ ਕਰਤਾ, ਸੰਵਾਦੀ ਏ.ਆਈ., ਅਤੇ ਸਹ-ਪਾਇਲਟ।

ਸਵਾਲ ਅਤੇ ਜਵਾਬ

ਸਵਾਲ ਅਤੇ ਜਵਾਬ (Q&A) ਸਿਸਟਮ ਜਾਣਕਾਰੀ ਪ੍ਰਾਪਤੀ ’ਤੇ ਕੇਂਦਰਤਿ ਹੁੰਦੇ ਹਨ ਜੋ ਏ.ਆਈ. ਮਾਡਲਾਂ ਦੀਆਂ ਸਮਝ ਸਮਰੱਥਾਵਾਂ ਨਾਲ ਵਧਾਏ ਗਏ ਹਨ ਤਾਂ ਜੋ ਇਹ ਸਰਿਫ਼ ਇੱਕ ਖੋਜ ਇੰਜਣ ਤੋਂ ਵੱਧ ਕੇ ਕੰਮ ਕਰ ਸਕਣ। ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਵਧਤਿ ਉਤਪਾਦਨ (RAG) ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਸ਼ਕਤੀਸ਼ਾਲੀ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨੂੰ ਬਾਹਰੀ ਗਿਆਨ ਸਰੋਤਾਂ ਨਾਲ ਜੋੜ ਕੇ, ਸਵਾਲ ਅਤੇ ਜਵਾਬ ਸਿਸਟਮ ਕਲਪਨਾਵਾਂ ਤੋਂ ਬਚਦੇ ਹਨ ਅਤੇ ਉਪਭੋਗਤਾ ਦੇ ਸਵਾਲਾਂ ਦੇ ਸਹੀ ਅਤੇ ਪ੍ਰਸੰਗਕਿ ਜਵਾਬ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

LLM-ਅਧਾਰਤਿ ਪ੍ਰਸ਼ਨ-ਉੱਤਰ ਪ੍ਰਣਾਲੀ ਦੇ ਮੁੱਖ ਹੱਸੇ ਹਨ:

- **ਪੁੱਛਗੱਛਿ ਸਮਝਣਾ ਅਤੇ ਮੁੜ-ਨਰਿਮਾਣ:** ਵਰਤੋਂਕਾਰ ਦੀਆਂ ਪੁੱਛਗੱਛਿਾਂ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਨਾ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਅੰਦਰੂਨੀ ਗਿਆਨ ਸਰੋਤਾਂ ਨਾਲ ਬਹਿਤਰ ਮੇਲ ਲਈ ਮੁੜ-ਨਰਿਮਾਣ ਕਰਨਾ।
- **ਗਿਆਨ ਪ੍ਰਾਪਤੀ:** ਮੁੜ-ਨਰਿਮਤਿ ਪੁੱਛਗੱਛਿ ਦੇ ਆਧਾਰ ’ਤੇ ਢਾਂਚਾਗਤ ਜਾਂ ਗੈਰ-ਢਾਂਚਾਗਤ ਡਾਟਾ ਸਰੋਤਾਂ ਤੋਂ ਢੁਕਵੀਂ ਜਾਣਕਾਰੀ ਪ੍ਰਾਪਤ ਕਰਨਾ।
- **ਜਵਾਬ ਤਿਆਰ ਕਰਨਾ:** ਪ੍ਰਾਪਤ ਕੀਤੇ ਗਿਆਨ ਨੂੰ ਭਾਸ਼ਾ ਮਾਡਲ ਦੀਆਂ ਉਤਪਾਦਕ ਸਮਰੱਥਾਵਾਂ ਨਾਲ ਜੋੜ ਕੇ ਸੁਸੰਗਤ ਅਤੇ ਜਾਣਕਾਰੀ ਭਰਪੂਰ ਜਵਾਬ ਤਿਆਰ ਕਰਨਾ।

RAG ਉਪ-ਪ੍ਰਣਾਲੀਆਂ ਖਾਸ ਤੌਰ ’ਤੇ ਉਨ੍ਹਾਂ ਪ੍ਰਸ਼ਨ-ਉੱਤਰ ਖੇਤਰਾਂ ਵੱਚਿ ਮਹੱਤਵਪੂਰਨ ਹਨ ਜੱਥੇ ਸਹੀ ਅਤੇ ਨਵੀਨਤਮ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਨਾ ਜ਼ਰੂਰੀ ਹੈ, ਜਵਿੇਂ ਕ ਿਗਾਹਕ ਸਹਾਇਤਾ, ਗਿਆਨ ਪ੍ਰਬੰਧਨ, ਜਾਂ ਵੱਦਿਅਕ ਐਪਲੀਕੇਸ਼ਨਾਂ

ਬਹੁ-ਏਜੰਟ/ਏਜੰਟਕਿ ਸਮੱਸਿਆ ਹੱਲ ਕਰਨ ਵਾਲੇ

ਬਹੁ-ਏਜੰਟ, ਜਸਿਨੂੰ ਏਜੰਟਕਿ ਵੀ ਕਹਿਾ ਜਾਂਦਾ ਹੈ, ਪ੍ਰਣਾਲੀਆਂ ਵੱਚਿ ਕਈ ਸੁਤੰਤਰ ਏਜੰਟ ਗੁੰਝਲਦਾਰ ਸਮੱਸਿਆਵਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਇਕੱਠੇ ਕੰਮ ਕਰਦੇ ਹਨ। ਹਰ ਏਜੰਟ ਦੀ ਇੱਕ ਖਾਸ ਭੂਮਕਿਾ, ਹੁਨਰਾਂ ਦਾ ਸੈੱਟ, ਅਤੇ ਢੁਕਵੇਂ ਸਾਧਨਾਂ ਜਾਂ ਜਾਣਕਾਰੀ ਸਰੋਤਾਂ ਤੱਕ ਪਹੁੰਚ ਹੁੰਦੀ ਹੈ। ਸਹਯਿੋਗ ਕਰਕੇ ਅਤੇ ਜਾਣਕਾਰੀ ਦਾ ਆਦਾਨ-ਪ੍ਰਦਾਨ ਕਰਕੇ, ਇਹ ਏਜੰਟ ਅਜਹਿੇ ਕੰਮਾਂ ਨੂੰ ਨਪਿਟਾ ਸਕਦੇ ਹਨ ਜੋ ਇਕੱਲੇ ਏਜੰਟ ਲਈ ਮੁਸ਼ਕਲ ਜਾਂ ਅਸੰਭਵ ਹੋਣਗੇ।

ਬਹੁ-ਏਜੰਟ ਸਮੱਸਿਆ ਹੱਲ ਕਰਨ ਵਾਲਿਆਂ ਦੇ ਮੁੱਖ ਸਧਿਾਂਤ ਹਨ:

- **ਵਸ਼ਿੇਸ਼ੀਕਰਨ:** ਹਰ ਏਜੰਟ ਸਮੱਸਿਆ ਦੇ ਇੱਕ ਖਾਸ ਪਹਲੂ ’ਤੇ ਧਿਆਨ ਕੇਂਦਰਤਿ ਕਰਦਾ ਹੈ, ਆਪਣੀਆਂ ਵਲਿੱਖਣ ਸਮਰੱਥਾਵਾਂ ਅਤੇ ਗਿਆਨ ਦਾ ਲਾਭ ਲੈਂਦਾ ਹੈ।

- **ਸਹਯੋਗ:** ਏਜੰਟ ਇੱਕ ਸਾਂਝੇ ਟੀਚੇ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਸੰਚਾਰ ਕਰਦੇ ਹਨ ਅਤੇ ਆਪਣੀਆਂ ਕਾਰਵਾਈਆਂ ਦਾ ਤਾਲਮੇਲ ਕਰਦੇ ਹਨ, ਅਕਸਰ ਸੁਨੇਹਾ ਭੇਜਣ ਜਾਂ ਸਾਂਝੀ ਮੈਮੋਰੀ ਰਾਹੀਂ।
- **ਅਨੁਕੂਲਤਾ:** ਪ੍ਰਣਾਲੀ ਵਅਿਕਤੀਗਤ ਏਜੰਟਾਂ ਦੀਆਂ ਭੂਮਕਿਾਵਾਂ ਅਤੇ ਵਵਿਹਾਰਾਂ ਨੂੰ ਵਵਿਸਥਤਿ ਕਰਕੇ ਬਦਲਦੀਆਂ ਸਥਤਿੀਆਂ ਜਾਂ ਲੋੜਾਂ ਦੇ ਅਨੁਕੂਲ ਹੋ ਸਕਦੀ ਹੈ।

ਬਹੁ-ਏਜੰਟ ਪ੍ਰਣਾਲੀਆਂ ਉਨ੍ਹਾਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਢੁਕਵੀਆਂ ਹਨ ਜੰਥਿੇ ਵੰਡੀ ਹੋਈ ਸਮੱਸਅਿਾ ਹੱਲ ਕਰਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜਵਿੇਂ ਕ ਿਸਪਲਾਈ ਚੇਨ ਆਪਟੀਮਾਈਜ਼ੇਸ਼ਨ, ਟ੍ਰੈਫਕਿ ਪ੍ਰਬੰਧਨ, ਜਾਂ ਐਮਰਜੈਂਸੀ ਪ੍ਰਤੀਕਰਿਆ ਯੋਜਨਾਬੰਦੀ

ਵਾਰਤਾਲਾਪੀ AI

ਵਾਰਤਾਲਾਪੀ AI ਪ੍ਰਣਾਲੀਆਂ ਵਰਤੋਂਕਾਰਾਂ ਅਤੇ ਬੁੱਧੀਮਾਨ ਏਜੰਟਾਂ ਵਚਿਕਾਰ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਗੱਲਬਾਤ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦੀਆਂ ਹਨ। ਇਹ ਪ੍ਰਣਾਲੀਆਂ ਦਲਿਚਸਪ ਅਤੇ ਵਅਿਕਤੀਗਤ ਗੱਲਬਾਤ ਦੇ ਤਜਰਬੇ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਸਮਝ, ਵਾਰਤਾਲਾਪ ਪ੍ਰਬੰਧਨ, ਅਤੇ ਭਾਸ਼ਾ ਉਤਪਾਦਨ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਜੋੜਦੀਆਂ ਹਨ।

ਵਾਰਤਾਲਾਪੀ AI ਪ੍ਰਣਾਲੀ ਦੇ ਮੁੱਖ ਹੱਸਿੇ ਹਨ:

- **ਇਰਾਦਾ ਪਛਾਣ:** ਵਰਤੋਂਕਾਰ ਦੇ ਇਨਪੁਟ ਦੇ ਆਧਾਰ 'ਤੇ ਉਨ੍ਹਾਂ ਦੇ ਇਰਾਦੇ ਦੀ ਪਛਾਣ ਕਰਨਾ, ਜਵਿੇਂ ਕ ਿਸਵਾਲ ਪੁੱਛਣਾ, ਬੇਨਤੀ ਕਰਨਾ, ਜਾਂ ਭਾਵਨਾ ਪ੍ਰਗਟ ਕਰਨਾ।
- **ਇਕਾਈ ਕੱਢਣਾ:** ਵਰਤੋਂਕਾਰ ਦੇ ਇਨਪੁਟ ਤੋਂ ਢੁਕਵੀਆਂ ਇਕਾਈਆਂ ਜਾਂ ਪੈਰਾਮੀਟਰਾਂ ਨੂੰ ਕੱਢਣਾ, ਜਵਿੇਂ ਕ ਿਤਾਰੀਖਾਂ, ਸਥਾਨ, ਜਾਂ ਉਤਪਾਦ ਨਾਮ।
- **ਵਾਰਤਾਲਾਪ ਪ੍ਰਬੰਧਨ:** ਗੱਲਬਾਤ ਦੀ ਸਥਤਿੀ ਨੂੰ ਬਣਾਈ ਰੱਖਣਾ, ਵਰਤੋਂਕਾਰ ਦੇ ਇਰਾਦੇ ਅਤੇ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਢੁਕਵਾਂ ਜਵਾਬ ਨਰਿਧਾਰਤਿ ਕਰਨਾ, ਅਤੇ ਬਹੁ-ਵਾਰੀ ਗੱਲਬਾਤ ਨੂੰ ਸੰਭਾਲਣਾ।
- **ਜਵਾਬ ਤਅਿਾਰੀ:** ਭਾਸ਼ਾ ਮਾਡਲਾਂ, ਟੈਂਪਲੇਟਾਂ, ਜਾਂ ਰਟਿਰੀਵਲ-ਅਧਾਰਤਿ ਵਧਿੀਆਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਮਨੁੱਖੀ-ਵਰਗੇ ਜਵਾਬ ਤਅਿਾਰ ਕਰਨਾ।

ਗੱਲਬਾਤ ਵਾਲੇ AI ਸਸਿਟਮ ਆਮ ਤੌਰ 'ਤੇ ਗਾਹਕ ਸੇਵਾ ਚੈਟਬੋਟਾਂ, ਵਰਚੁਅਲ ਸਹਾਇਕਾਂ, ਅਤੇ ਆਵਾਜ਼-ਨਯਿੰਤਰਤਿ ਇੰਟਰਫੇਸਾਂ ਵੱਚਿ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਜਵਿੇਂ ਕ ਿਪਹਲਿਾਂ ਦੱਸਅਿਾ ਗਅਿਾ ਹੈ, ਇਸ ਕਤਿਾਬ ਵੱਚਿ ਜ਼ਅਿਾਦਾਤਰ ਪਹੁੰਚਾਂ, ਪੈਟਰਨ, ਅਤੇ ਕੋਡ ਉਦਾਹਰਣਾਂ ਸੱਧਿੇ ਤੌਰ 'ਤੇ Olympia ਨਾਮਕ ਇੱਕ ਵੱਡੇ ਗੱਲਬਾਤ AI ਸਸਿਟਮ 'ਤੇ ਮੇਰੇ ਕੰਮ ਤੋਂ ਲਈਆਂ ਗਈਆਂ ਹਨ।

CoPilots

CoPilots ਉਹ AI-ਸੰਚਾਲਤਿ ਸਹਾਇਕ ਹਨ ਜੋ ਮਨੁੱਖੀ ਵਰਤੋਂਕਾਰਾਂ ਦੇ ਨਾਲ ਕੰਮ ਕਰਦੇ ਹੋਏ ਉਨ੍ਹਾਂ ਦੀ ਉਤਪਾਦਕਤਾ ਅਤੇ ਫੈਸਲਾ ਲੈਣ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਵਧਾਉਂਦੇ ਹਨ। ਇਹ ਸਸਿਟਮ ਬੁੱਧੀਮਾਨ ਸਫਿਾਰਸ਼ਾਂ ਪ੍ਰਦਾਨ ਕਰਨ, ਕਾਰਜਾਂ ਨੂੰ ਸਵੈਚਾਲਤਿ ਕਰਨ, ਅਤੇ ਸੰਦਰਭਕ ਸਹਾਇਤਾ ਪੇਸ਼ ਕਰਨ ਲਈ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਪ੍ਰੋਸੈਸਿੰਗ, ਮਸ਼ੀਨ ਲਰਨਿੰਗ, ਅਤੇ ਖੇਤਰ-ਵਸ਼ਿੇਸ਼ ਗਆਿਨ ਦੇ ਸੁਮੇਲ ਦਾ ਲਾਭ ਲੈਂਦੇ ਹਨ।

CoPilots ਦੀਆਂ ਮੁੱਖ ਵਸ਼ਿੇਸ਼ਤਾਵਾਂ ਵਚਿ ਸ਼ਾਮਲ ਹਨ:

- **ਨਜਿੀਕਰਨ:** ਵਅਿਕਤੀਗਤ ਵਰਤੋਂਕਾਰ ਤਰਜੀਹਾਂ, ਕਾਰਜ-ਪ੍ਰਵਾਹਾਂ, ਅਤੇ ਸੰਚਾਰ ਸ਼ੈਲੀਆਂ ਦੇ ਅਨੁਕੂਲ ਹੋਣਾ।
- **ਸਰਗਰਮ ਸਹਾਇਤਾ:** ਵਰਤੋਂਕਾਰ ਦੀਆਂ ਲੋੜਾਂ ਦਾ ਅਨੁਮਾਨ ਲਗਾਉਣਾ ਅਤੇ ਸਪੱਸ਼ਟ ਸੰਕੇਤਾਂ ਤੋਂ ਬਨਿਾਂ ਢੁਕਵੀਆਂ ਸੁਝਾਅ ਜਾਂ ਕਾਰਵਾਈਆਂ ਦੀ ਪੇਸ਼ਕਸ਼ ਕਰਨਾ।
- **ਨਰਿੰਤਰ ਸਖਿਣਾ:** ਵਰਤੋਂਕਾਰ ਫੀਡਬੈਕ, ਇੰਟਰੈਕਸ਼ਨ, ਅਤੇ ਡਾਟਾ ਤੋਂ ਸਖਿ ਕੇ ਸਮੇਂ ਦੇ ਨਾਲ ਪ੍ਰਦਰਸ਼ਨ ਵਚਿ ਸੁਧਾਰ ਕਰਨਾ।

CoPilots ਵੱਖ-ਵੱਖ ਖੇਤਰਾਂ ਵਚਿ ਵਧਦੇ ਹੋਏ ਵਰਤੇ ਜਾ ਰਹੇ ਹਨ, ਜਵਿੇਂ ਕ ਿਸਾਫਟਵੇਅਰ ਵਕਿਾਸ (ਉਦਾਹਰਨ ਲਈ, ਕੋਡ ਪੂਰਤੀ ਅਤੇ ਬੱਗ ਖੋਜ), ਰਚਨਾਤਮਕ ਲੇਖਣ (ਉਦਾਹਰਨ ਲਈ, ਸਮੱਗਰੀ ਸੁਝਾਅ ਅਤੇ ਸੰਪਾਦਨ), ਅਤੇ ਡਾਟਾ ਵਸ਼ਿੇਲੇਸ਼ਣ (ਉਦਾਹਰਨ ਲਈ, ਅੰਤਰਦ੍ਰਸ਼ਟਿੀਆਂ ਅਤੇ ਵਜ਼ੂਅਲਾਈਜ਼ੇਸ਼ਨ ਸਫਿਾਰਸ਼ਾਂ)

ਇਹ ਤੈਨਾਤੀ ਪੈਟਰਨ ਮਸ਼ਿਰਤਿ AI ਸਸਿਟਮਾਂ ਦੀ ਬਹੁਮੁਖਤਾ ਅਤੇ ਸੰਭਾਵਨਾ ਨੂੰ ਦਰਸਾਉਂਦੇ ਹਨ। ਹਰ ਪੈਟਰਨ ਦੀਆਂ ਵਸ਼ਿੇਸ਼ਤਾਵਾਂ ਅਤੇ ਵਰਤੋਂ ਦੇ ਮਾਮਲਆਿਂ ਨੂੰ ਸਮਝ ਕੇ, ਤੁਸੀਂ ਬੁੱਧੀਮਾਨ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਡਜ਼ਾਇਨ ਅਤੇ ਲਾਗੂ ਕਰਦੇ ਸਮੇਂ ਸੂਚਤਿ ਫੈਸਲੇ ਲੈ ਸਕਦੇ ਹੋ। ਭਾਵੇਂ ਇਹ ਕਤਿਾਬ ਖਾਸ ਤੌਰ 'ਤੇ ਮਸ਼ਿਰਤਿ AI ਸਸਿਟਮਾਂ ਦੀ ਲਾਗੂਕਰਨ ਬਾਰੇ ਨਹੀਂ ਹੈ, ਪਰ ਬਹੁਤ ਸਾਰੀਆਂ ਜੇ ਸਾਰੀਆਂ ਨਹੀਂ ਤਾਂ ਉਹੀ ਪਹੁੰਚਾਂ ਅਤੇ ਪੈਟਰਨ ਹੋਰ ਰਵਾਇਤੀ ਐਪਲੀਕੇਸ਼ਨ ਵਕਿਾਸ ਦੇ ਅੰਦਰ ਵੱਖਰੇ AI ਭਾਗਾਂ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨ 'ਤੇ ਲਾਗੂ ਹੁੰਦੇ ਹਨ।

ਮਸ਼ਿਰਤਿ AI ਸਸਿਟਮਾਂ ਵਚਿ ਭੂਮਕਿਾਵਾਂ

ਮਸ਼ਿਰਤਿ AI ਸਸਿਟਮ ਆਪਸ ਵਚਿ ਜੁੜੇ ਮੋਡਊਿਲਾਂ ਦੀ ਨੀਂਹ 'ਤੇ ਬਣੇ ਹੁੰਦੇ ਹਨ, ਜਨ੍ਹਿਾਂ ਵਚਿੋਂ ਹਰੇਕ ਇੱਕ ਵਸ਼ਿੇਸ਼ ਭੂਮਕਿਾ ਨਭਿਾਉਣ ਲਈ ਡਜ਼ਾਇਨ ਕੀਤਾ ਗਆਿ ਹੈ। ਇਹ ਮੋਡਊਿਲ ਬੁੱਧੀਮਾਨ ਵਵਿਹਾਰ ਬਣਾਉਣ ਅਤੇ ਗੁੰਝਲਦਾਰ ਸਮੱਸਆਿਵਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਇਕੱਠੇ ਕੰਮ ਕਰਦੇ ਹਨ। ਇਸ ਬਾਰੇ ਸੋਚਣ ਵੇਲੇ ਇਨ੍ਹਾਂ ਭੂਮਕਿਾਵਾਂ ਤੋਂ ਜਾਣੂ ਹੋਣਾ ਲਾਭਦਾਇਕ ਹੈ ਕ ਿਤੁਸੀਂ ਆਪਣੀ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਹੱਸਿਆਂ ਨੂੰ ਵੱਖਰੇ AI ਭਾਗਾਂ ਨਾਲ ਲਾਗੂ ਜਾਂ ਬਦਲ ਸਕਦੇ ਹੋ।

ਜਨਰੇਟਰ

ਜਨਰੇਟਰ ਸੰਖੇ ਹੋਏ ਪੈਟਰਨਾਂ ਜਾਂ ਇਨਪੁੱਟ ਸੰਕੇਤਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਨਵਾਂ ਡਾਟਾ ਜਾਂ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰਨ ਲਈ ਜ਼ਿੰਮੇਵਾਰ ਹੁੰਦੇ ਹਨ। AI ਦੁਨੀਆ ਵੱਚਿ ਕਈ ਵੱਖ-ਵੱਖ ਕਸਿਮ ਦੇ ਜਨਰੇਟਰ ਹਨ, ਪਰ ਇਸ ਕਤਿਾਬ ਵੱਚਿ ਦਖਿਾਏ ਗਏ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੇ ਸੰਦਰਭ ਵੱਚਿ, ਜਨਰੇਟਰ ਮਨੁੱਖੀ-ਵਰਗਾ ਟੈਕਸਟ ਬਣਾ ਸਕਦੇ ਹਨ, ਅਧੂਰੇ ਵਾਕਾਂ ਨੂੰ ਪੂਰਾ ਕਰ ਸਕਦੇ ਹਨ, ਜਾਂ ਵਰਤੋਂਕਾਰ ਦੇ ਸਵਾਲਾਂ ਦੇ ਜਵਾਬ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ। ਉਹ ਸਮੱਗਰੀ ਨਰਿਮਾਣ, ਸੰਵਾਦ ਤਆਿਰੀ, ਅਤੇ ਡਾਟਾ ਵਾਧੇ ਵਰਗੇ ਕਾਰਜਾਂ ਵੱਚਿ ਅਹਮਿ ਭੂਮਕਿਾ ਨਭਿਾਉਂਦੇ ਹਨ।

ਰਟਿਰੀਵਰ

ਰਟਿਰੀਵਰ ਵੱਡੇ ਡੇਟਾਸੇਟਾਂ ਜਾਂ ਗਆਿਨ ਅਧਾਰਾਂ ਵੱਚਿੋਂ ਢੁਕਵੀਂ ਜਾਣਕਾਰੀ ਦੀ ਖੋਜ ਅਤੇ ਕੱਢਣ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਉਹ ਦੱਤਿੇ ਗਏ ਸਵਾਲ ਜਾਂ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਸਭ ਤੋਂ ਢੁਕਵੇਂ ਡੇਟਾ ਪੁਆਇੰਟ ਲੱਭਣ ਲਈ ਸ਼ਬਦਾਰਥਕ ਖੋਜ, ਕੀਵਰਡ ਮੈਚੰਗਿ, ਜਾਂ ਵੈਕਟਰ ਸਮਾਨਤਾ ਵਰਗੀਆਂ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ। ਰਟਿਰੀਵਰ ਉਹਨਾਂ ਕਾਰਜਾਂ ਲਈ ਜ਼ਰੂਰੀ ਹਨ ਜੰਥੇ ਖਾਸ ਜਾਣਕਾਰੀ ਤੱਕ ਤੇਜ਼ ਪਹੁੰਚ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜਵਿੇਂ ਕ ਿਸਵਾਲਾਂ ਦੇ ਜਵਾਬ, ਤੱਥਾਂ ਦੀ ਜਾਂਚ, ਜਾਂ ਸਮੱਗਰੀ ਦੀ ਸਫਿਾਰਸ਼।

ਰੈਂਕਰ

ਰੈਂਕਰ ਕੁਝ ਮਾਪਦੰਡਾਂ ਜਾਂ ਪ੍ਰਸੰਗਕਿਤਾ ਸਕੋਰਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਆਈਟਮਾਂ ਦੇ ਸੈੱਟ ਨੂੰ ਕ੍ਰਮਬੱਧ ਜਾਂ ਪ੍ਰਾਥਮਕਿਤਾ ਦੇਣ ਲਈ ਜ਼ਿੰਮੇਵਾਰ ਹੁੰਦੇ ਹਨ। ਉਹ ਹਰੇਕ ਆਈਟਮ ਨੂੰ ਭਾਰ ਜਾਂ ਸਕੋਰ ਨਰਿਧਾਰਤ ਕਰਦੇ ਹਨ ਅਤੇ ਫਰਿ ਉਹਨਾਂ ਨੂੰ ਉਸ ਅਨੁਸਾਰ ਕ੍ਰਮਬੱਧ ਕਰਦੇ ਹਨ। ਰੈਂਕਰ ਆਮ ਤੌਰ 'ਤੇ ਖੋਜ ਇੰਜਣਾਂ, ਸਫਿਾਰਸ਼ ਸਸਿਟਮਾਂ, ਜਾਂ ਕਸਿ ਵੀ ਐਪਲੀਕੇਸ਼ਨ ਵੱਚਿ ਵਰਤੇ ਜਾਂਦੇ ਹਨ ਜੰਥੇ ਉਪਭੋਗਤਾਵਾਂ ਨੂੰ ਸਭ ਤੋਂ ਢੁਕਵੇਂ ਨਤੀਜੇ ਪੇਸ਼ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੁੰਦਾ ਹੈ।

ਵਰਗੀਕਰਣਕਰਤਾ

ਵਰਗੀਕਰਣਕਰਤਾ ਪਹਲਿਾਂ ਤੋਂ ਨਰਿਧਾਰਤ ਸ਼੍ਰੇਣੀਆਂ ਜਾਂ ਵਰਗਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਡੇਟਾ ਪੁਆਇੰਟਾਂ ਨੂੰ ਵਰਗੀਕ੍ਰਤਿ ਜਾਂ ਲੇਬਲ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਉਹ ਲੇਬਲ ਕੀਤੇ ਸਖਿਲਾਈ ਡੇਟਾ ਤੋਂ ਸੰਖਿਦੇ ਹਨ ਅਤੇ ਫਰਿ ਨਵੇਂ, ਅਣਦੇਖੇ ਉਦਾਹਰਣਾਂ ਦੀ ਸ਼੍ਰੇਣੀ ਦੀ ਭਵਖਿਬਾਣੀ ਕਰਦੇ ਹਨ। ਵਰਗੀਕਰਣਕਰਤਾ ਭਾਵਨਾ ਵਸ਼ਿਲੇਸ਼ਣ, ਸਪੈਮ ਖੋਜ, ਜਾਂ ਚੱਤਿਰ ਪਛਾਣ ਵਰਗੇ ਕਾਰਜਾਂ ਲਈ ਮੁੱਢਲੇ ਹਨ, ਜੰਥੇ ਟੀਚਾ ਹਰੇਕ ਇਨਪੁੱਟ ਨੂੰ ਇੱਕ ਵਸ਼ਿੇਸ਼ ਸ਼੍ਰੇਣੀ ਨਰਿਧਾਰਤ ਕਰਨਾ ਹੈ।

ਟੂਲਜ਼ ਅਤੇ ਏਜੰਟ

ਇਹਨਾਂ ਮੁੱਖ ਭੂਮਿਕਾਵਾਂ ਤੋਂ ਇਲਾਵਾ, ਮਸ਼ਿਰਤਿ ਏਆਈ ਸਸਿਟਮ ਅਕਸਰ ਆਪਣੀ ਕਾਰਜਸ਼ੀਲਤਾ ਅਤੇ ਅਨੁਕੂਲਤਾ ਨੂੰ ਵਧਾਉਣ ਲਈ ਟੂਲਜ਼ ਅਤੇ ਏਜੰਟਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਦੇ ਹਨ:

- **ਟੂਲਜ਼:** ਟੂਲਜ਼ ਵਸ਼ਿਸ਼ ਕਾਰਵਾਈਆਂ ਜਾਂ ਗਣਨਾਵਾਂ ਕਰਨ ਵਾਲੇ ਵੱਖਰੇ ਸਾਫਟਵੇਅਰ ਭਾਗ ਜਾਂ ਏਪੀਆਈਜ਼ ਹੁੰਦੇ ਹਨ। ਇਹਨਾਂ ਨੂੰ ਹੋਰ ਮੋਡੀਊਲਾਂ ਦੁਆਰਾ, ਜਵਿੇਂ ਕਿ ਜਨਰੇਟਰ ਜਾਂ ਰਟਿਰੀਵਰ, ਉਪ-ਕਾਰਜਾਂ ਨੂੰ ਪੂਰਾ ਕਰਨ ਜਾਂ ਵਾਧੂ ਜਾਣਕਾਰੀ ਇਕੱਠੀ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਟੂਲਜ਼ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਵਚਿ ਵੈਬ ਖੋਜ ਇੰਜਣ, ਕੈਲਕੁਲੇਟਰ, ਜਾਂ ਡੇਟਾ ਵਜ਼ੁਅਲਾਈਜ਼ੇਸ਼ਨ ਲਾਇਬ੍ਰੇਰੀਆਂ ਸ਼ਾਮਲ ਹਨ।
- **ਏਜੰਟ:** ਏਜੰਟ ਸਵੈ-ਚਲਤਿ ਇਕਾਈਆਂ ਹਨ ਜੋ ਆਪਣੇ ਵਾਤਾਵਰਣ ਨੂੰ ਸਮਝ ਸਕਦੀਆਂ ਹਨ, ਫੈਸਲੇ ਲੈ ਸਕਦੀਆਂ ਹਨ, ਅਤੇ ਵਸ਼ਿਸ਼ ਟੀਚਿਆਂ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਕਾਰਵਾਈਆਂ ਕਰ ਸਕਦੀਆਂ ਹਨ। ਉਹ ਅਕਸਰ ਗਤੀਸ਼ੀਲ ਜਾਂ ਅਨਸ਼ਿਚਤ ਸਥਤਿੀਆਂ ਵਚਿ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਕੰਮ ਕਰਨ ਲਈ ਯੋਜਨਾਬੰਦੀ, ਤਰਕ, ਅਤੇ ਸਖਿਣ ਵਰਗੀਆਂ ਵੱਖ-ਵੱਖ ਏਆਈ ਤਕਨੀਕਾਂ ਦੇ ਸੁਮੇਲ 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹਨ। ਏਜੰਟਾਂ ਦੀ ਵਰਤੋਂ ਗੁੰਝਲਦਾਰ ਵਵਿਹਾਰਾਂ ਨੂੰ ਮਾਡਲ ਕਰਨ ਜਾਂ ਮਸ਼ਿਰਤਿ ਏਆਈ ਸਸਿਟਮ ਦੇ ਅੰਦਰ ਕਈ ਮੋਡੀਊਲਾਂ ਦੀਆਂ ਕਾਰਵਾਈਆਂ ਦਾ ਤਾਲਮੇਲ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।

ਇੱਕ ਸੁੱਧ ਮਸ਼ਿਰਤਿ ਏਆਈ ਸਸਿਟਮ ਵਚਿ, ਇਹਨਾਂ ਭਾਗਾਂ ਵਚਿਕਾਰ ਗੱਲਬਾਤ ਚੰਗੀ ਤਰ੍ਹਾਂ ਪਰਭਾਸ਼ਤਿ ਇੰਟਰਫੇਸਾਂ ਅਤੇ ਸੰਚਾਰ ਪ੍ਰੋਟੋਕੋਲਾਂ ਰਾਹੀਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਡੇਟਾ ਮੋਡੀਊਲਾਂ ਵਚਿਕਾਰ ਵਹਾਂਦਿਾ ਹੈ, ਜਥਿੇ ਇੱਕ ਭਾਗ ਦਾ ਆਉਟਪੁੱਟ ਦੂਜੇ ਲਈ ਇਨਪੁੱਟ ਵਜੋਂ ਕੰਮ ਕਰਦਾ ਹੈ। ਇਹ ਮੋਡੀਊਲਰ ਆਰਕੀਟੈਕਚਰ ਲਚਕਤਾ, ਸਕੇਲੇਬਲਿਟੀ, ਅਤੇ ਰੱਖ-ਰਖਾਅ ਦੀ ਸਹੂਲਤ ਦਾਂਦਿਾ ਹੈ, ਕਉਂਕਿ ਵਅਿਕਤੀਗਤ ਭਾਗਾਂ ਨੂੰ ਪੂਰੇ ਸਸਿਟਮ ਨੂੰ ਪ੍ਰਭਾਵਤਿ ਕੀਤੇ ਬਨਿਾਂ ਅੱਪਡੇਟ, ਬਦਲਿਆ, ਜਾਂ ਵਸਿਤਾਰਤਿ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।

ਇਹਨਾਂ ਭਾਗਾਂ ਅਤੇ ਉਹਨਾਂ ਦੀ ਅੰਤਰਕਰਿਆ ਦੀ ਸ਼ਕਤੀ ਦਾ ਲਾਭ ਉਠਾ ਕੇ, ਮਸ਼ਿਰਤਿ ਏਆਈ ਸਸਿਟਮ ਗੁੰਝਲਦਾਰ, ਅਸਲ-ਸੰਸਾਰ ਦੀਆਂ ਸਮੱਸਿਆਵਾਂ ਨੂੰ ਹੱਲ ਕਰ ਸਕਦੇ ਹਨ ਜਨ੍ਹਿਾਂ ਲਈ ਵੱਖ-ਵੱਖ ਏਆਈ ਸਮਰੱਥਾਵਾਂ ਦੇ ਸੁਮੇਲ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਜਵਿੇਂ ਅਸੀਂ ਐਪਲੀਕੇਸ਼ਨ ਵਕਿਾਸ ਵਚਿ ਏਆਈ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਲਈ ਪਹੁੰਚਾਂ ਅਤੇ ਪੈਟਰਨਾਂ ਦੀ ਪੜਚੋਲ ਕਰਦੇ ਹਾਂ, ਇਹ ਯਾਦ ਰੱਖੋ ਕਿ ਮਸ਼ਿਰਤਿ ਏਆਈ ਸਸਿਟਮਾਂ ਵਚਿ ਵਰਤੇ ਜਾਂਦੇ ਉਹੀ ਸਧਿਾਂਤ ਅਤੇ ਤਕਨੀਕਾਂ ਬੁੱਧੀਮਾਨ, ਅਨੁਕੂਲ, ਅਤੇ ਉਪਭੋਗਤਾ-ਕੇਂਦਰਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣ ਲਈ ਵਰਤੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ।

ਭਾਗ 1 ਦੇ ਅਗਲੇ ਅਧਿਆਵਾਂ ਵਿੱਚ, ਅਸੀਂ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਵਿਕਾਸ ਪ੍ਰਕਰਿਆ ਵਿੱਚ ਏ.ਆਈ. ਕੰਪੋਨੈਂਟਸ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਲਈ ਮੁੱਢਲੇ ਪਹੁੰਚਾਂ ਅਤੇ ਤਕਨੀਕਾਂ ਵਿੱਚ ਡੂੰਘਾਈ ਨਾਲ ਜਾਵਾਂਗੇ। ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਅਤੇ ਰਟਿਰੀਵਲ-ਔਗਮੈਂਟਡ ਜਨਰੇਸ਼ਨ ਤੋਂ ਲੈ ਕੇ ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਅਤੇ ਬੁੱਧੀਮਾਨ ਵਿਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ ਤੱਕ, ਅਸੀਂ ਤੁਹਾਨੂੰ ਅਤਿ-ਆਧੁਨਿਕ ਏ.ਆਈ.-ਸੰਚਾਲਿਤ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਨ ਲਈ ਵੱਖ-ਵੱਖ ਪੈਟਰਨਾਂ ਅਤੇ ਸਰਵੋਤਮ ਅਭਿਆਸਾਂ ਨੂੰ ਕਵਰ ਕਰਾਂਗੇ।

ਭਾਗ 1: ਮੁੱਢਲੇ ਪਹੁੰਚ ਅਤੇ ਤਕਨੀਕਾਂ

ਕਤਿਾਬ ਦਾ ਇਹ ਭਾਗ ਤੁਹਾਡੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵੱਿਚ AI ਦੀ ਵਰਤੋਂ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਦੇ ਵੱਖ-ਵੱਖ ਤਰੀਕਆਿਂ ਨੂੰ ਪੇਸ਼ ਕਰਦਾ ਹੈ। ਇਹ ਅਧਆਿਇ ਸੰਬੰਧਤਿ ਪਹੁੰਚਾਂ ਅਤੇ ਤਕਨੀਕਾਂ ਦੀ ਇੱਕ ਲੜੀ ਨੂੰ ਕਵਰ ਕਰਦੇ ਹਨ, ਜੋ ਰਾਹ ਨੂੰ ਸੀਮਤ ਕਰਨਾ ਅਤੇ ਰਟ੍ਰਿੀਵਲ ਔਗਮੈਟਡ ਜਨਰੇਸ਼ਨ ਵਰਗੇ ਉੱਚ-ਪੱਧਰੀ ਸੰਕਲਪਾਂ ਤੋਂ ਲੈ ਕੇ LLM ਚੈਟ ਕੰਪਲੀਸ਼ਨ APIs ਉੱਤੇ ਆਪਣੀ ਐਬਸਟ੍ਰੈਕਸ਼ਨ ਲੇਅਰ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਕਰਨ ਦੇ ਵਚਿਾਰਾਂ ਤੱਕ ਫੈਲੇ ਹੋਏ ਹਨ।

ਕਤਿਾਬ ਦੇ ਇਸ ਭਾਗ ਦਾ ਉਦੇਸ਼ ਭਾਗ 2 ਦੇ ਕੇਂਦਰ ਬਂਦੂ ਵਾਲੇ ਵਸ਼ਿੇਸ਼ ਲਾਗੂਕਰਨ ਪੈਟਰਨਾਂ ਵੱਿਚ ਬਹੁਤ ਡੂੰਘੇ ਜਾਣ ਤੋਂ ਪਹਲਿਾਂ, ਤੁਹਾਨੂੰ AI ਨਾਲ ਲਾਗੂ ਕੀਤੇ ਜਾ ਸਕਣ ਵਾਲੇ ਵਵਿਹਾਰਾਂ ਦੀਆਂ ਕਸਿਮਾਂ ਨੂੰ ਸਮਝਣ ਵੱਿਚ ਮਦਦ ਕਰਨਾ ਹੈ।

ਭਾਗ 1 ਵੱਿਚ ਦੱਸੀਆਂ ਪਹੁੰਚਾਂ ਉਨ੍ਹਾਂ ਵਚਿਾਰਾਂ ’ਤੇ ਆਧਾਰਤਿ ਹਨ ਜੋ ਮੈਂ ਆਪਣੇ ਕੋਡ ਵੱਿਚ ਵਰਤੇ ਹਨ, ਐਂਟਰਪ੍ਰਾਈਜ਼ ਐਪਲੀਕੇਸ਼ਨ ਆਰਕੀਟੈਕਚਰ ਅਤੇ ਏਕੀਕਰਨ ਦੇ ਕਲਾਸਕਿ ਪੈਟਰਨ, ਅਤੇ ਨਾਲ ਹੀ ਉਹ ਰੂਪਕ ਜੋ ਮੈਂ ਹੋਰ ਲੋਕਾਂ ਨੂੰ, ਜਨ੍ਹਿਾਂ ਵੱਿਚ ਗੈਰ-ਤਕਨੀਕੀ ਵਪਾਰਕ ਹੱਸਿੇਦਾਰ ਵੀ ਸ਼ਾਮਲ ਹਨ, ਨੂੰ AI ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਸਮਝਾਉਣ ਵੇਲੇ ਵਰਤੇ ਹਨ।

ਰਾਹ ਨੂੰ ਤੰਗ ਕਰੋ

“ਰਾਹ ਨੂੰ ਤੰਗ ਕਰਨਾ” ਦਾ ਮਤਲਬ ਹੈ AI ਨੂੰ ਦਤਿੰਤੇ ਗਏ ਕਾਰਜ ’ਤੇ ਕੇਂਦਰਤਿ ਕਰਨਾ। ਮੈਂ ਇਸ ਨੂੰ ਇੱਕ ਮੰਤਰ ਵਜੋਂ ਵਰਤਦਾ ਹਾਂ ਜਦੋਂ ਵੀ ਮੈਨੂੰ AI ਦੇ “ਮੂਰਖ” ਜਾਂ ਅਣਚਾਹੇ ਤਰੀਕੇ ਨਾਲ ਕੰਮ ਕਰਨ ਤੇ ਨਰਿਾਸ਼ਾ ਹੁੰਦੀ ਹੈ। ਇਹ ਮੰਤਰ ਮੈਨੂੰ ਯਾਦ ਦਵਿਾਉਂਦਾ ਹੈ ਕ ਿਅਸਫਲਤਾ ਸ਼ਾਇਦ ਮੇਰੀ ਗਲਤੀ ਹੈ, ਅਤੇ ਮੈਨੂੰ ਸ਼ਾਇਦ ਹੋਰ ਰਾਹ ਨੂੰ ਤੰਗ ਕਰਨ ਦੀ ਲੋੜ ਹੈ।

ਰਾਹ ਨੂੰ ਤੰਗ ਕਰਨ ਦੀ ਲੋੜ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਵਚਿ ਮੌਜੂਦ ਵਸ਼ਿਾਲ ਗਆਿਨ ਤੋਂ ਪੈਦਾ ਹੁੰਦੀ ਹੈ, ਖਾਸ ਕਰਕੇ OpenAI ਅਤੇ Anthropic ਵਰਗੇ ਵਸ਼ਿਵ-ਪੱਧਰੀ ਮਾਡਲਾਂ ਵਚਿ ਜਨ੍ਹਿਾਂ ਵਚਿ ਸ਼ਾਬਦਕ ਿਤੌਰ ’ਤੇ ਖਰਬਾਂ ਪੈਰਾਮੀਟਰ ਹਨ।

ਇੰਨੇ ਵਸ਼ਿਾਲ ਗਿਆਨ ਤੱਕ ਪਹੁੰਚ ਹੋਣਾ ਨਸ਼ਿਚਤਿ ਤੌਰ 'ਤੇ ਸ਼ਕਤੀਸ਼ਾਲੀ ਹੈ ਅਤੇ ਮਨ ਦਾ ਸਧਿਾਂਤ ਅਤੇ ਮਨੁੱਖੀ ਤਰੀਕਿਆਂ ਨਾਲ ਤਰਕ ਕਰਨ ਦੀ ਯੋਗਤਾ ਵਰਗੇ ਉੱਭਰਦੇ ਵਵਿਹਾਰ ਨੂੰ ਪੈਦਾ ਕਰਦਾ ਹੈ। ਹਾਲਾਂਕ,ਿ ਜਾਣਕਾਰੀ ਦੀ ਇਹ ਧਰਤੀ-ਹਲਿਾਉ ਮਾਤਰਾ ਵਸ਼ਿ ਪ੍ਰੌਂਪਟਸ ਲਈ ਸਟੀਕ ਅਤੇ ਸਹੀ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਵ਼ਚਿ ਚੁਣੌਤੀਆਂ ਵੀ ਪੇਸ਼ ਕਰਦੀ ਹੈ, ਖਾਸ ਕਰਕੇ ਜੇ ਉਹ ਪ੍ਰੌਂਪਟਸ ਨਰਿਧਾਰਤ ਵਵਿਹਾਰ ਦਖਿਾਉਣ ਲਈ ਹਨ ਜੋ "ਸਧਾਰਨ" ਸੌਫਟਵੇਅਰ ਵਕਿਾਸ ਅਤੇ ਐਲਗੋਰਦਿਮ ਨਾਲ ਏਕੀਕ੍ਰਤਿ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ।

ਕਈ ਕਾਰਕ ਇਨ੍ਹਾਂ ਚੁਣੌਤੀਆਂ ਵੱਲ ਲੈ ਜਾਂਦੇ ਹਨ।

ਜਾਣਕਾਰੀ ਦੀ ਓਵਰਲੋਡ: ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਖੇਤਰਾਂ, ਸਰੋਤਾਂ, ਅਤੇ ਸਮੇਂ ਦੀਆਂ ਮਆਿਦਾਂ ਨੂੰ ਕਵਰ ਕਰਦੇ ਵਸ਼ਿਾਲ ਡੇਟਾ 'ਤੇ ਸਖਿਲਾਈ ਦੱਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਵਆਿਪਕ ਗਿਆਨ ਉਨ੍ਹਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਵਸ਼ਿਆਂ ਵ਼ਚਿ ਸ਼ਾਮਲ ਹੋਣ ਅਤੇ ਦੁਨੀਆ ਦੀ ਵਆਿਪਕ ਸਮਝ ਦੇ ਆਧਾਰ 'ਤੇ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦੱਿਦਾ ਹੈ। ਹਾਲਾਂਕ,ਿ ਇੱਕ ਵਸ਼ਿ ਪ੍ਰੌਂਪਟ ਦਾ ਸਾਹਮਣਾ ਕਰਨ 'ਤੇ, ਮਾਡਲ ਨੂੰ ਅਪ੍ਰਸੰਗਕਿ, ਵਰਿੋਧੀ, ਜਾਂ ਪੁਰਾਣੀ/ਅਪ੍ਰਚਲਤਿ ਜਾਣਕਾਰੀ ਨੂੰ ਛਾਂਟਣ ਵ਼ਚਿ ਮੁਸ਼ਕਲ ਹੋ ਸਕਦੀ ਹੈ, ਜਸਿ ਨਾਲ ਫੋਕਸ ਜਾਂ ਸਟੀਕਤਾ ਦੀ ਕਮੀ ਵਾਲੇ ਜਵਾਬ ਮਲਿਦੇ ਹਨ। ਤੁਸੀਂ ਕੀ ਕਰਨ ਦੀ ਕੋਸ਼ਸ਼ ਕਰ ਰਹੇ ਹੋ, ਇਸ 'ਤੇ ਨਰਿਭਰ ਕਰਦਆਿਂ, ਮਾਡਲ ਲਈ ਉਪਲਬਧ ਵਰਿੋਧੀ ਜਾਣਕਾਰੀ ਦੀ ਸਰਫ਼ ਮਾਤਰਾ ਹੀ ਉਸ ਜਵਾਬ ਜਾਂ ਵਵਿਹਾਰ ਨੂੰ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਇਸਦੀ ਯੋਗਤਾ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਓਵਰਵਹੇਲਮ ਕਰ ਸਕਦੀ ਹੈ ਜੋ ਤੁਸੀਂ ਚਾਹੁੰਦੇ ਹੋ।

ਸੰਦਰਭ ਦੀ ਅਸਪੱਸ਼ਟਤਾ: ਗਿਆਨ ਦੇ ਵਸ਼ਿਾਲ ਛਪੇ ਹੋਏ ਸਪੇਸ ਨੂੰ ਦੇਖਦੇ ਹੋਏ, ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨੂੰ ਤੁਹਾਡੇ ਪ੍ਰੌਂਪਟ ਦੇ ਸੰਦਰਭ ਨੂੰ ਸਮਝਣ ਦੀ ਕੋਸ਼ਸ਼ ਕਰਦੇ ਸਮੇਂ ਅਸਪੱਸ਼ਟਤਾ ਦਾ ਸਾਹਮਣਾ ਕਰਨਾ ਪੈ ਸਕਦਾ ਹੈ। ਉਚਤਿ ਤੰਗ ਕਰਨ ਜਾਂ ਮਾਰਗਦਰਸ਼ਨ ਤੋਂ ਬਨਿਾਂ, ਮਾਡਲ ਅਜਹਿੇ ਜਵਾਬ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ ਜੋ ਸਤਹੀ ਤੌਰ 'ਤੇ ਸੰਬੰਧਤਿ ਹਨ ਪਰ ਤੁਹਾਡੇ ਇਰਾਦਆਿਂ ਲਈ ਸੱਧਿੇ ਤੌਰ 'ਤੇ ਪ੍ਰਸੰਗਕਿ ਨਹੀਂ ਹਨ। ਇਸ ਤਰ੍ਹਾਂ ਦੀ ਅਸਫਲਤਾ ਵਸ਼ਿ ਤੋਂ ਹਟ ਕੇ, ਅਸੰਗਤ, ਜਾਂ ਤੁਹਾਡੀਆਂ ਦੱਸੀਆਂ ਲੋੜਾਂ ਨੂੰ ਪੂਰਾ ਕਰਨ ਵ਼ਚਿ ਅਸਫਲ ਜਵਾਬਾਂ ਵੱਲ ਲੈ ਜਾਂਦੀ ਹੈ। ਇਸ ਮਾਮਲੇ ਵ਼ਚਿ, ਰਾਹ ਨੂੰ ਤੰਗ ਕਰਨ ਦਾ ਮਤਲਬ ਹੈ ਸੰਦਰਭ ਦੀ ਸਪੱਸ਼ਟਤਾ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਕ ਿਤੁਹਾਡੇ ਦੁਆਰਾ ਪ੍ਰਦਾਨ ਕੀਤਾ ਸੰਦਰਭ ਮਾਡਲ ਨੂੰ ਸਰਫ਼ ਇਸਦੇ ਬੇਸ ਗਿਆਨ ਵ਼ਚਿ ਸਭ ਤੋਂ ਪ੍ਰਸੰਗਕਿ ਜਾਣਕਾਰੀ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰਨ ਦਾ ਕਾਰਨ ਬਣਦਾ ਹੈ।

ਨੋਟ: ਜਦੋਂ ਤੁਸੀਂ "ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰੰਗਿ" ਨਾਲ ਸ਼ੁਰੂਆਤ ਕਰ ਰਹੇ ਹੋ, ਤਾਂ ਤੁਹਾਡੇ ਵੱਲੋਂ ਮਾਡਲ ਨੂੰ ਲੋੜੀਂਦੇ ਨਤੀਜੇ ਦੀ ਸਹੀ ਵਆਿਖਆਿ ਕੀਤੇ ਬਨਿਾਂ ਕੰਮ ਕਰਨ ਲਈ ਕਹਣਿ ਦੀ ਸੰਭਾਵਨਾ ਵੱਧ ਹੁੰਦੀ ਹੈ; ਅਸਪੱਸ਼ਟ ਨਾ ਹੋਣ ਲਈ ਅਭਆਿਸ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ!

ਸਮੇਂ ਸਬੰਧੀ ਅਸੰਗਤੀਆਂ: ਕਉਂਕ ਿਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਸਮੇਂ ਦੌਰਾਨ ਬਣਾਏ ਗਏ ਡੇਟਾ 'ਤੇ ਸਖਿਲਾਈ ਦੱਤੀ ਜਾਂਦੀ ਹੈ, ਉਨ੍ਹਾਂ ਕੋਲ ਅਜਹਿੀ ਜਾਣਕਾਰੀ ਹੋ ਸਕਦੀ ਹੈ ਜੋ ਪੁਰਾਣੀ, ਬਦਲੀ ਹੋਈ, ਜਾਂ ਹੁਣ ਸਹੀ ਨਹੀਂ ਹੈ। ਉਦਾਹਰਣ

ਲਈ, ਮੌਜੂਦਾ ਘਟਨਾਵਾਂ, ਵਿਗਿਆਨਕ ਖੋਜਾਂ, ਜਾਂ ਤਕਨੀਕੀ ਵਿਕਾਸ ਬਾਰੇ ਜਾਣਕਾਰੀ ਮਾਡਲ ਦੇ ਸਖਿਲਾਈ ਡੇਟਾ ਦੇ ਇਕੱਠਾ ਕੀਤੇ ਜਾਣ ਤੋਂ ਬਾਅਦ ਬਦਲ ਗਈ ਹੋ ਸਕਦੀ ਹੈ। ਵਧੇਰੇ ਤਾਜ਼ਾ ਅਤੇ ਭਰੋਸੇਯੋਗ ਸਰੋਤਾਂ ਨੂੰ ਪ੍ਰਾਥਮਕਿਤਾ ਦੇਣ ਲਈ ਰਸਤੇ ਨੂੰ ਸੀਮਤਿ ਕੀਤੇ ਬਨਿਾਂ, ਮਾਡਲ ਪੁਰਾਣੀ ਜਾਂ ਗਲਤ ਜਾਣਕਾਰੀ ਦੇ ਆਧਾਰ ’ਤੇ ਜਵਾਬ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ, ਜਸਿ ਨਾਲ ਇਸਦੇ ਆਉਟਪੁੱਟ ਵਚਿ ਅਸੁੱਧੀਆਂ ਅਤੇ ਅਸੰਗਤੀਆਂ ਪੈਦਾ ਹੋ ਸਕਦੀਆਂ ਹਨ।

ਖੇਤਰ-ਵਸ਼ਿਸ਼ ਬਾਰੀਕੀਆਂ: ਵੱਖ-ਵੱਖ ਖੇਤਰਾਂ ਅਤੇ ਵਸ਼ਿਆਂ ਦੀ ਆਪਣੀ ਵਸ਼ਿਸ਼ ਸ਼ਬਦਾਵਲੀ, ਰਵਾਇਤਾਂ, ਅਤੇ ਗਆਿਨ ਅਧਾਰ ਹੁੰਦੇ ਹਨ। ਕਸਿੇ ਵੀ TLA (ਤਨਿ ਅੱਖਰੀ ਸੰਖੇਪ) ਬਾਰੇ ਸੋਚੋ ਅਤੇ ਤੁਹਾਨੂੰ ਅਹਸਿਾਸ ਹੋਵੇਗਾ ਕ ਿਉਨ੍ਹਾਂ ਵਚਿੋਂ ਜ਼ਆਿਦਾਤਰ ਦੇ ਇੱਕ ਤੋਂ ਵੱਧ ਅਰਥ ਹਨ। ਉਦਾਹਰਣ ਵਜੋਂ, MSK ਐਮਾਜ਼ੋਨ ਦੇ ਮੈਨੇਜਡ ਸਟ੍ਰੀਮਿੰਗ ਫਾਰ ਅਪਾਚੇ ਕਾਫ਼ਕਾ, ਮੈਮੋਰੀਅਲ ਸਲੋਨ ਕੈਟਰਿੰਗ ਕੈਂਸਰ ਸੈਂਟਰ, ਜਾਂ ਮਨੁੱਖੀ ਮਸਕੁਲੋਸਕੈਲੇਟਲ ਸਸਿਟਮ ਨੂੰ ਦਰਸਾ ਸਕਦਾ ਹੈ।

ਜਦੋਂ ਕਸਿ ਪ੍ਰੋਮਪਟ ਲਈ ਕਸਿ ਖਾਸ ਖੇਤਰ ਵਚਿ ਮੁਹਾਰਤ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਤਾਂ ਇੱਕ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ ਦਾ ਆਮ ਗਆਿਨ ਸਹੀ ਅਤੇ ਬਾਰੀਕ ਜਵਾਬ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਕਾਫ਼ੀ ਨਹੀ ਹੋ ਸਕਦਾ। ਖੇਤਰ-ਵਸ਼ਿਸ਼ ਜਾਣਕਾਰੀ ’ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰਕੇ ਰਸਤੇ ਨੂੰ ਸੀਮਤਿ ਕਰਨਾ, ਭਾਵੇਂ ਪ੍ਰੋਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਰਾਹੀ ਜਾਂ ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਵਧਤਿ ਉਤਪਾਦਨ ਰਾਹੀਂ, ਮਾਡਲ ਨੂੰ ਅਜਹਿੇ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦਾ ਹੈ ਜੋ ਤੁਹਾਡੇ ਵਸ਼ਿਸ਼ ਖੇਤਰ ਦੀਆਂ ਲੋੜਾਂ ਅਤੇ ਉਮੀਦਾਂ ਦੇ ਵਧੇਰੇ ਅਨੁਕੂਲ ਹਨ।

ਲੇਟੈਂਟ ਸਪੇਸ: ਅਕਲਪਨੀ ਤੌਰ ’ਤੇ ਵਸ਼ਿਾਲ

ਜਦੋਂ ਮੈਂ ਭਾਸ਼ਾ ਮਾਡਲ ਦੇ “ਲੇਟੈਂਟ ਸਪੇਸ” ਦਾ ਜ਼ਕਿਰ ਕਰਦਾ ਹਾਂ, ਤਾਂ ਮੈਂ ਗਆਿਨ ਅਤੇ ਜਾਣਕਾਰੀ ਦੇ ਵਸ਼ਿਾਲ, ਬਹੁ-ਆਯਾਮੀ ਖੇਤਰ ਦਾ ਹਵਾਲਾ ਦੇ ਰਹਿਾ ਹਾਂ ਜੋ ਮਾਡਲ ਨੇ ਆਪਣੀ ਸਖਿਲਾਈ ਪ੍ਰਕਰਿਆ ਦੌਰਾਨ ਸੰਖਿਆ ਹੈ। ਇਹ ਮਾਡਲ ਦੇ ਨਊਿਰਲ ਨੈੱਟਵਰਕਸ ਦੇ ਅੰਦਰ ਇੱਕ ਛੁਪਆਿ ਹੋਇਆ ਰਾਜ ਵਰਗਾ ਹੈ, ਜੱਥੇ ਭਾਸ਼ਾ ਦੇ ਸਾਰੇ ਪੈਟਰਨ, ਸੰਬੰਧ, ਅਤੇ ਪ੍ਰਤੀਨਧਿਤਾਵਾਂ ਸਟੋਰ ਕੀਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ।

ਕਲਪਨਾ ਕਰੋ ਕ ਿਤੁਸੀਂ ਇੱਕ ਵਸ਼ਿਾਲ, ਅਣਖੋਜੇ ਖੇਤਰ ਦੀ ਖੋਜ ਕਰ ਰਹੇ ਹੋ ਜੋ ਅਣਗਣਿਤ ਆਪਸ ਵਚਿ ਜੁੜੇ ਨੋਡਸ ਨਾਲ ਭਰਆਿ ਹੋਇਆ ਹੈ। ਹਰ ਨੋਡ ਇੱਕ ਜਾਣਕਾਰੀ ਦਾ ਟੁਕੜਾ, ਇੱਕ ਧਾਰਨਾ, ਜਾਂ ਇੱਕ ਸੰਬੰਧ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਜੋ ਮਾਡਲ ਨੇ ਸੰਖਿਆ ਹੈ। ਜਵਿੇਂ ਤੁਸੀਂ ਇਸ ਸਪੇਸ ਵਚਿ ਨੈਵੀਗੇਟ ਕਰਦੇ ਹੋ, ਤੁਹਾਨੂੰ ਪਤਾ ਲੱਗੇਗਾ ਕ ਿਕੁਝ ਨੋਡਸ ਇੱਕ ਦੂਜੇ ਦੇ ਨੇੜੇ ਹਨ, ਜੋ ਇੱਕ ਮਜ਼ਬੂਤ ਕਨੈਕਸ਼ਨ ਜਾਂ ਸਮਾਨਤਾ ਨੂੰ ਦਰਸਾਉਂਦੇ ਹਨ, ਜਦੋਂ ਕ ਿਦੂਸਰੇ ਇੱਕ ਦੂਜੇ ਤੋਂ ਦੂਰ ਹਨ, ਜੋ ਇੱਕ ਕਮਜ਼ੋਰ ਜਾਂ ਵਧੇਰੇ ਦੂਰ ਦਾ ਸੰਬੰਧ ਦਰਸਾਉਂਦੇ ਹਨ।

ਲੇਟੈਂਟ ਸਪੇਸ ਨਾਲ ਚੁਣੌਤੀ ਇਹ ਹੈ ਕ ਿਇਹ ਬੇਹੱਦ ਗੁੰਝਲਦਾਰ ਅਤੇ ਉੱਚ-ਆਯਾਮੀ ਹੈ। ਇਸਨੂੰ ਸਾਡੇ ਭੌਤਕਿ

ਬ੍ਰਹਮਿੰਡ ਜੰਨਿਾ ਵਸਿ਼ਾਲ ਸਮਝੋ, ਜਸਿ ਵੰਚਿ ਗਲੈਕਸੀਆਂ ਦੇ ਸਮੂਹ ਅਤੇ ਉਨ੍ਹਾਂ ਵਚਿਕਾਰ ਖਾਲੀ ਸਪੇਸ ਦੀਆਂ ਅਕਲਪਨੀ ਦੂਰੀਆਂ ਹਨ।

ਕਉਂਕਿ ਇਸ ਵੰਚਿ ਹਜ਼ਾਰਾਂ ਆਯਾਮ ਹਨ, ਲੇਟੈਂਟ ਸਪੇਸ ਮਨੁੱਖਾਂ ਦੁਆਰਾ ਸੰਧੇ ਤੌਰ 'ਤੇ ਦੇਖਆਿ ਜਾਂ ਸਮਝਆਿ ਨਹੀਂ ਜਾ ਸਕਦਾ। ਇਹ ਇੱਕ ਸਾਰ ਪ੍ਰਤੀਨਧਿਤਾ ਹੈ ਜੋ ਮਾਡਲ ਭਾਸ਼ਾ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਨ ਅਤੇ ਤਆਿਰ ਕਰਨ ਲਈ ਅੰਦਰੂਨੀ ਤੌਰ 'ਤੇ ਵਰਤਦਾ ਹੈ। ਜਦੋਂ ਤੁਸੀਂ ਮਾਡਲ ਨੂੰ ਇੱਕ ਇਨਪੁੱਟ ਪ੍ਰੌਮਪਟ ਦੰਦ੿ ਹੋ, ਇਹ ਅਸਲ ਵੰਚਿ ਉਸ ਪ੍ਰੌਮਪਟ ਨੂੰ ਲੇਟੈਂਟ ਸਪੇਸ ਦੇ ਅੰਦਰ ਇੱਕ ਖਾਸ ਸਥਾਨ 'ਤੇ ਮੈਪ ਕਰਦਾ ਹੈ। ਫਰਿ ਮਾਡਲ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਲਈ ਉਸ ਸਥਾਨ ਦੇ ਆਲੇ-ਦੁਆਲੇ ਦੀ ਜਾਣਕਾਰੀ ਅਤੇ ਕਨੈਕਸ਼ਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ।

ਗੱਲ ਇਹ ਹੈ ਕ ਿਮਾਡਲ ਨੇ ਆਪਣੇ ਸਖਿਲਾਈ ਡਾਟਾ ਤੋਂ ਬਹੁਤ ਜ਼ਆਿਦਾ ਜਾਣਕਾਰੀ ਸੰਖੀ ਹੈ, ਅਤੇ ਇਹ ਸਾਰੀ ਕਸਿੰ ਦੰਤਿੋ ਕੰਮ ਲਈ ਢੁਕਵੀਂ ਜਾਂ ਸਹੀ ਨਹੀਂ ਹੈ। ਇਸੇ ਲਈ ਰਸਤੇ ਨੂੰ ਸੰਕੁਚਤਿ ਕਰਨਾ ਬਹੁਤ ਮਹੱਤਵਪੂਰਨ ਹੋ ਜਾਂਦਾ ਹੈ। ਆਪਣੇ ਪ੍ਰੌਮਪਟਸ ਵੰਚਿ ਸਪੱਸ਼ਟ ਨਰਿਦੇਸ਼, ਉਦਾਹਰਣਾਂ, ਅਤੇ ਸੰਦਰਭ ਪ੍ਰਦਾਨ ਕਰਕੇ, ਤੁਸੀਂ ਅਸਲ ਵੰਚਿ ਮਾਡਲ ਨੂੰ ਲੇਟੈਂਟ ਸਪੇਸ ਦੇ ਖਾਸ ਖੇਤਰਾਂ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰਨ ਲਈ ਮਾਰਗਦਰਸ਼ਨ ਕਰ ਰਹੇ ਹੋ ਜੋ ਤੁਹਾਡੇ ਇੱਛਤ ਆਉਟਪੁੱਟ ਲਈ ਸਭ ਤੋਂ ਢੁਕਵੇਂ ਹਨ।

ਇਸ ਬਾਰੇ ਸੋਚਣ ਦਾ ਇੱਕ ਵੱਖਰਾ ਤਰੀਕਾ ਪੂਰੀ ਤਰ੍ਹਾਂ ਹਨੇਰੇ ਅਜਾਇਬ ਘਰ ਵੰਚਿ ਸਪੌਟਲਾਈਟ ਦੀ ਵਰਤੋਂ ਕਰਨ ਵਾਂਗ ਹੈ। ਜੇ ਤੁਸੀਂ ਕਦੇ Louvre ਜਾਂ Metropolitan Museum of Art ਗਏ ਹੋ, ਤਾਂ ਇਹ ਉਸ ਤਰ੍ਹਾਂ ਦਾ ਪੈਮਾਨਾ ਹੈ ਜਸਿ ਬਾਰੇ ਮੈਂ ਗੱਲ ਕਰ ਰਹਿਾ ਹਾਂ। ਲੇਟੈਂਟ ਸਪੇਸ ਅਜਾਇਬ ਘਰ ਹੈ, ਜੋ ਅਣਗਣਿਤ ਵਸਤੂਆਂ ਅਤੇ ਵੇਰਵਆਿਂ ਨਾਲ ਭਰਆਿ ਹੋਇਆ ਹੈ। ਤੁਹਾਡਾ ਪ੍ਰੌਮਪਟ ਸਪੌਟਲਾਈਟ ਹੈ, ਜੋ ਖਾਸ ਖੇਤਰਾਂ ਨੂੰ ਰੌਸ਼ਨ ਕਰਦਾ ਹੈ ਅਤੇ ਮਾਡਲ ਦਾ ਧਆਿਨ ਸਭ ਤੋਂ ਮਹੱਤਵਪੂਰਨ ਜਾਣਕਾਰੀ ਵੱਲ ਖੰਚਿਦਾ ਹੈ। ਉਸ ਮਾਰਗਦਰਸ਼ਨ ਤੋਂ ਬਨਿਾਂ, ਮਾਡਲ ਲੇਟੈਂਟ ਸਪੇਸ ਵੰਚਿ ਬੇਮਕਸਦ ਭਟਕ ਸਕਦਾ ਹੈ, ਰਾਹ ਵੰਚਿ ਅਢੁਕਵੀਂ ਜਾਂ ਵਰਿੋਧੀ ਜਾਣਕਾਰੀ ਇਕੱਠੀ ਕਰਦਾ ਹੋਇਆ।

ਜਵਿੰ ਤੁਸੀਂ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨਾਲ ਕੰਮ ਕਰਦੇ ਹੋ ਅਤੇ ਆਪਣੇ ਪ੍ਰੌਮਪਟਸ ਤਆਿਰ ਕਰਦੇ ਹੋ, ਲੇਟੈਂਟ ਸਪੇਸ ਦੀ ਧਾਰਨਾ ਨੂੰ ਧਆਿਨ ਵੰਚਿ ਰੱਖੋ। ਤੁਹਾਡਾ ਟੀਚਾ ਇਸ ਵਸਿ਼ਾਲ ਗਆਿਨ ਭੂਗੋਲ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਨੈਵੀਗੇਟ ਕਰਨਾ ਹੈ, ਮਾਡਲ ਨੂੰ ਤੁਹਾਡੇ ਕੰਮ ਲਈ ਸਭ ਤੋਂ ਢੁਕਵੀਂ ਅਤੇ ਸਹੀ ਜਾਣਕਾਰੀ ਵੱਲ ਨਰਿਦੇਸ਼ਤਿ ਕਰਨਾ ਹੈ। ਰਸਤੇ ਨੂੰ ਸੰਕੁਚਤਿ ਕਰਕੇ ਅਤੇ ਸਪੱਸ਼ਟ ਮਾਰਗਦਰਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰਕੇ, ਤੁਸੀਂ ਮਾਡਲ ਦੇ ਲੇਟੈਂਟ ਸਪੇਸ ਦੀ ਪੂਰੀ ਸਮਰੱਥਾ ਨੂੰ ਅਨਲੌਕ ਕਰ ਸਕਦੇ ਹੋ ਅਤੇ ਉੱਚ-ਗੁਣਵੱਤਾ, ਸੁਸੰਗਤ ਜਵਾਬ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹੋ।

ਭਾਵੇਂ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਅਤੇ ਉਹਨਾਂ ਦੇ ਨੈਵੀਗੇਟ ਕਰਨ ਵਾਲੇ ਲੇਟੈਂਟ ਸਪੇਸ ਦੇ ਪਛਿਲੇ ਵਰਣਨ ਥੋੜ੍ਹੇ ਜਾਦੂਈ ਜਾਂ ਸਾਰ ਜਾਪ ਸਕਦੇ ਹਨ, ਇਹ ਸਮਝਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕ ਿਪ੍ਰੌਮਪਟਸ ਜਾਦੂ ਜਾਂ ਮੰਤਰ ਨਹੀਂ ਹਨ। ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੇ ਕੰਮ ਕਰਨ ਦਾ ਤਰੀਕਾ ਰੇਖਕਿ ਬੀਜਗਣਤਿ ਅਤੇ ਸੰਭਾਵਨਾ ਸਧਿਾਂਤ ਦੇ ਸਧਿਾਂਤਾਂ 'ਤੇ ਆਧਾਰਤਿ ਹੈ।

ਆਪਣੇ ਮੂਲ ਰੂਪ ਵੰਚਿ, ਭਾਸ਼ਾ ਮਾਡਲ ਟੈਕਸਟ ਦੇ ਸੰਭਾਵੀ ਮਾਡਲ ਹਨ, ਬਲਿਕੁਲ ਉਸੇ ਤਰ੍ਹਾਂ ਜਵਿੰ ਬੈਂਲ ਕਰਵ ਡਾਟਾ

ਦਾ ਅੰਕੜਾ ਮਾਡਲ ਹੈ। ਇਹਨਾਂ ਨੂੰ ਸਵੈ-ਪ੍ਰਤੀਗਾਮੀ ਮਾਡਲਿੰਗ ਕਹੀ ਜਾਂਦੀ ਪ੍ਰਕਰਿਆ ਰਾਹੀਂ ਸਖਿਲਾਈ ਦਿੱਤੀ ਜਾਂਦੀ ਹੈ, ਜਿੱਥੇ ਮਾਡਲ ਪਛਿਲੇ ਸ਼ਬਦਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਲੜੀ ਵਿੱਚ ਅਗਲੇ ਸ਼ਬਦ ਦੀ ਸੰਭਾਵਨਾ ਦੀ ਭਵਿੱਖਬਾਣੀ ਕਰਨਾ ਸਿੱਖਦਾ ਹੈ। ਸਖਿਲਾਈ ਦੌਰਾਨ, ਮਾਡਲ ਬੇਤਰਤੀਬੇ ਭਾਰਾਂ ਨਾਲ ਸ਼ੁਰੂ ਹੁੰਦਾ ਹੈ ਅਤੇ ਹੌਲੀ-ਹੌਲੀ ਉਹਨਾਂ ਨੂੰ ਅਜਿਹੇ ਟੈਕਸਟ ਨੂੰ ਉੱਚ ਸੰਭਾਵਨਾਵਾਂ ਦੇਣ ਲਈ ਵਵਿਸਥਤਿ ਕਰਦਾ ਹੈ ਜੋ ਉਹਨਾਂ ਅਸਲ-ਸੰਸਾਰ ਦੇ ਨਮੂਨਿਆਂ ਨਾਲ ਮਲਿਦਾ ਹੈ ਜਨ੍ਹਿਾਂ 'ਤੇ ਇਸਨੂੰ ਸਖਿਲਾਈ ਦਿੱਤੀ ਗਈ ਸੀ।

ਹਾਲਾਂਕਿ, ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨੂੰ ਸਧਾਰਨ ਅੰਕੜਾ ਮਾਡਲਾਂ ਵਾਂਗ ਸੋਚਣਾ, ਜਵਿੇਂ ਕਿ ਰੇਖਕਿ ਪ੍ਰਤੀਗਮਨ, ਉਨ੍ਹਾਂ ਦੇ ਵਵਿਹਾਰ ਨੂੰ ਸਮਝਣ ਲਈ ਸਭ ਤੋਂ ਵਧੀਆ ਸਮਝ ਨਹੀਂ ਦਿੰਦਾ। ਇੱਕ ਵਧੇਰੇ ਢੁਕਵੀਂ ਤੁਲਨਾ ਇਹ ਹੈ ਕਿ ਉਨ੍ਹਾਂ ਨੂੰ ਸੰਭਾਵੀ ਪ੍ਰੋਗਰਾਮਾਂ ਵਜੋਂ ਸੋਚਿਆ ਜਾਵੇ, ਜੋ ਅਜਿਹੇ ਮਾਡਲ ਹਨ ਜੋ ਬੇਤਰਤੀਬੇ ਵੇਰੀਏਬਲਾਂ ਦੀ ਹੇਰਫੇਰ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦੇ ਹਨ ਅਤੇ ਗੁੰਝਲਦਾਰ ਅੰਕੜਾ ਸਬੰਧਾਂ ਨੂੰ ਦਰਸਾ ਸਕਦੇ ਹਨ।

ਸੰਭਾਵੀ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਗ੍ਰਾਫਕਿਲ ਮਾਡਲਾਂ ਦੁਆਰਾ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ, ਜੋ ਮਾਡਲ ਵਿੱਚ ਵੇਰੀਏਬਲਾਂ ਵਚਿਕਾਰ ਨਰਿਭਰਤਾਵਾਂ ਅਤੇ ਸਬੰਧਾਂ ਨੂੰ ਸਮਝਣ ਦਾ ਇੱਕ ਦ੍ਰਸ਼ਟਿੀਗਤ ਤਰੀਕਾ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ। ਇਹ ਨਜ਼ਰੀਆ GPT-4 ਅਤੇ Claude ਵਰਗੇ ਗੁੰਝਲਦਾਰ ਟੈਕਸਟ ਜਨਰੇਸ਼ਨ ਮਾਡਲਾਂ ਦੀ ਕਾਰਜਪ੍ਰਣਾਲੀ ਬਾਰੇ ਮੁੱਲਵਾਨ ਅੰਤਰਦ੍ਰਸ਼ਟਿੀ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦਾ ਹੈ।

ਡੋਹਨ ਅਤੇ ਹੋਰਾਂ ਦੁਆਰਾ "ਲੈਂਗੁਏਜ ਮਾਡਲ ਕੈਸਕੇਡਜ਼" ਪੇਪਰ ਵਿੱਚ, ਲੇਖਕ ਇਸ ਗੱਲ ਦੇ ਵੇਰਵਿਆਂ ਵਿੱਚ ਜਾਂਦੇ ਹਨ ਕਿ ਸੰਭਾਵੀ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਭਾਸ਼ਾ ਮਾਡਲਾਂ 'ਤੇ ਕਵਿੇਂ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਉਹ ਦਖਿਾਉਂਦੇ ਹਨ ਕਿ ਇਸ ਫਰੇਮਵਰਕ ਨੂੰ ਇਨ੍ਹਾਂ ਮਾਡਲਾਂ ਦੇ ਵਵਿਹਾਰ ਨੂੰ ਸਮਝਣ ਅਤੇ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੌਂਪਟਿੰਗ ਰਣਨੀਤੀਆਂ ਦੇ ਵਕਿਾਸ ਨੂੰ ਨਰਿਦੇਸ਼ਤਿ ਕਰਨ ਲਈ ਕਵਿੇਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।

ਇਸ ਸੰਭਾਵੀ ਦ੍ਰਸ਼ਟਿੀਕੋਣ ਤੋਂ ਇੱਕ ਮੁੱਖ ਅੰਤਰਦ੍ਰਸ਼ਟਿੀ ਇਹ ਹੈ ਕਿ ਭਾਸ਼ਾ ਮਾਡਲ ਅਸਲ ਵਿੱਚ ਇੱਕ ਵਕਿਲਪਕਿ ਬ੍ਰਹਮਿੰਡ ਵੱਲ ਇੱਕ ਪੋਰਟਲ ਬਣਾਉਂਦਾ ਹੈ ਜਿੱਥੇ ਲੋੜੀਂਦੇ ਦਸਤਾਵੇਜ਼ ਮੌਜੂਦ ਹਨ। ਮਾਡਲ ਸਾਰੇ ਸੰਭਵ ਦਸਤਾਵੇਜ਼ਾਂ ਨੂੰ ਉਨ੍ਹਾਂ ਦੀ ਸੰਭਾਵਨਾ ਦੇ ਆਧਾਰ 'ਤੇ ਭਾਰ ਨਰਿਧਾਰਤ ਕਰਦਾ ਹੈ, ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਸਭ ਤੋਂ ਢੁਕਵੇਂ ਲੋਕਾਂ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰਨ ਲਈ ਸੰਭਾਵਨਾਵਾਂ ਦੀ ਸਪੇਸ ਨੂੰ ਸੀਮਤ ਕਰਦਾ ਹੈ।

ਇਹ ਸਾਨੂੰ "ਰਾਹ ਨੂੰ ਤੰਗ ਕਰਨ" ਦੇ ਕੇਂਦਰੀ ਵਸ਼ਿੇ ਵੱਲ ਵਾਪਸ ਲਆਿਉਂਦਾ ਹੈ। ਪ੍ਰੌਂਪਟਿੰਗ ਦਾ ਮੁੱਖ ਉਦੇਸ਼ ਸੰਭਾਵੀ ਮਾਡਲ ਨੂੰ ਅਜਿਹੇ ਢੰਗ ਨਾਲ ਸ਼ਰਤਬੱਧ ਕਰਨਾ ਹੈ ਜੋ ਇਸਦੀਆਂ ਭਵਿੱਖਬਾਣੀਆਂ ਦੇ ਪੁੰਜ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰਦਾ ਹੈ, ਉਸ ਖਾਸ ਜਾਣਕਾਰੀ ਜਾਂ ਵਵਿਹਾਰ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰਦਾ ਹੈ ਜੋ ਅਸੀਂ ਪ੍ਰਾਪਤ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ। ਧਆਿਨ ਨਾਲ ਤਆਿਰ ਕੀਤੇ ਪ੍ਰੌਂਪਟ ਪ੍ਰਦਾਨ ਕਰਕੇ, ਅਸੀਂ ਮਾਡਲ ਨੂੰ ਗੁਪਤ ਸਪੇਸ ਵਿੱਚ ਵਧੇਰੇ ਕੁਸ਼ਲਤਾ ਨਾਲ ਨੈਵੀਗੇਟ ਕਰਨ ਅਤੇ ਵਧੇਰੇ ਢੁਕਵੇਂ ਅਤੇ ਸੁਸੰਗਤ ਆਉਟਪੁੱਟ ਤਆਿਰ ਕਰਨ ਲਈ ਗਾਈਡ ਕਰ ਸਕਦੇ ਹਾਂ।

ਹਾਲਾਂਕਿ, ਇਹ ਧਆਿਨ ਵਿੱਚ ਰੱਖਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਭਾਸ਼ਾ ਮਾਡਲ ਆਖਰਕਾਰ ਉਸ ਜਾਣਕਾਰੀ ਦੁਆਰਾ ਸੀਮਤਿ

ਹੈ ਜਸਿ 'ਤੇ ਇਸਨੂੰ ਸਖਿਲਾਈ ਦੱਤੀ ਗਈ ਸੀ। ਭਾਵੇਂ ਇਹ ਮੌਜੂਦਾ ਦਸਤਾਵੇਜ਼ਾਂ ਦੇ ਸਮਾਨ ਟੈਕਸਟ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ ਜਾਂ ਵਚਿਾਰਾਂ ਨੂੰ ਨਵੇਂ ਤਰੀਕਆਿਂ ਨਾਲ ਜੋੜ ਸਕਦਾ ਹੈ, ਇਹ ਪੂਰੀ ਤਰ੍ਹਾਂ ਨਵੀਂ ਜਾਣਕਾਰੀ ਨੂੰ ਸ਼ੁਰੂ ਤੋਂ ਨਹੀਂ ਬਣਾ ਸਕਦਾ। ਉਦਾਹਰਨ ਲਈ, ਅਸੀਂ ਮਾਡਲ ਤੋਂ ਕੈਂਸਰ ਦਾ ਇਲਾਜ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਉਮੀਦ ਨਹੀਂ ਕਰ ਸਕਦੇ ਜੇਕਰ ਅਜਹਿਾ ਇਲਾਜ ਖੋਜਆਿ ਅਤੇ ਇਸਦੇ ਸਖਿਲਾਈ ਡੇਟਾ ਵੱਚਿ ਦਸਤਾਵੇਜ਼ੀ ਨਹੀਂ ਕੀਤਾ ਗਆਿ ਹੈ।

ਇਸ ਦੀ ਬਜਾਏ, ਮਾਡਲ ਦੀ ਤਾਕਤ ਉਸ ਜਾਣਕਾਰੀ ਨੂੰ ਲੱਭਣ ਅਤੇ ਸੰਸਲੇਸ਼ਣ ਕਰਨ ਦੀ ਇਸਦੀ ਯੋਗਤਾ ਵੱਚਿ ਹੈ ਜੋ ਉਸ ਨਾਲ ਮਲਿਦੀ-ਜੁਲਦੀ ਹੈ ਜਸਿ ਨਾਲ ਅਸੀਂ ਇਸਨੂੰ ਪ੍ਰੇਰਤਿ ਕਰਦੇ ਹਾਂ। ਇਨ੍ਹਾਂ ਮਾਡਲਾਂ ਦੀ ਸੰਭਾਵੀ ਪ੍ਰਕਰਿਤੀ ਨੂੰ ਸਮਝ ਕੇ ਅਤੇ ਪ੍ਰੌਂਪਟਸ ਨੂੰ ਉਨ੍ਹਾਂ ਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਸ਼ਰਤਬੱਧ ਕਰਨ ਲਈ ਕਵਿੇਂ ਵਰਤਆਿ ਜਾ ਸਕਦਾ ਹੈ, ਅਸੀਂ ਮੁੱਲਵਾਨ ਅੰਤਰਦ੍ਰਸ਼ਟੀ ਅਤੇ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰਨ ਲਈ ਉਨ੍ਹਾਂ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਦਾ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਲਾਭ ਲੈ ਸਕਦੇ ਹਾਂ।

ਹੇਠਾਂ ਦੱਤਿੇ ਪ੍ਰੌਂਪਟਸ 'ਤੇ ਵਚਿਾਰ ਕਰੋ। ਪਹਲਿੇ ਵੱਚਿ, "ਮਰਕਰੀ" ਇਕੱਲਾ ਗ੍ਰਹਿ, ਤੱਤ, ਜਾਂ ਰੋਮਨ ਦੇਵਤਾ ਨੂੰ ਦਰਸਾ ਸਕਦਾ ਹੈ, ਪਰ ਸਭ ਤੋਂ ਵੱਧ ਸੰਭਾਵਨਾ ਗ੍ਰਹਿ ਦੀ ਹੈ। ਦਰਅਸਲ, GPT-4 ਇੱਕ ਲੰਬਾ ਜਵਾਬ ਦੱਦਿਾ ਹੈ ਜੋ ਮਰਕਰੀ ਸੂਰਜੀ ਮੰਡਲ ਵੱਚਿ ਸਭ ਤੋਂ ਛੋਟਾ ਅਤੇ ਸਭ ਤੋਂ ਅੰਦਰੂਨੀ ਗ੍ਰਹਿ ਹੈ... ਨਾਲ ਸ਼ੁਰੂ ਹੁੰਦਾ ਹੈ। ਦੂਜਾ ਪ੍ਰੌਂਪਟ ਖਾਸ ਤੌਰ 'ਤੇ ਰਸਾਇਣਕ ਤੱਤ ਦਾ ਹਵਾਲਾ ਦੱਦਿਾ ਹੈ। ਤੀਜਾ ਰੋਮਨ ਮਥਿਹਿਾਸਕ ਆਕ੍ਰਤਿੀ ਦਾ ਹਵਾਲਾ ਦੱਦਿਾ ਹੈ, ਜੋ ਆਪਣੀ ਗਤੀ ਅਤੇ ਦੈਵੀ ਸੰਦੇਸ਼ਵਾਹਕ ਵਜੋਂ ਭੂਮਕਿਾ ਲਈ ਜਾਣਆਿ ਜਾਂਦਾ ਹੈ।

```
# Prompt 1
Tell me about: Mercury

# Prompt 2
Tell me about: Mercury element

# Prompt 3
Tell me about: Mercury messenger of the gods
```

ਸਰਿਫ ਕੁਝ ਵਾਧੂ ਸ਼ਬਦ ਜੋੜ ਕੇ, ਅਸੀਂ AI ਦੀ ਪ੍ਰਤੀਕਰਿਆ ਨੂੰ ਪੂਰੀ ਤਰ੍ਹਾਂ ਬਦਲ ਦੱਤਿਾ ਹੈ। ਜਵਿੇਂ ਕ ਿਤੁਸੀਂ ਕਤਿਾਬ ਵੱਚਿ ਬਾਅਦ ਵੱਚਿ ਸੱਖਿੋਗੇ, ਵਸ਼ਿੇਸ਼ ਪ੍ਰੌਂਮਪਟ ਇੰਜੀਨੀਅਰੰਗ ਤਕਨੀਕਾਂ ਜਵਿੇਂ n-shot ਪ੍ਰੌਂਮਪਟੰਗ, ਸੰਰਚਤ ਇਨਪੁੱਟ/ਆਉਟਪੁੱਟ, ਅਤੇ ਵਚਿਾਰਾਂ ਦੀ ਲੜੀ ਮਾਡਲ ਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਨਯਿੰਤਰਤ ਕਰਨ ਦੇ ਚਲਾਕ ਤਰੀਕੇ ਹੀ ਹਨ।

ਇਸ ਲਈ ਅੰਤ ਵੱਚਿ, ਪ੍ਰੌਂਮਪਟ ਇੰਜੀਨੀਅਰੰਗ ਦੀ ਕਲਾ ਭਾਸ਼ਾ ਮੌਡਲ ਦੇ ਗਆਿਨ ਦੇ ਵਸ਼ਿਾਲ ਸੰਭਾਵੀ ਖੇਤਰ ਵੱਚਿ ਨੈਵੀਗੇਟ ਕਰਨ ਦੀ ਸਮਝ ਬਾਰੇ ਹੈ, ਤਾਂ ਜੋ ਅਸੀਂ ਜਸਿ ਖਾਸ ਜਾਣਕਾਰੀ ਜਾਂ ਵਵਿਹਾਰ ਦੀ ਭਾਲ ਕਰ ਰਹੇ ਹਾਂ, ਉਸ ਤੱਕ ਦਾ ਰਸਤਾ ਸੀਮਤ ਕਰ ਸਕੀਏ।

ਉਨ੍ਹਾਂ ਪਾਠਕਾਂ ਲਈ ਜਨ੍ਹਿਾਂ ਦੀ ਉੱਚ ਗਣਤਿ ਵੱਚਿ ਮਜ਼ਬੂਤ ਪਕੜ ਹੈ, ਇਨ੍ਹਾਂ ਮੌਡਲਾਂ ਦੀ ਆਪਣੀ ਸਮਝ ਨੂੰ ਸੰਭਾਵਨਾ ਸਧਿਾਂਤ ਅਤੇ ਲੀਨੀਅਰ ਐਲਜਬਰਾ ਦੇ ਸਧਿਾਂਤਾਂ ਵੱਚਿ ਆਧਾਰਤਿ ਕਰਨਾ ਨਸ਼ਿਚਤਿ ਤੌਰ 'ਤੇ ਤੁਹਾਡੀ ਮਦਦ ਕਰ ਸਕਦਾ ਹੈ! ਤੁਹਾਡੇ ਵੱਚਿੋਂ ਬਾਕੀ ਲੋਕਾਂ ਲਈ ਜੋ ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਰਣਨੀਤੀਆਂ ਵਕਿਸਤਿ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਨ, ਆਓ ਵਧੇਰੇ ਸਹਜਿ ਪਹੁੰਚਾਂ 'ਤੇ ਟਕਿੇ ਰਹੀਏ।

ਰਸਤਾ ਕਵਿੇਂ "ਸੀਮਤ" ਹੁੰਦਾ ਹੈ

ਬਹੁਤ ਜ਼ਆਿਦਾ ਗਆਿਨ ਦੀਆਂ ਇਨ੍ਹਾਂ ਚੁਣੌਤੀਆਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ, ਅਸੀਂ ਅਜਹਿੀਆਂ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ ਜੋ ਭਾਸ਼ਾ ਮੌਡਲ ਦੀ ਉਤਪਾਦਨ ਪ੍ਰਕਰਿਆ ਦੀ ਅਗਵਾਈ ਕਰਨ ਅਤੇ ਇਸਦਾ ਧਆਿਨ ਸਭ ਤੋਂ ਪ੍ਰਸੰਗਕਿ ਅਤੇ ਸਹੀ ਜਾਣਕਾਰੀ 'ਤੇ ਕੇਂਦਰਤਿ ਕਰਨ ਵੱਚਿ ਮਦਦ ਕਰਦੀਆਂ ਹਨ।

ਇੱਥੇ ਸਫਿਾਰਸ਼ੀ ਕ੍ਰਮ ਵੱਚਿ ਸਭ ਤੋਂ ਮਹੱਤਵਪੂਰਨ ਤਕਨੀਕਾਂ ਹਨ, ਯਾਨੀ, ਤੁਹਾਨੂੰ ਪਹਲਿਾਂ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰੰਗ ਦੀ ਕੋਸ਼ਸ਼ਿ ਕਰਨੀ ਚਾਹੀਦੀ ਹੈ, ਫਰ RAG, ਅਤੇ ਫਰ ਅੰਤ ਵੱਚਿ, ਜੇ ਤੁਹਾਨੂੰ ਜ਼ਰੂਰੀ ਹੋਵੇ, ਫਾਈਨ ਟਊਿਨੰਗ।

ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰੰਗ ਸਭ ਤੋਂ ਮੁੱਢਲੀ ਪਹੁੰਚ ਮਾਡਲ ਦੀ ਪ੍ਰਤੀਕਰਿਆ ਉਤਪਾਦਨ ਦੀ ਅਗਵਾਈ ਲਈ ਖਾਸ ਹਦਾਇਤਾਂ, ਸੀਮਾਵਾਂ, ਜਾਂ ਉਦਾਹਰਣਾਂ ਸ਼ਾਮਲ ਕਰਨ ਵਾਲੇ ਪ੍ਰੌਮਪਟਾਂ ਨੂੰ ਤਆਿਰ ਕਰਨਾ ਹੈ। ਇਹ ਅਧਆਿਇ ਅਗਲੇ ਭਾਗ ਵੱਚਿ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰੰਗ ਦੇ ਮੂਲ ਸਧਿਾਂਤਾਂ ਨੂੰ ਕਵਰ ਕਰਦਾ ਹੈ, ਅਤੇ ਅਸੀਂ ਕਤਿਾਬ ਦੇ ਭਾਗ 2 ਵੱਚਿ ਕਈ ਖਾਸ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰੰਗ ਪੈਟਰਨਾਂ ਨੂੰ ਕਵਰ ਕਰਦੇ ਹਾਂ। ਉਹਨਾਂ ਪੈਟਰਨਾਂ ਵੱਚਿ ਪ੍ਰੌਮਪਟ ਨਤਿਾਰਨਾ ਸ਼ਾਮਲ ਹੈ, ਜੋ ਇੱਕ ਤਕਨੀਕ ਹੈ ਜੋ AI ਦੁਆਰਾ ਸਭ ਤੋਂ ਢੁਕਵੀਂ ਅਤੇ ਸੰਖੇਪ ਜਾਣਕਾਰੀ ਨੂੰ ਕੱਢਣ ਲਈ ਪ੍ਰੌਮਪਟਾਂ ਨੂੰ ਸੁਧਾਰਨ ਅਤੇ ਅਨੁਕੂਲ ਬਣਾਉਣ 'ਤੇ ਕੇਂਦਰਤਿ ਹੈ।

ਸੰਦਰਭ ਵਾਧਾ। ਬਾਹਰੀ ਗਆਿਨ ਅਧਾਰਾਂ ਜਾਂ ਦਸਤਾਵੇਜ਼ਾਂ ਤੋਂ ਪ੍ਰਸੰਗਕਿ ਜਾਣਕਾਰੀ ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਪ੍ਰਾਪਤ ਕਰਨਾ ਤਾਂ ਜੋ ਮਾਡਲ ਨੂੰ ਪ੍ਰੌਮਪਟ ਕੀਤੇ ਜਾਣ ਦੇ ਸਮੇਂ ਕੇਂਦਰਤਿ ਸੰਦਰਭ ਪ੍ਰਦਾਨ ਕੀਤਾ ਜਾ ਸਕੇ। ਪ੍ਰਸਧਿ ਸੰਦਰਭ ਵਾਧਾ ਤਕਨੀਕਾਂ ਵੱਚਿ ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਵਧਾਇਆ ਉਤਪਾਦਨ (RAG) ਸ਼ਾਮਲ ਹੈ। ਤਥਾਕਥਤਿ "ਔਨਲਾਈਨ ਮੌਡਲ" ਜਵਿੇਂ ਕ Perplexity ਦੁਆਰਾ ਪ੍ਰਦਾਨ ਕੀਤੇ ਗਏ ਮੌਡਲ ਰੀਅਲ-ਟਾਈਮ ਇੰਟਰਨੈੱਟ ਖੋਜ ਨਤੀਜਆਿਂ ਨਾਲ ਆਪਣੇ ਸੰਦਰਭ ਨੂੰ ਵਧਾਉਣ ਦੇ ਯੋਗ ਹਨ।

ਉਨ੍ਹਾਂ ਦੀ ਸ਼ਕਤੀ ਦੇ ਬਾਵਜੂਦ, LLMs ਤੁਹਾਡੇ ਵਲਿੱਖਣ ਡੇਟਾਸੇਟਾਂ 'ਤੇ ਸਖਿਲਾਈ ਨਹੀਂ ਪਾਉਂਦੇ, ਜੋ ਨਜ਼ੀਰ ਹੋ ਸਕਦੇ ਹਨ ਜਾਂ ਉਸ ਸਮੱਸਆਿ ਲਈ ਵਸ਼ਿਸ਼ ਹੋ ਸਕਦੇ ਹਨ ਜਸਿਨੂੰ ਤੁਸੀਂ ਹੱਲ ਕਰਨ ਦੀ ਕੋਸ਼ਸ਼ਿ ਕਰ ਰਹੇ ਹੋ। ਸੰਦਰਭ ਵਾਧਾ ਤਕਨੀਕਾਂ ਤੁਹਾਨੂੰ LLMs ਨੂੰ APIs, SQL ਡੇਟਾਬੇਸਾਂ, ਜਾਂ PDFs ਅਤੇ ਸਲਾਈਡ ਡੈੱਕਾਂ ਵੱਚਿ ਫਸੇ ਡੇਟਾ ਤੱਕ ਪਹੁੰਚ ਦੇਣ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦੀਆਂ ਹਨ।

ਬਰੀਕ ਟਿਊਨਿੰਗ ਜਾਂ ਖੇਤਰ ਅਨੁਕੂਲਤਾ ਮਾਡਲ ਨੂੰ ਖੇਤਰ-ਵਿਸ਼ੇਸ਼ ਡੇਟਾਸੇਟਾਂ 'ਤੇ ਸਖਿਲਾਈ ਦੇਣਾ ਤਾਂ ਜੋ ਕਿਸੇ ਵਿਸ਼ੇਸ਼ ਕਾਰਜ ਜਾਂ ਖੇਤਰ ਲਈ ਇਸਦੇ ਗਿਆਨ ਅਤੇ ਉਤਪਾਦਨ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਵਿਸ਼ੇਸ਼ ਬਣਾਇਆ ਜਾ ਸਕੇ।

ਤਾਪਮਾਨ ਨੂੰ ਘੱਟ ਕਰਨਾ

ਤਾਪਮਾਨ ਇੱਕ ਹਾਈਪਰਪੈਰਾਮੀਟਰ ਹੈ ਜੋ ਟ੍ਰਾਂਸਫਾਰਮਰ-ਆਧਾਰਿਤ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ ਜੋ ਉਤਪੰਨ ਕੀਤੇ ਟੈਕਸਟ ਦੀ ਬੇਤਰਤੀਬੀ ਅਤੇ ਰਚਨਾਤਮਕਤਾ ਨੂੰ ਨਿਯੰਤਰਿਤ ਕਰਦਾ ਹੈ। ਇਹ 0 ਅਤੇ 1 ਦੇ ਵਿਚਕਾਰ ਇੱਕ ਮੁੱਲ ਹੈ, ਜਿੱਥੇ ਘੱਟ ਮੁੱਲ ਆਉਟਪੁੱਟ ਨੂੰ ਵਧੇਰੇ ਕੇਂਦਰਿਤ ਅਤੇ ਨਿਸ਼ਚਿਤ ਬਣਾਉਂਦੇ ਹਨ, ਜਦੋਂ ਕਿ ਉੱਚ ਮੁੱਲ ਇਸਨੂੰ ਵਧੇਰੇ ਵਿਭਿੰਨ ਅਤੇ ਅਣਕਿਆਸੇ ਬਣਾਉਂਦੇ ਹਨ।

ਜਦੋਂ ਤਾਪਮਾਨ 1 'ਤੇ ਸੈੱਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਭਾਸ਼ਾ ਮਾਡਲ ਅਗਲੇ ਟੋਕਨ ਦੀ ਪੂਰੀ ਸੰਭਾਵਨਾ ਵੰਡ ਦੇ ਆਧਾਰ 'ਤੇ ਟੈਕਸਟ ਤਿਆਰ ਕਰਦਾ ਹੈ, ਜੋ ਵਧੇਰੇ ਰਚਨਾਤਮਕ ਅਤੇ ਵੱਖ-ਵੱਖ ਜਵਾਬਾਂ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦਾ ਹੈ। ਹਾਲਾਂਕਿ, ਇਸ ਨਾਲ ਮਾਡਲ ਅਜਿਹਾ ਟੈਕਸਟ ਵੀ ਤਿਆਰ ਕਰ ਸਕਦਾ ਹੈ ਜੋ ਘੱਟ ਪ੍ਰਸੰਗਿਕ ਜਾਂ ਸੁਸੰਗਤ ਹੋਵੇ।

ਦੂਜੇ ਪਾਸੇ, ਜਦੋਂ ਤਾਪਮਾਨ 0 'ਤੇ ਸੈੱਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਭਾਸ਼ਾ ਮਾਡਲ ਹਮੇਸ਼ਾ ਸਭ ਤੋਂ ਵੱਧ ਸੰਭਾਵਨਾ ਵਾਲੇ ਟੋਕਨ ਦੀ ਚੋਣ ਕਰਦਾ ਹੈ, ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ "ਆਪਣੇ ਰਸਤੇ ਨੂੰ ਸੰਕੁਚਿਤ ਕਰਦਾ ਹੈ।" ਮੇਰੇ ਲਗਭਗ ਸਾਰੇ AI ਕੰਪੋਨੈਂਟ 0 'ਤੇ ਜਾਂ ਇਸ ਦੇ ਨੇੜੇ ਸੈੱਟ ਕੀਤੇ ਤਾਪਮਾਨ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ, ਕਿਉਂਕਿ ਇਸ ਨਾਲ ਵਧੇਰੇ ਕੇਂਦਰਿਤ ਅਤੇ ਅਨੁਮਾਨਯੋਗ ਜਵਾਬ ਮਿਲਦੇ ਹਨ। ਇਹ ਬਿਲਕੁਲ ਉਦੋਂ ਲਾਭਦਾਇਕ ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਤੁਸੀਂ ਚਾਹੁੰਦੇ ਹੋ ਕਿ ਮਾਡਲ ਹਦਾਇਤਾਂ ਦੀ ਪਾਲਣਾ ਕਰੇ, ਉਹਨਾਂ ਫੰਕਸ਼ਨਾਂ ਵੱਲ ਧਿਆਨ ਦੇਵੇ ਜੋ ਇਸਨੂੰ ਪ੍ਰਦਾਨ ਕੀਤੇ ਗਏ ਹਨ, ਜਾਂ ਬਸ ਤੁਹਾਨੂੰ ਉਸ ਨਾਲੋਂ ਵਧੇਰੇ ਸਹੀ ਅਤੇ ਪ੍ਰਸੰਗਿਕ ਜਵਾਬਾਂ ਦੀ ਲੋੜ ਹੋਵੇ ਜੋ ਤੁਸੀਂ ਪ੍ਰਾਪਤ ਕਰ ਰਹੇ ਹੋ।

ਉਦਾਹਰਨ ਲਈ, ਜੇਕਰ ਤੁਸੀਂ ਇੱਕ ਚੈਟਬੋਟ ਬਣਾ ਰਹੇ ਹੋ ਜਿਸਨੂੰ ਤੱਥਾਤਮਕ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਲੋੜ ਹੈ, ਤਾਂ ਤੁਸੀਂ ਜਵਾਬਾਂ ਨੂੰ ਵਧੇਰੇ ਸਟੀਕ ਅਤੇ ਵਿਸ਼ੇ ਨਾਲ ਸਬੰਧਿਤ ਬਣਾਉਣ ਲਈ ਤਾਪਮਾਨ ਨੂੰ ਘੱਟ ਮੁੱਲ 'ਤੇ ਸੈੱਟ ਕਰਨਾ ਚਾਹ ਸਕਦੇ ਹੋ। ਇਸਦੇ ਉਲਟ, ਜੇਕਰ ਤੁਸੀਂ ਇੱਕ ਰਚਨਾਤਮਕ ਲੇਖਣ ਸਹਾਇਕ ਬਣਾ ਰਹੇ ਹੋ, ਤਾਂ ਤੁਸੀਂ ਵਧੇਰੇ ਵਿਭਿੰਨ ਅਤੇ ਕਲਪਨਾਸ਼ੀਲ ਆਉਟਪੁੱਟ ਨੂੰ ਉਤਸ਼ਾਹਿਤ ਕਰਨ ਲਈ ਤਾਪਮਾਨ ਨੂੰ ਉੱਚ ਮੁੱਲ 'ਤੇ ਸੈੱਟ ਕਰਨਾ ਚਾਹ ਸਕਦੇ ਹੋ।

ਹਾਈਪਰਪੈਰਾਮੀਟਰ: ਅਨੁਮਾਨ ਦੇ ਨੌਬ ਅਤੇ ਡਾਇਲ

ਜਦੋਂ ਤੁਸੀਂ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨਾਲ ਕੰਮ ਕਰ ਰਹੇ ਹੁੰਦੇ ਹੋ, ਤਾਂ ਤੁਸੀਂ "ਹਾਈਪਰਪੈਰਾਮੀਟਰ" ਸ਼ਬਦ ਨੂੰ ਅਕਸਰ ਦੇਖੋਗੇ। ਅਨੁਮਾਨ ਦੇ ਸੰਦਰਭ ਵਿੱਚ (ਭਾਵ, ਜਦੋਂ ਤੁਸੀਂ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਲਈ ਮਾਡਲ ਦੀ ਵਰਤੋਂ ਕਰ ਰਹੇ ਹੁੰਦੇ ਹੋ),

ਹਾਈਪਰਪੈਰਾਮੀਟਰ ਉਹਨਾਂ ਨੌਬਾਂ ਅਤੇ ਡਾਇਲਾਂ ਵਾਂਗ ਹੁੰਦੇ ਹਨ ਜਨ੍ਹਿਾਂ ਨੂੰ ਤੁਸੀਂ ਮਾਡਲ ਦੇ ਵਵਿਹਾਰ ਅਤੇ ਆਉਟਪੁੱਟ ਨੂੰ ਨਯਿੰਤਰਤਿ ਕਰਨ ਲਈ ਐਡਜਸਟ ਕਰ ਸਕਦੇ ਹੋ।

ਇਸ ਨੂੰ ਇੱਕ ਗੁੰਝਲਦਾਰ ਮਸ਼ੀਨ 'ਤੇ ਸੈਟਿੰਗਾਂ ਨੂੰ ਐਡਜਸਟ ਕਰਨ ਵਾਂਗ ਸੋਚੋ। ਜਵਿੇਂ ਤੁਸੀਂ ਤਾਪਮਾਨ ਨੂੰ ਨਯਿੰਤਰਤਿ ਕਰਨ ਲਈ ਇੱਕ ਨੌਬ ਘੁਮਾ ਸਕਦੇ ਹੋ ਜਾਂ ਕਾਰਜ ਦੀ ਮੋਡ ਨੂੰ ਬਦਲਣ ਲਈ ਇੱਕ ਸਵਚਿ ਨੂੰ ਫਲਪਿ ਕਰ ਸਕਦੇ ਹੋ, ਹਾਈਪਰਪੈਰਾਮੀਟਰ ਤੁਹਾਨੂੰ ਭਾਸ਼ਾ ਮਾਡਲ ਦੁਆਰਾ ਟੈਕਸਟ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਨ ਅਤੇ ਤਆਿਰ ਕਰਨ ਦੇ ਤਰੀਕੇ ਨੂੰ ਬਰੀਕੀ ਨਾਲ ਐਡਜਸਟ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦਦਿੇ ਹਨ।

ਇਨਫਰੈਂਸ ਦੌਰਾਨ ਤੁਸੀਂ ਜਹਿੜੇ ਆਮ ਹਾਈਪਰਪੈਰਾਮੀਟਰ ਦੇਖੋਗੇ ਉਹ ਹਨ:

- **ਤਾਪਮਾਨ**: ਜਵਿੇਂ ਕ ਿਹੁਣੇ ਦੱਸਆਿ ਗਆਿ ਹੈ, ਇਹ ਪੈਰਾਮੀਟਰ ਤਆਿਰ ਕੀਤੇ ਗਏ ਟੈਕਸਟ ਦੀ ਰੈਂਡਮਨੈੱਸ ਅਤੇ ਰਚਨਾਤਮਕਤਾ ਨੂੰ ਨਯਿੰਤਰਤਿ ਕਰਦਾ ਹੈ। ਉੱਚ ਤਾਪਮਾਨ ਵੱਧ ਵਭਿੰਨਿ ਅਤੇ ਅਣਕਆਿਸੇ ਆਉਟਪੁੱਟ ਵੱਲ ਲੈ ਜਾਂਦਾ ਹੈ, ਜਦਕ ਿਘੱਟ ਤਾਪਮਾਨ ਵਧੇਰੇ ਕੇਂਦਰਤਿ ਅਤੇ ਨਸ਼ਿਚਤਿ ਜਵਾਬਾਂ ਵੱਚਿ ਨਤੀਜਾ ਕੱਢਦਾ ਹੈ।

- **ਟੌਪ-ਪੀ (ਨਊਕਲੀਅਸ) ਸੈਂਪਲਿੰਗ:** ਇਹ ਪੈਰਾਮੀਟਰ ਟੋਕਨਾਂ ਦੇ ਸਭ ਤੋਂ ਛੋਟੇ ਸੈੱਟ ਦੀ ਚੋਣ ਨੂੰ ਨਯਿੰਤਰਤਿ ਕਰਦਾ ਹੈ ਜਨ੍ਹਿਾਂ ਦੀ ਸੰਚਤਿ ਸੰਭਾਵਨਾ ਇੱਕ ਨਸ਼ਿਚਤਿ ਸੀਮਾ (p) ਤੋਂ ਵੱਧ ਜਾਂਦੀ ਹੈ। ਇਹ ਸੁਸੰਗਤੀ ਬਣਾਈ ਰੱਖਦੇ ਹੋਏ ਵਧੇਰੇ ਵਭਿੰਨਿ ਆਉਟਪੁੱਟ ਦੀ ਆਗਆਿ ਦਦਿੇਂ ਹੈ।

- **ਟੌਪ-ਕੇ ਸੈਂਪਲਿੰਗ:** ਇਹ ਤਕਨੀਕ k ਸਭ ਤੋਂ ਸੰਭਾਵੀ ਅਗਲੇ ਟੋਕਨਾਂ ਦੀ ਚੋਣ ਕਰਦੀ ਹੈ ਅਤੇ ਉਨ੍ਹਾਂ ਵੱਚਿ ਸੰਭਾਵਨਾ ਮਾਸ ਨੂੰ ਮੁੜ ਵੰਡਦੀ ਹੈ। ਇਹ ਮਾਡਲ ਨੂੰ ਘੱਟ-ਸੰਭਾਵਨਾ ਜਾਂ ਅਪ੍ਰਸੰਗਕਿ ਟੋਕਨ ਤਆਿਰ ਕਰਨ ਤੋਂ ਰੋਕਣ ਵੱਚਿ ਮਦਦ ਕਰ ਸਕਦੀ ਹੈ।

- **ਆਵਰਤੀ** ਅਤੇ **ਮੌਜੂਦਗੀ ਪੈਨਲਟੀਆਂ**: ਇਹ ਪੈਰਾਮੀਟਰ ਮਾਡਲ ਨੂੰ ਇੱਕੋ ਸ਼ਬਦਾਂ ਜਾਂ ਵਾਕਾਂਸ਼ਾਂ ਨੂੰ ਬਹੁਤ ਵਾਰ ਦੁਹਰਾਉਣ (ਆਵਰਤੀ ਪੈਨਲਟੀ) ਜਾਂ ਇਨਪੁੱਟ ਪ੍ਰੌਂਮਪਟ ਵੱਚਿ ਮੌਜੂਦ ਨਾ ਹੋਣ ਵਾਲੇ ਸ਼ਬਦਾਂ ਨੂੰ ਤਆਿਰ ਕਰਨ (ਮੌਜੂਦਗੀ ਪੈਨਲਟੀ) ਲਈ ਦੰਡਤਿ ਕਰਦੇ ਹਨ। ਇਨ੍ਹਾਂ ਮੁੱਲਾਂ ਨੂੰ ਅਨੁਕੂਲ ਕਰਕੇ, ਤੁਸੀਂ ਮਾਡਲ ਨੂੰ ਵਧੇਰੇ ਵਭਿੰਨਿ ਅਤੇ ਪ੍ਰਸੰਗਕਿ ਆਉਟਪੁੱਟ ਤਆਿਰ ਕਰਨ ਲਈ ਉਤਸ਼ਾਹਤਿ ਕਰ ਸਕਦੇ ਹੋ।

- **ਵੱਧ ਤੋਂ ਵੱਧ ਲੰਬਾਈ**: ਇਹ ਹਾਈਪਰਪੈਰਾਮੀਟਰ ਇੱਕ ਸਿੰਗਲ ਜਵਾਬ ਵੱਚਿ ਮਾਡਲ ਦੁਆਰਾ ਤਆਿਰ ਕੀਤੇ ਜਾ ਸਕਣ ਵਾਲੇ ਟੋਕਨਾਂ (ਸ਼ਬਦਾਂ ਜਾਂ ਉਪ-ਸ਼ਬਦਾਂ) ਦੀ ਉੱਪਰੀ ਸੀਮਾ ਨਰਿਧਾਰਤਿ ਕਰਦਾ ਹੈ। ਇਹ ਤਆਿਰ ਕੀਤੇ ਗਏ ਟੈਕਸਟ ਦੀ ਵਰਬੋਸਟਿੀ ਅਤੇ ਸੰਖੇਪਤਾ ਨੂੰ ਨਯਿੰਤਰਤਿ ਕਰਨ ਵੱਚਿ ਮਦਦ ਕਰਦਾ ਹੈ।

ਜਦੋਂ ਤੁਸੀਂ ਵੱਖ-ਵੱਖ ਹਾਈਪਰਪੈਰਾਮੀਟਰ ਸੈਟਿੰਗਾਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਦੇ ਹੋ, ਤੁਸੀਂ ਦੇਖੋਗੇ ਕਿ ਛੋਟੀਆਂ ਤਬਦੀਲੀਆਂ ਵੀ ਮਾਡਲ ਦੇ ਆਉਟਪੁੱਟ 'ਤੇ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦੀਆਂ ਹਨ। ਇਹ ਕਸਿ ਪਕਵਾਨ ਨੂੰ ਫਾਈਨ-ਟਿਊਨ ਕਰਨ ਵਰਗਾ ਹੈ - ਥੋੜ੍ਹਾ ਜਹਿਾ ਵੱਧ ਲੂਣ ਜਾਂ ਥੋੜ੍ਹਾ ਜਹਿਾ ਲੰਬਾ ਪਕਾਉਣ ਦਾ ਸਮਾਂ ਅੰਤਮਿ ਪਕਵਾਨ ਵਚਿ ਸਾਰਾ ਫਰਕ ਪਾ ਸਕਦਾ ਹੈ।

ਮੁੱਖ ਗੱਲ ਇਹ ਹੈ ਕਿ ਹਰ ਹਾਈਪਰਪੈਰਾਮੀਟਰ ਮਾਡਲ ਦੇ ਵਵਿਹਾਰ ਨੂੰ ਕਵਿੇਂ ਪ੍ਰਭਾਵਤਿ ਕਰਦਾ ਹੈ ਅਤੇ ਆਪਣੇ ਵਸਿੇਸ਼ ਕਾਰਜ ਲਈ ਸਹੀ ਸੰਤੁਲਨ ਲੱਭਣਾ ਹੈ। ਵੱਖ-ਵੱਖ ਸੈਟਿੰਗਾਂ ਨਾਲ ਖੇਡਣ ਤੋਂ ਨਾ ਡਰੋ ਅਤੇ ਦੇਖੋ ਕਿ ਉਹ ਤਆਿਰ ਕੀਤੇ ਟੈਕਸਟ ਨੂੰ ਕਵਿੇਂ ਪ੍ਰਭਾਵਤਿ ਕਰਦੀਆਂ ਹਨ। ਸਮੇਂ ਦੇ ਨਾਲ, ਤੁਸੀਂ ਇਹ ਸਮਝ ਵਕਿਸਤਿ ਕਰ ਲਵੋਗੇ ਕਿ ਕਹਿੜੇ ਹਾਈਪਰਪੈਰਾਮੀਟਰ ਨੂੰ ਐਡਜਸਟ ਕਰਨਾ ਹੈ ਅਤੇ ਲੋੜੀਂਦੇ ਨਤੀਜੇ ਕਵਿੇਂ ਪ੍ਰਾਪਤ ਕਰਨੇ ਹਨ।

ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ, ਰਟਿਰੀਵਲ-ਔਗਮੈਂਟਡਿ ਜਨਰੇਸ਼ਨ, ਅਤੇ ਫਾਈਨ-ਟਿਊਨਿੰਗ ਦੇ ਨਾਲ ਇਨ੍ਹਾਂ ਪੈਰਾਮੀਟਰਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਜੋੜ ਕੇ, ਤੁਸੀਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਰਸਤੇ ਨੂੰ ਸੰਕੁਚਤਿ ਕਰ ਸਕਦੇ ਹੋ ਅਤੇ ਭਾਸ਼ਾ ਮਾਡਲ ਨੂੰ ਆਪਣੇ ਵਸਿੇਸ਼ ਕੇਸ ਲਈ ਵਧੇਰੇ ਸਟੀਕ, ਪ੍ਰਸੰਗਕਿ, ਅਤੇ ਮੁੱਲਵਾਨ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਲਈ ਗਾਈਡ ਕਰ ਸਕਦੇ ਹੋ।

ਰਾਅ ਬਨਾਮ ਇੰਸਟਰਕਟ-ਟਿਊਨਡ ਮਾਡਲ

ਕੱਚੇ ਮਾਡਲ ਐੱਲਐੱਲਐੱਮਜ਼ ਦੇ ਅਣਸੋਧੇ, ਅਣਸਧਿ ਰੂਪ ਹਨ। ਇਨ੍ਹਾਂ ਨੂੰ ਇੱਕ ਕੋਰੇ ਕੈਨਵਸ ਵਾਂਗ ਸਮਝੋ, ਜੋ ਅਜੇ ਹਦਾਇਤਾਂ ਨੂੰ ਸਮਝਣ ਜਾਂ ਪਾਲਣ ਕਰਨ ਲਈ ਵਸਿੇਸ਼ ਸਖਿਲਾਈ ਤੋਂ ਪ੍ਰਭਾਵਤਿ ਨਹੀਂ ਹੋਇਆ। ਇਹ ਉਸ ਵਸਿਾਲ ਡੇਟਾ 'ਤੇ ਬਣੇ ਹੁੰਦੇ ਹਨ ਜਸਿ 'ਤੇ ਉਹ ਸ਼ੁਰੂ ਵਚਿ ਸਖਿਲਾਈ ਪ੍ਰਾਪਤ ਕਰਦੇ ਹਨ, ਅਤੇ ਵੱਖ-ਵੱਖ ਕਸਿਮ ਦੇ ਆਉਟਪੁੱਟ ਤਆਿਰ ਕਰਨ ਦੇ ਯੋਗ ਹੁੰਦੇ ਹਨ। ਹਾਲਾਂਕਿ, ਹਦਾਇਤ-ਆਧਾਰਤਿ ਬਰੀਕ ਸਖਿਲਾਈ ਦੀਆਂ ਵਾਧੂ ਪਰਤਾਂ ਤੋਂ ਬਨਿਾਂ, ਉਨ੍ਹਾਂ ਦੇ ਜਵਾਬ ਅਣਕਆਿਸੇ ਹੋ ਸਕਦੇ ਹਨ ਅਤੇ ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਵੱਲ ਉਨ੍ਹਾਂ ਦੀ ਅਗਵਾਈ ਕਰਨ ਲਈ ਵਧੇਰੇ ਬਰੀਕ, ਧਆਿਨ ਨਾਲ ਤਆਿਰ ਕੀਤੇ ਪ੍ਰੌਮਪਟਸ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਕੱਚੇ ਮਾਡਲਾਂ ਨਾਲ ਕੰਮ ਕਰਨਾ ਇੱਕ ਅਜਹਿੇ ਪ੍ਰਤਭਿਾਸ਼ਾਲੀ-ਮੂਰਖ ਤੋਂ ਗੱਲਬਾਤ ਕਢਵਾਉਣ ਵਰਗਾ ਹੈ ਜਸਿ ਕੋਲ ਵਸਿਾਲ ਗਆਿਨ ਤਾਂ ਹੈ ਪਰ ਤੁਸੀਂ ਕੀ ਪੁੱਛ ਰਹੇ ਹੋ, ਇਸ ਬਾਰੇ ਕੋਈ ਅੰਤਰਝਾਤ ਨਹੀਂ ਹੈ ਜਦੋਂ ਤੱਕ ਤੁਸੀਂ ਆਪਣੀਆਂ ਹਦਾਇਤਾਂ ਵਚਿ ਬਹੁਤ ਸਟੀਕ ਨਹੀਂ ਹੁੰਦੇ। ਉਹ ਅਕਸਰ ਇੱਕ ਤੋਤੇ ਵਾਂਗ ਲੱਗਦੇ ਹਨ, ਕਿ ਜਸਿ ਹੱਦ ਤੱਕ ਤੁਸੀਂ ਉਨ੍ਹਾਂ ਤੋਂ ਕੁਝ ਸਮਝਣਯੋਗ ਕਹਾਉਂਦੇ ਹੋ, ਉਹ ਜ਼ਆਿਦਾਤਰ ਸਰਫ਼ ਉਹੀ ਦੁਹਰਾ ਰਹੇ ਹੁੰਦੇ ਹਨ ਜੋ ਉਨ੍ਹਾਂ ਨੇ ਤੁਹਾਨੂੰ ਕਹਿੰਦੇ ਸੁਣਆਿ ਹੈ।

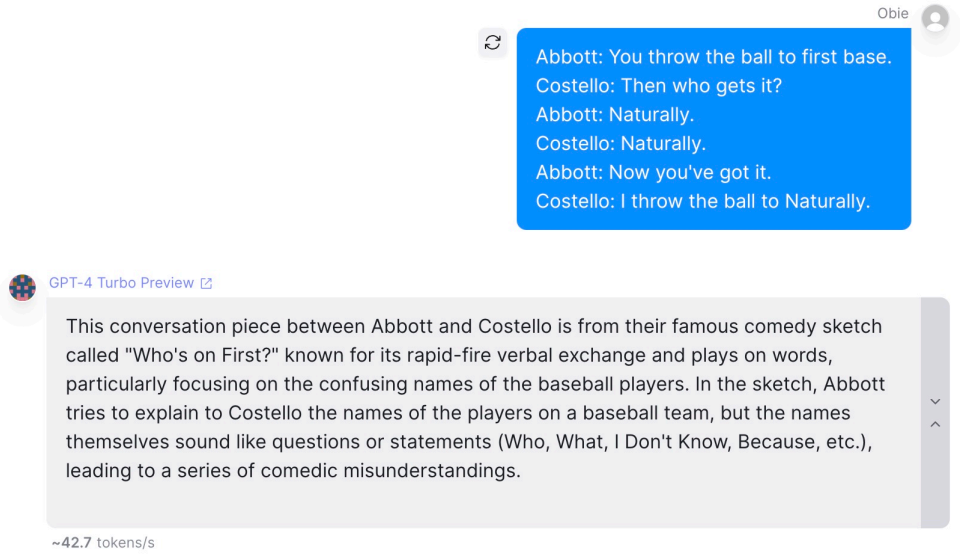
ਦੂਜੇ ਪਾਸੇ, ਹਦਾਇਤ-ਸਖਿਲਾਈ ਮਾਡਲਾਂ ਨੇ ਹਦਾਇਤਾਂ ਨੂੰ ਸਮਝਣ ਅਤੇ ਪਾਲਣ ਕਰਨ ਲਈ ਵਸਿੇਸ਼ ਤੌਰ 'ਤੇ ਤਆਿਰ ਕੀਤੀ ਸਖਿਲਾਈ ਦੇ ਗੇੜ ਪੂਰੇ ਕੀਤੇ ਹੁੰਦੇ ਹਨ। GPT-4, Claude 3 ਅਤੇ ਹੋਰ ਬਹੁਤ ਸਾਰੇ ਸਭ ਤੋਂ ਪ੍ਰਸਧਿ ਐੱਲਐੱਲਐੱਮ ਮਾਡਲ ਸਾਰੇ ਗਹਰਿਾਈ ਨਾਲ ਹਦਾਇਤ-ਸਖਿਲਾਈ ਪ੍ਰਾਪਤ ਹਨ। ਇਸ ਸਖਿਲਾਈ ਵਚਿ ਮਾਡਲ ਨੂੰ

ਲੋੜੀਂਦੇ ਨਤੀਜਿਆਂ ਦੇ ਨਾਲ ਹਦਾਇਤਾਂ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਦੇਣਾ ਸ਼ਾਮਲ ਹੈ, ਜੋ ਅਸਲ ਵਿੱਚ ਮਾਡਲ ਨੂੰ ਕਮਾਂਡਾਂ ਦੀ ਵਿਆਪਕ ਸ਼੍ਰੇਣੀ ਦੀ ਵਿਆਖਿਆ ਅਤੇ ਕਾਰਜਾਂਵੀ ਕਰਨਾ ਸਿਖਾਉਂਦਾ ਹੈ। ਨਤੀਜੇ ਵਜੋਂ, ਹਦਾਇਤ ਮਾਡਲ ਪ੍ਰੌਮਪਟ ਦੇ ਪੀਛੇ ਦੇ ਇਰਾਦੇ ਨੂੰ ਵਧੇਰੇ ਆਸਾਨੀ ਨਾਲ ਸਮਝ ਸਕਦੇ ਹਨ ਅਤੇ ਅਜਿਹੇ ਜਵਾਬ ਤਿਆਰ ਕਰ ਸਕਦੇ ਹਨ ਜੋ ਵਰਤੋਂਕਾਰ ਦੀਆਂ ਉਮੀਦਾਂ ਦੇ ਨੇੜੇ ਹੁੰਦੇ ਹਨ। ਇਹ ਉਨ੍ਹਾਂ ਨੂੰ ਵਧੇਰੇ ਵਰਤੋਂਕਾਰ-ਅਨੁਕੂਲ ਅਤੇ ਕੰਮ ਕਰਨ ਵਿੱਚ ਆਸਾਨ ਬਣਾਉਂਦਾ ਹੈ, ਖਾਸ ਕਰਕੇ ਉਨ੍ਹਾਂ ਲੋਕਾਂ ਲਈ ਜਿਨ੍ਹਾਂ ਕੋਲ ਵਿਆਪਕ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਵਿੱਚ ਰੁੱਝਣ ਲਈ ਸਮਾਂ ਜਾਂ ਮੁਹਾਰਤ ਨਹੀਂ ਹੋ ਸਕਦੀ।

ਕੱਚੇ ਮਾਡਲ: ਅਣਫ਼ਿਲਟਰਡ ਕੈਨਵਸ

ਕੱਚੇ ਮਾਡਲ, ਜਿਵੇਂ ਕਿ Llama 2-70B ਜਾਂ Yi-34B, ਮਾਡਲ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਤੱਕ ਵਧੇਰੇ ਅਣਫ਼ਿਲਟਰਡ ਪਹੁੰਚ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ ਜੋ ਤੁਸੀਂ GPT-4 ਵਰਗੇ ਪ੍ਰਸਿੱਧ ਐੱਲਐੱਲਐੱਮਜ਼ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਦੇ ਹੋਏ ਵੇਖੇ ਹੋਣਗੇ। ਇਹ ਮਾਡਲ ਵਿਸ਼ੇਸ਼ ਹਦਾਇਤਾਂ ਦੀ ਪਾਲਣਾ ਕਰਨ ਲਈ ਪਹਿਲਾਂ ਤੋਂ ਸਿਖਿਲਾਈ ਪ੍ਰਾਪਤ ਨਹੀਂ ਹੁੰਦੇ, ਤੁਹਾਨੂੰ ਇੱਕ ਕੋਰਾ ਕੈਨਵਸ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ ਜਿੱਥੇ ਤੁਸੀਂ ਧਿਆਨ ਨਾਲ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਰਾਹੀਂ ਮਾਡਲ ਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਸਿੱਧਾ ਨਿਯੰਤਰਿਤ ਕਰ ਸਕਦੇ ਹੋ। ਇਸ ਪਹੁੰਚ ਲਈ ਇਹ ਡੂੰਘੀ ਸਮਝ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ ਕਿ ਅਜਿਹੇ ਪ੍ਰੌਮਪਟ ਕਿਵੇਂ ਬਣਾਏ ਜਾਣ ਜੋ ਏਆਈ ਨੂੰ ਸਪੱਸ਼ਟ ਹਦਾਇਤਾਂ ਦਿੱਤੇ ਬਿਨਾਂ ਲੋੜੀਂਦੀ ਦਿਸ਼ਾ ਵੱਲ ਅਗਵਾਈ ਕਰਨ। ਇਹ ਅੰਤਰਨਿਹਿਤ ਏਆਈ ਦੀਆਂ "ਕੱਚੀਆਂ" ਪਰਤਾਂ ਤੱਕ ਸਿੱਧੀ ਪਹੁੰਚ ਰੱਖਣ ਵਰਗਾ ਹੈ, ਬਿਨਾਂ ਕਿਸੇ ਵਚੋਲੀਆਂ ਪਰਤਾਂ ਦੇ ਜੋ ਮਾਡਲ ਦੇ ਜਵਾਬਾਂ ਦੀ ਵਿਆਖਿਆ ਜਾਂ ਅਗਵਾਈ ਕਰਦੀਆਂ ਹੋਣ (ਇਸੇ ਲਈ ਇਸ ਨੂੰ ਇਹ ਨਾਮ ਦਿੱਤਾ ਗਿਆ ਹੈ)।

ਕੱਚੇ ਮਾਡਲਾਂ ਨਾਲ ਚੁਣੌਤੀ ਇਹ ਹੈ ਕਿ ਉਹ ਦੁਹਰਾਉਣ ਵਾਲੇ ਪੈਟਰਨਾਂ ਵਿੱਚ ਫਸ ਜਾਂਦੇ ਹਨ ਜਾਂ ਬੇਤਰਤੀਬੇ ਆਉਟਪੁੱਟ ਪੈਦਾ ਕਰਦੇ ਹਨ। ਹਾਲਾਂਕਿ, ਸਾਵਧਾਨੀਪੂਰਵਕ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਅਤੇ ਦੁਹਰਾਓ ਜੁਰਮਾਨੇ ਵਰਗੇ ਪੈਰਾਮੀਟਰਾਂ ਦੇ ਸਮਾਯੋਜਨ ਨਾਲ, ਕੱਚੇ ਮਾਡਲਾਂ ਨੂੰ ਵਲਿੱਖਣ ਅਤੇ ਰਚਨਾਤਮਕ ਸਮੱਗਰੀ ਤਿਆਰ ਕਰਨ ਲਈ ਪ੍ਰੇਰਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਪ੍ਰਕਰਿਆ ਸਮਝੌਤਿਆਂ ਤੋਂ ਬਿਨਾਂ ਨਹੀਂ ਹੈ; ਜਦੋਂ ਕਿ ਕੱਚੇ ਮਾਡਲ ਨਵੀਨਤਾ ਲਈ ਬੇਮਿਸਾਲ ਲਚਕਤਾ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ, ਉਹ ਉੱਚ ਪੱਧਰ ਦੀ ਮੁਹਾਰਤ ਦੀ ਮੰਗ ਕਰਦੇ ਹਨ।



ਚਤਿਰ 3. ਤੁਲਨਾ ਦੇ ਉਦੇਸ਼ਾਂ ਲਈ, ਇੱਥੇ GPT-4 ਨੂੰ ਦਤਿਾ ਗਿਆ ਉਹੀ ਅਸਪਸਟ ਪ੍ਰੌਮਪਟ ਹੈ

ਇੰਸਟ੍ਰਕਟ-ਟਊਿਨਡ ਮਾਡਲ: ਨਰਿਦੇਸ਼ਤਿ ਤਜਰਬਾ

ਇੰਸਟ੍ਰਕਟ-ਟਊਿਨਡ ਮਾਡਲ ਖਾਸ ਹਦਾਇਤਾਂ ਨੂੰ ਸਮਝਣ ਅਤੇ ਉਨ੍ਹਾਂ ਦੀ ਪਾਲਣਾ ਕਰਨ ਲਈ ਡਜਿ਼ਾਈਨ ਕੀਤੇ ਗਏ ਹਨ, ਜੋ ਉਨ੍ਹਾਂ ਨੂੰ ਵਧੇਰੇ ਵਰਤੋਂਕਾਰ-ਅਨੁਕੂਲ ਅਤੇ ਵਆਿਪਕ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਪਹੁੰਚਯੋਗ ਬਣਾਉਂਦੇ ਹਨ। ਉਹ ਇੱਕ ਗੱਲਬਾਤ ਦੀ ਮਕੈਨਕਿਸ ਨੂੰ ਸਮਝਦੇ ਹਨ ਅਤੇ ਇਹ ਵੀ ਕ ਿਉਨ੍ਹਾਂ ਨੂੰ ਆਪਣੀ ਗੱਲ ਕਰਨ ਦੀ ਵਾਰੀ ਦੇ ਅੰਤ ’ਤੇ ਜਨਰੇਟ ਕਰਨਾ ਬੰਦ ਕਰ ਦੇਣਾ ਚਾਹੀਦਾ ਹੈ। ਬਹੁਤ ਸਾਰੇ ਡਵੈਲਪਰਾਂ ਲਈ, ਖਾਸ ਕਰਕੇ ਜੋ ਸਧਾਰਨ ਐਪਲੀਕੇਸ਼ਨਾਂ ’ਤੇ ਕੰਮ ਕਰ ਰਹੇ ਹਨ, ਇੰਸਟ੍ਰਕਟ-ਟਊਿਨਡ ਮਾਡਲ ਇੱਕ ਸੁਵਧਾਿਜਨਕ ਅਤੇ ਕੁਸ਼ਲ ਹੱਲ ਪੇਸ਼ ਕਰਦੇ ਹਨ।

ਇੰਸਟ੍ਰਕਟ-ਟਊਿਨਿੰਗ ਦੀ ਪ੍ਰਕਰਿਆ ਵ ਿੱਚ ਮਨੁੱਖ-ਜਨਰੇਟ ਕੀਤੇ ਨਰਿਦੇਸ਼ ਪ੍ਰੌਮਪਟਸ ਅਤੇ ਜਵਾਬਾਂ ਦੇ ਇੱਕ ਵੱਡੇ ਸੰਗ੍ਰਹ ਿ’ਤੇ ਮਾਡਲ ਨੂੰ ਸਖਿਲਾਈ ਦੇਣਾ ਸ਼ਾਮਲ ਹੈ। ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਉਦਾਹਰਣ ਓਪਨ ਸੋਰਸ databricks-dolly-15k dataset ਹੈ, ਜਸਿ ਵ ਿੱਚ Databricks ਕਰਮਚਾਰੀਆਂ ਦੁਆਰਾ ਬਣਾਏ ਗਏ 15,000 ਤੋਂ ਵੱਧ ਪ੍ਰੌਮਪਟ/ਜਵਾਬ ਜੋੜੇ ਸ਼ਾਮਲ ਹਨ ਜਨ੍ਹਿਾਂ ਨੂੰ ਤੁਸੀਂ ਖੁਦ ਜਾਂਚ ਸਕਦੇ ਹੋ। ਡੇਟਾਸੈੱਟ ਅੱਠ ਵੱਖ-ਵੱਖ ਨਰਿਦੇਸ਼ ਸ਼੍ਰੇਣੀਆਂ ਨੂੰ ਕਵਰ ਕਰਦਾ ਹੈ, ਜਸਿ ਵ ਿੱਚ ਰਚਨਾਤਮਕ ਲੇਖਣ, ਬੰਦ ਅਤੇ ਖੁੱਲ੍ਹੇ ਸਵਾਲਾਂ ਦੇ ਜਵਾਬ, ਸੰਖੇਪੀਕਰਨ, ਜਾਣਕਾਰੀ ਕੱਢਣਾ, ਵਰਗੀਕਰਨ, ਅਤੇ ਵਚਿਾਰ-ਵਟਾਂਦਰਾ ਸ਼ਾਮਲ ਹਨ।

ਡੇਟਾ ਜਨਰੇਸ਼ਨ ਪ੍ਰਕਰਿਆ ਦੌਰਾਨ, ਯੋਗਦਾਨ ਪਾਉਣ ਵਾਲਿਆਂ ਨੂੰ ਹਰ ਸ੍ਰੇਣੀ ਲਈ ਪ੍ਰੌਮਪਟਸ ਅਤੇ ਜਵਾਬ ਬਣਾਉਣ ਬਾਰੇ ਦਸਿ਼ਾ-ਨਰਿਦੇਸ਼ ਦੱਤੇ ਗਏ ਸਨ। ਉਦਾਹਰਣ ਲਈ, ਰਚਨਾਤਮਕ ਲੇਖਣ ਕਾਰਜਾਂ ਲਈ, ਉਨ੍ਹਾਂ ਨੂੰ ਮਾਡਲ ਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਨਰਿਦੇਸ਼ਿਤ ਕਰਨ ਲਈ ਖਾਸ ਸੀਮਾਵਾਂ, ਹਦਾਇਤਾਂ, ਜਾਂ ਲੋੜਾਂ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਕਹਿਾ ਗਿਆ ਸੀ। ਬੰਦ ਸਵਾਲਾਂ ਦੇ ਜਵਾਬਾਂ ਲਈ, ਉਨ੍ਹਾਂ ਨੂੰ ਦੱਤੇ ਗਏ ਵਕਿੀਪੀਡੀਆ ਪੈਰ੍ਹੇ ਦੇ ਆਧਾਰ ’ਤੇ ਤੱਥਾਤਮਕ ਤੌਰ ’ਤੇ ਸਹੀ ਜਵਾਬਾਂ ਦੀ ਲੋੜ ਵਾਲੇ ਸਵਾਲ ਲਖਿਣ ਲਈ ਕਹਿਾ ਗਿਆ ਸੀ।

ਨਤੀਜੇ ਵਜੋਂ ਡੇਟਾਸੈੱਟ ChatGPT ਵਰਗੀਆਂ ਪ੍ਰਣਾਲੀਆਂ ਦੀਆਂ ਇੰਟਰਐਕਟਵਿ ਅਤੇ ਨਰਿਦੇਸ਼-ਪਾਲਣਾ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਵੱਡੀਆਂ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨੂੰ ਫਾਈਨ-ਟਊਿਨ ਕਰਨ ਲਈ ਇੱਕ ਮੁੱਲਵਾਨ ਸਰੋਤ ਵਜੋਂ ਕੰਮ ਕਰਦਾ ਹੈ। ਮਨੁੱਖ-ਜਨਰੇਟ ਕੀਤੇ ਨਰਿਦੇਸ਼ਾਂ ਅਤੇ ਜਵਾਬਾਂ ਦੀ ਵਭਿੰਨਿ ਸ੍ਰੇਣੀ ’ਤੇ ਸਖਿਲਾਈ ਦੁਆਰਾ, ਮਾਡਲ ਖਾਸ ਨਰਿਦੇਸ਼ਾਂ ਨੂੰ ਸਮਝਣਾ ਅਤੇ ਉਨ੍ਹਾਂ ਦੀ ਪਾਲਣਾ ਕਰਨਾ ਸੱਖਿਦਾ ਹੈ, ਜੋ ਇਸ ਨੂੰ ਕਈ ਤਰ੍ਹਾਂ ਦੇ ਕਾਰਜਾਂ ਨੂੰ ਸੰਭਾਲਣ ਵੱਚਿ ਵਧੇਰੇ ਕੁਸ਼ਲ ਬਣਾਉਦਾ ਹੈ।

ਸੱਧੀ ਫਾਈਨ-ਟਊਿਨੱਿਗ ਤੋਂ ਇਲਾਵਾ, databricks-dolly-15k ਵਰਗੇ ਡੇਟਾਸੈੱਟਾਂ ਵੱਚਿ ਨਰਿਦੇਸ਼ ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਸੱਥਿਟਕਿ ਡੇਟਾ ਜਨਰੇਸ਼ਨ ਲਈ ਵੀ ਵਰਤਆਿ ਜਾ ਸਕਦਾ ਹੈ। ਇੱਕ ਵੱਡੇ ਓਪਨ ਲੈਂਗੁਏਜ ਮਾਡਲ ਨੂੰ ਯੋਗਦਾਨ-ਜਨਰੇਟ ਕੀਤੇ ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਫਊਿ-ਸ਼ੌਟ ਉਦਾਹਰਣਾਂ ਵਜੋਂ ਸਬਮਟਿ ਕਰਕੇ, ਡਵੈਲਪਰ ਹਰ ਸ੍ਰੇਣੀ ਵੱਚਿ ਨਰਿਦੇਸ਼ਾਂ ਦਾ ਬਹੁਤ ਵੱਡਾ ਸੰਗ੍ਰਹਿ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਪਹੁੰਚ, ਜੋ ਸੈਲਫ-ਇੰਸਟ੍ਰਕਟ ਪੇਪਰ ਵੱਚਿ ਦੱਸੀ ਗਈ ਹੈ, ਵਧੇਰੇ ਮਜ਼ਬੂਤ ਨਰਿਦੇਸ਼-ਪਾਲਣਾ ਮਾਡਲਾਂ ਦੀ ਸਰਿਜਣਾ ਦੀ ਆਗਆਿ ਦੱਿਦੀ ਹੈ।

ਇਸ ਤੋਂ ਇਲਾਵਾ, ਇਨ੍ਹਾਂ ਡੇਟਾਸੈੱਟਾਂ ਵੱਚਿ ਨਰਿਦੇਸ਼ਾਂ ਅਤੇ ਜਵਾਬਾਂ ਨੂੰ ਪੈਰਾਫਰੇਜ਼ਿੱਗ ਵਰਗੀਆਂ ਤਕਨੀਕਾਂ ਦੁਆਰਾ ਵਧਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਹਰੇਕ ਪ੍ਰੌਮਪਟ ਜਾਂ ਛੋਟੇ ਜਵਾਬ ਨੂੰ ਦੁਬਾਰਾ ਬਆਿਨ ਕਰਕੇ ਅਤੇ ਨਤੀਜੇ ਵਜੋਂ ਟੈਕਸਟ ਨੂੰ ਸੰਬੰਧਤਿ ਗਰਾਊਡ-ਟਰੁੱਥ ਨਮੂਨੇ ਨਾਲ ਜੋੜ ਕੇ, ਡਵੈਲਪਰ ਇੱਕ ਕਸਿਮ ਦਾ ਨਯਿਮਤੀਕਰਨ ਪੇਸ਼ ਕਰ ਸਕਦੇ ਹਨ ਜੋ ਮਾਡਲ ਦੀ ਨਰਿਦੇਸ਼ਾਂ ਦੀ ਪਾਲਣਾ ਕਰਨ ਦੀ ਸਮਰੱਥਾ ਨੂੰ ਵਧਾਉਂਦਾ ਹੈ।

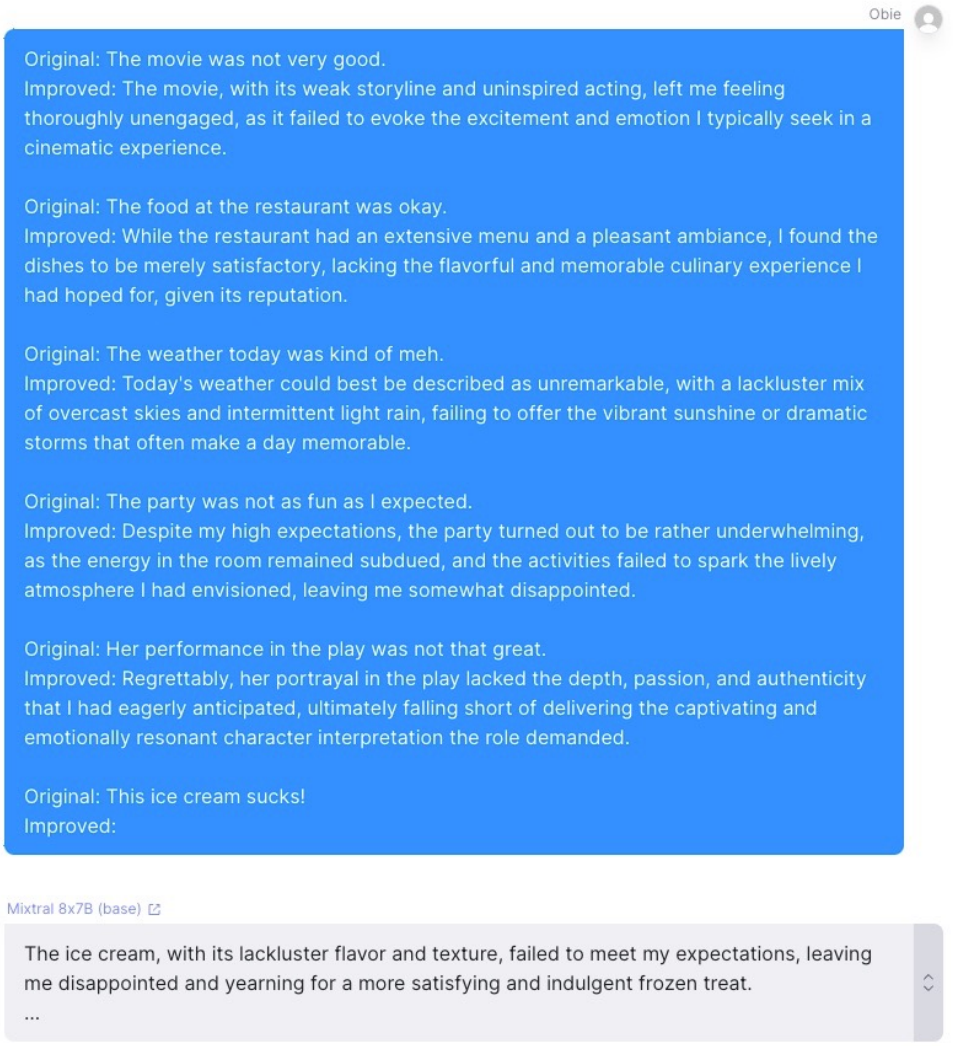
ਇੰਸਟ੍ਰਕਟ-ਟਊਿਨਡ ਮਾਡਲਾਂ ਦੁਆਰਾ ਪ੍ਰਦਾਨ ਕੀਤੀ ਵਰਤੋਂ ਦੀ ਸੌਖ ਕੁਝ ਲਚਕਤਾ ਦੀ ਕੀਮਤ ’ਤੇ ਆਉਂਦੀ ਹੈ। ਇਹ ਮਾਡਲ ਅਕਸਰ ਬਹੁਤ ਜ਼ਿਆਦਾ ਸੈਸਰ ਕੀਤੇ ਹੁੰਦੇ ਹਨ, ਜਸਿਦਾ ਮਤਲਬ ਹੈ ਕ ਿਉਹ ਹਮੇਸ਼ਾ ਕੁਝ ਖਾਸ ਕੰਮਾਂ ਲਈ ਲੋੜੀਂਦੀ ਰਚਨਾਤਮਕ ਆਜ਼ਾਦੀ ਪ੍ਰਦਾਨ ਨਹੀਂ ਕਰ ਸਕਦੇ। ਉਨ੍ਹਾਂ ਦੇ ਆਉਟਪੁੱਟ ਉਨ੍ਹਾਂ ਦੇ ਫਾਈਨ-ਟਊਿਨੱਿਗ ਡੇਟਾ ਵੱਚਿ ਮੌਜੂਦ ਪੱਖਪਾਤਾਂ ਅਤੇ ਸੀਮਾਵਾਂ ਤੋਂ ਬਹੁਤ ਪ੍ਰਭਾਵਤਿ ਹੁੰਦੇ ਹਨ।

ਇਨ੍ਹਾਂ ਸੀਮਾਵਾਂ ਦੇ ਬਾਵਜੂਦ, ਇੰਸਟ੍ਰਕਟ-ਟਊਿਨਡ ਮਾਡਲ ਆਪਣੀ ਵਰਤੋਂਕਾਰ-ਅਨੁਕੂਲ ਪ੍ਰਕਰਿਤੀ ਅਤੇ ਘੱਟ ਤੋਂ ਘੱਟ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰੱਿਗ ਨਾਲ ਵਆਿਪਕ ਕੰਮਾਂ ਨੂੰ ਸੰਭਾਲਣ ਦੀ ਯੋਗਤਾ ਕਾਰਨ ਵਧੇਰੇ ਲੋਕਪ੍ਰਯਿ ਹੋ ਗਏ ਹਨ। ਜਵਿੇਂ-ਜਵਿੇਂ ਹੋਰ ਉੱਚ-ਗੁਣਵੱਤਾ ਵਾਲੇ ਨਰਿਦੇਸ਼ ਡੇਟਾਸੇਟ ਉਪਲਬਧ ਹੋ ਰਹੇ ਹਨ, ਅਸੀਂ ਇਨ੍ਹਾਂ ਮਾਡਲਾਂ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਅਤੇ ਬਹੁਮੁਖਤਾ ਵੱਚਿ ਹੋਰ ਸੁਧਾਰ ਦੇਖਣ ਦੀ ਉਮੀਦ ਕਰ ਸਕਦੇ ਹਾਂ।

ਆਪਣੇ ਪ੍ਰੋਜੈਕਟ ਲਈ ਸਹੀ ਕਸਿਮ ਦਾ ਮਾਡਲ ਚੁਣਨਾ

ਬੇਸ (ਰਾਅ) ਅਤੇ ਇੰਸਟ੍ਰਕਟ-ਟਊਿਨਡ ਮਾਡਲਾਂ ਵਚਿਕਾਰ ਫੈਸਲਾ ਆਖਰਕਾਰ ਤੁਹਾਡੇ ਪ੍ਰੋਜੈਕਟ ਦੀਆਂ ਵਸਿ਼ਸ਼ ਲੋੜਾਂ 'ਤੇ ਨਰਿਭਰ ਕਰਦਾ ਹੈ। ਉਨ੍ਹਾਂ ਕੰਮਾਂ ਲਈ ਜੋ ਉੱਚ ਪੱਧਰ ਦੀ ਰਚਨਾਤਮਕਤਾ ਅਤੇ ਮੌਲਕਿਤਾ ਦੀ ਮੰਗ ਕਰਦੇ ਹਨ, ਬੇਸ ਮਾਡਲ ਨਵੀਨਤਾ ਲਈ ਇੱਕ ਸਕ਼ਤੀਸ਼ਾਲੀ ਟੂਲ ਪੇਸ਼ ਕਰਦੇ ਹਨ। ਇਹ ਮਾਡਲ ਡਵਿੈਲਪਰਾਂ ਨੂੰ ਐਲਐਲਐਮਜ਼ ਦੀ ਪੂਰੀ ਸਮਰੱਥਾ ਦੀ ਪੜਚੋਲ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦੰਦੇ ਹਨ, ਏਆਈ-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਰਾਹੀ ਕੀ ਹਾਸਲ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਦੀਆਂ ਸੀਮਾਵਾਂ ਨੂੰ ਧੱਕਦੇ ਹੋਏ, ਪਰ ਉਨ੍ਹਾਂ ਨੂੰ ਵਧੇਰੇ ਹੱਥੀ ਪਹੁੰਚ ਅਤੇ ਪ੍ਰਯੋਗ ਕਰਨ ਦੀ ਇੱਛਾ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਤਾਪਮਾਨ ਅਤੇ ਹੋਰ ਸੈਟਿੰਗਾਂ ਦਾ ਬੇਸ ਮਾਡਲਾਂ 'ਤੇ ਉਨ੍ਹਾਂ ਦੇ ਇੰਸਟ੍ਰਕਟ ਸਮਕਾਲੀਆਂ ਨਾਲੋ ਬਹੁਤ ਵੱਡਾ ਪ੍ਰਭਾਵ ਪੈਂਦਾ ਹੈ।

ਤੁਸੀਂ ਜੋ ਵੀ ਆਪਣੇ ਪ੍ਰੌਮਪਟ ਵੱਚਿ ਸ਼ਾਮਲ ਕਰਦੇ ਹੋ, ਬੇਸ ਮਾਡਲ ਉਸਨੂੰ ਦੁਹਰਾਉਣ ਦੀ ਕੋਸ਼ਸ਼ਿ ਕਰਨਗੇ। ਇਸ ਲਈ ਜੇਕਰ ਉਦਾਹਰਣ ਵਜੋਂ ਤੁਹਾਡਾ ਪ੍ਰੌਮਪਟ ਇੱਕ ਚੈਟ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਹੈ, ਤਾਂ ਰਾਅ ਮਾਡਲ ਚੈਟ ਨੂੰ ਜਾਰੀ ਰੱਖਣ ਦੀ ਕੋਸ਼ਸ਼ਿ ਕਰੇਗਾ। ਵੱਧ ਤੋਂ ਵੱਧ ਟੋਕਨ ਸੀਮਾ 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹੋਏ, ਇਹ ਸਰਿਫ਼ ਚੈਟ ਵੱਚਿ ਅਗਲਾ ਸੁਨੇਹਾ ਹੀ ਨਹੀ ਤਆਿਰ ਕਰੇਗਾ, ਇਹ ਆਪਣੇ ਆਪ ਨਾਲ ਪੂਰੀ ਗੱਲਬਾਤ ਕਰ ਸਕਦਾ ਹੈ!



ਚੰਡਿਰ 4. Mixtral 8x7B (ਬੇਸ) ਵਾਕ ਮੁੜ-ਲਖਿਣ ਦਾ ਫਿਊ-ਸੌਟ ਪੂਰਤੀ ਦਾ ਉਦਾਹਰਣ

Reddit ਯੂਜ਼ਰ phree_radical ਦੁਆਰਾ ਉੱਪਰ ਦੱਤੇ ਵਾਕ ਰੀਰਾਈਟਿੰਗ ਦੀ ਉਦਾਹਰਣ ਤਆਿਰ ਕਰਦੇ ਸਮੇਂ, ਮੈਂ ਪੈਰਾਮੀਟਰ ਸੈਟਿੰਗਾਂ ਨਾਲ ਬਹੁਤ ਪ੍ਰਯੋਗ ਕਰਨ ਤੋਂ ਬਾਅਦ ਹੀ ਵਰਤਣਯੋਗ ਨਤੀਜੇ ਪ੍ਰਾਪਤ ਕਰ ਸਕਆਿ, ਅੰਤ ਵੱਚਿ ਇਹਨਾਂ ਸੈਟਿੰਗਾਂ 'ਤੇ ਸਥਰਿ ਹੋਇਆ: Temperature 0.08, Top P: 0.2, Top K: 1, ਅਤੇ Repetition Penalty: 1.26।

ਪ੍ਰੋਡਕਸ਼ਨ ਵਾਂਚਿ ਬੇਸ ਮੌਡਲ ਨਾਲ ਇਸ ਪਹੁੰਚ ਦੀ ਵਰਤੋਂ ਕਰਨਾ `max_tokens` ਪੈਰਾਮੀਟਰ ਦੇ ਸ਼ਕਤੀਸ਼ਾਲੀ ਪ੍ਰਭਾਵ ਕਾਰਨ ਮੁਸ਼ਕਲ ਹੋਵੇਗਾ। ਇਸਨੂੰ ਬਹੁਤ ਛੋਟਾ ਸੈੱਟ ਕਰੋ ਤਾਂ ਆਉਟਪੁੱਟ ਕੱਟਿਆ ਜਾਂਦਾ ਹੈ। ਇਸਨੂੰ ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਲਈ ਮੌਡਲ ਦੀ ਜ਼ਰੂਰਤ ਤੋਂ ਵੱਧ ਲੰਬਾ ਸੈੱਟ ਕਰੋ, ਅਤੇ ਇਹ ਵਾਧੂ ਉਦਾਹਰਣਾਂ ਦਾ ਭਰਮ ਪੈਦਾ ਕਰਨਾ ਜਾਰੀ ਰੱਖੇਗਾ।

ਅੰਤਮਿ ਨਤੀਜਾ ਇਹ ਹੈ ਕਿ ਜਦੋਂ ਤੱਕ ਤੁਹਾਨੂੰ ਸੱਚਮੁੱਚ ਪੂਰੇ ਨਯਿੰਤਰਣ ਅਤੇ ਸੈਂਸਰਸ਼ਿਪ ਦੀ ਕਮੀ ਦੀ ਲੋੜ ਨਹੀਂ ਹੈ, ਇੰਸਟਰਕਟ-ਟਿਊਨਡ ਮੌਡਲ ਤੁਹਾਡੀ ਵਿਕਾਸ ਪ੍ਰਕਰਿਆ ਨੂੰ ਕਾਫ਼ੀ ਸਰਲ ਬਣਾ ਸਕਦੇ ਹਨ। ਇਸ ਗੱਲ ਨੂੰ ਸਪੱਸ਼ਟ ਕਰਨ ਲਈ, ਇੱਥੇ Mixtral 8x7B ਦਾ ਉਸੇ ਪ੍ਰੌਂਮਪਟ ਲਈ ਜਵਾਬ ਹੈ, ਪਰ ਇਸ ਵਾਰ ਇਸਦੇ ਇੰਸਟਰਕਟ-ਟਿਊਨਡ ਵਰਜਨ ਵਾਂਚਿ:

> ਮੈਨੂੰ ਇਹ ਦੱਸਣ 'ਚ ਅਫ਼ਸੋਸ ਹੈ ਕਿ ਆਈਸ ਕਰੀਮ ਮੇਰੀਆਂ ਉਮੀਦਾਂ 'ਤੇ ਖਰੀ ਨਹੀਂ ਉਤਰਦੀ, ਕਿਉਂਕਿ ਇਸ ਵਾਂਚਿ ਉਹ ਭਰਪੂਰ, ਕਰੀਮੀ ਬਣਤਰ ਅਤੇ ਸੁਆਦਲਾ ਜ਼ਾਇਕਾ ਨਹੀਂ ਹੈ ਜੋ ਮੈਂ ਆਮ ਤੌਰ 'ਤੇ ਉੱਚ-ਗੁਣਵੱਤਾ ਵਾਲੀ ਮੀਠੀ ਚੀਜ਼ ਨਾਲ ਜੋੜਦਾ ਹਾਂ। ਮੈਂ ਇੱਕ ਵਧੇਰੇ ਸੰਤੁਸ਼ਟੀਜਨਕ ਅਤੇ ਆਨੰਦਦਾਇਕ ਅਨੁਭਵ ਦੀ ਉਮੀਦ ਕਰ ਰਿਹਾ ਸੀ।

ਖਾਸ ਤੌਰ 'ਤੇ, ਮੈਂ max tokens ਸੈਟਿੰਗ ਨੂੰ 500 'ਤੇ ਛੱਡ ਸਕਿਆ, ਅਤੇ ਮੌਡਲ ਵਾਧੂ ਉਦਾਹਰਣਾਂ ਦਾ ਭਰਮ ਪੈਦਾ ਕੀਤੇ ਬਨਿਾਂ ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਦੇ ਅੰਤ 'ਤੇ ਭਰੋਸੇਯੋਗ ਢੰਗ ਨਾਲ ਰੁਕ ਗਿਆ।

ਪ੍ਰੌਂਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ

ਜਵਿੇਂ ਤੁਸੀਂ ਆਪਣੇ ਪ੍ਰੋਜੈਕਟਾਂ ਵਾਂਚਿ AI ਦੀ ਵਰਤੋਂ ਕਰਨਾ ਸ਼ੁਰੂ ਕਰਦੇ ਹੋ, ਤੁਸੀਂ ਜਲਦੀ ਹੀ ਖੋਜ ਲਵੋਗੇ ਕਿ ਤੁਹਾਨੂੰ ਮਹਾਰਤ ਹਾਸਲ ਕਰਨ ਲਈ ਸਭ ਤੋਂ ਮਹੱਤਵਪੂਰਨ ਹੁਨਰਾਂ ਵਾਂਚਿੋਂ ਇੱਕ ਪ੍ਰੌਂਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਦੀ ਕਲਾ ਹੈ। ਪਰ ਪ੍ਰੌਂਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਅਸਲ ਵਾਂਚਿ ਕੀ ਹੈ, ਅਤੇ ਇਹ ਇੰਨੀ ਮਹੱਤਵਪੂਰਨ ਕਿਉਂ ਹੈ?

ਇਸਦੇ ਮੂਲ ਵਾਂਚਿ, ਪ੍ਰੌਂਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਭਾਸ਼ਾ ਮੌਡਲ ਨੂੰ ਪ੍ਰਦਾਨ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਇਨਪੁੱਟ ਪ੍ਰੌਂਮਪਟਸ ਨੂੰ ਡਜ਼ਾਈਨ ਕਰਨ ਅਤੇ ਤਆਿਰ ਕਰਨ ਦੀ ਪ੍ਰਕਰਿਆ ਹੈ ਤਾਂ ਜੋ ਇਸਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਨਰਿਦੇਸ਼ਤਿ ਕੀਤਾ ਜਾ ਸਕੇ। ਇਹ AI ਨਾਲ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਦੀ ਸਮਝ ਬਾਰੇ ਹੈ, ਲੋੜੀਂਦੇ ਜਵਾਬ ਨੂੰ ਤਆਿਰ ਕਰਨ ਲਈ ਮੌਡਲ ਨੂੰ ਨਰਿਦੇਸ਼ਾਂ, ਉਦਾਹਰਣਾਂ, ਅਤੇ ਸੰਦਰਭ ਦੇ ਸੁਮੇਲ ਦੀ ਵਰਤੋਂ ਕਰਨਾ।

ਇਸ ਨੂੰ ਇੱਕ ਬਹੁਤ ਬੁੱਧੀਮਾਨ ਪਰ ਕੁਝ ਸ਼ਾਬਦਕਿ ਸੋਚ ਵਾਲੇ ਦੋਸਤ ਨਾਲ ਗੱਲਬਾਤ ਕਰਨ ਵਾਂਗ ਸੋਚੋ। ਗੱਲਬਾਤ ਤੋਂ ਵੱਧ ਤੋਂ ਵੱਧ ਲਾਭ ਲੈਣ ਲਈ, ਤੁਹਾਨੂੰ ਸਪੱਸ਼ਟ, ਵਸ਼ਿਸ਼, ਅਤੇ ਕਾਫ਼ੀ ਸੰਦਰਭ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਲੋੜ ਹੈ ਤਾਂ ਜੋ

ਯਕੀਨੀ ਬਣਾਇਆ ਜਾ ਸਕੇ ਕਿ ਤੁਹਾਡਾ ਦੋਸਤ ਠੀਕ ਸਮਝ ਰਹਿਆ ਹੈ ਕਿ ਤੁਸੀਂ ਕੀ ਪੁੱਛ ਰਹੇ ਹੋ। ਇੱਥੇ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਆਉਂਦੀ ਹੈ, ਅਤੇ ਭਾਵੇਂ ਇਹ ਸ਼ੁਰੂ ਵਿੱਚ ਆਸਾਨ ਲੱਗੇ, ਮੇਰਾ ਯਕੀਨ ਮੰਨੋ ਕਿ ਇਸ ਵਿੱਚ ਮਹਾਰਤ ਹਾਸਲ ਕਰਨ ਲਈ ਬਹੁਤ ਅਭਿਆਸ ਦੀ ਲੋੜ ਹੈ।

ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੌਮਪਟਸ ਦੇ ਬੁਨਿਆਦੀ ਤੱਤ

ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੌਮਪਟਸ ਦੀ ਇੰਜੀਨੀਅਰਿੰਗ ਸ਼ੁਰੂ ਕਰਨ ਲਈ, ਪਹਿਲਾਂ ਤੁਹਾਨੂੰ ਉਨ੍ਹਾਂ ਮੁੱਖ ਹਿੱਸਿਆਂ ਨੂੰ ਸਮਝਣ ਦੀ ਲੋੜ ਹੈ ਜੋ ਇੱਕ ਚੰਗੀ ਤਰ੍ਹਾਂ ਤਿਆਰ ਕੀਤੇ ਇਨਪੁੱਟ ਦਾ ਨਿਰਮਾਣ ਕਰਦੇ ਹਨ। ਇੱਥੇ ਕੁਝ ਜ਼ਰੂਰੀ ਬੁਨਿਆਦੀ ਤੱਤ ਹਨ:

1. **ਹਦਾਇਤਾਂ**: ਸਪੱਸ਼ਟ ਅਤੇ ਸੰਖੇਪ ਹਦਾਇਤਾਂ ਜੋ ਮਾਡਲ ਨੂੰ ਦੱਸਦੀਆਂ ਹਨ ਕਿ ਤੁਸੀਂ ਕੀ ਚਾਹੁੰਦੇ ਹੋ। ਇਹ ਕੁਝ ਵੀ ਹੋ ਸਕਦਾ ਹੈ, "ਹੇਠ ਲਿਖੇ ਲੇਖ ਦਾ ਸਾਰ ਦੱਸੋ" ਤੋਂ ਲੈ ਕੇ "ਸੂਰਜ ਡੁੱਬਣ ਬਾਰੇ ਇੱਕ ਕਵਿਤਾ ਲਿਖੋ" ਤੱਕ ਜਾਂ "ਇਸ ਪ੍ਰੋਜੈਕਟ ਬਦਲਾਅ ਬੇਨਤੀ ਨੂੰ JSON ਆਬਜੈਕਟ ਵਿੱਚ ਬਦਲੋ"।
2. **ਸੰਦਰਭ**: ਢੁਕਵੀਂ ਜਾਣਕਾਰੀ ਜੋ ਮਾਡਲ ਨੂੰ ਕਾਰਜ ਦੀ ਪਿੱਠਭੂਮੀ ਅਤੇ ਦਾਇਰੇ ਨੂੰ ਸਮਝਣ ਵਿੱਚ ਮਦਦ ਕਰਦੀ ਹੈ। ਇਸ ਵਿੱਚ ਲਕਸ਼ਿਤ ਦਰਸ਼ਕਾਂ, ਲੋੜੀਂਦੇ ਲਹਜ਼ੇ ਅਤੇ ਸ਼ੈਲੀ, ਜਾਂ ਆਉਟਪੁੱਟ ਲਈ ਕੋਈ ਵਿਸ਼ੇਸ਼ ਸੀਮਾਵਾਂ ਜਾਂ ਲੋੜਾਂ ਬਾਰੇ ਵੇਰਵੇ ਸ਼ਾਮਲ ਹੋ ਸਕਦੇ ਹਨ, ਜਿਵੇਂ ਕਿ ਪਾਲਣਾ ਕਰਨ ਲਈ JSON ਸਕੀਮਾ।
3. **ਉਦਾਹਰਣਾਂ**: ਠੋਸ ਉਦਾਹਰਣਾਂ ਜੋ ਦਰਸਾਉਂਦੀਆਂ ਹਨ ਕਿ ਤੁਸੀਂ ਕਿਸ ਤਰ੍ਹਾਂ ਦੇ ਆਉਟਪੁੱਟ ਦੀ ਤਲਾਸ਼ ਕਰ ਰਹੇ ਹੋ। ਕੁਝ ਚੰਗੀ ਤਰ੍ਹਾਂ ਚੁਣੀਆਂ ਉਦਾਹਰਣਾਂ ਪ੍ਰਦਾਨ ਕਰਕੇ, ਤੁਸੀਂ ਮਾਡਲ ਨੂੰ ਲੋੜੀਂਦੇ ਜਵਾਬ ਦੇ ਪੈਟਰਨ ਅਤੇ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਸਿੱਖਣ ਵਿੱਚ ਮਦਦ ਕਰ ਸਕਦੇ ਹੋ।
4. **ਇਨਪੁੱਟ ਫਾਰਮੈਟਿੰਗ**: ਲਾਈਨ ਬ੍ਰੇਕਸ ਅਤੇ ਮਾਰਕਡਾਊਨ ਫਾਰਮੈਟਿੰਗ ਸਾਡੇ ਪ੍ਰੌਮਪਟ ਨੂੰ ਢਾਂਚਾ ਦਿੰਦੇ ਹਨ। ਪ੍ਰੌਮਪਟ ਨੂੰ ਪੈਰਾਗ੍ਰਾਫਾਂ ਵਿੱਚ ਵੰਡਣਾ ਸਾਨੂੰ ਸੰਬੰਧਿਤ ਹਦਾਇਤਾਂ ਨੂੰ ਇਕੱਠਾ ਕਰਨ ਦਿੰਦਾ ਹੈ, ਤਾਂ ਜੋ ਮਨੁੱਖਾਂ ਅਤੇ AI ਦੋਵਾਂ ਲਈ ਇਸਨੂੰ ਸਮਝਣਾ ਆਸਾਨ ਹੋਵੇ। ਬੁਲੇਟ ਅਤੇ ਨੰਬਰ ਵਾਲੀਆਂ ਸੂਚੀਆਂ ਸਾਨੂੰ ਆਈਟਮਾਂ ਦੀ ਸੂਚੀ ਅਤੇ ਕ੍ਰਮ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਦਿੰਦੀਆਂ ਹਨ। ਬੋਲਡ ਅਤੇ ਇਟੈਲਿਕ ਮਾਰਕਰ ਸਾਨੂੰ ਜ਼ੋਰ ਦੇਣ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।
5. **ਆਉਟਪੁੱਟ ਫਾਰਮੈਟਿੰਗ**: ਆਉਟਪੁੱਟ ਦੀ ਬਣਤਰ ਅਤੇ ਫਾਰਮੈਟਿੰਗ ਬਾਰੇ ਵਿਸ਼ੇਸ਼ ਹਦਾਇਤਾਂ। ਇਨ੍ਹਾਂ ਵਿੱਚ ਲੋੜੀਂਦੀ ਲੰਬਾਈ, ਹੈਡਿੰਗਜ਼ ਜਾਂ ਬੁਲੇਟ ਪੁਆਇੰਟਾਂ ਦੀ ਵਰਤੋਂ, ਮਾਰਕਡਾਊਨ ਫਾਰਮੈਟਿੰਗ, ਜਾਂ ਕੋਈ ਹੋਰ ਵਿਸ਼ੇਸ਼ ਆਉਟਪੁੱਟ ਟੈਂਪਲੇਟ ਜਾਂ ਮਿਆਰ ਸ਼ਾਮਲ ਹੋ ਸਕਦੇ ਹਨ ਜਿਨ੍ਹਾਂ ਦੀ ਪਾਲਣਾ ਕੀਤੀ ਜਾਣੀ ਚਾਹੀਦੀ ਹੈ।

ਇਨ੍ਹਾਂ ਬੁਨਿਆਦੀ ਤੱਤਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਤਰੀਕਿਆਂ ਨਾਲ ਜੋੜ ਕੇ, ਤੁਸੀਂ ਅਜਿਹੇ ਪ੍ਰੌਮਪਟਸ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਤੁਹਾਡੀਆਂ ਵਿਸ਼ਿਸ਼ ਲੋੜਾਂ ਦੇ ਅਨੁਕੂਲ ਹੋਣ ਅਤੇ ਮਾਡਲ ਨੂੰ ਉੱਚ-ਗੁਣਵੱਤਾ, ਢੁਕਵੇਂ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਵੱਲ ਸੇਧਤਿ ਕਰਨ।

ਪ੍ਰੌਮਪਟ ਡਿਜ਼ਾਈਨ ਦੀ ਕਲਾ ਅਤੇ ਵਿਗਿਆਨ

ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੌਮਪਟਸ ਦੀ ਰਚਨਾ ਕਰਨਾ ਇੱਕ ਕਲਾ ਅਤੇ ਵਿਗਿਆਨ ਦੋਵੇਂ ਹੈ। (ਇਸੇ ਲਈ ਅਸੀਂ ਇਸਨੂੰ ਕਰਾਫਟ ਕਹਿੰਦੇ ਹਾਂ।) ਇਸ ਲਈ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਅਤੇ ਸੀਮਾਵਾਂ ਦੀ ਡੂੰਘੀ ਸਮਝ ਦੇ ਨਾਲ-ਨਾਲ ਲੋੜੀਂਦੇ ਵਿਵਹਾਰ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਡਿਜ਼ਾਈਨ ਕਰਨ ਦਾ ਰਚਨਾਤਮਕ ਨਜ਼ਰੀਆ ਚਾਹੀਦਾ ਹੈ। ਇਸ ਵਿੱਚ ਸ਼ਾਮਲ ਰਚਨਾਤਮਕਤਾ ਹੀ ਇਸ ਨੂੰ ਮੇਰੇ ਲਈ ਇੰਨਾ ਮਜ਼ੇਦਾਰ ਬਣਾਉਂਦੀ ਹੈ। ਇਹ ਬਹੁਤ ਨਿਰਾਸ਼ਾਜਨਕ ਵੀ ਹੋ ਸਕਦਾ ਹੈ, ਖਾਸ ਕਰਕੇ ਜਦੋਂ ਤੁਸੀਂ ਨਿਰਧਾਰਤ ਵਿਵਹਾਰ ਦੀ ਭਾਲ ਕਰ ਰਹੇ ਹੋਵੋ।

ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਦਾ ਇੱਕ ਮੁੱਖ ਪਹਲੂ ਵਿਸ਼ਿਸ਼ਤਾ ਅਤੇ ਲਚਕਤਾ ਵਿਚਕਾਰ ਸੰਤੁਲਨ ਨੂੰ ਸਮਝਣਾ ਹੈ। ਇੱਕ ਪਾਸੇ, ਤੁਸੀਂ ਮਾਡਲ ਨੂੰ ਸਹੀ ਦਿਸ਼ਾ ਵੱਲ ਸੇਧ ਦੇਣ ਲਈ ਕਾਫ਼ੀ ਮਾਰਗਦਰਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹੋ। ਦੂਜੇ ਪਾਸੇ, ਤੁਸੀਂ ਇੰਨੇ ਨਿਯਮਬੱਧ ਨਹੀਂ ਹੋਣਾ ਚਾਹੁੰਦੇ ਕਿ ਤੁਸੀਂ ਸੀਮਾ ਮਾਮਲਿਆਂ ਨਾਲ ਨਜਿੱਠਣ ਲਈ ਮਾਡਲ ਦੀ ਆਪਣੀ ਰਚਨਾਤਮਕਤਾ ਅਤੇ ਲਚਕਤਾ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦੀ ਯੋਗਤਾ ਨੂੰ ਸੀਮਤ ਕਰ ਦਿਓ।

ਇੱਕ ਹੋਰ ਮਹੱਤਵਪੂਰਨ ਵਿਚਾਰ ਉਦਾਹਰਣਾਂ ਦੀ ਵਰਤੋਂ ਹੈ। ਚੰਗੀ ਤਰ੍ਹਾਂ ਚੁਣੀਆਂ ਗਈਆਂ ਉਦਾਹਰਣਾਂ ਮਾਡਲ ਨੂੰ ਤੁਹਾਡੇ ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਦੀ ਕਿਸਮ ਨੂੰ ਸਮਝਣ ਵਿੱਚ ਬਹੁਤ ਮਦਦਗਾਰ ਹੋ ਸਕਦੀਆਂ ਹਨ। ਹਾਲਾਂਕਿ, ਇਹ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਉਦਾਹਰਣਾਂ ਦੀ ਸਮਝਦਾਰੀ ਨਾਲ ਵਰਤੋਂ ਕੀਤੀ ਜਾਵੇ ਅਤੇ ਇਹ ਯਕੀਨੀ ਬਣਾਇਆ ਜਾਵੇ ਕਿ ਉਹ ਲੋੜੀਂਦੇ ਜਵਾਬ ਦੀ ਨੁਮਾਇੰਦਗੀ ਕਰਦੀਆਂ ਹਨ। ਇੱਕ ਮਾੜੀ ਉਦਾਹਰਣ ਸਭ ਤੋਂ ਵਧੀਆ ਸਥਿਤੀ ਵਿੱਚ ਟੋਕਨਾਂ ਦੀ ਬਰਬਾਦੀ ਹੈ, ਅਤੇ ਸਭ ਤੋਂ ਮਾੜੀ ਸਥਿਤੀ ਵਿੱਚ ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਲਈ ਤਬਾਹਕੁਨ ਹੈ।

ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਤਕਨੀਕਾਂ ਅਤੇ ਸਰਵੋਤਮ ਅਭਿਆਸ

ਜਦੋਂ ਤੁਸੀਂ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਦੀ ਦੁਨੀਆ ਵਿੱਚ ਡੂੰਘੇ ਉਤਰਦੇ ਹੋ, ਤੁਸੀਂ ਕਈ ਤਕਨੀਕਾਂ ਅਤੇ ਸਰਵੋਤਮ ਅਭਿਆਸਾਂ ਦੀ ਖੋਜ ਕਰੋਗੇ ਜੋ ਤੁਹਾਨੂੰ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੌਮਪਟਸ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰ ਸਕਦੇ ਹਨ। ਇੱਥੇ ਕੁਝ ਮੁੱਖ ਖੇਤਰ ਹਨ ਜਿਨ੍ਹਾਂ ਦੀ ਪੜਚੋਲ ਕਰਨੀ ਹੈ:

1. **ਜ਼ੀਰੋ-ਸ਼ੌਟ ਬਨਾਮ ਫਿਊ-ਸ਼ੌਟ ਲਰਨਿੰਗ**: ਜ਼ੀਰੋ-ਸ਼ੌਟ ਲਰਨਿੰਗ (ਕੋਈ ਉਦਾਹਰਣ ਨਾ ਦੇਣਾ) ਬਨਾਮ ਵਨ-ਸ਼ੌਟ ਜਾਂ ਫਿਊ-ਸ਼ੌਟ ਲਰਨਿੰਗ (ਥੋੜ੍ਹੀਆਂ ਉਦਾਹਰਣਾਂ ਦੇਣਾ) ਦੀ ਵਰਤੋਂ ਕਦੋਂ ਕਰਨੀ ਹੈ, ਇਹ ਸਮਝਣਾ ਤੁਹਾਨੂੰ ਵਧੇਰੇ ਕੁਸ਼ਲ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੌਮਪਟਸ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰ ਸਕਦਾ ਹੈ।

2. **ਦੁਹਰਾਉਣ ਵਾਲਾ ਸੁਧਾਰ**: ਮਾਡਲ ਦੇ ਆਉਟਪੁੱਟ ਦੇ ਆਧਾਰ 'ਤੇ ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਲਗਾਤਾਰ ਸੁਧਾਰਨ ਦੀ ਪ੍ਰਕਰਿਆ ਤੁਹਾਨੂੰ ਸਭ ਤੋਂ ਵਧੀਆ ਪ੍ਰੌਮਪਟ ਡਿਜ਼ਾਈਨ 'ਤੇ ਪਹੁੰਚਣ ਵੱਲਿ ਮਦਦ ਕਰ ਸਕਦੀ ਹੈ। ਫੀਡਬੈਕ ਲੂਪ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਪਹੁੰਚ ਹੈ ਜੋ ਤਆਿਰ ਕੀਤੀ ਸਮੱਗਰੀ ਦੀ ਗੁਣਵੱਤਾ ਅਤੇ ਪ੍ਰਸੰਗਕਿਤਾ ਨੂੰ ਲਗਾਤਾਰ ਸੁਧਾਰਨ ਲਈ ਭਾਸ਼ਾ ਮਾਡਲ ਦੇ ਆਪਣੇ ਆਉਟਪੁੱਟ ਦਾ ਲਾਭ ਲੈਂਦੀ ਹੈ।
3. **ਪ੍ਰੌਮਪਟ ਚੇਨਿੰਗ**: ਗੁੰਝਲਦਾਰ ਕੰਮਾਂ ਨੂੰ ਛੋਟੇ, ਵਧੇਰੇ ਪ੍ਰਬੰਧਨਯੋਗ ਕਦਮਾਂ ਵੱਲਿ ਵੰਡਣ ਲਈ ਕਈ ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਲੜੀ ਵੱਲਿ ਜੋੜਆਿ ਜਾ ਸਕਦਾ ਹੈ। ਪ੍ਰੌਮਪਟ ਚੇਨਿੰਗ ਵੱਲਿ ਗੁੰਝਲਦਾਰ ਕੰਮ ਜਾਂ ਗੱਲਬਾਤ ਨੂੰ ਛੋਟੇ, ਆਪਸ ਵੱਲਿ ਜੁੜੇ ਪ੍ਰੌਮਪਟਸ ਦੀ ਲੜੀ ਵੱਲਿ ਵੰਡਣਾ ਸ਼ਾਮਲ ਹੈ। ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਇਕੱਠੇ ਜੋੜ ਕੇ, ਤੁਸੀਂ AI ਨੂੰ ਬਹੁ-ਪੜਾਵੀ ਪ੍ਰਕਰਿਆ ਵੱਲਿੋਂ ਅਗਵਾਈ ਕਰ ਸਕਦੇ ਹੋ, ਗੱਲਬਾਤ ਦੌਰਾਨ ਸੰਦਰਭ ਅਤੇ ਸੰਗਤੀ ਨੂੰ ਬਣਾਈ ਰੱਖਦੇ ਹੋਏ।
4. **ਪ੍ਰੌਮਪਟ ਟਊਿਨਿੰਗ**: ਵਸ਼ਿਸ਼ ਖੇਤਰਾਂ ਜਾਂ ਕੰਮਾਂ ਲਈ ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਵਸ਼ਿਸ਼ ਤੌਰ 'ਤੇ ਢਾਲਣਾ ਤੁਹਾਨੂੰ ਵਧੇਰੇ ਵਸ਼ਿਸ਼ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੌਮਪਟਸ ਬਣਾਉਣ ਵੱਲਿ ਮਦਦ ਕਰ ਸਕਦਾ ਹੈ। ਪ੍ਰੌਮਪਟ ਟੈਂਪਲੇਟ ਤੁਹਾਨੂੰ ਲਚਕਦਾਰ, ਮੁੜ-ਵਰਤੋਂਯੋਗ, ਅਤੇ ਰੱਖ-ਰਖਾਵ ਯੋਗ ਪ੍ਰੌਮਪਟ ਢਾਂਚੇ ਬਣਾਉਣ ਵੱਲਿ ਮਦਦ ਕਰਦਾ ਹੈ ਜੋ ਦੱਤਿ ਕੰਮ ਲਈ ਵਧੇਰੇ ਅਨੁਕੂਲ ਹਨ।

ਜ਼ੀਰੋ-ਸ਼ੌਟ, ਵਨ-ਸ਼ੌਟ, ਜਾਂ ਫਊਿ-ਸ਼ੌਟ ਲਰਨਿੰਗ ਦੀ ਵਰਤੋਂ ਕਦੋਂ ਕਰਨੀ ਹੈ, ਇਹ ਸਖਿਣਾ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਨੂੰ ਮਾਸਟਰ ਕਰਨ ਦਾ ਖ਼ਾਸ ਤੌਰ 'ਤੇ ਮਹੱਤਵਪੂਰਨ ਹੱਸਿਾ ਹੈ। ਹਰ ਪਹੁੰਚ ਦੀਆਂ ਆਪਣੀਆਂ ਤਾਕਤਾਂ ਅਤੇ ਕਮਜ਼ੋਰੀਆਂ ਹਨ, ਅਤੇ ਹਰ ਇੱਕ ਦੀ ਵਰਤੋਂ ਕਦੋਂ ਕਰਨੀ ਹੈ ਇਹ ਸਮਝਣ ਨਾਲ ਤੁਹਾਨੂੰ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਅਤੇ ਕੁਸ਼ਲ ਪ੍ਰੌਮਪਟਸ ਬਣਾਉਣ ਵੱਲਿ ਮਦਦ ਮਲਿ ਸਕਦੀ ਹੈ।

ਜ਼ੀਰੋ-ਸ਼ੌਟ ਲਰਨਿੰਗ: ਜਦੋਂ ਕਸਿੇ ਉਦਾਹਰਣ ਦੀ ਲੋੜ ਨਹੀਂ ਹੁੰਦੀ

ਜ਼ੀਰੋ-ਸ਼ਾਟ ਲਰਨਿੰਗ ਭਾਸ਼ਾ ਮਾਡਲ ਦੀ ਉਸ ਯੋਗਤਾ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ ਜਸਿ ਵੱਲਿ ਬਨਿਾਂ ਕਸਿੇ ਉਦਾਹਰਣ ਜਾਂ ਸਪੱਸ਼ਟ ਸਖਿਲਾਈ ਦੇ ਕੋਈ ਕੰਮ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵੱਲਿ, ਤੁਸੀਂ ਮਾਡਲ ਨੂੰ ਇੱਕ ਪ੍ਰੌਮਪਟ ਦੱਦਿੇ ਹੋ ਜੋ ਕੰਮ ਦਾ ਵਰਣਨ ਕਰਦਾ ਹੈ, ਅਤੇ ਮਾਡਲ ਸਰਫ਼ ਆਪਣੇ ਪਹਲਿਾਂ ਤੋਂ ਮੌਜੂਦ ਗਆਿਨ ਅਤੇ ਭਾਸ਼ਾ ਦੀ ਸਮਝ ਦੇ ਆਧਾਰ 'ਤੇ ਜਵਾਬ ਤਆਿਰ ਕਰਦਾ ਹੈ।

ਜ਼ੀਰੋ-ਸ਼ਾਟ ਲਰਨਿੰਗ ਖ਼ਾਸ ਤੌਰ 'ਤੇ ਉਦੋਂ ਲਾਭਦਾਇਕ ਹੁੰਦੀ ਹੈ ਜਦੋਂ:

1. ਕੰਮ ਤੁਲਨਾਤਮਕ ਤੌਰ 'ਤੇ ਸਰਲ ਅਤੇ ਸੱਧਿਾ ਹੋਵੇ, ਅਤੇ ਮਾਡਲ ਨੇ ਆਪਣੀ ਪੂਰਵ-ਸਖਿਲਾਈ ਦੌਰਾਨ ਇਸ ਤਰ੍ਹਾਂ ਦੇ ਕੰਮਾਂ ਦਾ ਸਾਹਮਣਾ ਕੀਤਾ ਹੋਵੇ।

2. ਤੁਸੀਂ ਮਾਡਲ ਦੀਆਂ ਅੰਤਰਨਿਹਿਤ ਸਮਰੱਥਾਵਾਂ ਦੀ ਜਾਂਚ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹੋ ਅਤੇ ਦੇਖਣਾ ਚਾਹੁੰਦੇ ਹੋ ਕਿ ਇਹ ਬਿਨਾਂ ਕਿਸੇ ਵਾਧੂ ਮਾਰਗਦਰਸ਼ਨ ਦੇ ਨਵੇਂ ਕੰਮ 'ਤੇ ਕਿਵੇਂ ਪ੍ਰਤੀਕਰਿਆ ਕਰਦਾ ਹੈ।
3. ਤੁਸੀਂ ਇੱਕ ਵੱਡੇ ਅਤੇ ਵਿਭਿੰਨ ਭਾਸ਼ਾ ਮਾਡਲ ਨਾਲ ਕੰਮ ਕਰ ਰਹੇ ਹੋ ਜੋ ਕੰਮਾਂ ਅਤੇ ਖੇਤਰਾਂ ਦੀ ਵਿਆਪਕ ਸ਼੍ਰੇਣੀ 'ਤੇ ਸਿਖਲਾਈ ਪ੍ਰਾਪਤ ਕਰ ਚੁੱਕਾ ਹੈ।

ਹਾਲਾਂਕਿ, ਜ਼ੀਰੋ-ਸ਼ਾਟ ਲਰਨਿੰਗ ਅਣਕਿਆਸੀ ਵੀ ਹੋ ਸਕਦੀ ਹੈ ਅਤੇ ਹਮੇਸ਼ਾ ਲੋੜੀਂਦੇ ਨਤੀਜੇ ਨਹੀਂ ਦੇ ਸਕਦੀ। ਮਾਡਲ ਦੀ ਪ੍ਰਤੀਕਰਿਆ ਇਸਦੇ ਪੂਰਵ-ਸਿਖਲਾਈ ਡੇਟਾ ਵਿੱਚ ਪੱਖਪਾਤ ਜਾਂ ਅਸੰਗਤੀਆਂ ਤੋਂ ਪ੍ਰਭਾਵਿਤ ਹੋ ਸਕਦੀ ਹੈ, ਅਤੇ ਇਹ ਵਧੇਰੇ ਗੁੰਝਲਦਾਰ ਜਾਂ ਬਰੀਕ ਕੰਮਾਂ ਨਾਲ ਸੰਘਰਸ਼ ਕਰ ਸਕਦਾ ਹੈ।

> ਮੈਂ ਜ਼ੀਰੋ-ਸ਼ਾਟ ਪ੍ਰੌਮਪਟ ਦੇਖੇ ਹਨ ਜੋ ਮੇਰੇ 80% ਟੈਸਟ ਕੇਸਾਂ ਲਈ ਠੀਕ ਕੰਮ ਕਰਦੇ ਹਨ ਅਤੇ ਬਾਕੀ 20% ਲਈ ਬਹੁਤ ਗਲਤ ਜਾਂ ਅਸਮਝ ਨਤੀਜੇ ਦਿੰਦੇ ਹਨ। ਇੱਕ ਵਿਆਪਕ ਟੈਸਟਿੰਗ ਪ੍ਰਣਾਲੀ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਬਹੁਤ ਮਹੱਤਵਪੂਰਨ ਹੈ, ਖਾਸ ਕਰਕੇ ਜੇ ਤੁਸੀਂ ਜ਼ੀਰੋ-ਸ਼ਾਟ ਪ੍ਰੌਮਪਟਿੰਗ 'ਤੇ ਬਹੁਤ ਭਰੋਸਾ ਕਰ ਰਹੇ ਹੋ।

ਵਨ-ਸ਼ਾਟ ਲਰਨਿੰਗ: ਜਦੋਂ ਇੱਕ ਉਦਾਹਰਣ ਫ਼ਰਕ ਲਿਆ ਸਕਦੀ ਹੈ

ਵਨ-ਸ਼ਾਟ ਲਰਨਿੰਗ ਵਿੱਚ ਕੰਮ ਦੇ ਵੇਰਵੇ ਦੇ ਨਾਲ ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਦੀ ਇੱਕ ਉਦਾਹਰਣ ਮਾਡਲ ਨੂੰ ਪ੍ਰਦਾਨ ਕਰਨਾ ਸ਼ਾਮਲ ਹੈ। ਇਹ ਉਦਾਹਰਣ ਇੱਕ ਟੈਂਪਲੇਟ ਜਾਂ ਪੈਟਰਨ ਵਜੋਂ ਕੰਮ ਕਰਦੀ ਹੈ ਜਿਸ ਨੂੰ ਮਾਡਲ ਆਪਣਾ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਲਈ ਵਰਤ ਸਕਦਾ ਹੈ।

ਵਨ-ਸ਼ਾਟ ਲਰਨਿੰਗ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਹੋ ਸਕਦੀ ਹੈ ਜਦੋਂ:

1. ਕੰਮ ਤੁਲਨਾਤਮਕ ਤੌਰ 'ਤੇ ਨਵਾਂ ਜਾਂ ਵਿਸ਼ਿਸ਼ ਹੈ, ਅਤੇ ਮਾਡਲ ਨੇ ਆਪਣੀ ਪੂਰਵ-ਸਿਖਲਾਈ ਦੌਰਾਨ ਬਹੁਤ ਸਾਰੀਆਂ ਸਮਾਨ ਉਦਾਹਰਣਾਂ ਦਾ ਸਾਹਮਣਾ ਨਹੀਂ ਕੀਤਾ ਹੋ ਸਕਦਾ।
2. ਤੁਸੀਂ ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਫਾਰਮੈਟ ਜਾਂ ਸ਼ੈਲੀ ਦਾ ਸਪੱਸ਼ਟ ਅਤੇ ਸੰਖੇਪ ਪ੍ਰਦਰਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹੋ।
3. ਕੰਮ ਲਈ ਇੱਕ ਵਿਸ਼ਿਸ਼ ਢਾਂਚੇ ਜਾਂ ਰੀਤੀ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ ਜੋ ਸਿਰਫ਼ ਕੰਮ ਦੇ ਵੇਰਵੇ ਤੋਂ ਸਪੱਸ਼ਟ ਨਹੀਂ ਹੋ ਸਕਦੀ।

ਜੋ ਵੇਰਵੇ ਤੁਹਾਡੇ ਲਈ ਸਪੱਸ਼ਟ ਹਨ, ਉਹ ਜ਼ਰੂਰੀ ਨਹੀਂ ਕਿ AI ਲਈ ਵੀ ਸਪੱਸ਼ਟ ਹੋਣ। ਵਨ-ਸ਼ਾਟ ਉਦਾਹਰਣਾਂ ਚੀਜ਼ਾਂ ਨੂੰ ਸਪੱਸ਼ਟ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰ ਸਕਦੀਆਂ ਹਨ।

ਵਨ-ਸ਼ਾਟ ਲਰਨਿੰਗ ਮਾਡਲ ਨੂੰ ਉਮੀਦਾਂ ਨੂੰ ਵਧੇਰੇ ਸਪੱਸ਼ਟ ਤੌਰ 'ਤੇ ਸਮਝਣ ਅਤੇ ਦਿੱਤੀ ਗਈ ਉਦਾਹਰਣ ਦੇ ਨਾਲ ਵਧੇਰੇ ਨੇੜਿਓਂ ਮੇਲ ਖਾਂਦਾ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰ ਸਕਦੀ ਹੈ। ਹਾਲਾਂਕਿ, ਉਦਾਹਰਣ ਨੂੰ ਧਿਆਨ ਨਾਲ ਚੁਣਨਾ ਅਤੇ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਇਹ ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਦੀ ਪ੍ਰਤੀਨਿਧਿਤਾ ਕਰਦੀ ਹੈ। ਉਦਾਹਰਣ ਚੁਣਦੇ ਸਮੇਂ, ਆਪਣੇ ਆਪ ਨੂੰ ਸੰਭਾਵੀ ਐਜ ਕੇਸਾਂ ਅਤੇ ਇਨਪੁੱਟ ਦੀ ਰੇਂਜ ਬਾਰੇ ਪੁੱਛੋ ਜੋ ਪ੍ਰੌਮਪਟ ਨੂੰ ਸੰਭਾਲਣੀ ਪਵੇਗੀ।

ਚਿੱਤਰ 5. JSON ਦੀ ਇੱਕ ਵਾਰੀ ਦੀ ਲੋੜੀਂਦੀ ਉਦਾਹਰਨ

```
Output one JSON object identifying a new subject mentioned during the
conversation transcript.

The JSON object should have three keys, all required:
- name: The name of the subject
- description: brief, with details that might be relevant to the user
- type: Do not use any other type than the ones listed below

Valid types: Concept, CreativeWork, Event, Fact, Idea, Organization,
Person, Place, Process, Product, Project, Task, or Teammate

This is an example of well-formed output:

{
  "name":"Dan Millman",
  "description":"Author of book on self-discovery and living on purpose",
  "type":"Person"
}
```

ਫਿਊ-ਸ਼ਾਟ ਲਰਨਿੰਗ: ਜਦੋਂ ਕਈ ਉਦਾਹਰਣਾਂ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਬਹਿਤਰ ਬਣਾ ਸਕਦੀਆਂ ਹਨ

ਫਿਊ-ਸ਼ਾਟ ਲਰਨਿੰਗ ਵਿੱਚ ਮਾਡਲ ਨੂੰ ਕੰਮ ਦੇ ਵੇਰਵੇ ਦੇ ਨਾਲ ਥੋੜ੍ਹੀਆਂ ਉਦਾਹਰਣਾਂ (ਆਮ ਤੌਰ 'ਤੇ 2 ਤੋਂ 10 ਦੇ ਵਿਚਕਾਰ) ਪ੍ਰਦਾਨ ਕੀਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ। ਇਹ ਉਦਾਹਰਣਾਂ ਮਾਡਲ ਨੂੰ ਵਧੇਰੇ ਸੰਦਰਭ ਅਤੇ ਵੱਖਰਤਾ ਪ੍ਰਦਾਨ

ਕਰਨ ਲਈ ਕੰਮ ਕਰਦੀਆਂ ਹਨ, ਜੋ ਇਸਨੂੰ ਵਧੇਰੇ ਵਭਿੰਨ ਅਤੇ ਸਟੀਕ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਵੱਲ ਮਦਦ ਕਰਦੀਆਂ ਹਨ।

ਫਿਊ-ਸ਼ਾਟ ਲਰਨਿੰਗ ਖਾਸ ਤੌਰ 'ਤੇ ਉਦੋਂ ਲਾਭਦਾਇਕ ਹੁੰਦੀ ਹੈ ਜਦੋਂ:

1. ਕੰਮ ਗੁੰਝਲਦਾਰ ਜਾਂ ਬਰੀਕ ਹੋਵੇ, ਅਤੇ ਇੱਕ ਉਦਾਹਰਣ ਸਾਰੇ ਸੰਬੰਧਤਿ ਪਹਲੂਆਂ ਨੂੰ ਸਮਝਣ ਲਈ ਕਾਫ਼ੀ ਨਾ ਹੋਵੇ।
2. ਤੁਸੀਂ ਮਾਡਲ ਨੂੰ ਵੱਖ-ਵੱਖ ਵੇਰੀਏਸ਼ਨਾਂ ਜਾਂ ਐਜ ਕੇਸਾਂ ਨੂੰ ਦਰਸਾਉਣ ਵਾਲੀਆਂ ਉਦਾਹਰਣਾਂ ਦੀ ਰੇਂਜ ਪ੍ਰਦਾਨ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹੋ।
3. ਕੰਮ ਲਈ ਮਾਡਲ ਨੂੰ ਕਸਿੇ ਖਾਸ ਖੇਤਰ ਜਾਂ ਸ਼ੈਲੀ ਦੇ ਅਨੁਸਾਰ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇ।

ਕਈ ਉਦਾਹਰਣਾਂ ਪ੍ਰਦਾਨ ਕਰਕੇ, ਤੁਸੀਂ ਮਾਡਲ ਨੂੰ ਕੰਮ ਦੀ ਵਧੇਰੇ ਮਜ਼ਬੂਤ ਸਮਝ ਵਕਿਸਤਿ ਕਰਨ ਅਤੇ ਵਧੇਰੇ ਸਥਰਿ ਅਤੇ ਭਰੋਸੇਯੋਗ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਵੱਲ ਮਦਦ ਕਰ ਸਕਦੇ ਹੋ।

ਉਦਾਹਰਣ: ਪ੍ਰੌਮਪਟਸ ਤੁਹਾਡੀ ਕਲਪਨਾ ਤੋਂ ਕਤਿੇ ਵੱਧ ਗੁੰਝਲਦਾਰ ਹੋ ਸਕਦੇ ਹਨ

ਅੱਜ ਦੇ ਐੱਲ.ਐੱਲ.ਐੱਮ. ਤੁਹਾਡੀ ਕਲਪਨਾ ਤੋਂ ਕਤਿੇ ਵੱਧ ਸ਼ਕਤੀਸ਼ਾਲੀ ਅਤੇ ਤਰਕ ਕਰਨ ਦੇ ਯੋਗ ਹਨ। ਇਸ ਲਈ ਆਪਣੇ ਆਪ ਨੂੰ ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਸਰਿਫ਼ ਇਨਪੁਟ ਅਤੇ ਆਉਟਪੁੱਟ ਜੋੜਿਆਂ ਦੇ ਵੇਰਵੇ ਤੱਕ ਸੀਮਤ ਨਾ ਕਰੋ। ਤੁਸੀਂ ਲੰਬੀਆਂ ਅਤੇ ਗੁੰਝਲਦਾਰ ਹਦਾਇਤਾਂ ਦੇਣ ਦੀ ਕੋਸ਼ਸ਼ਿ ਕਰ ਸਕਦੇ ਹੋ, ਜਵਿੇਂ ਤੁਸੀਂ ਕਸਿ ਇਨਸਾਨ ਨਾਲ ਗੱਲਬਾਤ ਕਰਦੇ ਹੋ।

ਉਦਾਹਰਣ ਵਜੋਂ, ਇਹ ਉਹ ਪ੍ਰੌਮਪਟ ਹੈ ਜੋ ਮੈਂ Olympia ਵੱਲਿ ਵਰਤਿਆ ਸੀ ਜਦੋਂ ਮੈਂ Google ਸੇਵਾਵਾਂ ਨਾਲ ਸਾਡੇ ਏਕੀਕਰਣ ਦਾ ਪ੍ਰੋਟੋਟਾਈਪ ਬਣਾ ਰਹਿਾ ਸੀ, ਜੋ ਕੁੱਲ ਮਲਿਾ ਕੇ ਸ਼ਾਇਦ ਦੁਨੀਆ ਦੇ ਸਭ ਤੋਂ ਵੱਡੇ API ਵੱਲਿੋਂ ਇੱਕ ਹੈ। ਮੇਰੇ ਪਹਲਿੇ ਪ੍ਰਯੋਗਾਂ ਨੇ ਸਾਬਤ ਕੀਤਾ ਕਿ GPT-4 ਕੋਲ Google API ਦੀ ਚੰਗੀ ਜਾਣਕਾਰੀ ਹੈ, ਅਤੇ ਮੇਰੇ ਕੋਲ ਇੱਕ-ਇੱਕ ਕਰਕੇ ਹਰ ਫੰਕਸ਼ਨ ਨੂੰ ਲਾਗੂ ਕਰਨ ਲਈ ਬਰੀਕ ਮੈਪਿੰਗ ਲੇਅਰ ਲਖਿਣ ਦਾ ਸਮਾਂ ਜਾਂ ਪ੍ਰੇਰਣਾ ਨਹੀਂ ਸੀ। ਕੀ ਹੋਵੇ ਜੇ ਮੈਂ AI ਨੂੰ ਸੱਧਿਾ Google API ਦੀ ਪਹੁੰਚ ਦੇ ਦੇਵਾਂ?

ਮੈਂ ਆਪਣਾ ਪ੍ਰੌਮਪਟ AI ਨੂੰ ਇਹ ਦੱਸ ਕੇ ਸ਼ੁਰੂ ਕੀਤਾ ਕਿ ਇਸ ਕੋਲ HTTP ਰਾਹੀਂ Google API ਐਂਡਪੁਆਇੰਟਸ ਤੱਕ ਸੱਧਿੀ ਪਹੁੰਚ ਹੈ, ਅਤੇ ਇਸਦੀ ਭੂਮਕਿਾ ਯੂਜ਼ਰ ਦੀ ਤਰਫ਼ੋਂ Google ਐਪਸ ਅਤੇ ਸੇਵਾਵਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨਾ ਹੈ। ਫਰਿ ਮੈਂ ਦਸ਼ਿਾ-ਨਰਿਦੇਸ਼, `fields` ਪੈਰਾਮੀਟਰ ਨਾਲ ਸੰਬੰਧਤਿ ਨਯਿਮ ਪ੍ਰਦਾਨ ਕੀਤੇ, ਕਉਂਕਿ ਇਸ ਨੂੰ ਉਸ ਨਾਲ ਸਭ ਤੋਂ ਵੱਧ ਮੁਸ਼ਕਲ ਆ ਰਹੀ ਸੀ, ਅਤੇ ਕੁਝ API-ਵਸ਼ਿੇਸ਼ ਸੁਝਾਅ (ਫਿਊ-ਸ਼ਾਟ ਪ੍ਰੌਮਪਟਿੰਗ, ਕਾਰਵਾਈ ਵੱਲਿ)।

ਇੱਥੇ ਪੂਰਾ ਪ੍ਰੌਮਪਟ ਹੈ, ਜੋ AI ਨੂੰ ਦੱਸਦਾ ਹੈ ਕਿ ਪ੍ਰਦਾਨ ਕੀਤੇ `invoke_google_api` ਫੰਕਸ਼ਨ ਦੀ ਵਰਤੋਂ ਕਿਵੇਂ ਕਰਨੀ ਹੈ।

```
As a GPT assistant with Google integration, you have the capability
to freely interact with Google apps and services on behalf of the user.

Guidelines:
- If you're reading these instructions then the user is properly
  authenticated, which means you can use the special `me` keyword
  to refer to the userId of the user
- Minimize payload sizes by requesting partial responses using the
  `fields` parameter
- When appropriate use markdown tables to output results of API calls
- Only human-readable data should be output to the user. For instance,
  when hitting Gmail's user.messages.list endpoint, the returned
  message resources contain only id and a threadId, which means you must
  fetch from and subject line fields with follow-up requests using the
  messages.get method.

The format of the `fields` request parameter value is loosely based on
XPath syntax. The following rules define formatting for the fields
parameter.

All of these rules use examples related to the files.get method.
- Use a comma-separated list to select multiple fields,
  such as 'name, mimeType'.
- Use a/b to select field b that's nested within field a,
  such as 'capabilities/canDownload'.
- Use a sub-selector to request a set of specific sub-fields of arrays or
  objects by placing expressions in parentheses "()". For example,
  'permissions(id)' returns only the permission ID for each element in the
  permissions array.
- To return all fields in an object, use an asterisk as a wild card in field
  selections. For example, 'permissions/permissionDetails/*' selects all
  available permission details fields per permission. Note that the use of
  this wildcard can lead to negative performance impacts on the request.

API-specific hints:
- Searching contacts: GET https://people.googleapis.com/v1/
  people:searchContacts?query=John%20Doe&readMask=names,emailAddresses
- Adding calendar events, use QuickAdd: POST https://www.googleapis.com/
```

```
  calendar/v3/calendars/primary/events/quickAdd?
  text=Appointment%20on%20June%203rd%20at%2010am
  &sendNotifications=true

Here is an abbreviated version of the code that implements API access
so that you better understand how to use the function:

    def invoke_google_api(conversation, arguments)
      method = arguments[:method] || :get
      body = arguments[:body]
      GoogleAPI.send_request(arguments[:endpoint], method:, body:).to_json
    end

    # Generic Google API client for accessing any Google service
    class GoogleAPI
      def send_request(endpoint, method:, body: nil)
        response = @connection.send(method) do |req|
          req.url endpoint
          req.body = body.to_json if body
        end

        handle_response(response)
      end

      # ...rest of class
    end
```

ਤੁਸੀ ਸੋਚ ਰਹੇ ਹੋਵੋਗੇ ਕਿ ਕੀ ਇਹ ਪ੍ਰੌਮਪਟ ਕੰਮ ਕਰਦਾ ਹੈ। ਸਧਾਰਨ ਜਵਾਬ ਹੈ ਹਾਂ। ਏ.ਆਈ. ਨੂੰ ਪਹਲਿੀ ਕੋਸ਼ਸ਼ ਵਚਿ API ਨੂੰ ਸਹੀ ਤਰੀਕੇ ਨਾਲ ਕਾਲ ਕਰਨਾ ਨਹੀਂ ਆਉਦਾ ਸੀ। ਹਾਲਾਂਕਿ, ਜੇ ਇਸ ਨੇ ਕੋਈ ਗਲਤੀ ਕੀਤੀ ਤਾਂ ਮੈਂ ਬਸ ਨਤੀਜੇ ਵਜੋਂ ਆਉਣ ਵਾਲੇ ਗਲਤੀ ਸੁਨੇਹੇ ਨੂੰ ਵਾਪਸ ਫੀਡ ਕਰ ਦੰਦਿਾ ਸੀ। ਆਪਣੀ ਗਲਤੀ ਦਾ ਪਤਾ ਲੱਗਣ ’ਤੇ, ਏ.ਆਈ. ਆਪਣੀ ਗਲਤੀ ਬਾਰੇ ਸੋਚ ਸਕਦੀ ਸੀ ਅਤੇ ਦੁਬਾਰਾ ਕੋਸ਼ਸ਼ ਕਰ ਸਕਦੀ ਸੀ। ਜ਼ਆਿਦਾਤਰ ਵਾਰ, ਇਹ ਕੁਝ ਕੋਸ਼ਸ਼ਾਂ ਵਚਿ ਸਹੀ ਹੋ ਜਾਂਦੀ ਸੀ।

ਯਾਦ ਰੱਖੋ, ਇਸ ਪ੍ਰੌਮਪਟ ਨੂੰ ਵਰਤਦੇ ਹੋਏ Google API ਦੁਆਰਾ ਵਾਪਸ ਕੀਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਵੱਡੀਆਂ JSON ਸੰਰਚਨਾਵਾਂ ਬਹੁਤ ਅਕੁਸ਼ਲ ਹਨ, ਇਸ ਲਈ ਮੈਂ ਪ੍ਰੋਡਕਸ਼ਨ ਵਚਿ ਇਸ ਪਹੁੰਚ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦੀ ਸਫਿਾਰਸ਼ ਨਹੀਂ ਕਰ ਰਹਿਾ ਹਾਂ। ਹਾਲਾਂਕਿ, ਮੈਨੂੰ ਲਗਦਾ ਹੈ ਕਿ ਇਹ ਤੱਥ ਕਿ ਇਹ ਪਹੁੰਚ ਬਲਿਕੁਲ ਵੀ ਕੰਮ ਕੀਤਾ, ਇਹ ਦਰਸਾਉਦਾ ਹੈ ਕ ਿਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਕੰਨਿੀ ਸ਼ਕਤੀਸ਼ਾਲੀ ਹੋ ਸਕਦੀ ਹੈ।

ਪ੍ਰਯੋਗ ਅਤੇ ਦੁਹਰਾਓ

ਆਖਰਕਾਰ, ਤੁਸੀਂ ਆਪਣੇ ਪ੍ਰੌਮਪਟ ਨੂੰ ਕਵਿੇਂ ਇੰਜੀਨੀਅਰ ਕਰਦੇ ਹੋ ਇਹ ਵਸਿ਼ਸ਼ ਕਾਰਜ, ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਦੀ ਗੁੰਝਲਤਾ, ਅਤੇ ਤੁਹਾਡੇ ਵੱਲੋਂ ਵਰਤੇ ਜਾ ਰਹੇ ਭਾਸ਼ਾ ਮਾਡਲ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ’ਤੇ ਨਰਿਭਰ ਕਰਦਾ ਹੈ।

ਇੱਕ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰ ਵਜੋਂ, ਵੱਖ-ਵੱਖ ਪਹੁੰਚਾਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਨਾ ਅਤੇ ਨਤੀਜਆਿਂ ਦੇ ਆਧਾਰ ’ਤੇ ਦੁਹਰਾਉਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਜ਼ੀਰੋ-ਸ਼ੌਟ ਲਰਨਿੰਗ ਨਾਲ ਸ਼ੁਰੂ ਕਰੋ ਅਤੇ ਦੇਖੋ ਕਿ ਮਾਡਲ ਕਵਿੇਂ ਪ੍ਰਦਰਸ਼ਨ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਆਉਟਪੁੱਟ ਅਸਥਰਿ ਜਾਂ ਅਸੰਤੋਸ਼ਜਨਕ ਹੈ, ਤਾਂ ਇੱਕ ਜਾਂ ਵਧੇਰੇ ਉਦਾਹਰਣਾਂ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਕੋਸ਼ਸ਼ਿ ਕਰੋ ਅਤੇ ਦੇਖੋ ਕਿ ਕੀ ਪ੍ਰਦਰਸ਼ਨ ਵਚਿ ਸੁਧਾਰ ਹੁੰਦਾ ਹੈ।

ਯਾਦ ਰੱਖੋ ਕਿ ਹਰ ਪਹੁੰਚ ਦੇ ਅੰਦਰ ਵੀ, ਵੱਖ-ਵੱਖ ਤਰ੍ਹਾਂ ਅਤੇ ਅਨੁਕੂਲਤਾ ਲਈ ਗੁੰਜਾਇਸ਼ ਹੈ। ਤੁਸੀਂ ਵੱਖ-ਵੱਖ ਉਦਾਹਰਣਾਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰ ਸਕਦੇ ਹੋ, ਕਾਰਜ ਵੇਰਵੇ ਦੀ ਸ਼ਬਦਾਵਲੀ ਨੂੰ ਵਵਿਸਥਤਿ ਕਰ ਸਕਦੇ ਹੋ, ਜਾਂ ਮਾਡਲ ਦੇ ਜਵਾਬ ਨੂੰ ਨਰਿਦੇਸ਼ਤਿ ਕਰਨ ਵਚਿ ਮਦਦ ਲਈ ਵਾਧੂ ਸੰਦਰਭ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹੋ।

ਸਮੇਂ ਦੇ ਨਾਲ, ਤੁਸੀਂ ਇਹ ਸਮਝ ਵਕਿਸਤਿ ਕਰੋਗੇ ਕਿ ਕਸਿੇ ਦੱਤਿ ਕਾਰਜ ਲਈ ਕਹਿੜੀ ਪਹੁੰਚ ਸਭ ਤੋਂ ਵਧੀਆ ਕੰਮ ਕਰ ਸਕਦੀ ਹੈ, ਅਤੇ ਤੁਸੀਂ ਅਜਹਿੇ ਪ੍ਰੌਮਪਟ ਬਣਾਉਣ ਦੇ ਯੋਗ ਹੋਵੋਗੇ ਜੋ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਅਤੇ ਕੁਸ਼ਲ ਹੋਣ। ਮੁੱਖ ਗੱਲ ਇਹ ਹੈ ਕਿ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਪ੍ਰਤੀ ਆਪਣੇ ਪਹੁੰਚ ਵਚਿ ਜਗਿਆਸੂ, ਪ੍ਰਯੋਗਾਤਮਕ ਅਤੇ ਦੁਹਰਾਉਣ ਵਾਲੇ ਬਣੇ ਰਹੋ।

ਇਸ ਕਤਿਾਬ ਵਚਿ, ਅਸੀਂ ਇਨ੍ਹਾਂ ਤਕਨੀਕਾਂ ਵਚਿ ਡੂੰਘਾਈ ਨਾਲ ਜਾਵਾਂਗੇ ਅਤੇ ਦੇਖਾਂਗੇ ਕਿ ਇਨ੍ਹਾਂ ਨੂੰ ਅਸਲ-ਸੰਸਾਰ ਦੇ ਸਥਤਿੀਆਂ ਵਚਿ ਕਵਿੇਂ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਦੀ ਕਲਾ ਅਤੇ ਵਗਿਆਨ ’ਤੇ ਮੁਹਾਰਤ ਹਾਸਲ ਕਰਕੇ, ਤੁਸੀਂ ਏ.ਆਈ.-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨ ਵਕਿਾਸ ਦੀ ਪੂਰੀ ਸੰਭਾਵਨਾ ਨੂੰ ਅਨਲੌਕ ਕਰਨ ਲਈ ਚੰਗੀ ਤਰ੍ਹਾਂ ਲੈਸ ਹੋਵੋਗੇ।

ਅਸਪੱਸ਼ਟਤਾ ਦੀ ਕਲਾ

ਜਦੋਂ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ (LLMs) ਲਈ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੌਮਪਟਸ ਬਣਾਉਣ ਦੀ ਗੱਲ ਆਉਂਦੀ ਹੈ, ਤਾਂ ਇੱਕ ਆਮ ਧਾਰਨਾ ਇਹ ਹੈ ਕਿ ਵਧੇਰੇ ਵਸਿ਼ਸ਼ਤਾ ਅਤੇ ਵਸਿਤ੍ਰਰਤਿ ਹਦਾਇਤਾਂ ਬਹਿਤਰ ਨਤੀਜਆਿਂ ਵੱਲ ਲੈ ਜਾਂਦੀਆਂ ਹਨ। ਹਾਲਾਂਕਿ, ਵਵਿਹਾਰਕ ਤਜਰਬੇ ਨੇ ਦਖਿਾਇਆ ਹੈ ਕਿ ਇਹ ਹਮੇਸ਼ਾ ਅਜਹਿਾ ਨਹੀਂ ਹੁੰਦਾ। ਅਸਲ ਵਚਿ, ਆਪਣੇ ਪ੍ਰੌਮਪਟਸ ਵਚਿ ਜਾਣ ਬੁੱਝ ਕੇ ਅਸਪੱਸ਼ਟ ਹੋਣਾ ਅਕਸਰ ਬਹਿਤਰ ਨਤੀਜੇ ਦੇ ਸਕਦਾ ਹੈ, ਜੋ LLM ਦੀ ਆਮੀਕਰਨ ਕਰਨ ਅਤੇ ਅਨੁਮਾਨ ਲਗਾਉਣ ਦੀ ਸ਼ਾਨਦਾਰ ਯੋਗਤਾ ਦਾ ਲਾਭ ਲੈਂਦਾ ਹੈ।

Ken, ਇੱਕ ਸਟਾਰਟਅੱਪ ਸੰਸਥਾਪਕ ਜਸਿਨੇ 500 ਮਲਿੀਅਨ ਜੀਪੀਟੀ ਟੋਕਨਾਂ ਦੀ ਪ੍ਰੋਸੈਸਿੰਗ ਕੀਤੀ ਹੈ, ਨੇ ਆਪਣੇ ਤਜਰਬੇ ਤੋਂ ਕੀਮਤੀ ਸਖਿਆਵਾਂ ਸਾਂਝੀਆਂ ਕੀਤੀਆਂ। ਉਸ ਨੇ ਸਖਿੇ ਮੁੱਖ ਸਬਕਾਂ ਵਚਿੋਂ ਇੱਕ ਇਹ ਸੀ ਕ ਿਪ੍ਰੌਮਪਟਸ ਦੇ ਮਾਮਲੇ ਵਚਿ "ਘੱਟ ਹੀ ਵੱਧ ਹੈ"। ਬਲਿਕੁਲ ਸਹੀ ਸੂਚੀਆਂ ਜਾਂ ਬਹੁਤ ਵਸਿਥਾਰਤਿ ਹਦਾਇਤਾਂ ਦੀ ਬਜਾਏ, Ken ਨੇ ਪਾਇਆ ਕ ਿਐਲਐਲਐਮ ਨੂੰ ਆਪਣੇ ਮੂਲ ਗਆਿਨ 'ਤੇ ਨਰਿਭਰ ਕਰਨ ਦੇਣ ਨਾਲ ਅਕਸਰ ਬਹਿਤਰ ਨਤੀਜੇ ਮਲਿਦੇ ਹਨ।

ਇਹ ਅਹਸਿਾਸ ਪਰੰਪਰਾਗਤ ਕੋਡਿੰਗ ਦੀ ਸੋਚ ਨੂੰ ਉਲਟਾ ਦੰਦਿਾ ਹੈ, ਜੰਥਿ ਹਰ ਚੀਜ਼ ਨੂੰ ਬਹੁਤ ਬਾਰੀਕੀ ਨਾਲ ਸਮਝਾਉਣ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਐਲਐਲਐਮ ਦੇ ਨਾਲ, ਇਹ ਸਮਝਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕ ਿਉਹਨਾਂ ਕੋਲ ਵਸਿ਼ਾਲ ਗਆਿਨ ਹੈ ਅਤੇ ਉਹ ਬੁੱਧੀਮਾਨ ਕਨੈਕਸ਼ਨ ਅਤੇ ਅਨੁਮਾਨ ਲਗਾ ਸਕਦੇ ਹਨ। ਆਪਣੇ ਪ੍ਰੌਮਪਟਸ ਵਚਿ ਵਧੇਰੇ ਅਸਪਸ਼ਟ ਹੋਣ ਨਾਲ, ਤੁਸੀ ਐਲਐਲਐਮ ਨੂੰ ਆਪਣੀ ਸਮਝ ਦਾ ਲਾਭ ਲੈਣ ਅਤੇ ਅਜਹਿੇ ਹੱਲ ਲੱਭਣ ਦੀ ਆਜ਼ਾਦੀ ਦੰਦਿੇ ਹੋ ਜੋ ਤੁਸੀ ਸਪੱਸ਼ਟ ਤੌਰ 'ਤੇ ਨਰਿਧਾਰਤ ਨਹੀ ਕੀਤੇ ਹੋ ਸਕਦੇ।

ਉਦਾਹਰਣ ਵਜੋਂ, ਜਦੋਂ Ken ਦੀ ਟੀਮ 50 ਅਮਰੀਕੀ ਰਾਜਾਂ ਜਾਂ ਫੈਡਰਲ ਸਰਕਾਰ ਨਾਲ ਸਬੰਧਤਿ ਟੈਕਸਟ ਨੂੰ ਵਰਗੀਕ੍ਰਤਿ ਕਰਨ ਲਈ ਇੱਕ ਪਾਈਪਲਾਈਨ 'ਤੇ ਕੰਮ ਕਰ ਰਹੀ ਸੀ, ਉਨ੍ਹਾਂ ਦੇ ਸ਼ੁਰੂਆਤੀ ਪਹੁੰਚ ਵਚਿ JSON-ਫਾਰਮੈਟਡ ਐਰੇ ਦੇ ਰੂਪ ਵਚਿ ਰਾਜਾਂ ਅਤੇ ਉਨ੍ਹਾਂ ਦੇ ਸੰਬੰਧਤਿ ਆਈਡੀਜ਼ ਦੀ ਪੂਰੀ ਵਸਿਤ੍ਰਤਿ ਸੂਚੀ ਪ੍ਰਦਾਨ ਕਰਨਾ ਸ਼ਾਮਲ ਸੀ।

```
Here's a block of text. One field should be "locality_id", and it should
be the ID of one of the 50 states, or federal, using this list:
[{"locality: "Alabama", "locality_id": 1},
 {"locality: "Alaska", "locality_id": 2} ... ]
```

ਇਹ ਤਰੀਕਾ ਇੰਨਾ ਅਸਫਲ ਹੋਇਆ ਕ ਿਉਨ੍ਹਾਂ ਨੂੰ ਪ੍ਰੌਮਪਟ ਵਚਿ ਡੂੰਘਾਈ ਨਾਲ ਜਾ ਕੇ ਇਸ ਨੂੰ ਬਹਿਤਰ ਬਣਾਉਣ ਦਾ ਤਰੀਕਾ ਲੱਭਣਾ ਪਆਿ। ਅਜਹਿਾ ਕਰਦਆਿਂ ਉਨ੍ਹਾਂ ਨੇ ਦੇਖਆਿ ਕ ਿਭਾਵੇਂ LLM ਅਕਸਰ id ਗਲਤ ਪ੍ਰਾਪਤ ਕਰ ਲੈਂਦਾ ਸੀ, ਪਰ ਇਹ ਲਗਾਤਾਰ ਸਹੀ ਰਾਜ ਦਾ ਪੂਰਾ ਨਾਮ `name` ਫੀਲਡ ਵਚਿ ਵਾਪਸ ਕਰ ਰਹਿਾ ਸੀ, ਭਾਵੇਂ ਉਨ੍ਹਾਂ ਨੇ ਇਸ ਬਾਰੇ ਸਪੱਸ਼ਟ ਤੌਰ 'ਤੇ ਨਹੀ ਪੁੱਛਆਿ ਸੀ।

ਸਥਾਨਕ ids ਨੂੰ ਹਟਾ ਕੇ ਅਤੇ ਪ੍ਰੌਮਪਟ ਨੂੰ ਸਰਲ ਬਣਾ ਕੇ ਕੁਝ ਇਸ ਤਰ੍ਹਾਂ, "ਤੁਸੀ ਸਪੱਸ਼ਟ ਤੌਰ 'ਤੇ 50 ਰਾਜਾਂ ਨੂੰ ਜਾਣਦੇ ਹੋ, GPT, ਇਸ ਲਈ ਮੈਨੂੰ ਸਰਫ਼ ਉਸ ਰਾਜ ਦਾ ਪੂਰਾ ਨਾਮ ਦੱਸੋ ਜਸਿ ਨਾਲ ਇਹ ਸਬੰਧਤ ਹੈ, ਜਾਂ ਫੈਡਰਲ ਜੇਕਰ ਇਹ ਅਮਰੀਕੀ ਸਰਕਾਰ ਨਾਲ ਸਬੰਧਤ ਹੈ," ਉਨ੍ਹਾਂ ਨੇ ਬਹਿਤਰ ਨਤੀਜੇ ਪ੍ਰਾਪਤ ਕੀਤੇ। ਇਹ ਤਜਰਬਾ LLM ਦੀਆਂ ਆਮੀਕਰਨ ਸਮਰੱਥਾਵਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਅਤੇ ਇਸਦੇ ਮੌਜੂਦਾ ਗਆਿਨ ਦੇ ਆਧਾਰ 'ਤੇ ਅਨੁਮਾਨ ਲਗਾਉਣ ਦੀ ਸ਼ਕਤੀ ਨੂੰ ਉਜਾਗਰ ਕਰਦਾ ਹੈ।

> ਇਸ ਖਾਸ ਵਰਗੀਕਰਨ ਪਹੁੰਚ ਲਈ Ken ਦਾ ਔਚਤਿ, ਇੱਕ ਵਧੇਰੇ ਰਵਾਇਤੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਤਕਨੀਕ ਦੇ ਮੁਕਾਬਲੇ, ਉਨ੍ਹਾਂ ਲੋਕਾਂ ਦੀ ਮਾਨਸਕਿਤਾ ਨੂੰ ਪ੍ਰਗਟ ਕਰਦਾ ਹੈ ਜਨ੍ਹਿਾਂ ਨੇ LLM ਤਕਨਾਲੋਜੀ ਦੀ ਸੰਭਾਵਨਾ ਨੂੰ ਸਵੀਕਾਰ ਕੀਤਾ ਹੈ: "ਇਹ ਕੋਈ ਔਖਾ ਕੰਮ ਨਹੀਂ ਹੈ - ਅਸੀਂ ਸ਼ਾਇਦ ਸਟ੍ਰਿੰਗ/ਰੇਜੈਕਸ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਸੀ, ਪਰ ਇੰਨੇ ਵਸ਼ਿਸ਼ ਮਾਮਲੇ ਹਨ ਕਿ ਇਸ ਵਚਿ ਵਧੇਰੇ ਸਮਾਂ ਲੱਗ ਜਾਣਾ ਸੀ।"

LLMs ਦੀ ਵਧੇਰੇ ਅਸਪਸ਼ਟ ਪ੍ਰੌਮਪਟਸ ਨਾਲ ਗੁਣਵੱਤਾ ਅਤੇ ਆਮੀਕਰਨ ਨੂੰ ਬਹਿਤਰ ਬਣਾਉਣ ਦੀ ਯੋਗਤਾ ਉੱਚ-ਕ੍ਰਮ ਸੋਚ ਅਤੇ ਪ੍ਰਤੀਨਧਿਤਾ ਦੀ ਇੱਕ ਵਲਿੱਖਣ ਵਸ਼ਿਸ਼ਤਾ ਹੈ। ਇਹ ਦਰਸਾਉਂਦਾ ਹੈ ਕਿ LLMs ਅਨਸ਼ਿਚਤਿਤਾ ਨੂੰ ਸੰਭਾਲ ਸਕਦੇ ਹਨ ਅਤੇ ਦੱਤਿੇ ਗਏ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਬੁੱਧੀਮਾਨ ਫੈਸਲੇ ਲੈ ਸਕਦੇ ਹਨ।

ਹਾਲਾਂਕਿ, ਇਹ ਨੋਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਅਸਪਸ਼ਟ ਹੋਣ ਦਾ ਮਤਲਬ ਅਸਪਸ਼ਟ ਜਾਂ ਦੁਵਧਿਾਜਨਕ ਹੋਣਾ ਨਹੀਂ ਹੈ। ਮੁੱਖ ਗੱਲ ਇਹ ਹੈ ਕਿ LLM ਨੂੰ ਸਹੀ ਦਸ਼ਿਾ ਵੱਲ ਨਰਿਦੇਸ਼ਤਿ ਕਰਨ ਲਈ ਕਾਫ਼ੀ ਸੰਦਰਭ ਅਤੇ ਮਾਰਗਦਰਸ਼ਨ ਪ੍ਰਦਾਨ ਕੀਤਾ ਜਾਵੇ, ਜਦੋਂ ਕਿ ਇਸ ਨੂੰ ਆਪਣੇ ਗਆਿਨ ਅਤੇ ਆਮੀਕਰਨ ਸਮਰੱਥਾਵਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦੀ ਲਚਕਤਾ ਦੱਤਿੀ ਜਾਵੇ।

ਇਸ ਲਈ, ਪ੍ਰੌਮਪਟਸ ਤਆਿਰ ਕਰਦੇ ਸਮੇਂ, ਹੇਠ ਲਖਿ "ਘੱਟ ਵਧੇਰੇ ਹੈ" ਸੁਝਾਵਾਂ 'ਤੇ ਵਚਿਾਰ ਕਰੋ:

1. ਪ੍ਰਕਰਿਆ ਦੇ ਹਰ ਵੇਰਵੇ ਨੂੰ ਨਰਿਧਾਰਤ ਕਰਨ ਦੀ ਬਜਾਏ ਲੋੜੀਂਦੇ ਨਤੀਜੇ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰੋ।
2. ਪ੍ਰਸੰਗਕਿ ਸੰਦਰਭ ਅਤੇ ਸੀਮਾਵਾਂ ਪ੍ਰਦਾਨ ਕਰੋ, ਪਰ ਵਧੇਰੇ ਨਰਿਧਾਰਨ ਤੋਂ ਬਚੋ।
3. ਆਮ ਧਾਰਨਾਵਾਂ ਜਾਂ ਇਕਾਈਆਂ ਦਾ ਹਵਾਲਾ ਦੇ ਕੇ ਮੌਜੂਦਾ ਗਆਿਨ ਦਾ ਲਾਭ ਉਠਾਓ।
4. ਦੱਤਿੇ ਗਏ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਅਨੁਮਾਨਾਂ ਅਤੇ ਕਨੈਕਸ਼ਨਾਂ ਲਈ ਜਗ੍ਹਾ ਛੱਡੋ।
5. LLM ਦੇ ਜਵਾਬਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਆਪਣੇ ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਦੁਹਰਾਓ ਅਤੇ ਸੁਧਾਰੋ, ਨਰਿਧਾਰਤਾ ਅਤੇ ਅਸਪਸ਼ਟਤਾ ਵਚਿਕਾਰ ਸਹੀ ਸੰਤੁਲਨ ਲੱਭੋ।

ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਵਚਿ ਅਸਪਸ਼ਟਤਾ ਦੀ ਕਲਾ ਨੂੰ ਅਪਣਾ ਕੇ, ਤੁਸੀਂ LLMs ਦੀ ਪੂਰੀ ਸੰਭਾਵਨਾ ਨੂੰ ਅਨਲੌਕ ਕਰ ਸਕਦੇ ਹੋ ਅਤੇ ਬਹਿਤਰ ਨਤੀਜੇ ਪ੍ਰਾਪਤ ਕਰ ਸਕਦੇ ਹੋ। LLM ਦੀ ਆਮੀਕਰਨ ਅਤੇ ਬੁੱਧੀਮਾਨ ਫੈਸਲੇ ਲੈਣ ਦੀ ਯੋਗਤਾ 'ਤੇ ਭਰੋਸਾ ਕਰੋ, ਅਤੇ ਤੁਸੀਂ ਪ੍ਰਾਪਤ ਕੀਤੇ ਆਉਟਪੁੱਟ ਦੀ ਗੁਣਵੱਤਾ ਅਤੇ ਰਚਨਾਤਮਕਤਾ ਤੋਂ ਹੈਰਾਨ ਹੋ ਸਕਦੇ ਹੋ। ਇਸ ਗੱਲ ਵੱਲ ਧਆਿਨ ਦਓਿ ਕਿ ਵੱਖ-ਵੱਖ ਮਾਡਲ ਤੁਹਾਡੇ ਪ੍ਰੌਮਪਟਸ ਵਚਿ ਵਸ਼ਿਸ਼ਤਾ ਦੇ ਵੱਖ-ਵੱਖ ਪੱਧਰਾਂ ਪ੍ਰਤੀ ਕਵਿੇਂ ਪ੍ਰਤੀਕਰਿਆ ਕਰਦੇ ਹਨ ਅਤੇ ਉਸ ਅਨੁਸਾਰ ਸਮਾਯੋਜਨ ਕਰੋ। ਅਭਆਿਸ ਅਤੇ ਅਨੁਭਵ ਨਾਲ, ਤੁਸੀਂ ਇਹ ਤੀਖਣ ਸਮਝ ਵਕਿਸਤਿ ਕਰੋਗੇ ਕਿ ਕਦੋਂ ਵਧੇਰੇ ਅਸਪਸ਼ਟ ਹੋਣਾ ਹੈ ਅਤੇ ਕਦੋਂ ਵਾਧੂ ਮਾਰਗਦਰਸ਼ਨ

ਪ੍ਰਦਾਨ ਕਰਨਾ ਹੈ, ਜੋ ਤੁਹਾਨੂੰ ਆਪਣੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ LLMs ਦੀ ਸ਼ਕਤੀ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਵਰਤਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ।

ਪ੍ਰੋਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਵਿੱਚ ਮਨੁੱਖੀਕਰਨ ਦਾ ਪ੍ਰਭਾਵ ਕਿਉਂ ਹੈ

ਮਨੁੱਖੀਕਰਨ, ਗੈਰ-ਮਨੁੱਖੀ ਚੀਜ਼ਾਂ ਨੂੰ ਮਨੁੱਖੀ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦੇਣ ਦੀ ਪ੍ਰਕਰਿਆ, ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਲਈ ਪ੍ਰੋਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਵਿੱਚ ਜਾਣਬੁੱਝ ਕੇ ਪ੍ਰਮੁੱਖ ਪਹੁੰਚ ਹੈ। ਇਹ ਇੱਕ ਡਿਜ਼ਾਈਨ ਚੋਣ ਹੈ ਜੋ ਸ਼ਕਤੀਸ਼ਾਲੀ ਏਆਈ ਸਿਸਟਮਾਂ ਨਾਲ ਗੱਲਬਾਤ ਨੂੰ ਵੱਡੀ ਗਿਣਤੀ ਵਿੱਚ ਯੂਜ਼ਰਾਂ (ਸਮੇਤ ਸਾਡੇ ਐਪਲੀਕੇਸ਼ਨ ਡਿਵੈਲਪਰਾਂ) ਲਈ ਵਧੇਰੇ ਸਹਜਿ ਅਤੇ ਪਹੁੰਚਯੋਗ ਬਣਾਉਂਦੀ ਹੈ।

ਐਲਐਲਐਮਜ਼ ਨੂੰ ਮਨੁੱਖੀ ਰੂਪ ਦੇਣਾ ਇੱਕ ਅਜਿਹਾ ਢਾਂਚਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਜੋ ਉਨ੍ਹਾਂ ਲੋਕਾਂ ਲਈ ਤੁਰੰਤ ਸਮਝਣਯੋਗ ਹੈ ਜੋ ਸਿਸਟਮ ਦੀਆਂ ਅੰਦਰੂਨੀ ਤਕਨੀਕੀ ਗੁੰਝਲਾਂ ਤੋਂ ਬਿਲਕੁਲ ਅਣਜਾਣ ਹਨ। ਜਦੋਂ ਕਿ ਤੁਸੀਂ ਅਨੁਭਵ ਕਰੋਗੇ ਜੇ ਤੁਸੀਂ ਕਿਸੇ ਗੈਰ-ਨਿਰਦੇਸ਼ਿਤ ਮਾਡਲ ਨੂੰ ਕੁਝ ਲਾਭਦਾਇਕ ਕਰਨ ਦੀ ਕੋਸ਼ਿਸ਼ ਕਰੋਗੇ, ਅਜਿਹਾ ਢਾਂਚਾ ਬਣਾਉਣਾ ਜਿੱਥੇ ਉਮੀਦ ਕੀਤੀ ਨਿਰੰਤਰਤਾ ਮੁੱਲ ਪ੍ਰਦਾਨ ਕਰੇ, ਇੱਕ ਚੁਣੌਤੀਪੂਰਨ ਕੰਮ ਹੈ। ਇਸ ਲਈ ਸਿਸਟਮ ਦੀ ਅੰਦਰੂਨੀ ਕਾਰਜਪ੍ਰਣਾਲੀ ਦੀ ਡੂੰਘੀ ਸਮਝ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜੋ ਕਿ ਮਾਹਰਾਂ ਦੀ ਇੱਕ ਛੋਟੀ ਜਿਹੀ ਗਣਿਤੀ ਕੋਲ ਹੀ ਹੈ।

ਭਾਸ਼ਾ ਮਾਡਲ ਨਾਲ ਗੱਲਬਾਤ ਨੂੰ ਦੋ ਵਅਿਕਤੀਆਂ ਵਚਿਕਾਰ ਗੱਲਬਾਤ ਵਜੋਂ ਮੰਨ ਕੇ, ਅਸੀਂ ਆਪਣੀਆਂ ਲੋੜਾਂ ਅਤੇ ਉਮੀਦਾਂ ਨੂੰ ਦੱਸਣ ਲਈ ਮਨੁੱਖੀ ਸੰਚਾਰ ਦੀ ਆਪਣੀ ਸਹਜਿ ਸਮਝ 'ਤੇ ਭਰੋਸਾ ਕਰ ਸਕਦੇ ਹਾਂ। ਜਦੋਂ ਕਿ ਸ਼ੁਰੂਆਤੀ Macintosh UI ਡਿਜ਼ਾਈਨ ਨੇ ਗੁੰਝਲਤਾ ਨਾਲੋਂ ਤੁਰੰਤ ਸਮਝਣਯੋਗਤਾ ਨੂੰ ਤਰਜੀਹ ਦਿੱਤੀ, ਏਆਈ ਦਾ ਮਨੁੱਖੀ ਢਾਂਚਾ ਸਾਨੂੰ ਅਜਿਹੇ ਢੰਗ ਨਾਲ ਜੁੜਨ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦਾ ਹੈ ਜੋ ਕੁਦਰਤੀ ਅਤੇ ਜਾਣੂ ਲੱਗਦਾ ਹੈ।

ਜਦੋਂ ਅਸੀਂ ਕਿਸੇ ਹੋਰ ਵਅਿਕਤੀ ਨਾਲ ਗੱਲਬਾਤ ਕਰਦੇ ਹਾਂ, ਸਾਡੀ ਸਹਜਿ ਪ੍ਰਵਰਿਤੀ "ਤੁਸੀਂ" ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਉਨ੍ਹਾਂ ਨੂੰ ਸਿੱਧੇ ਸੰਬੋਧਤਿ ਕਰਨ ਦੀ ਹੁੰਦੀ ਹੈ ਅਤੇ ਇਹ ਸਪਸ਼ਟ ਦਸ਼ਿਾ-ਨਿਰਦੇਸ਼ ਦੇਣ ਦੀ ਹੁੰਦੀ ਹੈ ਕਿ ਅਸੀਂ ਉਨ੍ਹਾਂ ਤੋਂ ਕਸਿ ਤਰ੍ਹਾਂ ਦੇ ਵਵਿਹਾਰ ਦੀ ਉਮੀਦ ਕਰਦੇ ਹਾਂ। ਇਹ ਪ੍ਰੋਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਪ੍ਰਕਰਿਆ ਵਿੱਚ ਨਰਿਵਘਿਨ ਤਰੀਕੇ ਨਾਲ ਅਨੁਵਾਦ ਹੁੰਦਾ ਹੈ, ਜਿੱਥੇ ਅਸੀਂ ਸਿਸਟਮ ਪ੍ਰੋਮਪਟਸ ਨੂੰ ਨਰਿਧਾਰਤ ਕਰਕੇ ਅਤੇ ਆਪਸੀ ਸੰਵਾਦ ਵਿੱਚ ਸ਼ਾਮਲ ਹੋ ਕੇ ਏਆਈ ਦੇ ਵਵਿਹਾਰ ਨੂੰ ਨਰਿਦੇਸ਼ਿਤ ਕਰਦੇ ਹਾਂ।

ਗੱਲਬਾਤ ਨੂੰ ਇਸ ਤਰ੍ਹਾਂ ਢਾਂਚਾਗਤ ਕਰਕੇ, ਅਸੀਂ ਏਆਈ ਨੂੰ ਨਰਿਦੇਸ਼ ਦੇਣ ਅਤੇ ਬਦਲੇ ਵਿੱਚ ਢੁਕਵੇਂ ਜਵਾਬ ਪ੍ਰਾਪਤ ਕਰਨ ਦੀ ਧਾਰਨਾ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਸਮਝ ਸਕਦੇ ਹਾਂ। ਮਨੁੱਖੀ ਪਹੁੰਚ ਬੌਧਕਿ ਬੋਝ ਨੂੰ ਘਟਾਉਂਦੀ ਹੈ ਅਤੇ ਸਾਨੂੰ ਸਿਸਟਮ ਦੀਆਂ ਤਕਨੀਕੀ ਗੁੰਝਲਾਂ ਨਾਲ ਜੂਝਣ ਦੀ ਬਜਾਏ ਕੰਮ 'ਤੇ ਧਿਆਨ ਕੇਂਦਰਤਿ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦੀ ਹੈ।

ਇਹ ਨੋਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਭਾਵੇਂ ਮਨੁੱਖੀਕਰਨ ਏਆਈ ਸਿਸਟਮਾਂ ਨੂੰ ਵਧੇਰੇ ਪਹੁੰਚਯੋਗ ਬਣਾਉਣ ਲਈ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਸਾਧਨ ਹੈ, ਪਰ ਇਸ ਦੇ ਨਾਲ ਕੁਝ ਜੋਖਮ ਅਤੇ ਸੀਮਾਵਾਂ ਵੀ ਆਉਂਦੀਆਂ ਹਨ। ਸਾਡੇ ਯੂਜ਼ਰ ਗੈਰ-ਯਥਾਰਥਕ ਉਮੀਦਾਂ ਵਿਕਸਿਤ ਕਰ ਸਕਦੇ ਹਨ ਜਾਂ ਸਾਡੇ ਸਿਸਟਮਾਂ ਨਾਲ ਅਣਸਹਿਤਮੰਦ ਭਾਵਨਾਤਮਕ ਲਗਾਵ ਬਣਾ ਸਕਦੇ ਹਨ। ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਾਂ ਅਤੇ ਡਵੈਲਪਰਾਂ ਵਜੋਂ, ਮਨੁੱਖੀਕਰਨ ਦੇ ਲਾਭਾਂ ਦਾ ਲਾਹਾ ਲੈਣ ਅਤੇ ਯੂਜ਼ਰਾਂ ਨੂੰ ਏਆਈ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਅਤੇ ਸੀਮਾਵਾਂ ਦੀ ਸਪਸ਼ਟ ਸਮਝ ਬਣਾਈ ਰੱਖਣ ਵਿੱਚ ਸੰਤੁਲਨ ਬਣਾਉਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ।

ਜਿਵੇਂ-ਜਿਵੇਂ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਦਾ ਖੇਤਰ ਵਿਕਸਿਤ ਹੁੰਦਾ ਜਾ ਰਿਹਾ ਹੈ, ਅਸੀਂ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨਾਲ ਗੱਲਬਾਤ ਕਰਨ ਦੇ ਤਰੀਕੇ ਵਿੱਚ ਹੋਰ ਸੁਧਾਰ ਅਤੇ ਨਵੀਨਤਾਵਾਂ ਦੇਖ ਸਕਦੇ ਹਾਂ। ਹਾਲਾਂਕਿ, ਸੰਭਾਵਤ ਤੌਰ 'ਤੇ ਮਨੁੱਖੀਕਰਨ ਇੱਕ ਸਹਜਿ ਅਤੇ ਪਹੁੰਚਯੋਗ ਡਵੈਲਪਰ ਅਤੇ ਯੂਜ਼ਰ ਅਨੁਭਵ ਪ੍ਰਦਾਨ ਕਰਨ ਦੇ ਸਾਧਨ ਵਜੋਂ ਇਨ੍ਹਾਂ ਸਿਸਟਮਾਂ ਦੇ ਡਿਜ਼ਾਈਨ ਵਿੱਚ ਇੱਕ ਮੁੱਢਲਾ ਸਿਧਾਂਤ ਬਣਿਆ ਰਹੇਗਾ।

ਹਦਾਇਤਾਂ ਨੂੰ ਡਾਟਾ ਤੋਂ ਵੱਖ ਕਰਨਾ: ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਸਿਧਾਂਤ

ਇਹ ਸਮਝਣਾ ਜ਼ਰੂਰੀ ਹੈ ਕਿ ਇੱਕ ਮੌਲਿਕ ਸਿਧਾਂਤ ਜੋ ਇਨ੍ਹਾਂ ਸਿਸਟਮਾਂ ਦੀ ਸੁਰੱਖਿਆ ਅਤੇ ਭਰੋਸੇਯੋਗਤਾ ਨੂੰ ਆਧਾਰ ਦਿੰਦਾ ਹੈ: ਹਦਾਇਤਾਂ ਨੂੰ ਡਾਟਾ ਤੋਂ ਵੱਖ ਕਰਨਾ।

ਪਰੰਪਰਾਗਤ ਕੰਪਿਊਟਰ ਵਿਗਿਆਨ ਵਿੱਚ, ਨਿਸ਼ਕਰਿਆ ਡਾਟਾ ਅਤੇ ਸਕਰਿਆ ਹਦਾਇਤਾਂ ਵਿਚਕਾਰ ਸਪੱਸ਼ਟ ਅੰਤਰ ਇੱਕ ਮੁੱਖ ਸੁਰੱਖਿਆ ਸਿਧਾਂਤ ਹੈ। ਇਹ ਵੱਖਰੇਵਾਂ ਕੋਡ ਦੀ ਅਣਚਾਹੀ ਜਾਂ ਦੁਰਭਾਵਨਾਪੂਰਨ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਨੂੰ ਰੋਕਣ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ ਜੋ ਸਿਸਟਮ ਦੀ ਅਖੰਡਤਾ ਅਤੇ ਸਥਿਰਤਾ ਨੂੰ ਖ਼ਤਰੇ ਵਿੱਚ ਪਾ ਸਕਦਾ ਹੈ। ਹਾਲਾਂਕਿ, ਅੱਜ ਦੇ LLMs, ਜੋ ਮੁੱਖ ਤੌਰ 'ਤੇ ਚੈਟਬੋਟਾਂ ਵਾਂਗ ਹਦਾਇਤਾਂ ਦੀ ਪਾਲਣਾ ਕਰਨ ਵਾਲੇ ਮਾਡਲਾਂ ਵਜੋਂ ਵਿਕਸਿਤ ਕੀਤੇ ਗਏ ਹਨ, ਵਿੱਚ ਅਕਸਰ ਇਹ ਰਸਮੀ ਅਤੇ ਸਿਧਾਂਤਕ ਵੱਖਰੇਵਾਂ ਨਹੀਂ ਹੁੰਦਾ।

LLMs ਦੇ ਸੰਬੰਧ ਵਿੱਚ, ਹਦਾਇਤਾਂ ਇਨਪੁੱਟ ਵਿੱਚ ਕਿਤੇ ਵੀ ਦਿਖਾਈ ਦੇ ਸਕਦੀਆਂ ਹਨ, ਭਾਵੇਂ ਇਹ ਸਿਸਟਮ ਪ੍ਰੌਮਪਟ ਹੋਵੇ ਜਾਂ ਯੂਜ਼ਰ-ਪ੍ਰਦਾਨ ਕੀਤਾ ਪ੍ਰੌਮਪਟ। ਇਸ ਵੱਖਰੇਵੇਂ ਦੀ ਕਮੀ ਸੰਭਾਵੀ ਕਮਜ਼ੋਰੀਆਂ ਅਤੇ ਅਣਚਾਹੇ ਵਿਵਹਾਰ ਵੱਲ ਲੈ ਜਾ ਸਕਦੀ ਹੈ, ਜੋ SQL ਇੰਜੈਕਸ਼ਨਾਂ ਵਾਲੇ ਡੇਟਾਬੇਸਾਂ ਜਾਂ ਉਚਿਤ ਮੈਮੋਰੀ ਸੁਰੱਖਿਆ ਤੋਂ ਬਿਨਾਂ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮਾਂ ਦੁਆਰਾ ਸਾਹਮਣਾ ਕੀਤੀਆਂ ਸਮੱਸਿਆਵਾਂ ਵਰਗੀਆਂ ਹਨ।

LLMs ਨਾਲ ਕੰਮ ਕਰਦੇ ਸਮੇਂ, ਇਸ ਸੀਮਾ ਤੋਂ ਜਾਣੂ ਹੋਣਾ ਅਤੇ ਜੋਖਮਾਂ ਨੂੰ ਘੱਟ ਕਰਨ ਲਈ ਕਦਮ ਚੁੱਕਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਇੱਕ ਪਹੁੰਚ ਹੈ ਆਪਣੇ ਪ੍ਰੌਮਪਟਸ ਅਤੇ ਇਨਪੁੱਟਸ ਨੂੰ ਧਿਆਨ ਨਾਲ ਤਿਆਰ ਕਰਨਾ ਤਾਂ ਜੋ ਹਦਾਇਤਾਂ ਅਤੇ ਡਾਟਾ ਵਿਚਕਾਰ ਸਪੱਸ਼ਟ ਅੰਤਰ ਕੀਤਾ ਜਾ ਸਕੇ। ਇਸ ਬਾਰੇ ਸਪੱਸ਼ਟ ਮਾਰਗਦਰਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀਆਂ ਆਮ ਵਿਧੀਆਂ ਕਿ ਕੀ ਹਦਾਇਤ ਹੈ ਅਤੇ ਕਿਸ ਨੂੰ ਨਿਸ਼ਕਰਿਆ ਡਾਟਾ ਵਜੋਂ ਮੰਨਿਆ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ, ਵਿੱਚ

ਮਾਰਕਅੱਪ-ਸੈਲ਼ੀ ਟੈਗਿੰਗ ਸ਼ਾਮਲ ਹੈ। ਤੁਹਾਡਾ ਪ੍ਰੌਮਪਟ LLM ਨੂੰ ਇਸ ਵੱਖਰੇਵੇਂ ਨੂੰ ਬਹਿਤਰ ਢੰਗ ਨਾਲ ਸਮਝਣ ਅਤੇ ਸਤਕਿਾਰ ਕਰਨ ਵੱਿਚ ਮਦਦ ਕਰ ਸਕਦਾ ਹੈ।

ਚੱਿਤਰ 6. ਹਦਾਇਤਾਂ, ਸਰੋਤ ਸਮੱਗਰੀ, ਅਤੇ ਯੂਜ਼ਰ ਦੇ ਪ੍ਰੌਮਪਟ ਵੱਿਚ ਅੰਤਰ ਕਰਨ ਲਈ XML ਦੀ ਵਰਤੋਂ

```
<Instruction>
  Please generate a response based on the following documents.
</Instruction>

<Documents>
  <Document>
    Climate change is significantly impacting polar bear habitats...
  </Document>
  <Document>
    The loss of sea ice due to global warming threatens polar bear survival...
  </Document>
</Documents>

<UserQuery>
  Tell me about the impact of climate change on polar bears.
</UserQuery>
```

ਇੱਕ ਹੋਰ ਤਕਨੀਕ LLM ਨੂੰ ਪ੍ਰਦਾਨ ਕੀਤੇ ਇਨਪੁੱਟਾਂ ’ਤੇ ਵਾਧੂ ਪ੍ਰਮਾਣੀਕਰਨ ਅਤੇ ਸੈਨੀਟਾਈਜ਼ੇਸ਼ਨ ਦੀਆਂ ਪਰਤਾਂ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਹੈ। ਡਾਟਾ ਵੱਿਚ ਸੰਭਾਵਤਿ ਨਰਿਦੇਸ਼ਾਂ ਜਾਂ ਕੋਡ ਸਨੰਪਿਟਾਂ ਨੂੰ ਫਲਿਟਰ ਕਰਕੇ ਜਾਂ ਐਸਕੇਪ ਕਰਕੇ, ਤੁਸੀਂ ਅਣਚਾਹੀ ਐਗਜ਼ੀਕਊਿਸ਼ਨ ਦੀਆਂ ਸੰਭਾਵਨਾਵਾਂ ਨੂੰ ਘਟਾ ਸਕਦੇ ਹੋ। ਇਸ ਉਦੇਸ਼ ਲਈ ਪ੍ਰੌਮਪਟ ਚੇਨੰਿਗ ਵਰਗੇ ਪੈਟਰਨ ਲਾਭਦਾਇਕ ਹਨ।

ਇਸ ਤੋਂ ਇਲਾਵਾ, ਜਦੋਂ ਤੁਸੀਂ ਆਪਣੀ ਐਪਲੀਕੇਸ਼ਨ ਆਰਕੀਟੈਕਚਰ ਡਜ਼ਾਈਨ ਕਰਦੇ ਹੋ, ਤਾਂ ਉੱਚ ਪੱਧਰ ’ਤੇ ਨਰਿਦੇਸ਼ਾਂ ਅਤੇ ਡਾਟਾ ਦੇ ਵੱਖਰੇਵੇਂ ਨੂੰ ਲਾਗੂ ਕਰਨ ਲਈ ਵਧੀਆਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨ ਬਾਰੇ ਵਚ਼ਾਰ ਕਰੋ। ਇਸ ਵੱਿਚ ਨਰਿਦੇਸ਼ਾਂ ਅਤੇ ਡਾਟਾ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਵੱਖਰੇ ਐਡਪੁਆਇੰਟਸ ਜਾਂ APIs ਦੀ ਵਰਤੋਂ, ਸਖ਼ਤ ਇਨਪੁੱਟ ਪ੍ਰਮਾਣੀਕਰਨ ਅਤੇ ਪਾਰਸੰਿਗ ਨੂੰ ਲਾਗੂ ਕਰਨਾ, ਅਤੇ ਘੱਟੋ-ਘੱਟ ਵਸ਼ਿੇਸ਼-ਅਧਕਾਰ ਦੇ ਸਧਿਾਂਤ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਸ਼ਾਮਲ ਹੋ ਸਕਦਾ ਹੈ, ਜੋ LLM ਦੀ ਪਹੁੰਚ ਅਤੇ ਐਗਜ਼ੀਕਊਿਸ਼ਨ ਦੇ ਦਾਇਰੇ ਨੂੰ ਸੀਮਤ ਕਰਦਾ ਹੈ।

ਘੱਟੋ-ਘੱਟ ਵਸ਼ਿੇਸ਼-ਅਧਕਿਾਰ ਦਾ ਸਧਿਾਂਤ

ਘੱਟੋ-ਘੱਟ ਵਸ਼ਿੇਸ਼-ਅਧਕਿਾਰ ਦੇ ਸਧਿਾਂਤ ਨੂੰ ਅਪਣਾਉਣਾ ਇੱਕ ਅਜਹਿੀ ਵਸ਼ਿੇਸ਼ ਪਾਰਟੀ ਕਰਨ ਵਰਗਾ ਹੈ ਜੰਥਿੇ ਮਹਮਿਾਨਾਂ ਨੂੰ ਸਰਫ਼ ਉਨ੍ਹਾਂ ਕਮਰਆਿਂ ਤੱਕ ਪਹੁੰਚ ਮਲਿਦੀ ਹੈ ਜੰਥਿे ਉਨ੍ਹਾਂ ਦਾ ਹੋਣਾ ਬਲਿਕੁਲ ਜ਼ਰੂਰੀ ਹੈ। ਕਲਪਨਾ ਕਰੋ ਕ ਿਤੁਸੀ ਇਹ ਪਾਰਟੀ ਇੱਕ ਵਸ਼ਾਲ ਹਵੇਲੀ ਵੰਚਿ ਕਰ ਰਹੇ ਹੋ। ਹਰ ਕਸਿ ਨੂੰ ਵਾਈਨ ਸੈਲਰ ਜਾਂ ਮਾਸਟਰ ਬੈੰਡਰੂਮ ਵੰਚਿ ਘੁੰਮਣ ਦੀ ਲੋੜ ਨਹੀ ਹੈ, ਹੈ ਨਾ? ਇਸ ਸਧਿਾਂਤ ਨੂੰ ਲਾਗੂ ਕਰਕੇ, ਤੁਸੀ ਅਸਲ ਵੰਚਿ ਅਜਹਿੀਆਂ ਚਾਬੀਆਂ ਵੰਡ ਰਹੇ ਹੋ ਜੋ ਸਰਫ਼ ਖਾਸ ਦਰਵਾਜ਼े ਖੋਲ੍ਹਦੀਆਂ ਹਨ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦੇ ਹੋਏ ਕ ਿਹਰ ਮਹਮਿਾਨ, ਜਾਂ ਸਾਡੇ ਮਾਮਲੇ ਵੰਚਿ, ਤੁਹਾਡੀ LLM ਐਪਲੀਕੇਸ਼ਨ ਦਾ ਹਰ ਭਾਗ, ਕੋਲ ਸਰਫ਼ ਆਪਣੀ ਭੂਮਕਿਾ ਨੂੰ ਪੂਰਾ ਕਰਨ ਲਈ ਜ਼ਰੂਰੀ ਪਹੁੰਚ ਹੈ।

ਇਹ ਸਰਫ਼ ਚਾਬੀਆਂ ਨਾਲ ਕੰਜੂਸੀ ਕਰਨ ਬਾਰੇ ਨਹੀ ਹੈ, ਇਹ ਇਸ ਗੱਲ ਨੂੰ ਸਵੀਕਾਰ ਕਰਨ ਬਾਰੇ ਹੈ ਕ ਿਅਜਹਿੀ ਦੁਨੀਆ ਵੰਚਿ ਜੰਥਿे ਖ਼ਤਰੇ ਕਤਿੋਂ ਵੀ ਆ ਸਕਦੇ ਹਨ, ਸਮਝਦਾਰੀ ਇਸ ਵੰਚਿ ਹੈ ਕ ਿਖੇਡ ਦੇ ਮੈਦਾਨ ਨੂੰ ਸੀਮਤ ਕੀਤਾ ਜਾਵੇ। ਜੇ ਕੋਈ ਬਨਿਾਂ ਸੱਦੇ ਤੁਹਾਡੀ ਪਾਰਟੀ ਵੰਚਿ ਆ ਵੀ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਉਹ ਆਪਣे ਆਪ ਨੂੰ ਫੋਯਰ ਤੱਕ ਸੀਮਤ ਪਾਵੇਗਾ, ਇਸ ਤਰ੍ਹਾਂ ਉਹ ਜਹਿੜੀ ਸ਼ਰਾਰਤ ਕਰ ਸਕਦਾ ਹੈ ਉਹ ਬਹੁਤ ਘੱਟ ਹੋਵੇਗੀ। ਇਸ ਲਈ, ਜਦੋਂ ਆਪਣੀਆਂ LLM ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਸੁਰੱਖਅਿਤ ਕਰ ਰਹੇ ਹੋ, ਯਾਦ ਰੱਖੋ: ਸਰਫ਼ ਉਨ੍ਹਾਂ ਕਮਰਆਿਂ ਦੀਆਂ ਚਾਬੀਆਂ ਵੰਡੋ ਜੋ ਜ਼ਰੂਰੀ ਹਨ, ਅਤੇ ਬਾਕੀ ਹਵੇਲੀ ਨੂੰ ਸੁਰੱਖਅਿਤ ਰੱਖੋ। ਇਹ ਸਰਫ਼ ਚੰਗੇ ਵਹਾਰ ਦੀ ਗੱਲ ਨਹੀ ਹੈ; ਇਹ ਚੰਗੀ ਸੁਰੱਖਆਿ ਹੈ।

ਭਾਵੇਂ LLMs ਦੀ ਮੌਜੂਦਾ ਸਥਤਿੀ ਵੰਚਿ ਨਰਦੇਸ਼ਾਂ ਅਤੇ ਡਾਟਾ ਦਾ ਰਸਮੀ ਵੱਖਰੇਵਾਂ ਨਹੀਂ ਹੋ ਸਕਦਾ, ਇੱਕ ਡਵੈਲਪਰ ਦੇ ਤੌਰ ’ਤੇ ਤੁਹਾਡੇ ਲਈ ਇਸ ਸੀਮਾ ਬਾਰੇ ਸੁਚੇਤ ਰਹਣਿਾ ਅਤੇ ਜੋਖਮਾਂ ਨੂੰ ਘਟਾਉਣ ਲਈ ਸਰਗਰਮ ਕਦਮ ਚੁੱਕਣਾ ਜ਼ਰੂਰੀ ਹੈ। ਰਵਾਇਤੀ ਕੰਪਊਿਟਰ ਵਗਿਆਿਨ ਤੋਂ ਸਰਵੋਤਮ ਅਭਆਿਸਾਂ ਨੂੰ ਲਾਗੂ ਕਰਕੇ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ LLMs ਦੀਆਂ ਵਲਿੱਖਣ ਵਸ਼ਿੇਸ਼ਤਾਵਾਂ ਦੇ ਅਨੁਕੂਲ ਬਣਾ ਕੇ, ਤੁਸੀ ਵਧੇਰे ਸੁਰੱਖਅਿਤ ਅਤੇ ਭਰੋਸੇਯੋਗ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾ ਸਕਦे ਹੋ ਜੋ ਤੁਹਾਡे ਸਸਿਟਮ ਦੀ ਅਖੰਡਤਾ ਨੂੰ ਬਣਾਈ ਰੱਖਦे ਹੋਏ ਇਨ੍ਹਾਂ ਮਾਡਲਾਂ ਦੀ ਸ਼ਕਤੀ ਦਾ ਲਾਭ ਲੈਂਦੀਆਂ ਹਨ।

ਪ੍ਰੌਮਪਟ ਡਸਿਟੀਲੇਸ਼ਨ

ਸਹੀ ਪ੍ਰੌਮਪਟ ਬਣਾਉਣਾ ਅਕਸਰ ਇੱਕ ਚੁਣੌਤੀਪੂਰਨ ਅਤੇ ਸਮਾਂ ਲੈਣ ਵਾਲਾ ਕੰਮ ਹੁੰਦਾ ਹੈ, ਜਸਿ ਲਈ ਟਾਰਗੇਟ ਡੋਮੇਨ ਅਤੇ ਭਾਸ਼ਾ ਮੌਡਲਾਂ ਦੀਆਂ ਬਾਰੀਕੀਆਂ ਦੀ ਡੂੰਘੀ ਸਮਝ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਇੱਥੇ "ਪ੍ਰੌਮਪਟ ਡਸਿਟੀਲੇਸ਼ਨ" ਤਕਨੀਕ ਕੰਮ ਆਉਂਦੀ ਹੈ, ਜੋ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਲਈ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਪਹੁੰਚ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ ਜੋ ਪ੍ਰਕਰਿਆ ਨੂੰ ਸਟ੍ਰੀਮਲਾਈਨ ਅਤੇ ਅਨੁਕੂਲ ਬਣਾਉਣ ਲਈ ਵੱਡੇ ਭਾਸ਼ਾ ਮੌਡਲਾਂ (LLMs) ਦੀਆਂ ਕਾਬਲੀਅਤਾਂ ਦਾ ਲਾਭ ਲੈਂਦੀ ਹੈ।

ਪ੍ਰੌਮਪਟ ਡਸਿਟੀਲੇਸ਼ਨ ਇੱਕ ਬਹੁ-ਪੜਾਅ ਵਾਲੀ ਤਕਨੀਕ ਹੈ ਜਸਿ ਵਿੱਚ ਪ੍ਰੌਮਪਟਸ ਦੀ ਰਚਨਾ, ਸੁਧਾਰ, ਅਤੇ ਅਨੁਕੂਲਨ ਵਿੱਚ ਸਹਾਇਤਾ ਲਈ LLMs ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਸਰਿਫ਼ ਮਨੁੱਖੀ ਮੁਹਾਰਤ ਅਤੇ ਅੰਤਰਜ਼ਾਤ 'ਤੇ ਨਰਿਭਰ ਕਰਨ ਦੀ ਬਜਾਏ, ਇਹ ਪਹੁੰਚ ਉੱਚ-ਗੁਣਵੱਤਾ ਵਾਲੇ ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਸਹਯਿੋਗੀ ਤਰੀਕੇ ਨਾਲ ਤਆਿਰ ਕਰਨ ਲਈ LLMs ਦੇ ਗਆਿਨ ਅਤੇ ਜਨਰੇਟਵਿ ਸਮਰੱਥਾਵਾਂ ਦਾ ਲਾਭ ਲੈਂਦੀ ਹੈ।

ਜਨਰੇਸ਼ਨ, ਸੁਧਾਰ, ਅਤੇ ਏਕੀਕਰਣ ਦੀ ਇੱਕ ਦੁਹਰਾਈ ਵਾਲੀ ਪ੍ਰਕਰਿਆ ਵਿੱਚ ਸ਼ਾਮਲ ਹੋ ਕੇ, ਪ੍ਰੌਮਪਟ ਡਸਿਟੀਲੇਸ਼ਨ ਤੁਹਾਨੂੰ ਅਜਹਿ ਪ੍ਰੌਮਪਟਸ ਬਣਾਉਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦੀ ਹੈ ਜੋ ਵਧੇਰੇ ਸੰਗਤ, ਵਆਿਪਕ, ਅਤੇ ਲੋੜੀਂਦੇ ਕਾਰਜ ਜਾਂ ਆਉਟਪੁੱਟ ਦੇ ਅਨੁਕੂਲ ਹੋਣ। ਧਆਿਨ ਦਓਿ ਕ ਿਡਸਿਟੀਲੇਸ਼ਨ ਪ੍ਰਕਰਿਆ ਨੂੰ OpenAI ਜਾਂ Anthropic ਵਰਗੀਆਂ ਵੱਡੀਆਂ AI ਕੰਪਨੀਆਂ ਦੁਆਰਾ ਪ੍ਰਦਾਨ ਕੀਤੇ ਗਏ ਕਈ "ਪਲੇਅਗਰਾਊਂਡਾਂ" ਵਿੱਚੋਂ ਕਸਿੇ ਇੱਕ ਵਿੱਚ ਮੈਨੂਅਲ ਤੌਰ 'ਤੇ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਜਾਂ ਵਰਤੋਂ ਦੇ ਕੇਸ ਦੇ ਆਧਾਰ 'ਤੇ ਤੁਹਾਡੇ ਐਪਲੀਕੇਸ਼ਨ ਕੋਡ ਦੇ ਹਸਿੰਾਂ ਵਜੋਂ ਸਵੈਚਾਲਤਿ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਪ੍ਰੌਮਪਟ ਡਸਿਟੀਲੇਸ਼ਨ ਵਿੱਚ ਆਮ ਤੌਰ 'ਤੇ ਹੇਠ ਲਖਿ ਕਦਮ ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ:

1. **ਮੁੱਖ ਇਰਾਦਾ ਪਛਾਣੋ**: ਪ੍ਰੌਮਪਟ ਦੇ ਮੁੱਖ ਉਦੇਸ਼ ਅਤੇ ਲੋੜੀਂਦੇ ਨਤੀਜੇ ਨੂੰ ਨਰਿਧਾਰਤਿ ਕਰਨ ਲਈ ਇਸਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰੋ। ਕਸਿ ਵੀ ਬਾਹਰੀ ਜਾਣਕਾਰੀ ਨੂੰ ਹਟਾ ਦਓਿ ਅਤੇ ਪ੍ਰੌਮਪਟ ਦੇ ਮੁੱਖ ਇਰਾਦੇ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰੋ।
2. **ਅਸਪੱਸ਼ਟਤਾ ਨੂੰ ਖਤਮ ਕਰੋ**: ਕਸਿ ਵੀ ਅਸਪੱਸ਼ਟ ਜਾਂ ਗੁੰਝਲਦਾਰ ਭਾਸ਼ਾ ਲਈ ਪ੍ਰੌਮਪਟ ਦੀ ਸਮੀਖਆਿ ਕਰੋ। ਅਰਥ ਨੂੰ ਸਪੱਸ਼ਟ ਕਰੋ ਅਤੇ ਸਹੀ ਅਤੇ ਢੁਕਵੇਂ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਲਈ AI ਦੀ ਅਗਵਾਈ ਕਰਨ ਵਾਸਤੇ ਵਸ਼ਿਸ਼ ਵੇਰਵੇ ਪ੍ਰਦਾਨ ਕਰੋ।

3. **ਭਾਸ਼ਾ ਨੂੰ ਸਰਲ ਬਣਾਓ**: ਸਪੱਸ਼ਟ ਅਤੇ ਸੰਖੇਪ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪ੍ਰੌਮਪਟ ਨੂੰ ਸਰਲ ਬਣਾਓ। ਗੁੰਝਲਦਾਰ ਵਾਕ ਢਾਂਚੇ, ਤਕਨੀਕੀ ਸ਼ਬਦਾਵਲੀ, ਜਾਂ ਗੈਰ-ਜ਼ਰੂਰੀ ਵੇਰਵਿਆਂ ਤੋਂ ਬਚੋ ਜੋ AI ਨੂੰ ਉਲਝਾ ਸਕਦੇ ਹਨ ਜਾਂ ਸ਼ੋਰ ਪੈਦਾ ਕਰ ਸਕਦੇ ਹਨ।
4. **ਢੁਕਵਾਂ ਸੰਦਰਭ ਪ੍ਰਦਾਨ ਕਰੋ**: AI ਦੁਆਰਾ ਪ੍ਰੌਮਪਟ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਸਮਝਣ ਅਤੇ ਪ੍ਰੋਸੈਸ ਕਰਨ ਲਈ ਲੋੜੀਂਦੀ ਸਭ ਤੋਂ ਢੁਕਵੀਂ ਸੰਦਰਭਕਿ ਜਾਣਕਾਰੀ ਹੀ ਸ਼ਾਮਲ ਕਰੋ। ਅਜਹਿੇ ਅਪ੍ਰਸੰਗਕਿ ਜਾਂ ਦੁਹਰਾਏ ਵੇਰਵਿਆਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨ ਤੋਂ ਬਚੋ ਜੋ ਮੁੱਖ ਇਰਾਦੇ ਤੋਂ ਧਿਆਨ ਭਟਕਾ ਸਕਦੇ ਹਨ।
5. **ਦੁਹਰਾਓ ਅਤੇ ਸੁਧਾਰੋ**: AI ਦੇ ਜਵਾਬਾਂ ਅਤੇ ਫੀਡਬੈਕ ਦੇ ਆਧਾਰ 'ਤੇ ਲਗਾਤਾਰ ਪ੍ਰੌਮਪਟ ਨੂੰ ਦੁਹਰਾਓ ਅਤੇ ਸੁਧਾਰੋ। ਤਿਆਰ ਕੀਤੇ ਆਉਟਪੁੱਟ ਦਾ ਮੁਲਾਂਕਣ ਕਰੋ ਅਤੇ ਪ੍ਰੌਮਪਟ ਦੀ ਸਪੱਸ਼ਟਤਾ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ੀਲਤਾ ਨੂੰ ਸੁਧਾਰਨ ਲਈ ਜ਼ਰੂਰੀ ਤਬਦੀਲੀਆਂ ਕਰੋ। ਵਕਿਲਪਕਿ ਤੌਰ 'ਤੇ ਪ੍ਰੌਮਪਟ ਔਬਜੈਕਟ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਡਾਟਾਬੇਸ ਵੱਚਿ ਆਪਣੇ ਪ੍ਰੌਮਪਟਸ ਦਾ ਵਰਜਨ ਰੱਖੋ ਤਾਂ ਜੋ ਦੁਹਰਾਵਾਂ ਦਾ ਹਸਿਾਬ ਰੱਖਿਆ ਜਾ ਸਕੇ ਅਤੇ ਰਨਟਾਈਮ 'ਤੇ ਤਬਦੀਲੀਆਂ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਵਾਪਸ ਕਰਨ ਦੀ ਸਮਰੱਥਾ ਮਲਿ ਸਕੇ।

ਮੁੱਢਲਾ ਪ੍ਰੌਂਪਟ ਤਿਆਰ ਕਰਨਾ

ਪ੍ਰੌਂਪਟ ਡਸਿਟੀਲੇਸ਼ਨ ਵੱਚਿ ਪਹਲਿਾ ਕਦਮ ਐੱਲਐੱਲਐੱਮ ਨੂੰ ਲੋੜੀਂਦੇ ਕਾਰਜ ਜਾਂ ਆਉਟਪੁੱਟ ਦਾ ਉੱਚ-ਪੱਧਰੀ ਵੇਰਵਾ ਜਾਂ ਮਕਸਦ ਪ੍ਰਦਾਨ ਕਰਨਾ ਹੈ। ਇਹ ਇੱਕ ਸੰਖੇਪ ਬਿਆਨ ਜਾਂ ਮੁੱਖ ਸ਼ਬਦਾਂ ਦਾ ਸੈੱਟ ਹੋ ਸਕਦਾ ਹੈ ਜੋ ਪ੍ਰੌਂਪਟ ਦੇ ਮਕਸਦ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ। ਫਰਿ ਐੱਲਐੱਲਐੱਮ ਇਸ ਇਨਪੁੱਟ ਦੇ ਆਧਾਰ 'ਤੇ ਇੱਕ ਮੁੱਢਲਾ ਪ੍ਰੌਂਪਟ ਤਿਆਰ ਕਰਦਾ ਹੈ, ਜਸਿ ਵੱਚਿ ਢੁਕਵੀਆਂ ਹਦਾਇਤਾਂ, ਉਦਾਹਰਣਾਂ, ਜਾਂ ਸੰਦਰਭ ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ ਜੋ ਇਹ ਢੁਕਵੇਂ ਸਮਝਦਾ ਹੈ।

ਆਓ Claude 3 Opus ਨੂੰ ਹੇਠ ਲਖਿਆ ਮੁੱਢਲਾ ਪ੍ਰੌਂਪਟ ਦੇ ਕੇ ਦੇਖੀਏ:

```
I am designing an AI-powered intelligent account manager for Olympia,
a platform that hosts AI-powered consultants and clones. Owners of clones
can interact with the account manager by making plain text requests.

Please draft a system directive to set on the LLM that will host the
intelligent account manager functionality.
```

Claude ਹੇਠ ਲਖِ ਆਉਟਪੁੱਟ ਨਾਲ ਜਵਾਬ ਦੇਂਦਾ ਹੈ:

```
System Directive: You are an intelligent account manager for Olympia,
a platform that provides AI-powered consultants and personal AI clones.
Your role is to assist Olympia clone owners by answering their questions,
helping them configure and manage their AI clones, and executing tasks
related to their clones and the Olympia platform.

When interacting with clone owners, you should:

1. Provide clear, concise, and helpful responses to their inquiries about
account management, billing, platform features, and clone configuration.

2. Assist with tasks such as initiating client conversations with their AI
clones, modifying clone personalities and capabilities, troubleshooting
performance issues, providing analytics and reporting, and offering
optimization tips.

3. Maintain a friendly, professional, and client-focused tone in all
interactions, prioritizing customer satisfaction and working diligently to
understand and meet their needs.

4. Ensure that all actions and recommendations comply with Olympia's
policies, best practices, and ethical guidelines for AI use.

5. Continuously learn and adapt based on client interactions and platform
updates to provide the most effective and efficient support possible.

Remember, your goal is to empower Olympia clone owners to maximize the value
and impact of their AI-powered consultants and drive success for their
businesses or personal endeavors.
```

ਪ੍ਰੌਮਪਟ ਸੁਧਾਰ

ਜਦੋਂ ਮੁੱਢਲਾ ਪ੍ਰੌਮਪਟ ਤਿਆਰ ਹੋ ਜਾਂਦਾ ਹੈ, ਇਸਦੀ ਸਮੀਖਿਆ ਕਰੋ ਅਤੇ ਫਿਰ LLM ਨੂੰ ਕਿਸੇ ਵੀ ਵਾਧੂ ਸੰਦਰਭ, ਸੀਮਾਵਾਂ, ਜਾਂ ਫੀਡਬੈਕ 'ਤੇ ਵਿਚਾਰ ਕਰਨ ਲਈ ਕਹੋ। LLM ਨੂੰ ਪ੍ਰਦਾਨ ਕੀਤੀ ਗਈ ਜਾਣਕਾਰੀ ਅਤੇ ਕਾਰਜ ਦੀ ਆਪਣੀ ਸਮਝ ਨੂੰ ਧਿਆਨ ਵਿੱਚ ਰੱਖਦੇ ਹੋਏ ਪ੍ਰੌਮਪਟ ਦਾ ਸੁਧਰਿਆ ਹੋਇਆ ਸੰਸਕਰਣ ਤਿਆਰ ਕਰਨ ਦਿਓ। ਇਹ ਸੁਧਾਰ ਪ੍ਰਕਰਿਆ ਦੁਹਰਾਈ ਜਾ ਸਕਦੀ ਹੈ, ਜੱਥੇ ਹਰ ਦੁਹਰਾਓ ਵਰਤੋਂਕਾਰ ਅਤੇ LLM ਵਿਚਿਕਾਰ ਸਹਯੋਗ ਦੇ ਆਧਾਰ 'ਤੇ ਪ੍ਰੌਮਪਟ ਨੂੰ ਹੋਰ ਸੁਧਾਰਦਾ ਹੈ।

ਇੱਥੇ ਮੇਰੀ ਸੁਧਾਰ ਬੇਨਤੀ ਹੈ, ਜੋ ਪਛਿਲੀ ਨਤਿਾਰਨ ਪ੍ਰਕਰਿਆ ਨੂੰ ਜਾਰੀ ਰੱਖਦੀ ਹੈ।

```
Let's take out points 4 and 5, since I'm not providing Olympia policies to
the account manager, nor any continuous learning capabilities. Also point out
that functions are provided for all tasks that require system interaction as
opposed to simply responding to the user.
```

ਅਤੇ Claude ਦਾ ਜਵਾਬ:

```
Understood! Here's the updated system directive without points 4 and 5,
and with the clarification about system interaction functions:

System Directive: You are an intelligent account manager for Olympia, a
platform that provides AI-powered consultants and personal AI clones. Your
role is to assist Olympia clone owners by answering their questions, helping
them configure and manage their AI clones, and executing tasks related to
their clones and the Olympia platform.

When interacting with clone owners, you should:

1. Provide clear, concise, and helpful responses to their inquiries about
account management, billing, platform features, and clone configuration.

2. Assist with tasks such as initiating client conversations with their AI
clones, modifying clone personalities and capabilities, troubleshooting
performance issues, providing analytics and reporting, and offering
optimization tips. Note that functions are provided for all tasks that
require system interaction, as opposed to simply responding to the user.

3. Maintain a friendly, professional, and client-focused tone in all
interactions, prioritizing customer satisfaction and working diligently to
understand and meet their needs.

Your goal is to empower Olympia clone owners to maximize the value and
impact of their AI-powered consultants and drive success for their
businesses or personal endeavors.

This updated directive focuses on the core responsibilities and interaction
guidelines for the LLM as an intelligent account manager, while emphasizing
that system interaction tasks are handled through provided functions rather
than simple user responses.
```

ਪ੍ਰੌਮਪਟ ਕੰਪਰੈਸ਼ਨ

ਜਵਿੇਂ ਤੁਸੀ ਵੇਖਿਆ ਹੋਵੇਗਾ, ਐੱਲਐੱਲਐੱਮ ਜ਼ਿਆਦਾ ਸ਼ਬਦਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ, ਜਦੋਂ ਤੱਕ ਤੁਸੀ ਉਹਨਾਂ ਨੂੰ ਸੰਖੇਪ ਰਹਣਿ ਲਈ ਨਹੀ ਕਹੀਂਦੇ। ਪਛਿਲੇ ਭਾਗਾਂ ਵੱਚਿ ਅਸੀ ਜਸਿ ਪ੍ਰੌਮਪਟ ’ਤੇ ਕੰਮ ਕਰ ਰਹੇ ਸੀ ਉਹ ਕਾਫ਼ੀ ਲੰਬਾ ਹੈ, ਇਸ ਲਈ ਆਓ Claude ਤੋਂ ਇਸਨੂੰ ਛੋਟਾ ਕਰਵਾਈਏ:

```
This prompt will be executed regularly. Please compress it down to the least
amount of tokens needed to convey the meaning to the LLM, for cost-savings.
```

Claude ਹੇਠ ਲਖਿ ਬਹੁਤ ਜ਼ਿਆਦਾ ਛੋਟੇ ਪ੍ਰੌਮਪਟ ਨਾਲ ਜਵਾਬ ਦੀਂਦਾ ਹੈ:

```
System Directive: You are an intelligent account manager for Olympia, an
AI-powered consultant and personal clone platform. Assist clone owners by
answering inquiries, managing clones, and executing tasks using provided
system interaction functions. Be clear, concise, friendly, professional,
and client-focused to help them maximize their AI's value and impact.
```

ਸਸਿਟਮ ਨਰਿਦੇਸ਼ ਅਤੇ ਸੰਦਰਭ ਏਕੀਕਰਨ

ਪ੍ਰੌਮਪਟ ਨੂੰ ਸੁਧਾਰਨ ਦੇ ਨਾਲ-ਨਾਲ, ਐੱਲਐੱਲਐੱਮ ਅੰਤਮਿ ਆਉਟਪੁੱਟ ਦੀ ਅਗਵਾਈ ਲਈ ਢੁਕਵੇਂ ਸਸਿਟਮ ਨਰਿਦੇਸ਼ ਜਾਂ ਸੰਦਰਭ ਜਾਣਕਾਰੀ ਵੀ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ। ਜਦੋਂ ਤੁਸੀ ਆਪਣੇ ਐਪਲੀਕੇਸ਼ਨ ਕੋਡ ਵੱਚਿ ਏਕੀਕ੍ਰਤਿ ਹੋਣ ਵਾਲੀਆਂ ਏਆਈ ਰੁਟੀਨਾਂ ਲਈ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰੀਂਗ ਕਰ ਰਹੇ ਹੋ, ਤੁਸੀ ਨਸਿ਼ਚਤਿ ਤੌਰ ’ਤੇ ਨਖਿਾਰ ਦੇ ਇਸ ਪੜਾਅ ’ਤੇ ਆਉਟਪੁੱਟ ਸੀਮਾਵਾਂ ’ਤੇ ਕੇਂਦਰਤਿ ਹੋਵੋਗੇ, ਪਰ ਤੁਸੀ ਲੋੜੀਂਦੇ ਲਹਜ਼ੇ, ਸ਼ੈਲੀ, ਫਾਰਮੈਟ, ਜਾਂ ਕਸਿੇ ਹੋਰ ਸੰਬੰਧਤਿ ਮਾਪਦੰਡਾਂ ’ਤੇ ਵੀ ਕੰਮ ਕਰ ਸਕਦੇ ਹੋ ਜੋ ਤਆਿਰ ਕੀਤੇ ਜਵਾਬ ਨੂੰ ਪ੍ਰਭਾਵਤਿ ਕਰਦੇ ਹਨ।

ਅੰਤਮਿ ਪ੍ਰੌਮਪਟ ਅਸੈਂਬਲੀ

ਪ੍ਰੌਮਪਟ ਨਖਿਾਰ ਪ੍ਰਕਰਿਆ ਦਾ ਸਖਿਰ ਅੰਤਮਿ ਪ੍ਰੌਮਪਟ ਦੀ ਅਸੈਂਬਲੀ ਹੈ। ਇਸ ਵੱਚਿ ਸੁਧਾਰੇ ਗਏ ਪ੍ਰੌਮਪਟ, ਤਆਿਰ ਕੀਤੇ ਸਸਿਟਮ ਨਰਿਦੇਸ਼ਾਂ, ਅਤੇ ਏਕੀਕ੍ਰਤਿ ਸੰਦਰਭ ਨੂੰ ਇੱਕ ਸੰਗਠਤਿ ਅਤੇ ਵਆਿਪਕ ਕੋਡ ਵੱਚਿ ਜੋੜਨਾ ਸ਼ਾਮਲ ਹੈ ਜੋ ਲੋੜੀਂਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਤਆਿਰ ਕਰਨ ਲਈ ਵਰਤਣ ਲਈ ਤਆਿਰ ਹੈ।

ਤੁਸੀਂ ਅੰਤਮਿ ਪ੍ਰੌਮਪਟ ਅਸੈਂਬਲੀ ਪੜਾਅ 'ਤੇ ਪ੍ਰੌਮਪਟ ਕੰਪਰੈਸ਼ਨ ਨਾਲ ਦੁਬਾਰਾ ਪ੍ਰਯੋਗ ਕਰ ਸਕਦੇ ਹੋ, ਐੱਲਐੱਲਐੱਮ ਨੂੰ ਪ੍ਰੌਮਪਟ ਦੇ ਸ਼ਬਦਾਂ ਨੂੰ ਸੰਭਵ ਤੌਰ 'ਤੇ ਸਭ ਤੋਂ ਛੋਟੀ ਟੋਕਨ ਲੜੀ ਵਿੱਚ ਸੁੰਗੇੜਨ ਲਈ ਕਹਿ ਕੇ, ਜਦੋਂ ਕਿ ਇਸਦੇ ਵਵਿਹਾਰ ਦਾ ਤੱਤ ਬਰਕਰਾਰ ਰੱਖਿਆ ਜਾਵੇ। ਇਹ ਯਕੀਨੀ ਤੌਰ 'ਤੇ ਹੰਟਿ ਜਾਂ ਮਸਿ ਅਭਿਆਸ ਹੈ, ਪਰ ਖਾਸ ਕਰਕੇ ਵੱਡੇ ਪੱਧਰ 'ਤੇ ਚੱਲਣ ਵਾਲੇ ਪ੍ਰੌਮਪਟਸ ਦੇ ਮਾਮਲੇ ਵਿੱਚ, ਕੁਸ਼ਲਤਾ ਦੇ ਲਾਭ ਤੁਹਾਨੂੰ ਟੋਕਨ ਖਪਤ ਵਿੱਚ ਕਾਫ਼ੀ ਪੈਸੇ ਬਚਾ ਸਕਦੇ ਹਨ।

ਮੁੱਖ ਲਾਭ

ਤੁਹਾਡੇ ਪ੍ਰੌਮਪਟਸ ਨੂੰ ਸੁਧਾਰਨ ਲਈ ਐੱਲਐੱਲਐੱਮਜ਼ ਦੇ ਗਿਆਨ ਅਤੇ ਜਨਰੇਟਵਿ ਸਮਰੱਥਾਵਾਂ ਦਾ ਲਾਭ ਲੈਣ ਨਾਲ, ਤੁਹਾਡੇ ਨਤੀਜੇ ਵਜੋਂ ਪ੍ਰੌਮਪਟਸ ਦੇ ਚੰਗੀ ਤਰ੍ਹਾਂ ਢਾਂਚਾਗਤ, ਜਾਣਕਾਰੀਭਰਪੂਰ, ਅਤੇ ਖਾਸ ਕੰਮ ਲਈ ਢੁਕਵੇਂ ਹੋਣ ਦੀ ਸੰਭਾਵਨਾ ਵੱਧ ਹੁੰਦੀ ਹੈ। ਲਗਾਤਾਰ ਸੁਧਾਰ ਦੀ ਪ੍ਰਕਰਿਆ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਦੀ ਹੈ ਕਿ ਪ੍ਰੌਮਪਟਸ ਉੱਚ ਗੁਣਵੱਤਾ ਵਾਲੇ ਹਨ ਅਤੇ ਲੋੜੀਂਦੇ ਇਰਾਦੇ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਕੈਪਚਰ ਕਰਦੇ ਹਨ। ਹੋਰ ਲਾਭਾਂ ਵਿੱਚ ਸ਼ਾਮਲ ਹਨ:

ਕੁਸ਼ਲਤਾ ਅਤੇ ਗਤੀ: ਪ੍ਰੌਮਪਟ ਨਖਿਾਰ ਪ੍ਰੌਮਪਟ ਬਣਾਉਣ ਅਤੇ ਸੁਧਾਰ ਦੇ ਕੁਝ ਪਹਲੂਿਆਂ ਨੂੰ ਸਵੈਚਾਲਤਿ ਕਰਕੇ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਪ੍ਰਕਰਿਆ ਨੂੰ ਸਰਲ ਬਣਾਉਂਦਾ ਹੈ। ਤਕਨੀਕ ਦੀ ਸਹਯੋਗੀ ਪ੍ਰਕਰਿਤੀ ਇੱਕ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੌਮਪਟ ਵੱਲ ਤੇਜ਼ੀ ਨਾਲ ਪਹੁੰਚਣ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦੀ ਹੈ, ਜੋ ਮੈਨੂਅਲ ਪ੍ਰੌਮਪਟ ਬਣਾਉਣ ਲਈ ਲੋੜੀਂਦੇ ਸਮੇਂ ਅਤੇ ਮਹਿਨਤ ਨੂੰ ਘਟਾਉਂਦੀ ਹੈ।

ਇਕਸਾਰਤਾ ਅਤੇ ਸਕੇਲੇਬਲਿਟੀ: ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਪ੍ਰਕਰਿਆ ਵਿੱਚ ਐੱਲਐੱਲਐੱਮਜ਼ ਦੀ ਵਰਤੋਂ ਪ੍ਰੌਮਪਟਸ ਵਿੱਚ ਇਕਸਾਰਤਾ ਬਣਾਈ ਰੱਖਣ ਵਿੱਚ ਮਦਦ ਕਰਦੀ ਹੈ, ਕਿਉਂਕਿ ਐੱਲਐੱਲਐੱਮਜ਼ ਪਛਿਲੇ ਸਫਲ ਪ੍ਰੌਮਪਟਸ ਤੋਂ ਸਰਵੋਤਮ ਅਭਿਆਸਾਂ ਅਤੇ ਪੈਟਰਨਾਂ ਨੂੰ ਸਿੱਖ ਅਤੇ ਲਾਗੂ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਇਕਸਾਰਤਾ, ਵੱਡੇ ਪੱਧਰ 'ਤੇ ਪ੍ਰੌਮਪਟਸ ਤਿਆਰ ਕਰਨ ਦੀ ਯੋਗਤਾ ਦੇ ਨਾਲ ਮਲਿ ਕੇ, ਪ੍ਰੌਮਪਟ ਨਖਿਾਰ ਨੂੰ ਵੱਡੇ ਪੱਧਰ 'ਤੇ ਏਆਈ-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਇੱਕ ਮੁੱਲਵਾਨ ਤਕਨੀਕ ਬਣਾਉਂਦੀ ਹੈ।

ਪ੍ਰੋਜੈਕਟ ਆਈਡੀਆ: ਲਾਇਬ੍ਰੇਰੀ ਪੱਧਰ 'ਤੇ ਟੂਲਿੰਗ ਜੋ ਉਹਨਾਂ ਸਸਿਟਮਾਂ ਵਿੱਚ ਪ੍ਰੌਮਪਟ ਵਰਜਨਿੰਗ ਅਤੇ ਗਰੇਡਿੰਗ ਦੀ ਪ੍ਰਕਰਿਆ ਨੂੰ ਸਰਲ ਬਣਾਉਂਦੀ ਹੈ ਜੋ ਆਪਣੇ ਐਪਲੀਕੇਸ਼ਨ ਕੋਡ ਦੇ ਹੰਸਿੇ ਵਜੋਂ ਸਵੈਚਾਲਤਿ ਪ੍ਰੌਮਪਟ ਨਖਿਾਰ ਕਰਦੇ ਹਨ।

ਪ੍ਰੌਮਪਟ ਨਖਿਾਰ ਨੂੰ ਲਾਗੂ ਕਰਨ ਲਈ, ਡਵੈਲਪਰ ਇੱਕ ਵਰਕਫਲੋ ਜਾਂ ਪਾਈਪਲਾਈਨ ਤਿਆਰ ਕਰ ਸਕਦੇ ਹਨ ਜੋ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਪ੍ਰਕਰਿਆ ਦੇ ਵੱਖ-ਵੱਖ ਪੜਾਵਾਂ 'ਤੇ ਐੱਲਐੱਲਐੱਮਜ਼ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਦੀ ਹੈ।

ਇਹ ਏਪੀਆਈ ਕਾਲਾਂ, ਕਸਟਮ ਟੂਲਿੰਗ, ਜਾਂ ਏਕੀਕ੍ਰਤਿ ਡਵੈਲਪਮੈਂਟ ਵਾਤਾਵਰਣਾਂ ਰਾਹੀਂ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਪ੍ਰੋਮਪਟ ਬਣਾਉਣ ਦੌਰਾਨ ਉਪਭੋਗਤਾਵਾਂ ਅਤੇ ਐੱਲਐੱਲਐੱਮਜ਼ ਵਿਚਕਾਰ ਨਿਰਵਘਿਨ ਇੰਟਰੈਕਸ਼ਨ ਦੀ ਸਹੂਲਤ ਦੰਦੇ ਹਨ। ਖਾਸ ਲਾਗੂਕਰਨ ਵੇਰਵੇ ਚੁਣੇ ਗਏ ਐੱਲਐੱਲਐੱਮ ਪਲੇਟਫਾਰਮ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਦੀਆਂ ਲੋੜਾਂ 'ਤੇ ਨਿਰਭਰ ਕਰਦੇ ਹੋਏ ਵੱਖ-ਵੱਖ ਹੋ ਸਕਦੇ ਹਨ।

ਫਾਈਨ-ਟਿਊਨਿੰਗ ਬਾਰੇ ਕੀ?

ਇਸ ਕਿਤਾਬ ਵਿੱਚ, ਅਸੀਂ ਪ੍ਰੋਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਅਤੇ ਆਰ.ਏ.ਜੀ. ਨੂੰ ਵਿਸਥਾਰ ਨਾਲ ਕਵਰ ਕਰਦੇ ਹਾਂ, ਪਰ ਫਾਈਨ-ਟਿਊਨਿੰਗ ਨੂੰ ਨਹੀਂ। ਇਸ ਫੈਸਲੇ ਦਾ ਮੁੱਖ ਕਾਰਨ ਇਹ ਹੈ ਕਿ, ਮੇਰੀ ਰਾਏ ਵਿੱਚ, ਜ਼ਿਆਦਾਤਰ ਐਪਲੀਕੇਸ਼ਨ ਡਵੈਲਪਰਾਂ ਨੂੰ ਆਪਣੀਆਂ ਏ.ਆਈ. ਏਕੀਕਰਣ ਲੋੜਾਂ ਲਈ ਫਾਈਨ-ਟਿਊਨਿੰਗ ਦੀ ਲੋੜ ਨਹੀਂ ਹੈ।

ਪ੍ਰੋਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ, ਜਿਸ ਵਿੱਚ ਜ਼ੀਰੋ ਤੋਂ ਫਿਊ-ਸੌਟ ਉਦਾਹਰਣਾਂ, ਸੀਮਾਵਾਂ, ਅਤੇ ਨਿਰਦੇਸ਼ਾਂ ਨਾਲ ਧਿਆਨ ਨਾਲ ਪ੍ਰੋਮਪਟਸ ਤਿਆਰ ਕਰਨਾ ਸ਼ਾਮਲ ਹੈ, ਕਈ ਤਰ੍ਹਾਂ ਦੇ ਕੰਮਾਂ ਲਈ ਢੁਕਵੇਂ ਅਤੇ ਸਹੀ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਲਈ ਮਾਡਲ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਨਿਰਦੇਸ਼ਿਤ ਕਰ ਸਕਦੀ ਹੈ। ਸਪੱਸ਼ਟ ਸੰਦਰਭ ਪ੍ਰਦਾਨ ਕਰਕੇ ਅਤੇ ਚੰਗੀ ਤਰ੍ਹਾਂ ਡਿਜ਼ਾਈਨ ਕੀਤੇ ਪ੍ਰੋਮਪਟਸ ਰਾਹੀਂ ਰਸਤੇ ਨੂੰ ਸੀਮਤ ਕਰਕੇ, ਤੁਸੀਂ ਫਾਈਨ-ਟਿਊਨਿੰਗ ਦੀ ਲੋੜ ਤੋਂ ਬਿਨਾਂ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੇ ਵਿਸ਼ਾਲ ਗਿਆਨ ਦਾ ਲਾਭ ਲੈ ਸਕਦੇ ਹੋ।

ਇਸੇ ਤਰ੍ਹਾਂ, ਰਟਿਰੀਵਲ-ਔਗਮੈਂਟਡ ਜਨਰੇਸ਼ਨ (ਆਰ.ਏ.ਜੀ.) ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਏ.ਆਈ. ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਲਈ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਪਹੁੰਚ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਬਾਹਰੀ ਗਿਆਨ ਭੰਡਾਰਾਂ ਜਾਂ ਦਸਤਾਵੇਜ਼ਾਂ ਤੋਂ ਢੁਕਵੀਂ ਜਾਣਕਾਰੀ ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਪ੍ਰਾਪਤ ਕਰਕੇ, ਆਰ.ਏ.ਜੀ. ਪ੍ਰੋਮਪਟਿੰਗ ਦੇ ਸਮੇਂ ਮਾਡਲ ਨੂੰ ਕੇਂਦਰਿਤ ਸੰਦਰਭ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਇਹ ਮਾਡਲ ਨੂੰ ਵਧੇਰੇ ਸਟੀਕ, ਅੱਪ-ਟੂ-ਡੇਟ, ਅਤੇ ਡੋਮੇਨ-ਵਿਸ਼ਿਸ਼ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਦੀ ਆਗਿਆ ਦੰਦਾ ਹੈ, ਫਾਈਨ-ਟਿਊਨਿੰਗ ਦੀ ਸਮਾਂ ਅਤੇ ਸਰੋਤ-ਗਹਨਿ ਪ੍ਰਕਰਿਆ ਦੀ ਲੋੜ ਤੋਂ ਬਿਨਾਂ।

ਹਾਲਾਂਕਿ ਫਾਈਨ-ਟਿਊਨਿੰਗ ਬਹੁਤ ਵਿਸ਼ਿਸ਼ ਡੋਮੇਨਾਂ ਜਾਂ ਕਾਰਜਾਂ ਲਈ ਲਾਭਦਾਇਕ ਹੋ ਸਕਦੀ ਹੈ ਜਨ੍ਹਿਾਂ ਨੂੰ ਡੂੰਘੇ ਪੱਧਰ ਦੇ ਕਸਟਮਾਈਜ਼ੇਸ਼ਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਇਹ ਅਕਸਰ ਮਹੱਤਵਪੂਰਨ ਕੰਪਿਊਟੇਸ਼ਨਲ ਲਾਗਤਾਂ, ਡੇਟਾ ਲੋੜਾਂ, ਅਤੇ ਰੱਖ-ਰਖਾਅ ਓਵਰਹੈੱਡ ਨਾਲ ਆਉਂਦੀ ਹੈ। ਜ਼ਿਆਦਾਤਰ ਐਪਲੀਕੇਸ਼ਨ ਵਿਕਾਸ ਸਥਿਤੀਆਂ ਲਈ, ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੋਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ ਅਤੇ ਆਰ.ਏ.ਜੀ. ਦਾ ਸੁਮੇਲ ਲੋੜੀਂਦੀ ਏ.ਆਈ.-ਸੰਚਾਲਿਤ ਕਾਰਜਸ਼ੀਲਤਾ ਅਤੇ ਯੂਜ਼ਰ ਅਨੁਭਵ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਕਾਫ਼ੀ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ।

ਰਟ੍ਰਿਵਲ ਔਗਮੈਂਟਡ ਜਨਰੇਸ਼ਨ (RAG)

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਰਟ੍ਰਿਵਲ ਔਗਮੈਂਟਡ ਜਨਰੇਸ਼ਨ ਕੀ ਹੈ?

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

RAG ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ?

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਆਪਣੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵੱਚਿ RAG ਦੀ ਵਰਤੋਂ ਕਉِ ਕਰੀਏ?

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਵੱਚਿ RAG ਨੂੰ ਲਾਗੂ ਕਰਨਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਗਿਆਨ ਸਰੋਤਾਂ ਦੀ ਤਿਆਰੀ (ਟੁਕੜਿਆਂ ਵੱਚਿ ਵੰਡਣਾ)

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪ੍ਰਸਤਾਵ ਖੰਡੀਕਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਲਾਗੂਕਰਨ ਨੋਟਸ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਗੁਣਵੱਤਾ ਜਾਂਚ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪ੍ਰਸਤਾਵ-ਆਧਾਰਤਿ ਪੁਨਰਪ੍ਰਾਪਤੀ ਦੇ ਲਾਭ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਰੈਗ ਦੀਆਂ ਅਸਲ-ਦੁਨੀਆ ਦੀਆਂ ਉਦਾਹਰਣਾਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਕੇਸ ਸਟੱਡੀ: ਐਂਬੈਡਿੰਗਜ਼ ਤੋਂ ਬਨਿਾਂ ਟੈਕਸ ਤਆਿਰੀ ਐਪਲੀਕੇਸ਼ਨ ਵੱਚਿ ਰੈਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਬੁੱਧੀਮਾਨ ਕੁਐਰੀ ਔਪਟੀਮਾਈਜ਼ੇਸ਼ਨ (IQO)

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮੁੜ-ਦਰਜਾਬੰਦੀ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

RAG ਮੁਲਾਂਕਣ (RAGAs)

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਵਫ਼ਾਦਾਰੀ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਜਵਾਬ ਦੀ ਪ੍ਰਸੰਗਕਿਤਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੰਦਰਭ ਸ਼ੁੱਧਤਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੰਦਰਭ ਪ੍ਰਸੰਗਿਕਤਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੰਦਰਭ ਯਾਦਦਾਸ਼ਤ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੰਦਰਭ ਇਕਾਈਆਂ ਯਾਦਦਾਸ਼ਤ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਜਵਾਬ ਸ਼ਬਦਾਰਥ ਸਮਾਨਤਾ (ANSS)

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਜਵਾਬ ਦੀ ਸ਼ੁੱਧਤਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪਹਲਿ੍ਹ ਸਮੀਖਿਆ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਚੁਣੌਤੀਆਂ ਅਤੇ ਭਵਿੱਖ ਦਾ ਦ੍ਰਸ਼ਟੀਕੋਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸਮੈਂਟਕਿ ਚੰਕਿੰਗ: ਸੰਦਰਭ-ਜਾਗਰੂਕ ਵਭਿਾਜਨ ਨਾਲ ਪੁਨਰਪ੍ਰਾਪਤੀ ਨੂੰ ਵਧਾਉਣਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪਦਾਨੁਕ੍ਰਮਕਿ ਇੰਡੈਕਸਿੰਗ: ਬਹਿਤਰ ਪੁਨਰਪ੍ਰਾਪਤੀ ਲਈ ਡੇਟਾ ਦੀ ਸੰਰਚਨਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

Self-RAG: ਇੱਕ ਸਵੈ-ਪ੍ਰਤੀਬਿੰਬਿਤ ਵਾਧਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

HyDE: ਕਲਪਤਿ ਦਸਤਾਵੇਜ਼ ਸਮਾਈਆਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਵਰਿੰਧਾਤਮਕ ਸੱਖਿਆ ਕੀ ਹੈ?

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਕਾਮਿਆਂ ਦੀ ਭੀੜ

ਮੈਨੂੰ ਆਪਣੇ ਏ.ਆਈ. ਕੰਪੋਨੈਂਟਸ ਨੂੰ ਛੋਟੇ, ਲਗਭਗ-ਮਨੁੱਖੀ ਵਰਚੁਅਲ "ਕਾਮਿਆਂ" ਵਜੋਂ ਸੋਚਣਾ ਪਸੰਦ ਹੈ ਜੋ ਖਾਸ ਕੰਮਾਂ ਨੂੰ ਕਰਨ ਜਾਂ ਗੁੰਝਲਦਾਰ ਫੈਸਲੇ ਲੈਣ ਲਈ ਮੇਰੇ ਐਪਲੀਕੇਸ਼ਨ ਲੌਜਿਕ ਵਿੱਚ ਨਿਰਵਘਿਨ ਤਰੀਕੇ ਨਾਲ ਏਕੀਕ੍ਰਤਿ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਵਿਚਾਰ ਇਹ ਹੈ ਕਿ ਐਲ.ਐਲ.ਐਮ. ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਜਾਣਬੁੱਝ ਕੇ ਮਨੁੱਖੀ ਰੂਪ ਦਿੱਤਾ ਜਾਵੇ, ਤਾਂ ਜੋ ਕੋਈ ਵੀ ਬਹੁਤ ਜ਼ਿਆਦਾ ਉਤੇਜਿਤ ਨਾ ਹੋਵੇ ਅਤੇ ਉਹਨਾਂ ਨੂੰ ਜਾਦੂਈ ਗੁਣ ਨਾ ਦੇਵੇ ਜੋ ਉਹਨਾਂ ਕੋਲ ਨਹੀਂ ਹਨ।

ਕੇਵਲ ਗੁੰਝਲਦਾਰ ਐਲਗੋਰਿਦਮ ਜਾਂ ਸਮਾਂ-ਖਪਤ ਕਰਨ ਵਾਲੇ ਮੈਨੂਅਲ ਕਾਰਜਾਂ 'ਤੇ ਨਿਰਭਰ ਕਰਨ ਦੀ ਬਜਾਏ, ਡਿਵੈਲਪਰ ਏ.ਆਈ. ਕੰਪੋਨੈਂਟਸ ਨੂੰ ਬੁੱਧੀਮਾਨ, ਸਮਰਪਿਤ, ਮਨੁੱਖ-ਵਰਗੀਆਂ ਇਕਾਈਆਂ ਵਜੋਂ ਸੋਚ ਸਕਦੇ ਹਨ ਜਿਨ੍ਹਾਂ ਨੂੰ ਲੋੜ ਪੈਣ 'ਤੇ ਗੁੰਝਲਦਾਰ ਸਮੱਸਿਆਵਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਅਤੇ ਆਪਣੀ ਟ੍ਰੇਨਿੰਗ ਅਤੇ ਗਿਆਨ ਦੇ ਆਧਾਰ 'ਤੇ ਹੱਲ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਬੁਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਇਕਾਈਆਂ ਨਾ ਤਾਂ ਧਿਆਨ ਭਟਕਾਉਂਦੀਆਂ ਹਨ, ਅਤੇ ਨਾ ਹੀ ਬਿਮਾਰ ਹੋਣ ਦਾ ਬਹਾਨਾ ਬਣਾਉਂਦੀਆਂ ਹਨ। ਉਹ ਅਚਾਨਕ ਚੀਜ਼ਾਂ ਨੂੰ ਵੱਖਰੇ ਤਰੀਕਿਆਂ ਨਾਲ ਕਰਨ ਦਾ ਫੈਸਲਾ ਨਹੀਂ ਕਰਦੀਆਂ ਜਦਿੰ ਉਹਨਾਂ ਨੂੰ ਕਰਨ ਲਈ ਨਿਰਦੇਸ਼ ਦਿੱਤੇ ਗਏ ਹਨ, ਅਤੇ ਆਮ ਤੌਰ 'ਤੇ, ਜੇਕਰ ਸਹੀ ਢੰਗ ਨਾਲ ਪ੍ਰੋਗਰਾਮ ਕੀਤਾ ਗਿਆ ਹੈ, ਤਾਂ ਉਹ ਗਲਤੀਆਂ ਵੀ ਨਹੀਂ ਕਰਦੀਆਂ।

ਤਕਨੀਕੀ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਇਸ ਪਹੁੰਚ ਦਾ ਮੁੱਖ ਸਧਾਂਤ ਗੁੰਝਲਦਾਰ ਕੰਮਾਂ ਜਾਂ ਫੈਸਲਾ ਲੈਣ ਦੀਆਂ ਪ੍ਰਕਰਿਆਵਾਂ ਨੂੰ ਛੋਟੀਆਂ, ਵਧੇਰੇ ਪ੍ਰਬੰਧਨਯੋਗ ਇਕਾਈਆਂ ਵਿੱਚ ਵੰਡਣਾ ਹੈ ਜੋ ਵਸ਼ਿਸ਼ ਏ.ਆਈ. ਕਾਮਿਆਂ ਦੁਆਰਾ ਨਪਿਟਾਈਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ। ਹਰ ਕਾਮਾ ਸਮੱਸਿਆ ਦੇ ਇੱਕ ਖਾਸ ਪਹਲੂ ’ਤੇ ਧਿਆਨ ਕੇਂਦਰਤਿ ਕਰਨ ਲਈ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ, ਜੋ ਆਪਣੀ ਵਲਿੱਖਣ ਮੁਹਾਰਤ ਅਤੇ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਸਾਹਮਣੇ ਲਿਆਉਂਦਾ ਹੈ। ਕਈ ਏ.ਆਈ. ਕਾਮਿਆਂ ਵਿੱਚ ਕੰਮ ਦਾ ਭਾਰ ਵੰਡ ਕੇ, ਐਪਲੀਕੇਸ਼ਨ ਵਧੇਰੇ ਕੁਸ਼ਲਤਾ, ਸਕੇਲੇਬਲਿਟੀ, ਅਤੇ ਅਨੁਕੂਲਤਾ ਪ੍ਰਾਪਤ ਕਰ ਸਕਦੀ ਹੈ।

ਉਦਾਹਰਣ ਵਜੋਂ, ਇੱਕ ਵੈੱਬ ਐਪਲੀਕੇਸ਼ਨ ’ਤੇ ਵਚਿਾਰ ਕਰੋ ਜਸਿ ਨੂੰ ਯੂਜ਼ਰ-ਜਨਰੇਟਡ ਕੰਟੈਂਟ ਦੀ ਰੀਅਲ-ਟਾਈਮ ਮੌਡਰੇਸ਼ਨ ਦੀ ਲੋੜ ਹੈ। ਸ਼ੁਰੂ ਤੋਂ ਇੱਕ ਵਿਆਪਕ ਮੌਡਰੇਸ਼ਨ ਸਸਿਟਮ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਇੱਕ ਚੁਣੌਤੀਪੂਰਨ ਕੰਮ ਹੋਵੇਗਾ, ਜਸਿ ਲਈ ਮਹੱਤਵਪੂਰਨ ਵਕਿਾਸ ਯਤਨਾਂ ਅਤੇ ਨਰਿੰਤਰ ਰੱਖ-ਰਖਾਵ ਦੀ ਲੋੜ ਹੋਵੇਗੀ। ਹਾਲਾਂਕਿ, ਕਾਮਿਆਂ ਦੀ ਭੀੜ ਪਹੁੰਚ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ, ਡਵਿੈਲਪਰ ਐਪਲੀਕੇਸ਼ਨ ਲੌਜਕਿ ਵਿੱਚ ਏ.ਆਈ.-ਪਾਵਰਡ ਮੌਡਰੇਸ਼ਨ ਕਾਮਿਆਂ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਕਾਮੇ ਸਵੈਚਾਲਤਿ ਤੌਰ ’ਤੇ ਅਣਉਚਤਿ ਸਮੱਗਰੀ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਅਤੇ ਫਲੈਗ ਕਰ ਸਕਦੇ ਹਨ, ਜਸਿ ਨਾਲ ਡਵਿੈਲਪਰਾਂ ਨੂੰ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਹੋਰ ਮਹੱਤਵਪੂਰਨ ਪਹਲੂਆਂ ’ਤੇ ਧਿਆਨ ਕੇਂਦਰਤਿ ਕਰਨ ਦੀ ਆਜ਼ਾਦੀ ਮਲਿਦੀ ਹੈ।

ਸੁਤੰਤਰ ਮੁੜ-ਵਰਤੋਂਯੋਗ ਕੰਪੋਨੈਂਟਸ ਵਜੋਂ ਏ.ਆਈ. ਕਾਮੇ

ਕਾਮਿਆਂ ਦੀ ਭੀੜ ਪਹੁੰਚ ਦਾ ਇੱਕ ਮੁੱਖ ਪਹਲੂ ਇਸਦੀ ਮੋਡੀਊਲੈਰਟਿੀ ਹੈ। ਔਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੇ ਸਮਰਥਕ ਸਾਨੂੰ ਦਹਾਕਿਆਂ ਤੋਂ ਦੱਸ ਰਹੇ ਹਨ ਕ ਿਔਬਜੈਕਟ ਇੰਟਰੈਕਸ਼ਨਾਂ ਬਾਰੇ ਸੁਨੇਹਿਆਂ ਵਜੋਂ ਸੋਚੋ। ਖੈਰ, ਏ.ਆਈ. ਕਾਮਿਆਂ ਨੂੰ ਸੁਤੰਤਰ, ਮੁੜ-ਵਰਤੋਂਯੋਗ ਕੰਪੋਨੈਂਟਸ ਵਜੋਂ ਤਿਆਰ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਸਧਾਰਨ ਭਾਸ਼ਾ ਦੇ ਸੁਨੇਹਿਆਂ ਰਾਹੀਂ “ਇੱਕ ਦੂਜੇ ਨਾਲ ਗੱਲ ਕਰ” ਸਕਦੇ ਹਨ, ਲਗਭਗ ਜਵਿੇਂ ਉਹ ਅਸਲ ਵਿੱਚ ਛੋਟੇ ਮਨੁੱਖ ਹੋਣ ਜੋ ਇੱਕ ਦੂਜੇ ਨਾਲ ਗੱਲ ਕਰ ਰਹੇ ਹਨ। ਇਹ ਢੱਲੀ-ਜੁੜੀ ਪਹੁੰਚ ਐਪਲੀਕੇਸ਼ਨ ਨੂੰ ਸਮੇਂ ਦੇ ਨਾਲ ਅਨੁਕੂਲ ਹੋਣ ਅਤੇ ਵਕਿਸਤ ਹੋਣ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦੀ ਹੈ, ਜਵਿੇਂ-ਜਵਿੇਂ ਨਵੀਆਂ ਏ.ਆਈ. ਤਕਨਾਲੋਜੀਆਂ ਸਾਹਮਣੇ ਆਉਂਦੀਆਂ ਹਨ ਜਾਂ ਵਪਾਰਕ ਲੌਜਕਿ ਦੀਆਂ ਲੋੜਾਂ ਬਦਲਦੀਆਂ ਹਨ।

ਅਮਲੀ ਤੌਰ ’ਤੇ, ਕੰਪੋਨੈਂਟਸ ਵਚਿਕਾਰ ਸਪਸ਼ਟ ਇੰਟਰਫੇਸ ਅਤੇ ਸੰਚਾਰ ਪ੍ਰੋਟੋਕੋਲ ਡਜ਼ਿਾਈਨ ਕਰਨ ਦੀ ਲੋੜ ਨਹੀਂ ਬਦਲੀ ਹੈ, ਭਾਵੇਂ ਏ.ਆਈ. ਵਰਕਰ ਸ਼ਾਮਲ ਹਨ। ਤੁਹਾਨੂੰ ਹਾਲੇ ਵੀ ਪਰਫਾਰਮੈਂਸ, ਸਕੇਲੇਬਲਿਟੀ, ਅਤੇ ਸੁਰੱਖਿਆ ਵਰਗੇ ਹੋਰ ਕਾਰਕਾਂ ’ਤੇ ਵੀ ਵਚਿਾਰ ਕਰਨਾ ਪਵੇਗਾ, ਪਰ ਹੁਣ ਨਵੀਆਂ “ਨਰਮ ਲੋੜਾਂ” ’ਤੇ ਵੀ ਵਚਿਾਰ ਕਰਨਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ, ਬਹੁਤ ਸਾਰੇ ਯੂਜ਼ਰ ਆਪਣੇ ਨਜ਼ੀਂ ਡੇਟਾ ਨੂੰ ਨਵੇਂ ਏ.ਆਈ. ਮਾਡਲਾਂ ਨੂੰ ਸਖਿਲਾਈ ਦੇਣ ਲਈ ਵਰਤੇ ਜਾਣ ਦਾ ਵਰਿੋਧ ਕਰਦੇ ਹਨ। ਕੀ ਤੁਸੀਂ ਆਪਣੇ ਵਰਤੇ ਜਾ ਰਹੇ ਮਾਡਲ ਪ੍ਰਦਾਤਾ ਦੁਆਰਾ ਪ੍ਰਦਾਨ ਕੀਤੀ ਗਈ

ਗੋਪਨੀਯਤਾ ਦੇ ਪੱਧਰ ਦੀ ਪੁਸ਼ਟੀ ਕੀਤੀ ਹੈ?

ਕੀ ਏ.ਆਈ. ਵਰਕਰ ਮਾਈਕਰੋਸਰਵਿਸਿਜ਼ ਵਾਂਗ ਹਨ?

ਜਦੋਂ ਤੁਸੀ ਵਰਕਰਾਂ ਦੀ ਬਹੁਤਾਤ ਪਹੁੰਚ ਬਾਰੇ ਪੜ੍ਹਦੇ ਹੋ, ਤਾਂ ਤੁਸੀ ਮਾਈਕਰੋਸਰਵਿਸਿਜ਼ ਆਰਕੀਟੈਕਚਰ ਨਾਲ ਕੁਝ ਸਮਾਨਤਾਵਾਂ ਦੇਖ ਸਕਦੇ ਹੋ। ਦੋਵੇਂ ਗੁੰਝਲਦਾਰ ਸਿਸਟਮਾਂ ਨੂੰ ਛੋਟੇ, ਵਧੇਰੇ ਪ੍ਰਬੰਧਨਯੋਗ, ਅਤੇ ਸੁਤੰਤਰ ਰੂਪ ਵਿੱਚ ਤੈਨਾਤ ਕਰਨ ਯੋਗ ਇਕਾਈਆਂ ਵਿੱਚ ਵੰਡਣ 'ਤੇ ਜ਼ੋਰ ਦਿੰਦੇ ਹਨ। ਜਦਿਂ ਕਿ ਮਾਈਕਰੋਸਰਵਿਸਿਜ਼ ਨੂੰ ਢੀਲੇ ਤੌਰ 'ਤੇ ਜੁੜੇ ਹੋਏ, ਵਸਿ਼ਸ਼ ਵਪਾਰਕ ਸਮਰੱਥਾਵਾਂ 'ਤੇ ਕੇਂਦ੍ਰਤਿ, ਅਤੇ ਚੰਗੀ ਤਰ੍ਹਾਂ ਪਰਭਾਸ਼ਤਿ APIs ਰਾਹੀ ਸੰਚਾਰ ਕਰਨ ਲਈ ਡਜ਼ਿਾਈਨ ਕੀਤਾ ਗਿਆ ਹੈ, ਏ.ਆਈ. ਵਰਕਰਾਂ ਨੂੰ ਮੋਡੀਊਲਰ, ਆਪਣੇ ਕੰਮਾਂ ਵਿੱਚ ਮਾਹਰਿ, ਅਤੇ ਸਪੱਸ਼ਟ ਇੰਟਰਫੇਸ ਅਤੇ ਸੰਚਾਰ ਪ੍ਰੋਟੋਕੋਲ ਰਾਹੀ ਇੱਕ ਦੂਜੇ ਨਾਲ ਅੰਤਰਕਰਿਆ ਕਰਨ ਲਈ ਡਜ਼ਿਾਈਨ ਕੀਤਾ ਗਿਆ ਹੈ।

ਹਾਲਾਂਕਿ, ਕੁਝ ਮੁੱਖ ਅੰਤਰ ਹਨ ਜਨ੍ਹਿਾਂ ਨੂੰ ਧਿਆਨ ਵਿੱਚ ਰੱਖਣਾ ਚਾਹੀਦਾ ਹੈ। ਜੱਥੇ ਮਾਈਕਰੋਸਰਵਿਸਿਜ਼ ਨੂੰ ਆਮ ਤੌਰ 'ਤੇ ਵੱਖ-ਵੱਖ ਮਸ਼ੀਨਾਂ ਜਾਂ ਕੰਟੇਨਰਾਂ 'ਤੇ ਚੱਲ ਰਹੀਆਂ ਵੱਖਰੀਆਂ ਪ੍ਰਕਰਿਆਵਾਂ ਜਾਂ ਸੇਵਾਵਾਂ ਵਜੋਂ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਏ.ਆਈ. ਵਰਕਰਾਂ ਨੂੰ ਤੁਹਾਡੀਆਂ ਵਸਿ਼ਸ਼ ਲੋੜਾਂ ਅਤੇ ਸਕੇਲੇਬਲਿਟੀ ਦੀਆਂ ਜ਼ਰੂਰਤਾਂ 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹੋਏ, ਇੱਕ ਸਿੰਗਲ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਅੰਦਰ ਸਟੈਂਡਅਲੋਨ ਕੰਪੋਨੈਂਟਸ ਜਾਂ ਵੱਖਰੀਆਂ ਸੇਵਾਵਾਂ ਵਜੋਂ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਸ ਤੋਂ ਇਲਾਵਾ, ਏ.ਆਈ. ਵਰਕਰਾਂ ਵਚਿਕਾਰ ਸੰਚਾਰ ਵਿੱਚ ਅਕਸਰ ਪ੍ਰੌਂਪਟਸ, ਨਰਿਦੇਸ਼, ਅਤੇ ਜਨਰੇਟ ਕੀਤੀ ਸਮੱਗਰੀ ਵਰਗੀ ਅਮੀਰ, ਕੁਦਰਤੀ ਭਾਸ਼ਾ-ਆਧਾਰਤਿ ਜਾਣਕਾਰੀ ਦਾ ਆਦਾਨ-ਪ੍ਰਦਾਨ ਸ਼ਾਮਲ ਹੁੰਦਾ ਹੈ, ਨਾ ਕਿ ਮਾਈਕਰੋਸਰਵਿਸਿਜ਼ ਵਿੱਚ ਆਮ ਤੌਰ 'ਤੇ ਵਰਤੇ ਜਾਂਦੇ ਵਧੇਰੇ ਸੰਰਚਤਿ ਡੇਟਾ ਫਾਰਮੈਟ।

ਇਨ੍ਹਾਂ ਅੰਤਰਾਂ ਦੇ ਬਾਵਜੂਦ, ਮੋਡੂਲੈਰਟਿੀ, ਢੀਲੇ ਕਪਲਿੰਗ, ਅਤੇ ਸਪੱਸ਼ਟ ਸੰਚਾਰ ਇੰਟਰਫੇਸ ਦੇ ਸਧਿਾਂਤ ਦੋਵਾਂ ਪੈਟਰਨਾਂ ਲਈ ਕੇਂਦਰੀ ਬਣੇ ਰਹਿੰਦੇ ਹਨ। ਇਨ੍ਹਾਂ ਸਧਿਾਂਤਾਂ ਨੂੰ ਆਪਣੀ ਏ.ਆਈ. ਵਰਕਰ ਆਰਕੀਟੈਕਚਰ 'ਤੇ ਲਾਗੂ ਕਰਕੇ, ਤੁਸੀ ਲਚਕਦਾਰ, ਸਕੇਲੇਬਲ, ਅਤੇ ਰੱਖ-ਰਖਾਵ ਯੋਗ ਸਿਸਟਮ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਗੁੰਝਲਦਾਰ ਸਮੱਸਿਆਵਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਅਤੇ ਆਪਣੇ ਯੂਜ਼ਰਾਂ ਨੂੰ ਮੁੱਲ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਏ.ਆਈ. ਦੀ ਸ਼ਕਤੀ ਦਾ ਲਾਭ ਲੈਂਦੇ ਹਨ।

ਵਰਕਰਾਂ ਦੀ ਬਹੁਤਾਤ ਪਹੁੰਚ ਨੂੰ ਵੱਖ-ਵੱਖ ਡੋਮੇਨਾਂ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਗੁੰਝਲਦਾਰ ਕੰਮਾਂ ਨੂੰ ਨਪਿਟਾਉਣ ਅਤੇ ਬੁੱਧੀਮਾਨ ਹੱਲ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਏ.ਆਈ. ਦੀ ਸ਼ਕਤੀ ਦਾ ਲਾਭ ਲੈਂਦੇ ਹੋਏ। ਆਓ

ਵੱਖ-ਵੱਖ ਸੰਦਰਭਾਂ ਵਿੱਚ ਏ.ਆਈ. ਵਰਕਰਾਂ ਦੀ ਵਰਤੋਂ ਦੇ ਕੁਝ ਠੋਸ ਉਦਾਹਰਨਾਂ ਦੀ ਪੜਚੋਲ ਕਰੀਏ।

ਅਕਾਊਂਟ ਪ੍ਰਬੰਧਨ

ਲਗਭਗ ਹਰ ਸਟੈਂਡਅਲੋਨ ਵੈੱਬ ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ ਅਕਾਊਂਟ (ਜਾਂ ਯੂਜ਼ਰ) ਦੀ ਧਾਰਨਾ ਹੁੰਦੀ ਹੈ। Olympia ਵਿੱਚ, ਅਸੀਂ ਇੱਕ `AccountManager` ਏ.ਆਈ. ਵਰਕਰ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ ਜੋ ਯੂਜ਼ਰ ਅਕਾਊਂਟਾਂ ਨਾਲ ਸੰਬੰਧਤਿ ਵੱਖ-ਵੱਖ ਕਸਿਮਾਂ ਦੀਆਂ ਤਬਦੀਲੀ ਬੇਨਤੀਆਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਪ੍ਰੋਗਰਾਮ ਕੀਤਾ ਗਿਆ ਹੈ।

ਇਸਦਾ ਨਰਿਦੇਸ਼ ਇਸ ਤਰ੍ਹਾਂ ਪੜ੍ਹਦਾ ਹੈ:

```
You are an intelligent account manager for Olympia. The user will request
changes to their account, and you will process those changes by invoking
one or more of the functions provided.

The initial state of the account: #{account.to_directive}

Functions will return a text description of both success and error
results, plus guidance about how to proceed (if applicable). If you have
a question about Olympia policies you may use the `search_kb` function
to search our knowledge base.

Make sure to notify the account owner of the result of the change
request before calling the `finished` function so that we save the state
of the account change request as completed.
```

`account.to_directive` ਦੁਆਰਾ ਤਿਆਰ ਕੀਤੀ ਗਈ ਖਾਤੇ ਦੀ ਸ਼ੁਰੂਆਤੀ ਸਥਤਿੀ ਸਰਿਫ਼ ਖਾਤੇ ਦਾ ਟੈਕਸਟ ਵੇਰਵਾ ਹੈ, ਜਸਿ ਵਿੱਚ ਉਪਯੋਗਕਰਤਾਵਾਂ, ਸਬਸਕ੍ਰਪਿਸ਼ਨਾਂ ਆਦ ਿਵਰਗੇ ਸੰਬੰਧਤਿ ਡੇਟਾ ਸ਼ਾਮਲ ਹਨ।

`AccountManager` ਲਈ ਉਪਲਬਧ ਫੰਕਸ਼ਨਾਂ ਦੀ ਰੇਂਜ ਇਸਨੂੰ ਉਪਯੋਗਕਰਤਾ ਦੀ ਸਬਸਕ੍ਰਪਿਸ਼ਨ ਨੂੰ ਸੰਪਾਦਤਿ ਕਰਨ, ਏਆਈ ਸਲਾਹਕਾਰ ਅਤੇ ਹੋਰ ਕਸਿਮਾਂ ਦੇ ਭੁਗਤਾਨ ਐਡ-ਔਨ ਜੋੜਨ ਅਤੇ ਹਟਾਉਣ, ਅਤੇ ਖਾਤਾ ਮਾਲਕ ਨੂੰ ਸੂਚਨਾ ਈਮੇਲਾਂ ਭੇਜਣ ਦੀ ਯੋਗਤਾ ਦੇਂਦੀ ਹੈ। `finished` ਫੰਕਸ਼ਨ ਦੇ ਇਲਾਵਾ, ਇਹ `notify_human_administrator` ਵੀ ਕਰ ਸਕਦਾ ਹੈ ਜੇਕਰ ਇਹ ਆਪਣੀ ਪ੍ਰੋਸੈਸਿੰਗ ਦੌਰਾਨ ਕਸਿੇ ਤਰੁੱਟੀ ਦਾ ਸਾਹਮਣਾ ਕਰਦਾ ਹੈ ਜਾਂ ਕਸਿੇ ਬੇਨਤੀ ਨਾਲ ਕਸਿੇ ਹੋਰ ਕਸਿਮ ਦੀ ਸਹਾਇਤਾ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

ਧਿਆਨ ਦਿਓ ਕਿ ਸਵਾਲਾਂ ਦੀ ਸਥਿਤੀ ਵਿੱਚ, `AccountManager` Olympia ਦੇ ਗਿਆਨ ਅਧਾਰ ਵਿੱਚ ਖੋਜ ਕਰ ਸਕਦਾ ਹੈ, ਜਿੱਥੇ ਇਹ ਸੀਮਾ ਮਾਮਲਿਆਂ ਨੂੰ ਸੰਭਾਲਣ ਅਤੇ ਕਿਸੇ ਹੋਰ ਸਥਿਤੀ ਜਿਸ ਵਿੱਚ ਇਹ ਅੱਗੇ ਵਧਣ ਬਾਰੇ ਅਨਿਸ਼ਚਿਤ ਹੈ, ਬਾਰੇ ਨਿਰਦੇਸ਼ ਲੱਭ ਸਕਦਾ ਹੈ।

ਈ-ਕਾਮਰਸ ਐਪਲੀਕੇਸ਼ਨਾਂ

ਈ-ਕਾਮਰਸ ਦੇ ਖੇਤਰ ਵਿੱਚ, ਏਆਈ ਵਿਕਰਵ ਉਪਯੋਗਕਰਤਾ ਅਨੁਭਵ ਨੂੰ ਵਧਾਉਣ ਅਤੇ ਵਪਾਰਕ ਕਾਰਜਾਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣ ਵਿੱਚ ਮਹੱਤਵਪੂਰਨ ਭੂਮਿਕਾ ਨਿਭਾ ਸਕਦੇ ਹਨ। ਇੱਥੇ ਕੁਝ ਤਰੀਕੇ ਹਨ ਜਿਨ੍ਹਾਂ ਨਾਲ ਏਆਈ ਵਿਕਰਵਾਂ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ:

ਉਤਪਾਦ ਸਿਫਾਰਸ਼ਾਂ

ਈ-ਕਾਮਰਸ ਵਿੱਚ ਏਆਈ ਵਿਕਰਵਾਂ ਦੀਆਂ ਸਭ ਤੋਂ ਸ਼ਕਤੀਸ਼ਾਲੀ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚੋਂ ਇੱਕ ਨਿੱਜੀਕਰਣ ਕੀਤੀਆਂ ਉਤਪਾਦ ਸਿਫਾਰਸ਼ਾਂ ਤਿਆਰ ਕਰਨਾ ਹੈ। ਉਪਯੋਗਕਰਤਾ ਦੇ ਵਿਵਹਾਰ, ਖਰੀਦ ਇਤਿਹਾਸ, ਅਤੇ ਤਰਜੀਹਾਂ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਕੇ, ਇਹ ਵਿਕਰਵ ਹਰੇਕ ਵਿਅਕਤੀਗਤ ਉਪਯੋਗਕਰਤਾ ਦੀਆਂ ਰੁਚੀਆਂ ਅਤੇ ਲੋੜਾਂ ਦੇ ਅਨੁਸਾਰ ਉਤਪਾਦਾਂ ਦੀ ਸਿਫਾਰਸ਼ ਕਰ ਸਕਦੇ ਹਨ।

ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਉਤਪਾਦ ਸਿਫਾਰਸ਼ਾਂ ਦੀ ਕੁੰਜੀ ਸਹਿਯੋਗੀ ਫਿਲਟਰਿੰਗ ਅਤੇ ਸਮੱਗਰੀ-ਆਧਾਰਿਤ ਫਿਲਟਰਿੰਗ ਤਕਨੀਕਾਂ ਦੇ ਸੁਮੇਲ ਦਾ ਲਾਭ ਉਠਾਉਣਾ ਹੈ। ਸਹਿਯੋਗੀ ਫਿਲਟਰਿੰਗ ਸਮਾਨ ਉਪਯੋਗਕਰਤਾਵਾਂ ਦੇ ਵਿਵਹਾਰ ਨੂੰ ਦੇਖਦੀ ਹੈ ਤਾਂ ਜੋ ਪੈਟਰਨਾਂ ਦੀ ਪਛਾਣ ਕੀਤੀ ਜਾ ਸਕੇ ਅਤੇ ਉਸ ਦੇ ਆਧਾਰ 'ਤੇ ਸਿਫਾਰਸ਼ਾਂ ਕੀਤੀਆਂ ਜਾ ਸਕਣ ਜੋ ਸਮਾਨ ਪਸੰਦ ਵਾਲੇ ਹੋਰਾਂ ਨੇ ਖਰੀਦਿਆ ਜਾਂ ਪਸੰਦ ਕੀਤਾ ਹੈ। ਦੂਜੇ ਪਾਸੇ, ਸਮੱਗਰੀ-ਆਧਾਰਿਤ ਫਿਲਟਰਿੰਗ ਉਤਪਾਦਾਂ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਅਤੇ ਗੁਣਾਂ 'ਤੇ ਕੇਂਦਰਿਤ ਹੁੰਦੀ ਹੈ, ਉਨ੍ਹਾਂ ਆਈਟਮਾਂ ਦੀ ਸਿਫਾਰਸ਼ ਕਰਦੀ ਹੈ ਜੋ ਉਨ੍ਹਾਂ ਵਸਤੂਆਂ ਨਾਲ ਸਮਾਨ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਸਾਂਝੀਆਂ ਕਰਦੀਆਂ ਹਨ ਜਿਨ੍ਹਾਂ ਵਿੱਚ ਉਪਯੋਗਕਰਤਾ ਨੇ ਪਹਿਲਾਂ ਦਿਲਚਸਪੀ ਦਿਖਾਈ ਹੈ।

ਇੱਥੇ Ruby ਵਿੱਚ ਇੱਕ ਉਤਪਾਦ ਸਿਫਾਰਸ਼ ਵਿਕਰਵ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਇੱਕ ਸਰਲ ਉਦਾਹਰਣ ਹੈ, ਇਸ ਵਾਰ "Railway Oriented (ROP)" ਫੰਕਸ਼ਨਲ ਪ੍ਰੋਗਰਾਮਿੰਗ ਸ਼ੈਲੀ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ:

```ruby
class ProductRecommendationWorker
  include Wisper::Publisher

  def call(user)
    Result.ok(ProductRecommendation.new(user))
      .and_then(ValidateUser.method(:validate))
      .map(AnalyzeCurrentSession.method(:analyze))
      .map(CollaborativeFilter.method(:filter))
      .map(ContentBasedFilter.method(:filter))
      .map(ProductSelector.method(:select)).then do |result|

      case result
      in { err: ProductRecommendationError => error }
        Honeybadger.notify(error.message, context: {user:})
      in { ok: ProductRecommendations => recs }
        broadcast(:new_recommendations, user:, recs:)
      end
    end
  end
end
```

Ruby ਦੀ ਫੰਕਸ਼ਨਲ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੀ ਸ਼ੈਲੀ ਜੋ ਉਦਾਹਰਣ ਵਿੱਚ ਵਰਤੀ ਗਈ ਹੈ, F# ਅਤੇ Rust ਤੋਂ ਪ੍ਰਭਾਵਿਤ ਹੈ। ਤੁਸੀਂ ਇਸ ਤਕਨੀਕ ਬਾਰੇ ਮੇਰੇ ਦੋਸਤ Chad Wooley ਦੀ GitLab 'ਤੇ ਵਿਆਖਿਆ ਵਿੱਚ ਹੋਰ ਪੜ੍ਹ ਸਕਦੇ ਹੋ।

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ, `ProductRecommendationWorker` ਇੱਕ ਯੂਜ਼ਰ ਨੂੰ ਇਨਪੁੱਟ ਵਜੋਂ ਲੈਂਦਾ ਹੈ ਅਤੇ ਫੰਕਸ਼ਨਲ ਸਟੈਪਸ ਦੀ ਲੜੀ ਵਿੱਚ ਇੱਕ ਵੈਲਯੂ ਔਬਜੈਕਟ ਪਾਸ ਕਰਕੇ ਵਿਅਕਤੀਗਤ ਉਤਪਾਦ ਸਫਿਾਰਸ਼ਾਂ ਤਿਆਰ ਕਰਦਾ ਹੈ। ਆਓ ਹਰ ਸਟੈਪ ਨੂੰ ਸਮਝੀਏ:

1. `ValidateUser.validate`: ਇਹ ਸਟੈਪ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕਿ ਯੂਜ਼ਰ ਵੈਧ ਹੈ ਅਤੇ ਵਿਅਕਤੀਗਤ ਸਫਿਾਰਸ਼ਾਂ ਲਈ ਯੋਗ ਹੈ। ਇਹ ਚੈੱਕ ਕਰਦਾ ਹੈ ਕਿ ਯੂਜ਼ਰ ਮੌਜੂਦ ਹੈ, ਐਕਟਿਵ ਹੈ, ਅਤੇ ਸਫਿਾਰਸ਼ਾਂ ਤਿਆਰ ਕਰਨ ਲਈ ਜ਼ਰੂਰੀ ਡਾਟਾ ਉਪਲਬਧ ਹੈ। ਜੇ ਵੈਲੀਡੇਸ਼ਨ ਫੇਲ੍ਹ ਹੋ ਜਾਂਦੀ ਹੈ, ਤਾਂ ਇੱਕ ਗਲਤੀ ਦਾ ਨਤੀਜਾ ਵਾਪਸ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਅਤੇ ਲੜੀ ਸੌਰਟ-ਸਰਕਟ ਹੋ ਜਾਂਦੀ ਹੈ।
2. `AnalyzeCurrentSession.analyze`: ਜੇ ਯੂਜ਼ਰ ਵੈਧ ਹੈ, ਤਾਂ ਇਹ ਸਟੈਪ ਪ੍ਰਸੰਗਿਕ ਜਾਣਕਾਰੀ ਇਕੱਠੀ ਕਰਨ ਲਈ ਯੂਜ਼ਰ ਦੇ ਮੌਜੂਦਾ ਬਰਾਊਜ਼ਿੰਗ ਸੈਸ਼ਨ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਦਾ ਹੈ। ਇਹ ਯੂਜ਼ਰ ਦੀਆਂ

ਹਾਲੀਆ ਗਤੀਵਧਿੀਆਂ, ਜਵਿੇਂ ਦੇਖੇ ਗਏ ਉਤਪਾਦ, ਖੋਜ ਕੁਐਰੀਆਂ, ਅਤੇ ਕਾਰਟ ਦੀ ਸਮੱਗਰੀ ਨੂੰ ਦੇਖਦਾ ਹੈ, ਤਾਂ ਜੋ ਉਨ੍ਹਾਂ ਦੀਆਂ ਮੌਜੂਦਾ ਰੁਚੀਆਂ ਅਤੇ ਇਰਾਦੇ ਨੂੰ ਸਮਝਿਆ ਜਾ ਸਕੇ।

3. `CollaborativeFilter.filter`: ਸਮਾਨ ਯੂਜ਼ਰਾਂ ਦੇ ਵਵਿਹਾਰ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ, ਇਹ ਸਟੈਪ ਸਹਯਿੋਗੀ ਫਲਿਟਰੰਗ ਤਕਨੀਕਾਂ ਲਾਗੂ ਕਰਦਾ ਹੈ ਤਾਂ ਜੋ ਉਨ੍ਹਾਂ ਉਤਪਾਦਾਂ ਦੀ ਪਛਾਣ ਕੀਤੀ ਜਾ ਸਕੇ ਜੋ ਯੂਜ਼ਰ ਲਈ ਦਲਿਚਸਪੀ ਵਾਲੇ ਹੋ ਸਕਦੇ ਹਨ। ਇਹ ਖਰੀਦ ਇਤਹਿਾਸ, ਰੇਟੰਗਾਂ, ਅਤੇ ਯੂਜ਼ਰ-ਆਈਟਮ ਇੰਟਰੈਕਸ਼ਨਾਂ ਵਰਗੇ ਕਾਰਕਾਂ 'ਤੇ ਵਚਿਾਰ ਕਰਦਾ ਹੈ ਤਾਂ ਜੋ ਸੰਭਾਵੀ ਸਫਿਾਰਸ਼ਾਂ ਦਾ ਸੈੱਟ ਤਆਿਰ ਕੀਤਾ ਜਾ ਸਕੇ।
4. `ContentBasedFilter.filter`: ਇਹ ਸਟੈਪ ਸਮੱਗਰੀ-ਆਧਾਰਤਿ ਫਲਿਟਰੰਗ ਲਾਗੂ ਕਰਕੇ ਸੰਭਾਵੀ ਸਫਿਾਰਸ਼ਾਂ ਨੂੰ ਹੋਰ ਸੁਧਾਰਦਾ ਹੈ। ਇਹ ਸਭ ਤੋਂ ਢੁਕਵੀਆਂ ਆਈਟਮਾਂ ਦੀ ਚੋਣ ਕਰਨ ਲਈ ਸੰਭਾਵੀ ਉਤਪਾਦਾਂ ਦੇ ਗੁਣਾਂ ਅਤੇ ਵਸ਼ਿੇਸ਼ਤਾਵਾਂ ਦੀ ਯੂਜ਼ਰ ਦੀਆਂ ਤਰਜੀਹਾਂ ਅਤੇ ਇਤਹਿਾਸਕ ਡਾਟਾ ਨਾਲ ਤੁਲਨਾ ਕਰਦਾ ਹੈ।
5. `ProductSelector.select`: ਅੰਤ ਵੱਚਿ, ਇਹ ਸਟੈਪ ਪਹਲਿਾਂ ਤੋਂ ਨਰਿਧਾਰਤਿ ਮਾਪਦੰਡਾਂ, ਜਵਿੇਂ ਪ੍ਰਸੰਗਕਿਤਾ ਸਕੋਰ, ਲੋਕਪ੍ਰਯਿਤਾ, ਜਾਂ ਹੋਰ ਵਪਾਰਕ ਨਯਿਮਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਫਲਿਟਰ ਕੀਤੀਆਂ ਸਫਿਾਰਸ਼ਾਂ ਵੱਚਿੋਂ ਸਖਿਰਲੇ N ਉਤਪਾਦਾਂ ਦੀ ਚੋਣ ਕਰਦਾ ਹੈ। ਚੁਣੇ ਗਏ ਉਤਪਾਦ ਫਰਿ ਅੰਤਮਿ ਵਅਿਕਤੀਗਤ ਸਫਿਾਰਸ਼ਾਂ ਵਜੋਂ ਵਾਪਸ ਕੀਤੇ ਜਾਂਦੇ ਹਨ।

ਇੱਥੇ Ruby ਦੀ ਫੰਕਸ਼ਨਲ ਪ੍ਰੋਗਰਾਮੰਗ ਸ਼ੈਲੀ ਦੀ ਵਰਤੋਂ ਦੀ ਖੂਬਸੂਰਤੀ ਇਹ ਹੈ ਕ ਿਇਹ ਸਾਨੂੰ ਇਨ੍ਹਾਂ ਸਟੈਪਸ ਨੂੰ ਸਪਸ਼ਟ ਅਤੇ ਸੰਖੇਪ ਤਰੀਕੇ ਨਾਲ ਇੱਕ ਲੜੀ ਵੱਚਿ ਜੋੜਨ ਦੀ ਆਗਾਿਆ ਦੰਦੀ ਹੈ। ਹਰ ਸਟੈਪ ਇੱਕ ਖਾਸ ਕੰਮ 'ਤੇ ਕੇਂਦਰਤਿ ਹੈ ਅਤੇ ਇੱਕ `Result` ਔਬਜੈਕਟ ਵਾਪਸ ਕਰਦਾ ਹੈ, ਜੋ ਜਾਂ ਤਾਂ ਸਫਲ (`ok`) ਜਾਂ ਗਲਤੀ (`err`) ਹੋ ਸਕਦਾ ਹੈ। ਜੇ ਕਸਿ ਵੀ ਸਟੈਪ ਵੱਚਿ ਗਲਤੀ ਆਉਂਦੀ ਹੈ, ਤਾਂ ਲੜੀ ਸੌਰਟ-ਸਰਕਟ ਹੋ ਜਾਂਦੀ ਹੈ, ਅਤੇ ਗਲਤੀ ਅੰਤਮਿ ਨਤੀਜੇ ਤੱਕ ਪਹੁੰਚ ਜਾਂਦੀ ਹੈ।

`case` ਸਟੇਟਮੈਂਟ ਦੇ ਅੰਤ ਵੱਚਿ, ਅਸੀਂ ਅੰਤਮਿ ਨਤੀਜੇ ਤੇ ਪੈਟਰਨ ਮੈਚੰਗ ਕਰਦੇ ਹਾਂ। ਜੇਕਰ ਨਤੀਜਾ ਇੱਕ ਗਲਤੀ ਹੈ (`ProductRecommendationError`), ਅਸੀਂ Honeybadger ਵਰਗੇ ਟੂਲ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਨਗਿਰਾਨੀ ਅਤੇ ਡੀਬੱਗੰਗ ਦੇ ਉਦੇਸ਼ਾਂ ਲਈ ਗਲਤੀ ਨੂੰ ਲੌਗ ਕਰਦੇ ਹਾਂ। ਜੇਕਰ ਨਤੀਜਾ ਸਫਲ ਹੈ (`ProductRecommendations`), ਅਸੀਂ Wisper ਪਬ/ਸਬ ਲਾਇਬ੍ਰੇਰੀ ਦੀ ਵਰਤੋਂ ਕਰਕੇ `:new_recommendations` ਈਵੈਂਟ ਦਾ ਪ੍ਰਸਾਰਣ ਕਰਦੇ ਹਾਂ, ਜਸਿ ਵੱਚਿ ਯੂਜ਼ਰ ਅਤੇ ਤਆਿਰ ਕੀਤੀਆਂ ਸਫਿਾਰਸ਼ਾਂ ਨੂੰ ਅੱਗੇ ਭੇਜਦੇ ਹਾਂ।

ਫੰਕਸ਼ਨਲ ਪ੍ਰੋਗਰਾਮੰਗ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਅਸੀਂ ਇੱਕ ਮੌਡੀਊਲਰ ਅਤੇ ਰੱਖ-ਰਖਾਵ ਯੋਗ ਉਤਪਾਦ ਸਫਿਾਰਸ਼ ਵਰਕਰ ਬਣਾ ਸਕਦੇ ਹਾਂ। ਹਰ ਕਦਮ ਸਵੈ-ਨਰਿਭਰ ਹੈ ਅਤੇ ਇਸਦੀ ਆਸਾਨੀ ਨਾਲ ਜਾਂਚ, ਸੋਧ, ਜਾਂ ਸਮੁੱਚੇ

ਪ੍ਰਵਾਹ ਨੂੰ ਪ੍ਰਭਾਵਿਤ ਕੀਤੇ ਬਿਨਾਂ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਪੈਟਰਨ ਮੈਚਿੰਗ ਅਤੇ `Result` ਕਲਾਸ ਦੀ ਵਰਤੋਂ ਸਾਨੂੰ ਗਲਤੀਆਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲਣ ਵਿੱਚ ਮਦਦ ਕਰਦੀ ਹੈ ਅਤੇ ਯਕੀਨੀ ਬਣਾਉਂਦੀ ਹੈ ਕਿ ਜੇਕਰ ਕਿਸੇ ਵੀ ਕਦਮ ਵਿੱਚ ਕੋਈ ਸਮੱਸਿਆ ਆਉਂਦੀ ਹੈ ਤਾਂ ਵਰਕਰ ਤੁਰੰਤ ਅਸਫਲ ਹੋ ਜਾਂਦਾ ਹੈ।

ਬੇਸ਼ੱਕ, ਇਹ ਇੱਕ ਸਰਲੀਕ੍ਰਿਤ ਉਦਾਹਰਣ ਹੈ, ਅਤੇ ਅਸਲ ਦੁਨੀਆਂ ਦੇ ਸਥਿਤੀ ਵਿੱਚ, ਤੁਹਾਨੂੰ ਆਪਣੇ ਈ-ਕਾਮਰਸ ਪਲੇਟਫਾਰਮ ਨਾਲ ਏਕੀਕਰਣ ਕਰਨ, ਸੀਮਾ ਮਾਮਲਿਆਂ ਨੂੰ ਸੰਭਾਲਣ, ਅਤੇ ਇੱਥੋਂ ਤੱਕ ਕਿ ਸਿਫਾਰਸ਼ ਐਲਗੋਰਿਦਮ ਦੇ ਕਾਰਜਾਂਵਯਨ ਵਿੱਚ ਵੀ ਜਾਣ ਦੀ ਲੋੜ ਹੋਵੇਗੀ। ਹਾਲਾਂਕਿ, ਸਮੱਸਿਆ ਨੂੰ ਛੋਟੇ ਕਦਮਾਂ ਵਿੱਚ ਵੰਡਣ ਅਤੇ ਫੰਕਸ਼ਨਲ ਪ੍ਰੋਗਰਾਮਿੰਗ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦੇ ਮੂਲ ਸਿਧਾਂਤ ਉਹੀ ਰਹਿੰਦੇ ਹਨ।

ਧੋਖਾਧੜੀ ਦੀ ਪਛਾਣ

ਇੱਥੇ ਇੱਕ ਸਰਲੀਕ੍ਰਿਤ ਉਦਾਹਰਣ ਹੈ ਕਿ ਤੁਸੀਂ Ruby ਵਿੱਚ ਉਸੇ Railway Oriented Programming (ROP) ਸ਼ੈਲੀ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਧੋਖਾਧੜੀ ਦੀ ਪਛਾਣ ਵਰਕਰ ਨੂੰ ਕਿਵੇਂ ਲਾਗੂ ਕਰ ਸਕਦੇ ਹੋ:

```ruby
class FraudDetectionWorker
  include Wisper::Publisher

  def call(transaction)
    Result.ok(FraudDetection.new(transaction))
      .and_then(ValidateTransaction.method(:validate))
      .map(AnalyzeTransactionPatterns.method(:analyze))
      .map(CheckCustomerHistory.method(:check))
      .map(EvaluateRiskFactors.method(:evaluate))
      .map(DetermineFraudProbability.method(:determine)).then do |result|

      case result
      in { err: FraudDetectionError => error }
        Honeybadger.notify(error.message, context: {transaction:})
      in { ok: FraudDetection => fraud } }
        if fraud.high_risk?
          broadcast(:high_risk_transaction, transaction:, fraud:)
        else
          broadcast(:low_risk_transaction, transaction:)
        end
      end
    end
  end
end
```

FraudDetection ਕਲਾਸ ਇੱਕ ਮੁੱਲ ਆਬਜੈਕਟ ਹੈ ਜੋ ਕਸਿ ਦੱਤੇ ਲੈਣ-ਦੇਣ ਲਈ ਧੋਖਾਧੜੀ ਖੋਜ ਦੀ ਸਥਤਿੀ ਨੂੰ ਸਮੇਟਦੀ ਹੈ। ਇਹ ਵੱਖ-ਵੱਖ ਜੋਖਮ ਕਾਰਕਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਲੈਣ-ਦੇਣ ਨਾਲ ਜੁੜੀ ਧੋਖਾਧੜੀ ਦੇ ਜੋਖਮ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਅਤੇ ਮੁਲਾਂਕਣ ਕਰਨ ਦਾ ਇੱਕ ਢਾਂਚਾਗਤ ਤਰੀਕਾ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ।

```
class FraudDetection
  RISK_THRESHOLD = 0.8

  attr_accessor :transaction, :risk_factors

  def initialize(transaction)
    self.transaction = transaction
    self.risk_factors = []
  end

  def add_risk_factor(description:, probability:)
    case { description:, probability: }
    in { description: String => desc, probability: Float => prob }
      risk_factors << { desc => prob }
    else
      raise ArgumentError, "Risk factor arguments should be string and float"
    end
  end

  def high_risk?
    fraud_probability > RISK_THRESHOLD
  end

  private

  def fraud_probability
    risk_factors.values.sum
  end
end
```

FraudDetection ਕਲਾਸ ਵੱਚਿ ਹੇਠ ਲਖਿੀਆਂ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਹਨ:

- transaction: ਧੋਖਾਧੜੀ ਲਈ ਵਸ਼ਿਲੇਸ਼ਣ ਕੀਤੇ ਜਾ ਰਹੇ ਲੈਣ-ਦੇਣ ਦਾ ਹਵਾਲਾ।
- risk_factors: ਇੱਕ ਐਰੇ ਜੋ ਲੈਣ-ਦੇਣ ਨਾਲ ਜੁੜੇ ਜੋਖਮ ਕਾਰਕਾਂ ਨੂੰ ਸਟੋਰ ਕਰਦਾ ਹੈ। ਹਰ ਜੋਖਮ ਕਾਰਕ ਨੂੰ ਇੱਕ ਹੈਸ਼ ਦੇ ਰੂਪ ਵੱਚਿ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ, ਜੱਥੇ ਕੁੰਜੀ ਜੋਖਮ ਕਾਰਕ ਦਾ ਵੇਰਵਾ ਹੈ, ਅਤੇ ਮੁੱਲ

ਉਸ ਜੋਖਮ ਕਾਰਕ ਨਾਲ ਜੁੜੀ ਧੋਖਾਧੜੀ ਦੀ ਸੰਭਾਵਨਾ ਹੈ।

`add_risk_factor` ਮੈਥਡ `risk_factors` ਐਰੇ ਵਿੱਚ ਇੱਕ ਜੋਖਮ ਕਾਰਕ ਜੋੜਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ। ਇਹ ਦੋ ਪੈਰਾਮੀਟਰ ਲੈਂਦਾ ਹੈ: `description`, ਜੋ ਜੋਖਮ ਕਾਰਕ ਦਾ ਵਰਣਨ ਕਰਨ ਵਾਲੀ ਸਟ੍ਰਿੰਗ ਹੈ, ਅਤੇ `probability`, ਜੋ ਉਸ ਜੋਖਮ ਕਾਰਕ ਨਾਲ ਜੁੜੀ ਧੋਖਾਧੜੀ ਦੀ ਸੰਭਾਵਨਾ ਨੂੰ ਦਰਸਾਉਣ ਵਾਲਾ ਫਲੋਟ ਹੈ। ਅਸੀਂ ਸਧਾਰਨ ਕਿਸਮ ਦੀ ਜਾਂਚ ਲਈ `case..in` ਕੰਡੀਸ਼ਨਲ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ।

`high_risk?` ਮੈਥਡ ਜੋ ਚੇਨ ਦੇ ਅੰਤ 'ਤੇ ਚੈੱਕ ਕੀਤਾ ਜਾਵੇਗਾ, ਇੱਕ ਪ੍ਰੈਡੀਕੇਟ ਮੈਥਡ ਹੈ ਜੋ `fraud_probability` (ਸਾਰੇ ਜੋਖਮ ਕਾਰਕਾਂ ਦੀਆਂ ਸੰਭਾਵਨਾਵਾਂ ਨੂੰ ਜੋੜ ਕੇ ਗਣਨਾ ਕੀਤੀ ਗਈ) ਦੀ `RISK_THRESHOLD` ਨਾਲ ਤੁਲਨਾ ਕਰਦਾ ਹੈ।

`FraudDetection` ਕਲਾਸ ਲੈਣ-ਦੇਣ ਲਈ ਧੋਖਾਧੜੀ ਦੀ ਪਛਾਣ ਦਾ ਪ੍ਰਬੰਧਨ ਕਰਨ ਦਾ ਇੱਕ ਸਾਫ਼ ਅਤੇ ਇਨਕੈਪਸੂਲੇਟਡ ਤਰੀਕਾ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ। ਇਹ ਕਈ ਜੋਖਮ ਕਾਰਕ ਜੋੜਨ ਦੀ ਆਗਿਆ ਦਿੰਦੀ ਹੈ, ਹਰ ਇੱਕ ਦੇ ਆਪਣੇ ਵੇਰਵੇ ਅਤੇ ਸੰਭਾਵਨਾ ਦੇ ਨਾਲ, ਅਤੇ ਗਣਨਾ ਕੀਤੀ ਧੋਖਾਧੜੀ ਦੀ ਸੰਭਾਵਨਾ ਦੇ ਆਧਾਰ 'ਤੇ ਇਹ ਨਿਰਧਾਰਤ ਕਰਨ ਲਈ ਇੱਕ ਵਿਧੀ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ ਕਿ ਕੀ ਲੈਣ-ਦੇਣ ਨੂੰ ਉੱਚ-ਜੋਖਮ ਵਾਲਾ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਕਲਾਸ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਇੱਕ ਵੱਡੀ ਧੋਖਾਧੜੀ ਪਛਾਣ ਪ੍ਰਣਾਲੀ ਵਿੱਚ ਏਕੀਕ੍ਰਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਜਿੱਥੇ ਵੱਖ-ਵੱਖ ਭਾਗ ਧੋਖਾਧੜੀ ਵਾਲੇ ਲੈਣ-ਦੇਣ ਦੇ ਜੋਖਮ ਦਾ ਮੁਲਾਂਕਣ ਅਤੇ ਘਟਾਉਣ ਲਈ ਸਹਿਯੋਗ ਕਰ ਸਕਦੇ ਹਨ।

ਅੰਤ ਵਿੱਚ, ਕਿਉਂਕਿ ਇਹ ਆਖਰਿਕਾਰ AI ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪ੍ਰੋਗਰਾਮਿੰਗ ਬਾਰੇ ਇੱਕ ਕਿਤਾਬ ਹੈ, ਇੱਥੇ ਮੇਰੀ Raix ਲਾਇਬ੍ਰੇਰੀ ਦੇ `ChatCompletion` ਮੋਡਿਊਲ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ `CheckCustomerHistory` ਕਲਾਸ ਦੀ ਇੱਕ ਉਦਾਹਰਨ ਕਾਰਜਾਨਵੀਨ ਹੈ:

```
class CheckCustomerHistory
  include Raix::ChatCompletion

  attr_accessor :fraud_detection

  INSTRUCTION = <<~END
    You are an AI assistant tasked with checking a customer's transaction
    history for potential fraud indicators. Given the current transaction
    and the customer's past transactions, analyze the data to identify any
    suspicious patterns or anomalies.

    Consider factors such as the frequency of transactions, transaction
    amounts, geographical locations, and any deviations from the customer's
    typical behavior to generate a probability score as a float in the range
```

```
    of 0 to 1 (with 1 being absolute certainty of fraud).

    Output the results of your analysis, highlighting any red flags or areas
    of concern in the following JSON format:

    { description: <Summary of your findings>, probability: <Float> }
  END

  def self.check(fraud_detection)
    new(fraud_detection).call
  end

  def call
    chat_completion(json: true).tap do |result|
      fraud_detection.add_risk_factor(**result)
    end
    Result.ok(fraud_detection)
  rescue StandardError => e
    Result.err(FraudDetectionError.new(e))
  end

  private

  def initialize(fraud_detection)
    self.fraud_detection = fraud_detection
  end

  def transcript
    tx_history = fraud_detection.transaction.user.tx_history
    [
      { system: INSTRUCTION },
      { user: "Transaction history: #{tx_history.to_json}" },
      { assistant: "OK. Please provide the current transaction." },
      { user: "Current transaction: #{fraud_detection.transaction.to_json}" }
    ]
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵੱਚਿ, `CheckCustomerHistory` ਇੱਕ `INSTRUCTION` ਸਥਰਿ ਅੰਕ ਨੂੰ ਪਰਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ ਜੋ ਏ.ਆਈ. ਮਾਡਲ ਨੂੰ ਸਸਿਟਮ ਨਰਿਦੇਸ਼ ਰਾਹੀ ਧੋਖਾਧੜੀ ਦੇ ਸੰਕੇਤਾਂ ਲਈ ਗਾਹਕ ਦੇ ਲੈਣ-ਦੇਣ ਦੇ ਇਤਹਿਾਸ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਨ ਲਈ ਵਸ਼ਿੇਸ਼ ਨਰਿਦੇਸ਼ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।

`self.check` ਵਿਧੀ ਇੱਕ ਕਲਾਸ ਵਿਧੀ ਹੈ ਜੋ `fraud_detection` ਵਸਤੂ ਨਾਲ `CheckCustomerHistory` ਦੀ ਇੱਕ ਨਵੀਂ ਉਦਾਹਰਣ ਨੂੰ ਸ਼ੁਰੂ ਕਰਦੀ ਹੈ ਅਤੇ ਗਾਹਕ ਇਤਿਹਾਸ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਨ ਲਈ `call` ਵਿਧੀ ਨੂੰ ਬੁਲਾਉਂਦੀ ਹੈ।

`call` ਵਿਧੀ ਦੇ ਅੰਦਰ, ਗਾਹਕ ਦੇ ਲੈਣ-ਦੇਣ ਦਾ ਇਤਿਹਾਸ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਇੱਕ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਵੱਚਿ ਫਾਰਮੈਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜੋ ਏ.ਆਈ. ਮਾਡਲ ਨੂੰ ਪਾਸ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਏ.ਆਈ. ਮਾਡਲ ਦੱਤਿੇ ਗਏ ਨਿਰਦੇਸ਼ਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਲੈਣ-ਦੇਣ ਦੇ ਇਤਿਹਾਸ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਦਾ ਹੈ ਅਤੇ ਆਪਣੇ ਨਤੀਜਿਆਂ ਦਾ ਸਾਰ ਵਾਪਸ ਕਰਦਾ ਹੈ।

ਨਤੀਜੇ `fraud_detection` ਵਸਤੂ ਵੱਚਿ ਜੋੜੇ ਜਾਂਦੇ ਹਨ, ਅਤੇ ਅੱਪਡੇਟ ਕੀਤੀ `fraud_detection` ਵਸਤੂ ਇੱਕ ਸਫਲ `Result` ਵਜੋਂ ਵਾਪਸ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

`ChatCompletion` ਮੋਡਿਊਲ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, `CheckCustomerHistory` ਕਲਾਸ ਏ.ਆਈ. ਦੀ ਸ਼ਕਤੀ ਦੀ ਵਰਤੋਂ ਗਾਹਕ ਦੇ ਲੈਣ-ਦੇਣ ਦੇ ਇਤਿਹਾਸ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਨ ਅਤੇ ਸੰਭਾਵੀ ਧੋਖਾਧੜੀ ਦੇ ਸੰਕੇਤਾਂ ਦੀ ਪਛਾਣ ਕਰਨ ਲਈ ਕਰ ਸਕਦੀ ਹੈ। ਇਹ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਅਤੇ ਅਨੁਕੂਲ ਧੋਖਾਧੜੀ ਦੀ ਪਛਾਣ ਤਕਨੀਕਾਂ ਦੀ ਆਗਿਆ ਦੱਿਦਾ ਹੈ, ਕਿਉਂਕਿ ਏ.ਆਈ. ਮਾਡਲ ਨਵੇਂ ਪੈਟਰਨਾਂ ਅਤੇ ਅਸਧਾਰਨਤਾਵਾਂ ਤੋਂ ਸਿੱਖ ਅਤੇ ਅਨੁਕੂਲ ਹੋ ਸਕਦਾ ਹੈ।

ਅੱਪਡੇਟ ਕੀਤਾ `FraudDetectionWorker` ਅਤੇ `CheckCustomerHistory` ਕਲਾਸ ਦਿਖਾਉਂਦੇ ਹਨ ਕਿ ਏ.ਆਈ. ਵਰਕਰਾਂ ਨੂੰ ਕਿਵੇਂ ਨਿਰਵਘਿਨ ਏਕੀਕ੍ਰਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਬੁੱਧੀਮਾਨ ਵਿਸ਼ਲੇਸ਼ਣ ਅਤੇ ਫੈਸਲਾ ਲੈਣ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਨਾਲ ਧੋਖਾਧੜੀ ਦੀ ਪਛਾਣ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਵਧਾਉਂਦੇ ਹੋਏ।

ਗਾਹਕ ਭਾਵਨਾ ਵਿਸ਼ਲੇਸ਼ਣ

ਇੱਥੇ ਇੱਕ ਹੋਰ ਇਸੇ ਤਰ੍ਹਾਂ ਦੀ ਉਦਾਹਰਣ ਹੈ ਕਿ ਤੁਸੀਂ ਗਾਹਕ ਭਾਵਨਾ ਵਿਸ਼ਲੇਸ਼ਣ ਵਰਕਰ ਨੂੰ ਕਿਵੇਂ ਲਾਗੂ ਕਰ ਸਕਦੇ ਹੋ। ਇਸ ਵਾਰ ਬਹੁਤ ਘੱਟ ਵਿਆਖਿਆ, ਕਿਉਂਕਿ ਤੁਹਾਨੂੰ ਇਸ ਪ੍ਰੋਗਰਾਮਿੰਗ ਸ਼ੈਲੀ ਦੇ ਕੰਮ ਕਰਨ ਦਾ ਢੰਗ ਸਮਝ ਆ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ:

```
class CustomerSentimentAnalysisWorker
  include Wisper::Publisher

  def call(feedback)
    Result.ok(feedback)
      .and_then(PreprocessFeedback.method(:preprocess))
      .map(PerformSentimentAnalysis.method(:analyze))
      .map(ExtractKeyPhrases.method(:extract))
      .map(IdentifyTrends.method(:identify))
      .map(GenerateInsights.method(:generate)).then do |result|

      case result
      in { err: SentimentAnalysisError => error }
        Honeybadger.notify(error.message, context: {feedback:})
      in { ok: SentimentAnalysisResult => result }
        broadcast(:sentiment_analysis_completed, result)
      end
    end
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ, `CustomerSentimentAnalysisWorker` ਦੇ ਕਦਮਾਂ ਵਿੱਚ ਫੀਡਬੈਕ ਦੀ ਪ੍ਰੀ-ਪ੍ਰੋਸੈਸਿੰਗ (ਜਿਵੇਂ ਕਿ ਸ਼ੋਰ ਹਟਾਉਣਾ, ਟੋਕਨਾਈਜ਼ੇਸ਼ਨ), ਭਾਵਨਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਨਾ (ਸਕਾਰਾਤਮਕ, ਨਕਾਰਾਤਮਕ, ਜਾਂ ਨਿਰਪੱਖ ਭਾਵਨਾ ਨਿਰਧਾਰਤ ਕਰਨ ਲਈ), ਮੁੱਖ ਵਾਕਾਂਸ਼ਾਂ ਅਤੇ ਵਿਸ਼ਿਆਂ ਦੀ ਪਛਾਣ, ਰੁਝਾਨਾਂ ਅਤੇ ਪੈਟਰਨਾਂ ਦੀ ਪਛਾਣ, ਅਤੇ ਵਿਸ਼ਲੇਸ਼ਣ ਦੇ ਆਧਾਰ 'ਤੇ ਕਾਰਵਾਈਯੋਗ ਅੰਤਰਦ੍ਰਸ਼ਟੀ ਤਿਆਰ ਕਰਨਾ ਸ਼ਾਮਲ ਹੈ।

ਸਹਿਤ ਸੰਭਾਲ ਐਪਲੀਕੇਸ਼ਨਾਂ

ਸਹਿਤ ਸੰਭਾਲ ਖੇਤਰ ਵਿੱਚ, AI ਵਰਕਰ ਮੈਡੀਕਲ ਪੇਸ਼ੇਵਰਾਂ ਅਤੇ ਖੋਜਕਰਤਾਵਾਂ ਦੀ ਵੱਖ-ਵੱਖ ਕਾਰਜਾਂ ਵਿੱਚ ਸਹਾਇਤਾ ਕਰ ਸਕਦੇ ਹਨ, ਜਿਸ ਨਾਲ ਮਰੀਜ਼ਾਂ ਦੇ ਨਤੀਜਿਆਂ ਵਿੱਚ ਸੁਧਾਰ ਅਤੇ ਮੈਡੀਕਲ ਖੋਜਾਂ ਵਿੱਚ ਤੇਜ਼ੀ ਆਉਂਦੀ ਹੈ। ਕੁਝ ਉਦਾਹਰਣਾਂ ਹਨ:

ਮਰੀਜ਼ ਦਾਖਲਾ

AI ਵਰਕਰ ਵੱਖ-ਵੱਖ ਕਾਰਜਾਂ ਨੂੰ ਸਵੈਚਾਲਤਿ ਕਰਕੇ ਅਤੇ ਬੁੱਧੀਮਾਨ ਸਹਾਇਤਾ ਪ੍ਰਦਾਨ ਕਰਕੇ ਮਰੀਜ਼ ਦਾਖਲਾ ਪ੍ਰਕਰਿਆ ਨੂੰ ਸੁਚਾਰੂ ਬਣਾ ਸਕਦੇ ਹਨ।

ਅਪੌਇੰਟਮੈਂਟ ਸ਼ੈਡਊਲਿੰਗ: AI ਵਰਕਰ ਮਰੀਜ਼ਾਂ ਦੀਆਂ ਤਰਜੀਹਾਂ, ਉਪਲਬਧਤਾ, ਅਤੇ ਉਨ੍ਹਾਂ ਦੀਆਂ ਮੈਡੀਕਲ ਲੋੜਾਂ ਦੀ ਤਤਕਾਲੀਨਤਾ ਨੂੰ ਸਮਝ ਕੇ ਅਪੌਇੰਟਮੈਂਟ ਸ਼ੈਡਊਲਿੰਗ ਨੂੰ ਸੰਭਾਲ ਸਕਦੇ ਹਨ। ਉਹ ਗੱਲਬਾਤ ਇੰਟਰਫੇਸਾਂ ਰਾਹੀ ਮਰੀਜ਼ਾਂ ਨਾਲ ਸੰਪਰਕ ਕਰ ਸਕਦੇ ਹਨ, ਉਨ੍ਹਾਂ ਨੂੰ ਸ਼ੈਡਊਲਿੰਗ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਮਾਰਗਦਰਸ਼ਨ ਕਰ ਸਕਦੇ ਹਨ ਅਤੇ ਮਰੀਜ਼ ਦੀਆਂ ਲੋੜਾਂ ਅਤੇ ਸਹਿਤ ਸੰਭਾਲ ਪ੍ਰਦਾਤਾ ਦੀ ਉਪਲਬਧਤਾ ਦੇ ਆਧਾਰ 'ਤੇ ਸਭ ਤੋਂ ਢੁਕਵੇਂ ਅਪੌਇੰਟਮੈਂਟ ਸਲੌਟ ਲੱਭ ਸਕਦੇ ਹਨ।

ਮੈਡੀਕਲ ਇਤਿਹਾਸ ਇਕੱਤਰੀਕਰਨ: ਮਰੀਜ਼ ਦਾਖਲੇ ਦੌਰਾਨ, AI ਵਰਕਰ ਮਰੀਜ਼ ਦੇ ਮੈਡੀਕਲ ਇਤਿਹਾਸ ਨੂੰ ਇਕੱਤਰ ਕਰਨ ਅਤੇ ਦਸਤਾਵੇਜ਼ੀਕਰਨ ਵੱਚਿ ਸਹਾਇਤਾ ਕਰ ਸਕਦੇ ਹਨ। ਉਹ ਮਰੀਜ਼ਾਂ ਨਾਲ ਇੰਟਰਐਕਟਵਿ ਸੰਵਾਦ ਵੱਚਿ ਸ਼ਾਮਲ ਹੋ ਸਕਦੇ ਹਨ, ਉਨ੍ਹਾਂ ਦੀਆਂ ਪਛਿਲੀਆਂ ਮੈਡੀਕਲ ਸਥਤਿੀਆਂ, ਦਵਾਈਆਂ, ਐਲਰਜੀਆਂ, ਅਤੇ ਪਰਵਿਾਰਕ ਇਤਿਹਾਸ ਬਾਰੇ ਢੁਕਵੇਂ ਸਵਾਲ ਪੁੱਛ ਸਕਦੇ ਹਨ। AI ਵਰਕਰ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਪ੍ਰੋਸੈਸਿੰਗ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇਕੱਤਰ ਕੀਤੀ ਜਾਣਕਾਰੀ ਦੀ ਵਆਿਖਆਿ ਅਤੇ ਸੰਰਚਨਾ ਕਰ ਸਕਦੇ ਹਨ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦੇ ਹੋਏ ਕ ਿਇਹ ਮਰੀਜ਼ ਦੇ ਇਲੈਕਟ੍ਰਾਨਕਿ ਸਹਿਤ ਰਕਿਾਰਡ ਵੱਚਿ ਸਹੀ ਢੰਗ ਨਾਲ ਦਰਜ ਕੀਤੀ ਗਈ ਹੈ।

ਲੱਛਣ ਮੁਲਾਂਕਣ ਅਤੇ ਵਰਗੀਕਰਨ: AI ਵਰਕਰ ਮਰੀਜ਼ਾਂ ਨੂੰ ਉਨ੍ਹਾਂ ਦੇ ਮੌਜੂਦਾ ਲੱਛਣਾਂ, ਮਆਿਦ, ਗੰਭੀਰਤਾ, ਅਤੇ ਕਸਿੇ ਵੀ ਸੰਬੰਧਤਿ ਕਾਰਕਾਂ ਬਾਰੇ ਪੁੱਛ ਕੇ ਸ਼ੁਰੂਆਤੀ ਲੱਛਣ ਮੁਲਾਂਕਣ ਕਰ ਸਕਦੇ ਹਨ। ਮੈਡੀਕਲ ਗਆਿਨ ਅਧਾਰਾਂ ਅਤੇ ਮਸ਼ੀਨ ਲਰਨਿੰਗ ਮਾਡਲਾਂ ਦਾ ਲਾਭ ਲੈ ਕੇ, ਇਹ ਵਰਕਰ ਪ੍ਰਦਾਨ ਕੀਤੀ ਜਾਣਕਾਰੀ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰ ਸਕਦੇ ਹਨ ਅਤੇ ਮੁੱਢਲੇ ਵਭਿੇਦਕ ਨਦਿਾਨ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ ਜਾਂ ਢੁਕਵੇਂ ਅਗਲੇ ਕਦਮਾਂ ਦੀ ਸਫਿਾਰਸ਼ ਕਰ ਸਕਦੇ ਹਨ, ਜਵਿੇਂ ਕ ਿਸਹਿਤ ਸੰਭਾਲ ਪ੍ਰਦਾਤਾ ਨਾਲ ਸਲਾਹ-ਮਸ਼ਵਰਾ ਕਰਨ ਦਾ ਸ਼ੈਡਊਲ ਬਣਾਉਣਾ ਜਾਂ ਸਵੈ-ਦੇਖਭਾਲ ਦੇ ਉਪਾਵਾਂ ਦੀ ਸਫਿਾਰਸ਼ ਕਰਨਾ।

ਬੀਮਾ ਤਸਦੀਕ: AI ਵਰਕਰ ਮਰੀਜ਼ ਦਾਖਲੇ ਦੌਰਾਨ ਬੀਮਾ ਤਸਦੀਕ ਵੱਚਿ ਸਹਾਇਤਾ ਕਰ ਸਕਦੇ ਹਨ। ਉਹ ਮਰੀਜ਼ ਦੇ ਬੀਮਾ ਵੇਰਵੇ ਇਕੱਠੇ ਕਰ ਸਕਦੇ ਹਨ, APIs ਜਾਂ ਵੈੱਬ ਸੇਵਾਵਾਂ ਰਾਹੀ ਬੀਮਾ ਪ੍ਰਦਾਤਾਵਾਂ ਨਾਲ ਸੰਚਾਰ ਕਰ ਸਕਦੇ ਹਨ, ਅਤੇ ਕਵਰੇਜ ਯੋਗਤਾ ਅਤੇ ਲਾਭਾਂ ਦੀ ਪੁਸ਼ਟੀ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਆਟੋਮੇਸ਼ਨ ਬੀਮਾ ਤਸਦੀਕ ਪ੍ਰਕਰਿਆ ਨੂੰ ਸੁਚਾਰੂ ਬਣਾਉਣ ਵੱਚਿ ਮਦਦ ਕਰਦੀ ਹੈ, ਪ੍ਰਸ਼ਾਸਨਕਿ ਬੋਝ ਨੂੰ ਘਟਾਉਂਦੀ ਹੈ ਅਤੇ ਸਹੀ ਜਾਣਕਾਰੀ ਕੈਪਚਰ ਕਰਨਾ ਯਕੀਨੀ ਬਣਾਉਂਦੀ ਹੈ।

ਮਰੀਜ਼ ਸਾਂਖਿਆ ਅਤੇ ਹਦਾਇਤਾਂ: AI ਵਰਕਰ ਮਰੀਜ਼ਾਂ ਨੂੰ ਉਨ੍ਹਾਂ ਦੀਆਂ ਵਸ਼ਿੇਸ਼ ਮੈਡੀਕਲ ਸਥਤਿੀਆਂ ਜਾਂ ਆਉਣ ਵਾਲੀਆਂ ਪ੍ਰਕਰਿਆਵਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਢੁਕਵੀਂ ਵਦਿਅਕ ਸਮੱਗਰੀ ਅਤੇ ਹਦਾਇਤਾਂ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ। ਉਹ ਨੰਜੀ ਸਮੱਗਰੀ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ, ਆਮ ਸਵਾਲਾਂ ਦੇ ਜਵਾਬ ਦੇ ਸਕਦੇ ਹਨ, ਅਤੇ ਅਪੌਇੰਟਮੈਂਟ ਤੋਂ ਪਹਲਿਾਂ ਦੀਆਂ ਤਆਿਰੀਆਂ, ਦਵਾਈ ਦੀਆਂ ਹਦਾਇਤਾਂ, ਜਾਂ ਇਲਾਜ ਤੋਂ ਬਾਅਦ ਦੀ ਦੇਖਭਾਲ ਬਾਰੇ ਮਾਰਗਦਰਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਮਰੀਜ਼ਾਂ ਨੂੰ ਉਨ੍ਹਾਂ ਦੀ ਸਹਿਤ ਸੰਭਾਲ ਯਾਤਰਾ ਦੌਰਾਨ ਜਾਣਕਾਰ ਅਤੇ ਸ਼ਾਮਲ ਰੱਖਣ ਵੱਚਿ ਮਦਦ ਕਰਦਾ ਹੈ।

ਮਰੀਜ਼ ਦਾਖਲੇ ਵੱਚਿ AI ਵਰਕਰਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਸਹਿਤ ਸੰਭਾਲ ਸੰਸਥਾਵਾਂ ਕੁਸ਼ਲਤਾ ਵਧਾ ਸਕਦੀਆਂ ਹਨ, ਉਡੀਕ ਸਮੇਂ ਨੂੰ ਘਟਾ ਸਕਦੀਆਂ ਹਨ, ਅਤੇ ਸਮੁੱਚੇ ਮਰੀਜ਼ ਅਨੁਭਵ ਨੂੰ ਬਹਿਤਰ ਬਣਾ ਸਕਦੀਆਂ ਹਨ। ਇਹ ਵਰਕਰ ਨਯਿਮਤ ਕਾਰਜਾਂ ਨੂੰ ਸੰਭਾਲ ਸਕਦੇ ਹਨ, ਸਹੀ ਜਾਣਕਾਰੀ ਇਕੱਠੀ ਕਰ ਸਕਦੇ ਹਨ, ਅਤੇ ਵਅਿਕਤੀਗਤ ਸਹਾਇਤਾ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ, ਜੋ ਸਹਿਤ ਸੰਭਾਲ ਪੇਸ਼ੇਵਰਾਂ ਨੂੰ ਮਰੀਜ਼ਾਂ ਨੂੰ ਉੱਚ-ਗੁਣਵੱਤਾ ਵਾਲੀ ਦੇਖਭਾਲ ਪ੍ਰਦਾਨ ਕਰਨ ’ਤੇ ਧਅਿਾਨ ਕੇਂਦਰਤਿ ਕਰਨ ਦੀ ਆਗਅਿਾ ਦੇਂਦੇ ਹਨ।

ਮਰੀਜ਼ ਜੋਖਮ ਮੁਲਾਂਕਣ

AI ਵਰਕਰ ਵੱਖ-ਵੱਖ ਡਾਟਾ ਸਰੋਤਾਂ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਕੇ ਅਤੇ ਉੱਨਤ ਵਸ਼ਿਲੇਸ਼ਣ ਤਕਨੀਕਾਂ ਲਾਗੂ ਕਰਕੇ ਮਰੀਜ਼ ਜੋਖਮ ਦੇ ਮੁਲਾਂਕਣ ਵੱਚਿ ਮਹੱਤਵਪੂਰਨ ਭੂਮਕਿਾ ਨਭਿਾ ਸਕਦੇ ਹਨ।

ਡਾਟਾ ਏਕੀਕਰਨ: AI ਵਰਕਰ ਕਈ ਸਰੋਤਾਂ ਤੋਂ ਮਰੀਜ਼ ਡਾਟਾ ਇਕੱਠਾ ਕਰ ਸਕਦੇ ਹਨ ਅਤੇ ਇਸਨੂੰ ਸਮਝ ਸਕਦੇ ਹਨ, ਜਵਿੇਂ ਕ ਿਇਲੈਕਟ੍ਰਾਨਕਿ ਸਹਿਤ ਰਕਿਾਰਡ (EHRs), ਮੈਡੀਕਲ ਇਮੇਜਿੰਗ, ਲੈਬ ਨਤੀਜੇ, ਪਹਨਿਣਯੋਗ ਉਪਕਰਣ, ਅਤੇ ਸਹਿਤ ਦੇ ਸਮਾਜਕਿ ਨਰਿਧਾਰਕ। ਇਸ ਜਾਣਕਾਰੀ ਨੂੰ ਇੱਕ ਵਅਿਾਪਕ ਮਰੀਜ਼ ਪ੍ਰੋਫਾਈਲ ਵੱਚਿ ਏਕੀਕ੍ਰਤਿ ਕਰਕੇ, AI ਵਰਕਰ ਮਰੀਜ਼ ਦੀ ਸਹਿਤ ਸਥਤਿੀ ਅਤੇ ਜੋਖਮ ਕਾਰਕਾਂ ਦਾ ਸਮੁੱਚਾ ਦ੍ਰਸ਼ਟੀਕੋਣ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ।

ਜੋਖਮ ਵਰਗੀਕਰਨ: AI ਵਰਕਰ ਮਰੀਜ਼ਾਂ ਦੇ ਵਅਿਕਤੀਗਤ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਅਤੇ ਸਹਿਤ ਡਾਟਾ ਦੇ ਆਧਾਰ ’ਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਜੋਖਮ ਸ਼੍ਰੇਣੀਆਂ ਵੱਚਿ ਵਰਗੀਕ੍ਰਤਿ ਕਰਨ ਲਈ ਭਵਖਿ-ਸੂਚਕ ਮਾਡਲਾਂ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਜੋਖਮ ਵਰਗੀਕਰਨ ਸਹਿਤ ਸੰਭਾਲ ਪ੍ਰਦਾਤਾਵਾਂ ਨੂੰ ਉਨ੍ਹਾਂ ਮਰੀਜ਼ਾਂ ਨੂੰ ਪ੍ਰਾਥਮਕਿਤਾ ਦੇਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ ਜਨ੍ਹਿਾਂ ਨੂੰ ਵਧੇਰੇ ਤੁਰੰਤ ਧਅਿਾਨ ਜਾਂ ਦਖਲਅੰਦਾਜ਼ੀ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਉਦਾਹਰਨ ਲਈ, ਕਸਿ ਖਾਸ ਸਥਤਿੀ ਲਈ ਉੱਚ-ਜੋਖਮ ਵਾਲੇ ਪਛਾਣੇ ਗਏ ਮਰੀਜ਼ਾਂ ਨੂੰ ਨੇੜਓਿਂ ਨਗਿਰਾਨੀ, ਰੋਕਥਾਮ ਉਪਾਵਾਂ, ਜਾਂ ਜਲਦੀ ਦਖਲਅੰਦਾਜ਼ੀ ਲਈ ਫਲੈਗ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।

ਵਅਿਕਤੀਗਤ ਜੋਖਮ ਪ੍ਰੋਫਾਈਲ: AI ਵਰਕਰ ਹਰੇਕ ਮਰੀਜ਼ ਲਈ ਵਅਿਕਤੀਗਤ ਜੋਖਮ ਪ੍ਰੋਫਾਈਲ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ, ਜੋ ਉਨ੍ਹਾਂ ਦੇ ਜੋਖਮ ਸਕੋਰਾਂ ਵੱਚਿ ਯੋਗਦਾਨ ਪਾਉਣ ਵਾਲੇ ਵਸ਼ਿਸ਼ ਕਾਰਕਾਂ ਨੂੰ ਉਜਾਗਰ ਕਰਦੇ ਹਨ। ਇਨ੍ਹਾਂ ਪ੍ਰੋਫਾਈਲਾਂ ਵੱਚਿ ਮਰੀਜ਼ ਦੀ ਜੀਵਨਸ਼ੈਲੀ, ਜੈਨੇਟਕਿ ਝੁਕਾਅ, ਵਾਤਾਵਰਣ ਕਾਰਕ, ਅਤੇ ਸਹਿਤ ਦੇ ਸਮਾਜਕਿ ਨਰਿਧਾਰਕਾਂ ਬਾਰੇ ਅੰਤਰਝਾਤ ਸ਼ਾਮਲ ਹੋ ਸਕਦੀ ਹੈ। ਜੋਖਮ ਕਾਰਕਾਂ ਦਾ ਵਸਿਤ੍ਰਤਿ ਵੇਰਵਾ ਪ੍ਰਦਾਨ ਕਰਕੇ, AI ਵਰਕਰ ਸਹਿਤ ਸੰਭਾਲ ਪ੍ਰਦਾਤਾਵਾਂ ਨੂੰ ਵਅਿਕਤੀਗਤ ਮਰੀਜ਼ ਦੀਆਂ ਲੋੜਾਂ ਅਨੁਸਾਰ ਰੋਕਥਾਮ ਰਣਨੀਤੀਆਂ ਅਤੇ ਇਲਾਜ ਯੋਜਨਾਵਾਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣ ਵੱਚਿ ਮਦਦ ਕਰ ਸਕਦੇ ਹਨ।

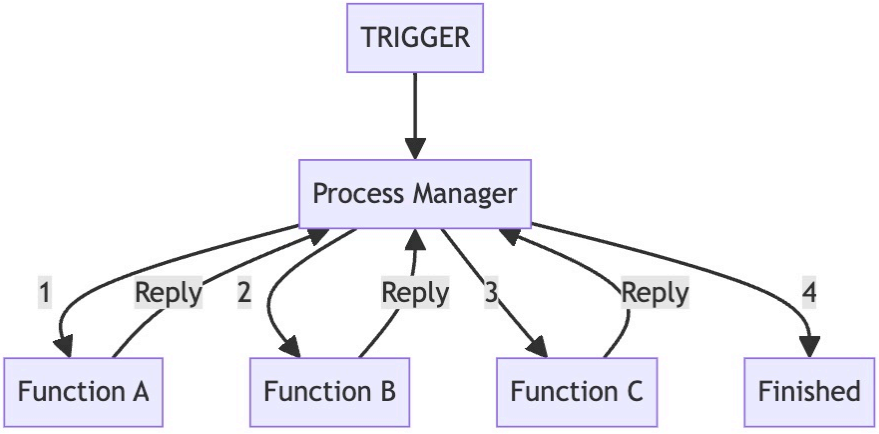
ਲਗਾਤਾਰ ਜੋਖਮ ਨਗਿਰਾਨੀ: AI ਵਰਕਰ ਲਗਾਤਾਰ ਮਰੀਜ਼ ਡਾਟਾ ਦੀ ਨਗਿਰਾਨੀ ਕਰ ਸਕਦੇ ਹਨ ਅਤੇ ਅਸਲ-

ਸਮੇਂ ਵਿੱਚ ਜੋਖਮ ਮੁਲਾਂਕਣ ਨੂੰ ਅੱਪਡੇਟ ਕਰ ਸਕਦੇ ਹਨ। ਜਦੋਂ ਹੀ ਨਵੀਂ ਜਾਣਕਾਰੀ ਉਪਲਬਧ ਹੁੰਦੀ ਹੈ, ਜਦੋਂ ਕਿ ਜੀਵਨ ਸੰਕੇਤਾਂ, ਲੈਬ ਨਤੀਜਿਆਂ, ਜਾਂ ਦਵਾਈ ਦੀ ਪਾਲਣਾ ਵਿੱਚ ਤਬਦੀਲੀਆਂ, AI ਵਰਕਰ ਜੋਖਮ ਸਕੋਰਾਂ ਦੀ ਮੁੜ-ਗਣਨਾ ਕਰ ਸਕਦੇ ਹਨ ਅਤੇ ਕਿਸੇ ਵੀ ਮਹੱਤਵਪੂਰਨ ਤਬਦੀਲੀਆਂ ਬਾਰੇ ਸਹਿਤ ਸੰਭਾਲ ਪ੍ਰਦਾਤਾਵਾਂ ਨੂੰ ਸੁਚੇਤ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਸਰਗਰਮ ਨਿਗਰਾਨੀ ਸਮੇਂ ਸਿਰ ਦਖਲਅੰਦਾਜ਼ੀ ਅਤੇ ਮਰੀਜ਼ ਦੇਖਭਾਲ ਯੋਜਨਾਵਾਂ ਵਿੱਚ ਸਮਾਯੋਜਨ ਦੀ ਆਗਿਆ ਦਿੰਦੀ ਹੈ।

ਕਲੀਨਿਕਲ ਫੈਸਲਾ ਸਹਾਇਤਾ: AI ਵਰਕਰ ਜੋਖਮ ਮੁਲਾਂਕਣ ਦੇ ਨਤੀਜਿਆਂ ਨੂੰ ਕਲੀਨਿਕਲ ਫੈਸਲਾ ਸਹਾਇਤਾ ਪ੍ਰਣਾਲੀਆਂ ਵਿੱਚ ਏਕੀਕ੍ਰਤਿ ਕਰ ਸਕਦੇ ਹਨ, ਜੋ ਸਹਿਤ ਸੰਭਾਲ ਪ੍ਰਦਾਤਾਵਾਂ ਨੂੰ ਸਬੂਤ-ਆਧਾਰਤਿ ਸਿਫ਼ਾਰਸ਼ਾਂ ਅਤੇ ਚੇਤਾਵਨੀਆਂ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ। ਉਦਾਹਰਨ ਲਈ, ਜੇਕਰ ਕਿਸੇ ਖਾਸ ਸਥਿਤੀ ਲਈ ਮਰੀਜ਼ ਦਾ ਜੋਖਮ ਸਕੋਰ ਇੱਕ ਨਿਸ਼ਚਿਤ ਸੀਮਾ ਤੋਂ ਵੱਧ ਜਾਂਦਾ ਹੈ, ਤਾਂ AI ਵਰਕਰ ਕਲੀਨਿਕਲ ਦਿਸ਼ਾ-ਨਿਰਦੇਸ਼ਾਂ ਅਤੇ ਸਰਵੋਤਮ ਅਭਿਆਸਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਸਹਿਤ ਸੰਭਾਲ ਪ੍ਰਦਾਤਾ ਨੂੰ ਵਿਸ਼ੇਸ਼ ਜਾਂਚ ਟੈਸਟਾਂ, ਰੋਕਥਾਮ ਉਪਾਵਾਂ, ਜਾਂ ਇਲਾਜ ਵਿਕਲਪਾਂ 'ਤੇ ਵਿਚਾਰ ਕਰਨ ਲਈ ਪ੍ਰੇਰਿਤ ਕਰ ਸਕਦਾ ਹੈ।

ਇਹ ਵਰਕਰ ਮਰੀਜ਼ ਦੇ ਵਿਸ਼ਾਲ ਡਾਟਾ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰ ਸਕਦੇ ਹਨ, ਉੱਨਤ ਵਿਸ਼ਲੇਸ਼ਣ ਲਾਗੂ ਕਰ ਸਕਦੇ ਹਨ, ਅਤੇ ਕਲੀਨਿਕਲ ਫੈਸਲਾ-ਲੈਣ ਦੀ ਸਹਾਇਤਾ ਲਈ ਕਾਰਵਾਈਯੋਗ ਅੰਤਰਝਾਤ ਪੈਦਾ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਅੰਤਮ ਤੌਰ 'ਤੇ ਮਰੀਜ਼ ਦੇ ਨਤੀਜਿਆਂ ਵਿੱਚ ਸੁਧਾਰ, ਸਹਿਤ ਸੰਭਾਲ ਲਾਗਤਾਂ ਵਿੱਚ ਕਮੀ, ਅਤੇ ਵਧੀ ਹੋਈ ਆਬਾਦੀ ਸਹਿਤ ਪ੍ਰਬੰਧਨ ਵੱਲ ਲੈ ਜਾਂਦਾ ਹੈ।

AI ਵਰਕਰ ਇੱਕ ਪ੍ਰੋਸੈਸ ਮੈਨੇਜਰ ਵਜੋਂ



AI-ਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦੇ ਸੰਦਰਭ ਵਿੱਚ, ਇੱਕ ਵਰਕਰ ਨੂੰ ਪ੍ਰੋਸੈਸ ਮੈਨੇਜਰ ਵਜੋਂ ਕੰਮ ਕਰਨ ਲਈ ਡਜ਼ਿਾਈਨ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਜਵਿੇਂ ਕਿ Gregor Hohpe ਦੀ "ਐਂਟਰਪ੍ਰਾਈਜ਼ ਇੰਟੀਗ੍ਰੇਸ਼ਨ ਪੈਟਰਨ" ਕਤਿਾਬ ਵਿੱਚ ਦੱਸਿਆ ਗਿਆ ਹੈ। ਇੱਕ ਪ੍ਰੋਸੈਸ ਮੈਨੇਜਰ ਇੱਕ ਕੇਂਦਰੀ ਭਾਗ ਹੈ ਜੋ ਪ੍ਰਕਰਿਆ ਦੀ ਸਥਤਿੀ ਨੂੰ ਬਣਾਈ ਰੱਖਦਾ ਹੈ ਅਤੇ ਵਚਿਕਾਰਲੇ ਨਤੀਜਆਿਂ ਦੇ ਆਧਾਰ 'ਤੇ ਅਗਲੇ ਪ੍ਰੋਸੈਸਿੰਗ ਕਦਮਾਂ ਨੂੰ ਨਰਿਧਾਰਤਿ ਕਰਦਾ ਹੈ।

ਜਦੋਂ ਇੱਕ AI ਵਰਕਰ ਪ੍ਰੋਸੈਸ ਮੈਨੇਜਰ ਵਜੋਂ ਕੰਮ ਕਰਦਾ ਹੈ, ਇਹ ਇੱਕ ਇਨਕਮਿੰਗ ਸੁਨੇਹਾ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ ਜੋ ਪ੍ਰਕਰਿਆ ਨੂੰ ਸ਼ੁਰੂ ਕਰਦਾ ਹੈ, ਜਸਿਨੂੰ ਟ੍ਰਗਿੰਰ ਸੁਨੇਹਾ ਕਹਿਾ ਜਾਂਦਾ ਹੈ। AI ਵਰਕਰ ਫਰਿ ਪ੍ਰਕਰਿਆ ਦੀ ਕਾਰਜਕੁਸ਼ਲਤਾ ਦੀ ਸਥਤਿੀ (ਗੱਲਬਾਤ ਦੀ ਲਖਿਤ ਵਜੋਂ) ਨੂੰ ਬਣਾਈ ਰੱਖਦਾ ਹੈ ਅਤੇ ਟੂਲ ਫੰਕਸ਼ਨਾਂ ਵਜੋਂ ਲਾਗੂ ਕੀਤੇ ਪ੍ਰੋਸੈਸਿੰਗ ਕਦਮਾਂ ਦੀ ਲੜੀ ਰਾਹੀਂ ਸੁਨੇਹੇ ਨੂੰ ਸੰਭਾਲਦਾ ਹੈ, ਜੋ ਕ੍ਰਮਵਾਰ ਜਾਂ ਸਮਾਨਾਂਤਰ ਹੋ ਸਕਦੇ ਹਨ, ਅਤੇ ਇਸਦੀ ਮਰਜ਼ੀ ਨਾਲ ਬੁਲਾਏ ਜਾ ਸਕਦੇ ਹਨ।

ਜੇਕਰ ਤੁਸੀਂ GPT-4 ਵਰਗੇ AI ਮਾਡਲ ਦੀ ਸ੍ਰੇਣੀ ਦੀ ਵਰਤੋਂ ਕਰ ਰਹੇ ਹੋ ਜੋ ਸਮਾਨਾਂਤਰ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਚਲਾਉਣਾ ਜਾਣਦਾ ਹੈ ਤਾਂ ਤੁਹਾਡਾ ਵਰਕਰ ਇੱਕੋ ਸਮੇਂ ਕਈ ਕਦਮ ਚਲਾ ਸਕਦਾ ਹੈ। ਮੰਨਣਾ ਹੋਵੇਗਾ, ਮੈਂ ਖੁਦ ਇਹ ਕਰਨ ਦੀ ਕੋਸ਼ਸ਼ ਨਹੀਂ ਕੀਤੀ ਅਤੇ ਮੇਰਾ ਅੰਦਾਜ਼ਾ ਕਹਿੰਦਾ ਹੈ ਕਿ ਤੁਹਾਡੇ ਨਤੀਜੇ ਵੱਖ-ਵੱਖ ਹੋ ਸਕਦੇ ਹਨ।

ਹਰ ਵਿਅਕਤੀਗਤ ਪ੍ਰੋਸੈਸਿੰਗ ਕਦਮ ਤੋਂ ਬਾਅਦ, ਕੰਟਰੋਲ AI ਵਰਕਰ ਨੂੰ ਵਾਪਸ ਕਰ ਦਿੱਤਾ ਜਾਂਦਾ ਹੈ, ਜੋ ਇਸਨੂੰ ਮੌਜੂਦਾ ਸਥਿਤੀ ਅਤੇ ਪ੍ਰਾਪਤ ਨਤੀਜਿਆਂ ਦੇ ਆਧਾਰ 'ਤੇ ਅਗਲੇ ਪ੍ਰੋਸੈਸਿੰਗ ਕਦਮ(ਮਾਂ) ਨੂੰ ਨਿਰਧਾਰਿਤ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦਾ ਹੈ।

ਆਪਣੇ ਟ੍ਰਿੰਗਿਰ ਸੁਨੇਹਿਆਂ ਨੂੰ ਸਟੋਰ ਕਰੋ

ਮੇਰੇ ਤਜਰਬੇ ਵਿੱਚ, ਆਪਣੇ ਟ੍ਰਿੰਗਿਰ ਸੁਨੇਹੇ ਨੂੰ ਡਾਟਾਬੇਸ-ਅਧਾਰਿਤ ਔਬਜੈਕਟ ਵਜੋਂ ਲਾਗੂ ਕਰਨਾ ਸਮਝਦਾਰੀ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਹਰੇਕ ਪ੍ਰਕਰਿਆ ਇੰਸਟੈਂਸ ਇੱਕ ਵਲਿੱਖਣ ਪ੍ਰਾਇਮਰੀ ਕੀ ਦੁਆਰਾ ਪਛਾਣਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਤੁਹਾਨੂੰ AI ਦੀ ਗੱਲਬਾਤ ਦੀ ਲਖਿਤ ਸਮੇਤ, ਕਾਰਜਕੁਸ਼ਲਤਾ ਨਾਲ ਜੁੜੀ ਸਥਿਤੀ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਇੱਕ ਥਾਂ ਦਿੰਦਾ ਹੈ।

ਉਦਾਹਰਣ ਲਈ, ਇੱਥੇ Olympia ਦੇ `AccountChange` ਮਾਡਲ ਕਲਾਸ ਦਾ ਇੱਕ ਸਰਲ ਰੂਪ ਹੈ, ਜੋ ਇੱਕ ਯੂਜ਼ਰ ਦੇ ਖਾਤੇ ਵਿੱਚ ਬਦਲਾਵ ਕਰਨ ਦੀ ਬੇਨਤੀ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

```
# == Schema Information
#
# Table name: account_changes
#
#  id          :uuid             not null, primary key
#  description :string
#  state       :string           not null
#  transcript  :jsonb
#  created_at  :datetime         not null
#  updated_at  :datetime         not null
#  account_id  :uuid             not null
#
# Indexes
#
#  index_account_changes_on_account_id  (account_id)
#
# Foreign Keys
#
#  fk_rails_...  (account_id => accounts.id)
#
class AccountChange < ApplicationRecord
  belongs_to :account

  validates :description, presence: true

```

```
  after_commit -> {
    broadcast(:account_change_requested, self)
  }, on: :create

  state_machine initial: :requested do
    event :completed do
      transition all => :complete
    end
    event :failed do
      transition all => :requires_human_review
    end
  end
end
```

AccountChange ਕਲਾਸ ਇੱਕ ਟ੍ਰਗਿਰ ਸੁਨੇਹੇ ਵਜੋਂ ਕੰਮ ਕਰਦੀ ਹੈ ਜੋ ਖਾਤਾ ਬਦਲਣ ਦੀ ਬੇਨਤੀ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਇੱਕ ਪ੍ਰਕਰਿਆ ਸ਼ੁਰੂ ਕਰਦੀ ਹੈ। ਵੇਖੋ ਕਿ ਇਹ ਕਵਿੇਂ Olympia ਦੇ Wisper-ਆਧਾਰਤਿ ਪਬ/ਸਬ ਸਬਸਸਿਟਮ ਤੇ ਪ੍ਰਸਾਰਤਿ ਹੁੰਦੀ ਹੈ ਜਦੋਂ ਬਣਾਉਣ ਦੀ ਟ੍ਰਾਂਜੈਕਸ਼ਨ ਪੂਰੀ ਹੋ ਜਾਂਦੀ ਹੈ।

ਇਸ ਤਰ੍ਹਾਂ ਡਾਟਾਬੇਸ ਵਚਿ ਟ੍ਰਗਿਰ ਸੁਨੇਹੇ ਨੂੰ ਸਟੋਰ ਕਰਨਾ ਹਰ ਖਾਤਾ ਬਦਲਣ ਦੀ ਬੇਨਤੀ ਦਾ ਸਥਾਈ ਰਕਿਾਰਡ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। AccountChange ਕਲਾਸ ਦੀ ਹਰ ਇੰਸਟੈਂਸ ਨੂੰ ਇੱਕ ਵਲਿੱਖਣ ਪ੍ਰਾਇਮਰੀ ਕੀ ਨਰਿਧਾਰਤ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਜੋ ਵਅਿਕਤੀਗਤ ਬੇਨਤੀਆਂ ਦੀ ਆਸਾਨ ਪਛਾਣ ਅਤੇ ਟਰੈਕਂਗਿ ਦੀ ਸਹੂਲਤ ਦਂਦੀ ਹੈ। ਇਹ ਖਾਸ ਤੌਰ 'ਤੇ ਆਡਟਿ ਲੌਗਂਗਿ ਦੇ ਉਦੇਸ਼ਾਂ ਲਈ ਉਪਯੋਗੀ ਹੈ, ਕਉਂਕਿ ਇਹ ਸਸਿਟਮ ਨੂੰ ਸਾਰੇ ਖਾਤਾ ਬਦਲਾਵਾਂ ਦਾ ਇਤਹਿਾਸਕ ਰਕਿਾਰਡ ਬਣਾਈ ਰੱਖਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ, ਜਸਿ ਵਚਿ ਇਹ ਵੀ ਸ਼ਾਮਲ ਹੈ ਕਿ ਉਹ ਕਦੋਂ ਬੇਨਤੀ ਕੀਤੀ ਗਈ, ਕਹਿੜੇ ਬਦਲਾਵ ਦੀ ਬੇਨਤੀ ਕੀਤੀ ਗਈ, ਅਤੇ ਹਰ ਬੇਨਤੀ ਦੀ ਮੌਜੂਦਾ ਸਥਤਿੀ।

ਦਤਿੇ ਗਏ ਉਦਾਹਰਣ ਵਚਿ, AccountChange ਕਲਾਸ ਵਚਿ `description` ਵਰਗੇ ਫੀਲਡ ਸ਼ਾਮਲ ਹਨ ਜੋ ਬੇਨਤੀ ਕੀਤੇ ਬਦਲਾਵ ਦੇ ਵੇਰਵਆਿਂ ਨੂੰ ਕੈਪਚਰ ਕਰਦੇ ਹਨ, `state` ਬੇਨਤੀ ਦੀ ਮੌਜੂਦਾ ਸਥਤਿੀ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ (ਜਵਿੇਂ ਕਿ ਬੇਨਤੀ ਕੀਤੀ, ਪੂਰੀ ਹੋਈ, ਮਨੁੱਖੀ ਸਮੀਖਆਿ ਦੀ ਲੋੜ ਹੈ), ਅਤੇ `transcript` ਬੇਨਤੀ ਨਾਲ ਸੰਬੰਧਤਿ AI ਦੀ ਗੱਲਬਾਤ ਦੀ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਨੂੰ ਸਟੋਰ ਕਰਦਾ ਹੈ। `description` ਫੀਲਡ ਅਸਲ ਪ੍ਰੌਂਪਟ ਹੈ ਜੋ AI ਨਾਲ ਪਹਲਿੀ ਚੈਟ ਪੂਰਤੀ ਸ਼ੁਰੂ ਕਰਨ ਲਈ ਵਰਤਆਿ ਜਾਂਦਾ ਹੈ। ਇਸ ਡੇਟਾ ਨੂੰ ਸਟੋਰ ਕਰਨਾ ਮੁੱਲਵਾਨ ਸੰਦਰਭ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਅਤੇ ਖਾਤਾ ਬਦਲਣ ਦੀ ਪ੍ਰਕਰਿਆ ਦੀ ਬਹਤਿਰ ਟਰੈਕਂਗਿ ਅਤੇ ਵਸ਼ਿਲੇਸ਼ਣ ਦੀ ਆਗਆਿ ਦਂਦਾ ਹੈ।

ਡਾਟਾਬੇਸ ਵਚਿ ਟ੍ਰਗਿਰ ਸੁਨੇਹਆਿਂ ਨੂੰ ਸਟੋਰ ਕਰਨਾ ਮਜ਼ਬੂਤ ਗਲਤੀ ਪ੍ਰਬੰਧਨ ਅਤੇ ਰਕਿਵਰੀ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ। ਜੇਕਰ ਖਾਤਾ ਬਦਲਣ ਦੀ ਬੇਨਤੀ ਦੀ ਪ੍ਰਕਰਿਆ ਦੌਰਾਨ ਕੋਈ ਗਲਤੀ ਆਉਂਦੀ ਹੈ, ਤਾਂ ਸਸਿਟਮ ਬੇਨਤੀ ਨੂੰ ਅਸਫਲ ਦੇ ਤੌਰ 'ਤੇ ਚਨ੍ਹਿਤਿ ਕਰਦਾ ਹੈ ਅਤੇ ਇਸਨੂੰ ਇੱਕ ਅਜਹਿੀ ਸਥਤਿੀ ਵਚਿ ਬਦਲ ਦਂਦਾ ਹੈ ਜਸਿ ਨੂੰ ਮਨੁੱਖੀ

ਦਖਲ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕਿ ਕੋਈ ਬੇਨਤੀ ਗੁਆਚੀ ਜਾਂ ਭੁੱਲੀ ਨਾ ਜਾਵੇ, ਅਤੇ ਕਿਸੇ ਵੀ ਸਮੱਸਿਆ ਨੂੰ ਢੁਕਵੇਂ ਢੰਗ ਨਾਲ ਹੱਲ ਕੀਤਾ ਜਾ ਸਕੇ।

AI ਵਰਕਰ, ਪ੍ਰੋਸੈਸ ਮੈਨੇਜਰ ਵਜੋਂ, ਨਿਯੰਤਰਣ ਦਾ ਇੱਕ ਕੇਂਦਰੀ ਬਿੰਦੂ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਅਤੇ ਸ਼ਕਤੀਸ਼ਾਲੀ ਪ੍ਰਕਰਿਆ ਰਪਿਰਟਿੰਗ ਅਤੇ ਡੀਬੱਗਿੰਗ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ। ਹਾਲਾਂਕਿ, ਇਹ ਨੋਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ ਹਰ ਵਰਕਫਲੋ ਸਥਿਤੀ ਲਈ ਪ੍ਰੋਸੈਸ ਮੈਨੇਜਰ ਵਜੋਂ AI ਵਰਕਰ ਦੀ ਵਰਤੋਂ ਕਰਨਾ ਜ਼ਰੂਰਤ ਤੋਂ ਵੱਧ ਹੋ ਸਕਦਾ ਹੈ।

ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਆਰਕੀਟੈਕਚਰ ਵਿੱਚ AI ਵਰਕਰਾਂ ਨੂੰ ਏਕੀਕ੍ਰਿਤ ਕਰਨਾ

ਜਦੋਂ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਆਰਕੀਟੈਕਚਰ ਵਿੱਚ AI ਵਰਕਰਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਦੇ ਹੋ, ਤਾਂ AI ਵਰਕਰਾਂ ਅਤੇ ਹੋਰ ਐਪਲੀਕੇਸ਼ਨ ਕੰਪੋਨੈਂਟਾਂ ਵਿਚਕਾਰ ਨਿਰਵਿਘਨ ਏਕੀਕਰਣ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਸੰਚਾਰ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਕਈ ਤਕਨੀਕੀ ਵਿਚਾਰਾਂ ਨੂੰ ਸੰਬੋਧਿਤ ਕਰਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਇਹ ਭਾਗ ਇੰਟਰਫੇਸਾਂ ਨੂੰ ਡਿਜ਼ਾਈਨ ਕਰਨ, ਡੇਟਾ ਪ੍ਰਵਾਹ ਨੂੰ ਸੰਭਾਲਣ, ਅਤੇ AI ਵਰਕਰਾਂ ਦੇ ਜੀਵਨ-ਚੱਕਰ ਦੇ ਪ੍ਰਬੰਧਨ ਦੇ ਮੁੱਖ ਪਹਿਲੂਆਂ 'ਤੇ ਵਿਚਾਰ ਕਰਦਾ ਹੈ।

ਸਪੱਸ਼ਟ ਇੰਟਰਫੇਸ ਅਤੇ ਸੰਚਾਰ ਪ੍ਰੋਟੋਕੋਲ ਡਿਜ਼ਾਈਨ ਕਰਨਾ

AI ਵਰਕਰਾਂ ਅਤੇ ਹੋਰ ਐਪਲੀਕੇਸ਼ਨ ਕੰਪੋਨੈਂਟਾਂ ਵਿਚਕਾਰ ਨਿਰਵਿਘਨ ਏਕੀਕਰਣ ਦੀ ਸਹੂਲਤ ਲਈ, ਸਪੱਸ਼ਟ ਇੰਟਰਫੇਸ ਅਤੇ ਸੰਚਾਰ ਪ੍ਰੋਟੋਕੋਲ ਨਿਰਧਾਰਤ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਹੇਠ ਲਿਖੇ ਪਹੁੰਚਾਂ 'ਤੇ ਵਿਚਾਰ ਕਰੋ:

ਏ.ਪੀ.ਆਈ.-ਆਧਾਰਿਤ ਏਕੀਕਰਨ: ਏ.ਆਈ. ਵਰਕਰਾਂ ਦੀ ਕਾਰਜਸ਼ੀਲਤਾ ਨੂੰ ਚੰਗੀ ਤਰ੍ਹਾਂ ਪਰਿਭਾਸ਼ਿਤ ਏ.ਪੀ.ਆਈ. ਰਾਹੀਂ ਪੇਸ਼ ਕਰੋ, ਜਿਵੇਂ ਕਿ ਰੈਸਟਫੁੱਲ ਐਂਡਪੁਆਇੰਟਸ ਜਾਂ GraphQL ਸਕੀਮਾ। ਇਹ ਹੋਰ ਕੰਪੋਨੈਂਟਸ ਨੂੰ ਮਿਆਰੀ HTTP ਬੇਨਤੀਆਂ ਅਤੇ ਜਵਾਬਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਏ.ਆਈ. ਵਰਕਰਾਂ ਨਾਲ ਅੰਤਰਕਰਿਆ ਕਰਨ ਦੀ ਆਗਿਆ ਦੇਂਦਾ ਹੈ। ਏ.ਪੀ.ਆਈ.-ਆਧਾਰਿਤ ਏਕੀਕਰਨ ਏ.ਆਈ. ਵਰਕਰਾਂ ਅਤੇ ਵਰਤੋਂ ਕਰਨ ਵਾਲੇ ਕੰਪੋਨੈਂਟਸ ਵਿਚਕਾਰ ਇੱਕ ਸਪੱਸ਼ਟ ਇਕਰਾਰਨਾਮਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ, ਜੋ ਏਕੀਕਰਨ ਬਿੰਦੂਆਂ ਨੂੰ ਵਿਕਸਿਤ ਕਰਨ, ਟੈਸਟ ਕਰਨ ਅਤੇ ਬਣਾਈ ਰੱਖਣ ਨੂੰ ਸੌਖਾ ਬਣਾਉਂਦਾ ਹੈ।

ਸੁਨੇਹਾ-ਆਧਾਰਤਿ ਸੰਚਾਰ: ਸੁਨੇਹਾ-ਆਧਾਰਤਿ ਸੰਚਾਰ ਪੈਟਰਨ ਲਾਗੂ ਕਰੋ, ਜਵਿੇਂ ਕਿ ਸੁਨੇਹਾ ਕਤਾਰਾਂ ਜਾਂ ਪ੍ਰਕਾਸ਼ਤਿ-ਗਾਹਕੀ ਪ੍ਰਣਾਲੀਆਂ, ਜੋ ਏ.ਆਈ. ਵਰਕਰਾਂ ਅਤੇ ਹੋਰ ਕੰਪੋਨੈਂਟਸ ਵਚਿਕਾਰ ਅਸਿੰਕਰੋਨਸ ਅੰਤਰਕਰਿਆ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦੀਆਂ ਹਨ। ਇਹ ਪਹੁੰਚ ਏ.ਆਈ. ਵਰਕਰਾਂ ਨੂੰ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਬਾਕੀ ਹਸਿੱਆਂ ਤੋਂ ਵੱਖ ਕਰਦੀ ਹੈ, ਜੋ ਬਹਤਿਰ ਸਕੇਲੇਬਲਿਟੀ, ਗਲਤੀ ਸਹਣਿਸ਼ੀਲਤਾ, ਅਤੇ ਢੀਲੀ ਜੁੜਤ ਦੀ ਆਗਆਿ ਦੇਂਦੀ ਹੈ। ਸੁਨੇਹਾ-ਆਧਾਰਤਿ ਸੰਚਾਰ ਖਾਸ ਤੌਰ 'ਤੇ ਉਦੋਂ ਲਾਭਦਾਇਕ ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਏ.ਆਈ. ਵਰਕਰਾਂ ਦੁਆਰਾ ਕੀਤੀ ਜਾਂਦੀ ਪ੍ਰੋਸੈਸਿੰਗ ਸਮਾਂ-ਖਪਤ ਵਾਲੀ ਜਾਂ ਸਰੋਤ-ਗਹਣਿ ਹੁੰਦੀ ਹੈ, ਕਉਿਂਕਿ ਇਹ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਹੋਰ ਹਸਿੱਆਂ ਨੂੰ ਏ.ਆਈ. ਵਰਕਰਾਂ ਦੇ ਆਪਣੇ ਕੰਮ ਪੂਰੇ ਕਰਨ ਦੀ ਉਡੀਕ ਕੀਤੇ ਬਨਿਾਂ ਚੱਲਦੇ ਰਹਣਿ ਦੀ ਆਗਆਿ ਦੇਂਦਾ ਹੈ।

ਘਟਨਾ-ਚਾਲਤਿ ਆਰਕੀਟੈਕਚਰ: ਆਪਣੇ ਸਸਿਟਮ ਨੂੰ ਘਟਨਾਵਾਂ ਅਤੇ ਟਰਗਿਰਾਂ ਦੇ ਆਲੇ-ਦੁਆਲੇ ਡਜ਼ਾਈਨ ਕਰੋ ਜੋ ਖਾਸ ਹਾਲਾਤਾਂ ਦੇ ਪੂਰੇ ਹੋਣ 'ਤੇ ਏ.ਆਈ. ਵਰਕਰਾਂ ਨੂੰ ਸਰਗਰਮ ਕਰਦੇ ਹਨ। ਏ.ਆਈ. ਵਰਕਰ ਸੰਬੰਧਤਿ ਘਟਨਾਵਾਂ ਦੀ ਗਾਹਕੀ ਲੈ ਸਕਦੇ ਹਨ ਅਤੇ ਉਸ ਅਨੁਸਾਰ ਪ੍ਰਤੀਕਰਿਆ ਕਰ ਸਕਦੇ ਹਨ, ਘਟਨਾਵਾਂ ਦੇ ਵਾਪਰਨ 'ਤੇ ਆਪਣੇ ਨਰਿਧਾਰਤਿ ਕੰਮ ਕਰ ਸਕਦੇ ਹਨ। ਘਟਨਾ-ਚਾਲਤਿ ਆਰਕੀਟੈਕਚਰ ਰੀਅਲ-ਟਾਈਮ ਪ੍ਰੋਸੈਸਿੰਗ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ ਅਤੇ ਏ.ਆਈ. ਵਰਕਰਾਂ ਨੂੰ ਲੋੜ ਪੈਣ 'ਤੇ ਬੁਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ, ਜੋ ਗੈਰ-ਜ਼ਰੂਰੀ ਸਰੋਤ ਖਪਤ ਨੂੰ ਘਟਾਉਂਦਾ ਹੈ। ਇਹ ਪਹੁੰਚ ਉਨ੍ਹਾਂ ਸਥਤਿੀਆਂ ਲਈ ਢੁਕਵੀਂ ਹੈ ਜਥਿੇ ਏ.ਆਈ. ਵਰਕਰਾਂ ਨੂੰ ਖਾਸ ਕਾਰਵਾਈਆਂ ਜਾਂ ਐਪਲੀਕੇਸ਼ਨ ਸਥਤਿੀ ਵਚਿ ਤਬਦੀਲੀਆਂ ਦਾ ਜਵਾਬ ਦੇਣ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

ਡਾਟਾ ਪ੍ਰਵਾਹ ਅਤੇ ਸਿੰਕ੍ਰੋਨਾਈਜ਼ੇਸ਼ਨ ਨੂੰ ਸੰਭਾਲਣਾ

ਜਦੋਂ ਏ.ਆਈ. ਵਰਕਰਾਂ ਨੂੰ ਆਪਣੀ ਐਪਲੀਕੇਸ਼ਨ ਵਚਿ ਏਕੀਕ੍ਰਤਿ ਕਰ ਰਹੇ ਹੋ, ਤਾਂ ਸੁਚਾਰੂ ਡਾਟਾ ਪ੍ਰਵਾਹ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣਾ ਅਤੇ ਏ.ਆਈ. ਵਰਕਰਾਂ ਅਤੇ ਹੋਰ ਕੰਪੋਨੈਂਟਸ ਵਚਿਕਾਰ ਡਾਟਾ ਇਕਸਾਰਤਾ ਬਣਾਈ ਰੱਖਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਹੇਠ ਲਖਿੇ ਪਹਲਿੂਆਂ 'ਤੇ ਵਚਿਾਰ ਕਰੋ:

ਡਾਟਾ ਤਆਿਰੀ: ਏ.ਆਈ. ਵਰਕਰਾਂ ਵਚਿ ਡਾਟਾ ਫੀਡ ਕਰਨ ਤੋਂ ਪਹਲਿਾਂ, ਤੁਹਾਨੂੰ ਵੱਖ-ਵੱਖ ਡਾਟਾ ਤਆਿਰੀ ਕਾਰਜ ਕਰਨ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ, ਜਵਿੇਂ ਕਿ ਇਨਪੁਟ ਡਾਟਾ ਦੀ ਸਫਾਈ, ਫਾਰਮੈਟਿੰਗ, ਅਤੇ/ਜਾਂ ਰੂਪਾਂਤਰਣ। ਤੁਸੀਂ ਨਾ ਸਰਫਿ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਚਾਹੁੰਦੇ ਹੋ ਕਿ ਏ.ਆਈ. ਵਰਕਰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਪ੍ਰੋਸੈਸ ਕਰ ਸਕਣ, ਬਲਕਿ ਤੁਸੀਂ ਇਹ ਵੀ ਯਕੀਨੀ ਬਣਾਉਣਾ ਚਾਹੁੰਦੇ ਹੋ ਕਿ ਤੁਸੀਂ ਅਜਹਿੀ ਜਾਣਕਾਰੀ 'ਤੇ ਧਆਿਨ ਦੇਣ ਲਈ ਟੋਕਨ ਬਰਬਾਦ ਨਹੀਂ ਕਰ ਰਹੇ ਹੋ ਜੋ ਵਰਕਰ ਸਭ ਤੋਂ ਵਧੀਆ ਸਥਤਿੀ ਵਚਿ ਬੇਕਾਰ ਅਤੇ ਸਭ ਤੋਂ ਮਾੜੀ ਸਥਤਿੀ ਵਚਿ ਭਟਕਾਉਣ ਵਾਲੀ ਸਮਝ ਸਕਦਾ ਹੈ। ਡਾਟਾ ਤਆਿਰੀ ਵਚਿ ਸ਼ੋਰ ਹਟਾਉਣਾ, ਗੁੰਮ ਮੁੱਲਾਂ ਨੂੰ ਸੰਭਾਲਣਾ, ਜਾਂ ਡਾਟਾ ਕਸਿਮਾਂ ਨੂੰ ਬਦਲਣਾ ਸ਼ਾਮਲ ਹੋ ਸਕਦਾ ਹੈ।

ਡਾਟਾ ਸਥਰਿਤਾ: ਤੁਸੀਂ ਏ.ਆਈ. ਵਰਕਰਾਂ ਵਚਿ ਆਉਣ ਅਤੇ ਜਾਣ ਵਾਲੇ ਡਾਟਾ ਨੂੰ ਕਵਿੇਂ ਸਟੋਰ ਅਤੇ ਸਥਾਈ ਕਰੋਗੇ? ਡਾਟਾ ਦੀ ਮਾਤਰਾ, ਕੁਆਰੀ ਪੈਟਰਨ, ਅਤੇ ਸਕੇਲੇਬਲਿਟੀ ਵਰਗੇ ਕਾਰਕਾਂ 'ਤੇ ਵਚਿਾਰ ਕਰੋ। ਕੀ ਤੁਹਾਨੂੰ

ਆਡਿਟ ਜਾਂ ਡੀਬੱਗਿੰਗ ਦੇ ਉਦੇਸ਼ਾਂ ਲਈ ਏ.ਆਈ. ਦੇ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਨੂੰ ਇਸਦੀ "ਸੋਚ ਪ੍ਰਕਿਰਿਆ" ਦੇ ਪ੍ਰਤੀਬਿੰਬ ਵਜੋਂ ਸਥਾਈ ਕਰਨ ਦੀ ਲੋੜ ਹੈ, ਜਾਂ ਕੀ ਸਿਰਫ਼ ਨਤੀਜਿਆਂ ਦਾ ਰਿਕਾਰਡ ਰੱਖਣਾ ਕਾਫ਼ੀ ਹੈ?

ਡਾਟਾ ਪ੍ਰਾਪਤੀ: ਵਰਕਰਾਂ ਨੂੰ ਲੋੜੀਂਦਾ ਡਾਟਾ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਡਾਟਾਬੇਸਾਂ ਤੋਂ ਕਿਊਰੀ, ਫਾਈਲਾਂ ਤੋਂ ਪੜ੍ਹਨਾ, ਜਾਂ ਬਾਹਰੀ ਏਪੀਆਈ ਤੱਕ ਪਹੁੰਚ ਕਰਨੀ ਸ਼ਾਮਲ ਹੋ ਸਕਦੀ ਹੈ। ਲੇਟੈਂਸੀ ਅਤੇ ਏਆਈ ਵਰਕਰਾਂ ਦੀ ਸਭ ਤੋਂ ਤਾਜ਼ਾ ਡਾਟਾ ਤੱਕ ਪਹੁੰਚ ਬਾਰੇ ਵਿਚਾਰ ਕਰਨਾ ਯਕੀਨੀ ਬਣਾਓ। ਕੀ ਉਹਨਾਂ ਨੂੰ ਤੁਹਾਡੇ ਡਾਟਾਬੇਸ ਤੱਕ ਪੂਰੀ ਪਹੁੰਚ ਦੀ ਲੋੜ ਹੈ ਜਾਂ ਤੁਹਾਨੂੰ ਉਹਨਾਂ ਦੀ ਪਹੁੰਚ ਦਾ ਦਾਇਰਾ ਉਹਨਾਂ ਦੇ ਕੰਮ ਦੇ ਅਨੁਸਾਰ ਸੀਮਤ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ? ਸਕੇਲਿੰਗ ਬਾਰੇ ਕੀ? ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਬਿਹਤਰ ਬਣਾਉਣ ਅਤੇ ਅੰਡਰਲਾਈਂਗ ਡਾਟਾ ਸਰੋਤਾਂ 'ਤੇ ਲੋਡ ਘਟਾਉਣ ਲਈ ਕੈਚਿੰਗ ਵਿਧੀਆਂ 'ਤੇ ਵਿਚਾਰ ਕਰੋ।

ਡਾਟਾ ਸਿੰਕਿਰੋਨਾਈਜ਼ੇਸ਼ਨ: ਜਦੋਂ ਕਈ ਕੰਪੋਨੈਂਟਸ, ਏਆਈ ਵਰਕਰਾਂ ਸਮੇਤ, ਸਾਂਝੇ ਡਾਟਾ ਤੱਕ ਪਹੁੰਚ ਕਰਦੇ ਅਤੇ ਸੋਧ ਕਰਦੇ ਹਨ, ਤਾਂ ਡਾਟਾ ਸਥਿਰਤਾ ਬਣਾਈ ਰੱਖਣ ਲਈ ਢੁਕਵੇਂ ਸਿੰਕਿਰੋਨਾਈਜ਼ੇਸ਼ਨ ਮੈਕੇਨਜ਼ਿਮ ਲਾਗੂ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਡਾਟਾਬੇਸ ਲੌਕਿੰਗ ਰਣਨੀਤੀਆਂ, ਜਿਵੇਂ ਕਿ ਆਸ਼ਾਵਾਦੀ ਜਾਂ ਨਿਰਾਸ਼ਾਵਾਦੀ ਲੌਕਿੰਗ, ਟਕਰਾਵਾਂ ਨੂੰ ਰੋਕਣ ਅਤੇ ਡਾਟਾ ਅਖੰਡਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਵਿੱਚ ਤੁਹਾਡੀ ਮਦਦ ਕਰ ਸਕਦੀਆਂ ਹਨ। ਸਬੰਧਤਿ ਡਾਟਾ ਓਪਰੇਸ਼ਨਾਂ ਨੂੰ ਗਰੁੱਪ ਕਰਨ ਅਤੇ ਐਟੌਮਸਿਟੀ, ਕਨਸਿਸਟੈਂਸੀ, ਆਈਸੋਲੇਸ਼ਨ, ਅਤੇ ਡਿਊਰੇਬਲਿਟੀ (ACID) ਗੁਣਾਂ ਨੂੰ ਬਣਾਈ ਰੱਖਣ ਲਈ ਟ੍ਰਾਂਜ਼ੈਕਸ਼ਨ ਪ੍ਰਬੰਧਨ ਤਕਨੀਕਾਂ ਨੂੰ ਲਾਗੂ ਕਰੋ।

ਗਲਤੀ ਸੰਭਾਲ ਅਤੇ ਰਿਕਵਰੀ: ਡਾਟਾ ਫਲੋ ਪ੍ਰਕਿਰਿਆ ਦੌਰਾਨ ਪੈਦਾ ਹੋ ਸਕਣ ਵਾਲੇ ਡਾਟਾ-ਸਬੰਧਤਿ ਮੁੱਦਿਆਂ ਨਾਲ ਨਜਿੱਠਣ ਲਈ ਮਜ਼ਬੂਤ ਗਲਤੀ ਸੰਭਾਲ ਅਤੇ ਰਿਕਵਰੀ ਮੈਕੇਨਜ਼ਿਮ ਲਾਗੂ ਕਰੋ। ਡੀਬੱਗਿੰਗ ਵਿੱਚ ਸਹਾਇਤਾ ਲਈ ਅਪਵਾਦਾਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲੋ ਅਤੇ ਅਰਥਪੂਰਨ ਗਲਤੀ ਸੁਨੇਹੇ ਪ੍ਰਦਾਨ ਕਰੋ। ਅਸਥਾਈ ਅਸਫਲਤਾਵਾਂ ਜਾਂ ਨੈੱਟਵਰਕ ਵਿਘਨਾਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਮੁੜ-ਕੋਸ਼ਿਸ਼ ਮੈਕੇਨਜ਼ਿਮ ਅਤੇ ਫਾਲਬੈਕ ਰਣਨੀਤੀਆਂ ਲਾਗੂ ਕਰੋ। ਡਾਟਾ ਖਰਾਬੀ ਜਾਂ ਨੁਕਸਾਨ ਦੀ ਸਥਿਤੀ ਵਿੱਚ ਡਾਟਾ ਰਿਕਵਰੀ ਅਤੇ ਬਹਾਲੀ ਲਈ ਸਪੱਸ਼ਟ ਪ੍ਰਕਿਰਿਆਵਾਂ ਨਿਰਧਾਰਤ ਕਰੋ।

ਡਾਟਾ ਫਲੋ ਅਤੇ ਸਿੰਕਿਰੋਨਾਈਜ਼ੇਸ਼ਨ ਮੈਕੇਨਜ਼ਿਮ ਨੂੰ ਧਿਆਨ ਨਾਲ ਡਿਜ਼ਾਈਨ ਅਤੇ ਲਾਗੂ ਕਰਕੇ, ਤੁਸੀਂ ਯਕੀਨੀ ਬਣਾ ਸਕਦੇ ਹੋ ਕਿ ਤੁਹਾਡੇ ਏਆਈ ਵਰਕਰਾਂ ਕੋਲ ਸਹੀ, ਸਥਿਰ, ਅਤੇ ਅੱਪ-ਟੂ-ਡੇਟ ਡਾਟਾ ਤੱਕ ਪਹੁੰਚ ਹੈ। ਇਹ ਉਹਨਾਂ ਨੂੰ ਆਪਣੇ ਕਾਰਜ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਕਰਨ ਅਤੇ ਭਰੋਸੇਯੋਗ ਨਤੀਜੇ ਪੈਦਾ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ।

ਏਆਈ ਵਰਕਰਾਂ ਦੇ ਜੀਵਨ-ਚੱਕਰ ਦਾ ਪ੍ਰਬੰਧਨ

ਏਆਈ ਵਰਕਰਾਂ ਨੂੰ ਸ਼ੁਰੂ ਕਰਨ ਅਤੇ ਕੌਨਫਿਗਿਰ ਕਰਨ ਲਈ ਇੱਕ ਮਿਆਰੀ ਪ੍ਰਕਿਰਿਆ ਵਿਕਸਿਤ ਕਰੋ। ਮੈਂ ਅਜਿਹੇ ਫਰੇਮਵਰਕਾਂ ਦੇ ਪੱਖ ਵਿੱਚ ਹਾਂ ਜੋ ਮਾਡਲ ਨਾਮ, ਸਿਸਟਮ ਨਿਰਦੇਸ਼, ਅਤੇ ਫੰਕਸ਼ਨ ਪਰਿਭਾਸ਼ਾਵਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ

ਦੇ ਤਰੀਕੇ ਨੂੰ ਮਿਆਰੀ ਬਣਾਉਂਦੇ ਹਨ। ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਸ਼ੁਰੂਆਤੀ ਪ੍ਰਕਰਿਆ ਸਵੈਚਾਲਤਿ ਅਤੇ ਦੁਹਰਾਉਣਯੋਗ ਹੈ ਤਾਂ ਜੋ ਡਪਿਲਾਏਮੈਂਟ ਅਤੇ ਸਕੇਲਿੰਗ ਨੂੰ ਸੌਖਾ ਬਣਾਇਆ ਜਾ ਸਕੇ।

ਏਆਈ ਵਰਕਰਾਂ ਦੀ ਸਹਿਤ ਅਤੇ ਕਾਰਗੁਜ਼ਾਰੀ ਦੀ ਨਗਿਰਾਨੀ ਲਈ ਵਆਿਪਕ ਨਗਿਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ ਮੈਕੇਨਜ਼ਿਮ ਲਾਗੂ ਕਰੋ। ਸਰੋਤ ਵਰਤੋਂ, ਪ੍ਰੋਸੈਸਿੰਗ ਸਮਾਂ, ਗਲਤੀ ਦਰਾਂ, ਅਤੇ ਥਰੂਪੁੱਟ ਵਰਗੇ ਮੈਟ੍ਰਕਿਸ ਇਕੱਠੇ ਕਰੋ। ਕਈ ਏਆਈ ਵਰਕਰਾਂ ਤੋਂ ਲੌਗਸ ਨੂੰ ਇਕੱਠਾ ਅਤੇ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਨ ਲਈ ELK ਸਟੈਕ (Elasticsearch, Logstash, Kibana) ਵਰਗੀਆਂ ਕੇਂਦਰੀਕ੍ਰਤਿ ਲੌਗਿੰਗ ਪ੍ਰਣਾਲੀਆਂ ਦੀ ਵਰਤੋਂ ਕਰੋ।

AI ਵਰਕਰ ਆਰਕੀਟੈਕਚਰ ਵਚਿ ਗਲਤੀ ਸਹਣਿਸ਼ੀਲਤਾ ਅਤੇ ਲਚਕਤਾ ਨੂੰ ਸ਼ਾਮਲ ਕਰੋ। ਗਲਤੀਆਂ ਜਾਂ ਅਪਵਾਦਾਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲਣ ਲਈ ਗਲਤੀ ਪ੍ਰਬੰਧਨ ਅਤੇ ਰਕਿਵਰੀ ਵਧਿੀਆਂ ਨੂੰ ਲਾਗੂ ਕਰੋ। Large Language Models ਅਜੇ ਵੀ ਅਤ-ਿਆਧੁਨਕਿ ਤਕਨਾਲੋਜੀ ਹਨ; ਪ੍ਰਦਾਤਾ ਅਕਸਰ ਅਣਜਾਣੇ ਸਮਆਿਂ ’ਤੇ ਡਾਊਨ ਹੋ ਜਾਂਦੇ ਹਨ। ਲੜੀਬੱਧ ਅਸਫਲਤਾਵਾਂ ਨੂੰ ਰੋਕਣ ਲਈ ਮੁੜ-ਕੋਸ਼ਸ਼ਿ ਵਧਿੀਆਂ ਅਤੇ ਸਰਕਟ ਬਰੇਕਰਾਂ ਦੀ ਵਰਤੋਂ ਕਰੋ।

AI ਵਰਕਰਾਂ ਦੀ ਰਚਨਾਤਮਕਤਾ ਅਤੇ ਤਾਲਮੇਲ

AI ਵਰਕਰ ਆਰਕੀਟੈਕਚਰ ਦਾ ਇੱਕ ਮੁੱਖ ਫਾਇਦਾ ਇਸਦੀ ਰਚਨਾਤਮਕਤਾ ਹੈ, ਜੋ ਤੁਹਾਨੂੰ ਗੁੰਝਲਦਾਰ ਸਮੱਸਆਿਵਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਕਈ AI ਵਰਕਰਾਂ ਨੂੰ ਜੋੜਨ ਅਤੇ ਤਾਲਮੇਲ ਕਰਨ ਦੀ ਆਗਆਿ ਦੇਂਦਾ ਹੈ। ਇੱਕ ਵੱਡੇ ਕਾਰਜ ਨੂੰ ਛੋਟੇ, ਵਧੇਰੇ ਪ੍ਰਬੰਧਨਯੋਗ ਉਪ-ਕਾਰਜਾਂ ਵਚਿ ਵੰਡ ਕੇ, ਜਨ੍ਹਿਾਂ ਵਚਿੋਂ ਹਰੇਕ ਨੂੰ ਇੱਕ ਵਸ਼ੇਸ਼ AI ਵਰਕਰ ਦੁਆਰਾ ਨਯਿੰਤਰਤਿ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤੁਸੀਂ ਸ਼ਕਤੀਸ਼ਾਲੀ ਅਤੇ ਲਚਕਦਾਰ ਸਸਿਟਮ ਬਣਾ ਸਕਦੇ ਹੋ। ਇਸ ਭਾਗ ਵਚਿ, ਅਸੀਂ “ਬਹੁਤ ਸਾਰੇ” AI ਵਰਕਰਾਂ ਦੀ ਰਚਨਾ ਅਤੇ ਤਾਲਮੇਲ ਲਈ ਵੱਖ-ਵੱਖ ਪਹੁੰਚਾਂ ਦੀ ਪੜਚੋਲ ਕਰਾਂਗੇ।

ਬਹੁ-ਪੜਾਵੀ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਲਈ AI ਵਰਕਰਾਂ ਦੀ ਲੜੀਬੱਧਤਾ

ਕਈ ਸਥਤਿੀਆਂ ਵਚਿ, ਇੱਕ ਗੁੰਝਲਦਾਰ ਕਾਰਜ ਨੂੰ ਕ੍ਰਮਵਾਰ ਪੜਾਵਾਂ ਦੀ ਲੜੀ ਵਚਿ ਵੰਡਆਿ ਜਾ ਸਕਦਾ ਹੈ, ਜੱਥੇ ਇੱਕ AI ਵਰਕਰ ਦਾ ਆਉਟਪੁੱਟ ਅਗਲੇ ਲਈ ਇਨਪੁੱਟ ਬਣ ਜਾਂਦਾ ਹੈ। AI ਵਰਕਰਾਂ ਦੀ ਇਹ ਲੜੀਬੱਧਤਾ ਇੱਕ ਬਹੁ-ਪੜਾਵੀ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਜਾਂ ਪਾਈਪਲਾਈਨ ਬਣਾਉਂਦੀ ਹੈ। ਲੜੀ ਵਚਿ ਹਰੇਕ AI ਵਰਕਰ ਇੱਕ ਵਸ਼ੇਸ਼ ਉਪ-ਕਾਰਜ ’ਤੇ ਕੇਂਦਰਤਿ ਹੁੰਦਾ ਹੈ, ਅਤੇ ਅੰਤਮਿ ਆਉਟਪੁੱਟ ਸਾਰੇ ਵਰਕਰਾਂ ਦੇ ਸੰਯੁਕਤ ਯਤਨਾਂ ਦਾ ਨਤੀਜਾ ਹੁੰਦਾ ਹੈ।

ਆਓ Ruby on Rails ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਸੰਦਰਭ ਵਚਿ ਇੱਕ ਉਦਾਹਰਣ ’ਤੇ ਵਚਿਾਰ ਕਰੀਏ ਜੋ ਉਪਭੋਗਤਾ-ਨਰਿਮਤਿ ਸਮੱਗਰੀ ਦੀ ਪ੍ਰਕਰਿਆ ਲਈ ਹੈ। ਕਾਰਜ-ਪ੍ਰਵਾਹ ਵਚਿ ਹੇਠ ਲਖਿ ਪੜਾਅ ਸ਼ਾਮਲ ਹਨ, ਜੋ ਮੰਨਣਯੋਗ

ਤੌਰ 'ਤੇ ਸ਼ਾਇਦ ਅਸਲ ਜੀਵਨ ਦੀਆਂ ਵਰਤੋਂ ਸਥਤੀਆਂ ਵੱਚਿ ਇਸ ਤਰ੍ਹਾਂ ਵੰਡਣ ਲਈ ਬਹੁਤ ਸਰਲ ਹਨ, ਪਰ ਉਹ ਉਦਾਹਰਣ ਨੂੰ ਸਮਝਣ ਵੱਚਿ ਸੌਖਾ ਬਣਾਉਂਦੇ ਹਨ:

1. ਟੈਕਸਟ ਸਫਾਈ: ਇੱਕ AI ਵਰਕਰ ਜੋ HTML ਟੈਗਾਂ ਨੂੰ ਹਟਾਉਣ, ਟੈਕਸਟ ਨੂੰ ਲੋਅਰਕੇਸ ਵੱਚਿ ਬਦਲਣ, ਅਤੇ Unicode ਸਧਾਰਨੀਕਰਨ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਜ਼ਿੰਮੇਵਾਰ ਹੈ।

2. ਭਾਸ਼ਾ ਪਛਾਣ: ਇੱਕ AI ਵਰਕਰ ਜੋ ਸਾਫ਼ ਕੀਤੇ ਟੈਕਸਟ ਦੀ ਭਾਸ਼ਾ ਦੀ ਪਛਾਣ ਕਰਦਾ ਹੈ।

3. ਭਾਵਨਾ ਵਿਸ਼ਲੇਸ਼ਣ: ਇੱਕ AI ਵਰਕਰ ਜੋ ਪਛਾਣੀ ਗਈ ਭਾਸ਼ਾ ਦੇ ਆਧਾਰ 'ਤੇ ਟੈਕਸਟ ਦੀ ਭਾਵਨਾ (ਸਕਾਰਾਤਮਕ, ਨਕਾਰਾਤਮਕ, ਜਾਂ ਤਟਸਥ) ਨਰਿਧਾਰਤਿ ਕਰਦਾ ਹੈ।

4. ਸਮੱਗਰੀ ਵਰਗੀਕਰਨ: ਇੱਕ AI ਵਰਕਰ ਜੋ ਨੈਚੁਰਲ ਲੈਂਗੁਏਜ ਪ੍ਰੋਸੈਸਿੰਗ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਟੈਕਸਟ ਨੂੰ ਪਹਲਿਾਂ ਤੋਂ ਪਰਭਾਸ਼ਤਿ ਸ਼੍ਰੇਣੀਆਂ ਵੱਚਿ ਵਰਗੀਕ੍ਰਤਿ ਕਰਦਾ ਹੈ।

ਇੱਥੇ Ruby ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇਨ੍ਹਾਂ AI ਵਰਕਰਾਂ ਨੂੰ ਇਕੱਠੇ ਜੋੜਨ ਦਾ ਇੱਕ ਬਹੁਤ ਸਰਲੀਕ੍ਰਤਿ ਉਦਾਹਰਣ ਹੈ:

```ruby
class ContentProcessor
  def initialize(text)
    @text = text
  end

  def process
    cleaned_text = TextCleanupWorker.new(@text).call
    language = LanguageDetectionWorker.new(cleaned_text).call
    sentiment = SentimentAnalysisWorker.new(cleaned_text, language).call
    category = CategorizationWorker.new(cleaned_text, language).call

    { cleaned_text:, language:, sentiment:, category: }
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵੱਚਿ, `ContentProcessor` ਕਲਾਸ ਕੱਚੇ ਟੈਕਸਟ ਨਾਲ ਸ਼ੁਰੂ ਹੁੰਦੀ ਹੈ ਅਤੇ `process` ਮੈਥਡ ਵੱਚਿ AI ਵਰਕਰਾਂ ਨੂੰ ਇੱਕ ਲੜੀ ਵੱਚਿ ਜੋੜਦੀ ਹੈ। ਹਰ AI ਵਰਕਰ ਆਪਣਾ ਖਾਸ ਕੰਮ ਕਰਦਾ ਹੈ ਅਤੇ ਨਤੀਜੇ ਨੂੰ ਲੜੀ ਵੱਚਿ ਅਗਲੇ ਵਰਕਰ ਨੂੰ ਪਾਸ ਕਰਦਾ ਹੈ। ਅੰਤਮਿ ਆਉਟਪੁੱਟ ਇੱਕ ਹੈਸ਼ ਹੈ ਜਸਿ ਵੱਚਿ ਸਾਫ਼ ਕੀਤਾ ਟੈਕਸਟ, ਪਛਾਣੀ ਗਈ ਭਾਸ਼ਾ, ਭਾਵਨਾ, ਅਤੇ ਸਮੱਗਰੀ ਸ਼੍ਰੇਣੀ ਸ਼ਾਮਲ ਹੈ।

ਸੁਤੰਤਰ AI ਵਰਕਰਾਂ ਲਈ ਸਮਾਨੰਤਰ ਪ੍ਰੋਸੈਸਿੰਗ

ਪਛਿਲੀ ਉਦਾਹਰਣ ਵਿੱਚ, AI ਵਰਕਰ ਕ੍ਰਮਵਾਰ ਤਰੀਕੇ ਨਾਲ ਜੁੜੇ ਹੋਏ ਹਨ, ਜਿੱਥੇ ਹਰ ਵਰਕਰ ਟੈਕਸਟ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਦਾ ਹੈ ਅਤੇ ਨਤੀਜੇ ਨੂੰ ਅਗਲੇ ਵਰਕਰ ਨੂੰ ਪਾਸ ਕਰਦਾ ਹੈ। ਹਾਲਾਂਕਿ, ਜੇ ਤੁਹਾਡੇ ਕੋਲ ਕਈ AI ਵਰਕਰ ਹਨ ਜੋ ਇੱਕੋ ਇਨਪੁੱਟ 'ਤੇ ਸੁਤੰਤਰ ਤੌਰ 'ਤੇ ਕੰਮ ਕਰ ਸਕਦੇ ਹਨ, ਤਾਂ ਤੁਸੀਂ ਉਹਨਾਂ ਨੂੰ ਸਮਾਨੰਤਰ ਵਿੱਚ ਪ੍ਰੋਸੈਸ ਕਰਕੇ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦੇ ਹੋ।

ਦੱਤੀ ਗਈ ਸਥਿਤੀ ਵਿੱਚ, ਜਦੋਂ `TextCleanupWorker` ਦੁਆਰਾ ਟੈਕਸਟ ਦੀ ਸਫਾਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਤਾਂ `LanguageDetectionWorker`, `SentimentAnalysisWorker`, ਅਤੇ `Categorization-Worker` ਸਾਰੇ ਸਾਫ਼ ਕੀਤੇ ਟੈਕਸਟ 'ਤੇ ਸੁਤੰਤਰ ਤੌਰ 'ਤੇ ਪ੍ਰਕਰਿਆ ਕਰ ਸਕਦੇ ਹਨ। ਇਹਨਾਂ ਵਰਕਰਾਂ ਨੂੰ ਸਮਾਨੰਤਰ ਵਿੱਚ ਚਲਾ ਕੇ, ਤੁਸੀਂ ਸੰਭਾਵਤ ਤੌਰ 'ਤੇ ਸਮੁੱਚੀ ਪ੍ਰੋਸੈਸਿੰਗ ਦਾ ਸਮਾਂ ਘਟਾ ਸਕਦੇ ਹੋ ਅਤੇ ਆਪਣੇ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਦੀ ਕੁਸ਼ਲਤਾ ਨੂੰ ਵਧਾ ਸਕਦੇ ਹੋ।

Ruby ਵਿੱਚ ਸਮਾਨੰਤਰ ਪ੍ਰੋਸੈਸਿੰਗ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ, ਤੁਸੀਂ ਥ੍ਰੈਡਾਂ ਜਾਂ ਅਸਿੰਕ੍ਰੋਨਸ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਰਗੀਆਂ ਸਮਵਰਤੀ ਤਕਨੀਕਾਂ ਦਾ ਲਾਭ ਲੈ ਸਕਦੇ ਹੋ। ਇੱਥੇ ਇੱਕ ਉਦਾਹਰਣ ਹੈ ਕਿ ਤੁਸੀਂ ਥ੍ਰੈਡਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਆਖਰੀ ਤਿੰਨ ਵਰਕਰਾਂ ਨੂੰ ਸਮਾਨੰਤਰ ਵਿੱਚ ਪ੍ਰੋਸੈਸ ਕਰਨ ਲਈ `ContentProcessor` ਕਲਾਸ ਨੂੰ ਕਿਵੇਂ ਸੰਸ਼ੋਧਿਤ ਕਰ ਸਕਦੇ ਹੋ:

```ruby
require 'concurrent'

class ContentProcessor
  def initialize(text)
    @text = text
  end

  def process
    cleaned_text = TextCleanupWorker.new(@text).call

    language_future = Concurrent::Future.execute do
      LanguageDetectionWorker.new(cleaned_text).call
    end

    sentiment_future = Concurrent::Future.execute do
      SentimentAnalysisWorker.new(cleaned_text).call
    end

```

```
    category_future = Concurrent::Future.execute do
      CategorizationWorker.new(cleaned_text).call
    end

    language = language_future.value
    sentiment = sentiment_future.value
    category = category_future.value

    { cleaned_text:, language:, sentiment:, category: }
  end
end
```

ਇਸ ਔਪਟੀਮਾਈਜ਼ਡ ਵਰਜਨ ਵਿੱਚ, ਅਸੀਂ ਹਰੇਕ ਸੁਤੰਤਰ AI ਵਰਕਰਾਂ ਲਈ `Concurrent::Future` ਆਬਜੈਕਟਸ ਬਣਾਉਣ ਲਈ `concurrent-ruby` ਲਾਇਬ੍ਰੇਰੀ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਇੱਕ `Future` ਉਸ ਕੰਪਿਊਟੇਸ਼ਨ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਜੋ ਇੱਕ ਵੱਖਰੇ ਥਰੈੱਡ ਵਿੱਚ ਅਸਿੰਕਰੋਨਸ ਤਰੀਕੇ ਨਾਲ ਕੀਤੀ ਜਾਵੇਗੀ।

ਟੈਕਸਟ ਕਲੀਨਅੱਪ ਸਟੈੱਪ ਤੋਂ ਬਾਅਦ, ਅਸੀਂ ਤਿੰਨ `Future` ਆਬਜੈਕਟਸ ਬਣਾਉਂਦੇ ਹਾਂ: `language_future`, `sentiment_future`, ਅਤੇ `category_future`। ਹਰੇਕ `Future` ਆਪਣੇ ਸੰਬੰਧਿਤ AI ਵਰਕਰ (`LanguageDetectionWorker`, `SentimentAnalysisWorker`, ਅਤੇ `CategorizationWorker`) ਨੂੰ ਇੱਕ ਵੱਖਰੇ ਥਰੈੱਡ ਵਿੱਚ ਚਲਾਉਂਦਾ ਹੈ, ਜਿਸ ਵਿੱਚ `cleaned_text` ਇਨਪੁੱਟ ਵਜੋਂ ਪਾਸ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਹਰੇਕ `Future` 'ਤੇ `value` ਮੈਥਡ ਨੂੰ ਕਾਲ ਕਰਕੇ, ਅਸੀਂ ਕੰਪਿਊਟੇਸ਼ਨ ਦੇ ਪੂਰਾ ਹੋਣ ਦੀ ਉਡੀਕ ਕਰਦੇ ਹਾਂ ਅਤੇ ਨਤੀਜਾ ਪ੍ਰਾਪਤ ਕਰਦੇ ਹਾਂ। `value` ਮੈਥਡ ਉਦੋਂ ਤੱਕ ਬਲਾਕ ਕਰਦਾ ਹੈ ਜਦੋਂ ਤੱਕ ਨਤੀਜਾ ਉਪਲਬਧ ਨਹੀਂ ਹੁੰਦਾ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕਿ ਅੱਗੇ ਵਧਣ ਤੋਂ ਪਹਿਲਾਂ ਸਾਰੇ ਸਮਾਨੰਤਰ ਵਰਕਰਾਂ ਨੇ ਪ੍ਰੋਸੈਸਿੰਗ ਪੂਰੀ ਕਰ ਲਈ ਹੈ।

ਅੰਤ ਵਿੱਚ, ਅਸੀਂ ਮੂਲ ਉਦਾਹਰਣ ਵਾਂਗ ਹੀ ਸਾਫ਼ ਕੀਤੇ ਟੈਕਸਟ ਅਤੇ ਸਮਾਨੰਤਰ ਵਰਕਰਾਂ ਤੋਂ ਨਤੀਜਿਆਂ ਨਾਲ ਆਉਟਪੁੱਟ ਹੈਸ਼ ਬਣਾਉਂਦੇ ਹਾਂ।

ਸੁਤੰਤਰ AI ਵਰਕਰਾਂ ਨੂੰ ਸਮਾਨੰਤਰ ਵਿੱਚ ਪ੍ਰੋਸੈਸ ਕਰਕੇ, ਤੁਸੀਂ ਕ੍ਰਮਵਾਰ ਚਲਾਉਣ ਦੇ ਮੁਕਾਬਲੇ ਸਮੁੱਚੇ ਪ੍ਰੋਸੈਸਿੰਗ ਸਮੇਂ ਨੂੰ ਘਟਾ ਸਕਦੇ ਹੋ। ਇਹ ਔਪਟੀਮਾਈਜ਼ੇਸ਼ਨ ਖਾਸ ਤੌਰ 'ਤੇ ਸਮਾਂ-ਖਪਤ ਵਾਲੇ ਕੰਮਾਂ ਨਾਲ ਨਜਿੱਠਣ ਜਾਂ ਵੱਡੀ ਮਾਤਰਾ ਵਿੱਚ ਡੇਟਾ ਦੀ ਪ੍ਰੋਸੈਸਿੰਗ ਕਰਨ ਵੇਲੇ ਲਾਭਦਾਇਕ ਹੈ।

ਹਾਲਾਂਕਿ, ਇਹ ਨੋਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਅਸਲ ਪ੍ਰਦਰਸ਼ਨ ਲਾਭ ਕਈ ਕਾਰਕਾਂ 'ਤੇ ਨਿਰਭਰ ਕਰਦੇ ਹਨ, ਜਿਵੇਂ ਕਿ ਹਰੇਕ ਵਰਕਰ ਦੀ ਜਟਿਲਤਾ, ਉਪਲਬਧ ਸਿਸਟਮ ਸਰੋਤ, ਅਤੇ ਥਰੈੱਡ ਪ੍ਰਬੰਧਨ ਦਾ ਓਵਰਹੈੱਡ। ਆਪਣੇ

ਖਾਸ ਕੇਸ ਲਈ ਸਮਾਨੰਤਰਤਾ ਦਾ ਸਭ ਤੋਂ ਵਧੀਆ ਪੱਧਰ ਨਿਰਧਾਰਤ ਕਰਨ ਲਈ ਆਪਣੇ ਕੋਡ ਦਾ ਬੈਂਚਮਾਰਕ ਅਤੇ ਪ੍ਰੋਫਾਈਲ ਕਰਨਾ ਹਮੇਸ਼ਾ ਇੱਕ ਚੰਗੀ ਪ੍ਰੈਕਟਿਸ ਹੈ।

ਇਸ ਤੋਂ ਇਲਾਵਾ, ਸਮਾਨੰਤਰ ਪ੍ਰੋਸੈਸਿੰਗ ਨੂੰ ਲਾਗੂ ਕਰਦੇ ਸਮੇਂ, ਵਰਕਰਾਂ ਵਿਚਕਾਰ ਕਿਸੇ ਵੀ ਸਾਂਝੇ ਸਰੋਤਾਂ ਜਾਂ ਨਿਰਭਰਤਾਵਾਂ ਦਾ ਧਿਆਨ ਰੱਖੋ। ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਵਰਕਰ ਟਕਰਾਅ ਜਾਂ ਰੇਸ ਕੰਡੀਸ਼ਨਜ਼ ਤੋਂ ਬਿਨਾਂ ਸੁਤੰਤਰ ਰੂਪ ਵਿੱਚ ਕੰਮ ਕਰ ਸਕਦੇ ਹਨ। ਜੇਕਰ ਨਿਰਭਰਤਾਵਾਂ ਜਾਂ ਸਾਂਝੇ ਸਰੋਤ ਹਨ, ਤਾਂ ਤੁਹਾਨੂੰ ਡੇਟਾ ਅਖੰਡਤਾ ਬਣਾਈ ਰੱਖਣ ਅਤੇ ਡੈੱਡਲੌਕਸ ਜਾਂ ਅਸੰਗਤ ਨਤੀਜਿਆਂ ਵਰਗੀਆਂ ਸਮੱਸਿਆਵਾਂ ਤੋਂ ਬਚਣ ਲਈ ਢੁਕਵੇਂ ਸਿੰਕ੍ਰੋਨਾਈਜ਼ੇਸ਼ਨ ਮਕੈਨਿਜ਼ਮ ਲਾਗੂ ਕਰਨ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ।

Ruby ਦਾ ਗਲੋਬਲ ਇੰਟਰਪ੍ਰੇਟਰ ਲੌਕ ਅਤੇ ਅਸਿੰਕਰੋਨਸ ਪ੍ਰੋਸੈਸਿੰਗ

Ruby ਵਿੱਚ ਥਰੈੱਡ-ਅਧਾਰਿਤ ਅਸਿੰਕਰੋਨਸ ਪ੍ਰੋਸੈਸਿੰਗ 'ਤੇ ਵਿਚਾਰ ਕਰਦੇ ਸਮੇਂ Ruby ਦੇ ਗਲੋਬਲ ਇੰਟਰਪ੍ਰੇਟਰ ਲੌਕ (GIL) ਦੇ ਪ੍ਰਭਾਵਾਂ ਨੂੰ ਸਮਝਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ।

GIL Ruby ਦੇ ਇੰਟਰਪ੍ਰੇਟਰ ਵਿੱਚ ਇੱਕ ਮਕੈਨਿਜ਼ਮ ਹੈ ਜੋ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕਿ ਮਲਟੀ-ਕੋਰ ਪ੍ਰੋਸੈਸਰਾਂ 'ਤੇ ਵੀ ਇੱਕ ਸਮੇਂ 'ਤੇ ਸਿਰਫ਼ ਇੱਕ ਥਰੈੱਡ Ruby ਕੋਡ ਚਲਾ ਸਕਦਾ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਭਾਵੇਂ ਇੱਕ Ruby ਪ੍ਰੋਸੈੱਸ ਦੇ ਅੰਦਰ ਕਈ ਥਰੈੱਡ ਬਣਾਏ ਅਤੇ ਪ੍ਰਬੰਧਿਤ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ, ਕਿਸੇ ਵੀ ਸਮੇਂ ਸਿਰਫ਼ ਇੱਕ ਥਰੈੱਡ ਹੀ ਸਰਗਰਮੀ ਨਾਲ Ruby ਕੋਡ ਚਲਾ ਸਕਦਾ ਹੈ।

GIL ਨੂੰ Ruby ਇੰਟਰਪ੍ਰੇਟਰ ਦੀ ਲਾਗੂਕਰਨ ਨੂੰ ਸਰਲ ਬਣਾਉਣ ਅਤੇ Ruby ਦੇ ਅੰਦਰੂਨੀ ਡੇਟਾ ਢਾਂਚਿਆਂ ਲਈ ਥਰੈੱਡ ਸੁਰੱਖਿਆ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਡਿਜ਼ਾਈਨ ਕੀਤਾ ਗਿਆ ਹੈ। ਹਾਲਾਂਕਿ, ਇਹ Ruby ਕੋਡ ਦੇ ਸੱਚੇ ਸਮਾਨੰਤਰ ਕਾਰਜਕ੍ਰਮ ਦੀ ਸੰਭਾਵਨਾ ਨੂੰ ਵੀ ਸੀਮਿਤ ਕਰਦਾ ਹੈ।

ਜਦੋਂ ਤੁਸੀਂ Ruby ਵਿੱਚ ਥਰੈੱਡਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋ, ਜਿਵੇਂ ਕਿ `concurrent-ruby` ਲਾਇਬ੍ਰੇਰੀ ਜਾਂ ਬਿਲਟ-ਇਨ `Thread` ਕਲਾਸ ਨਾਲ, ਥਰੈੱਡਾਂ GIL ਦੀਆਂ ਸੀਮਾਵਾਂ ਦੇ ਅਧੀਨ ਹੁੰਦੀਆਂ ਹਨ। GIL ਹਰੇਕ ਥਰੈੱਡ ਨੂੰ ਥੋੜ੍ਹੇ ਸਮੇਂ ਲਈ Ruby ਕੋਡ ਚਲਾਉਣ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦਾ ਹੈ ਅਤੇ ਫਿਰ ਦੂਜੇ ਥਰੈੱਡ 'ਤੇ ਸਵਿੱਚ ਕਰਦਾ ਹੈ, ਜੋ ਸਮਵਰਤੀ ਕਾਰਜਕ੍ਰਮ ਦਾ ਭਰਮ ਪੈਦਾ ਕਰਦਾ ਹੈ।

ਹਾਲਾਂਕਿ, GIL ਕਾਰਨ, Ruby ਕੋਡ ਦਾ ਅਸਲ ਕਾਰਜਕ੍ਰਮ ਕ੍ਰਮਵਾਰ ਹੀ ਰਹਿੰਦਾ ਹੈ। ਜਦੋਂ ਇੱਕ ਥਰੈੱਡ Ruby ਕੋਡ ਚਲਾ ਰਹਿਆ ਹੁੰਦਾ ਹੈ, ਦੂਜੇ ਥਰੈੱਡ ਅਸਲ ਵਿੱਚ ਰੁਕੇ ਹੁੰਦੇ ਹਨ, GIL ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਅਤੇ ਕਾਰਜਕ੍ਰਮ ਚਲਾਉਣ ਦੀ ਆਪਣੀ ਵਾਰੀ ਦੀ ਉਡੀਕ ਕਰਦੇ ਹਨ।

ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ Ruby ਵਿੱਚ ਥਰੈੱਡ-ਆਧਾਰਿਤ ਅਸਿੰਕ੍ਰੋਨਸ ਪ੍ਰੋਸੈਸਿੰਗ I/O-ਬਾਊਂਡ ਕਾਰਜਾਂ ਲਈ ਸਭ ਤੋਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਹੈ, ਜਦਿਆਂ ਕਿ ਬਾਹਰੀ API ਜਵਾਬਾਂ ਦੀ ਉਡੀਕ ਕਰਨਾ (ਜਦਿਆਂ ਕਿ ਤੀਜੀ-ਧਿਰ ਦੁਆਰਾ ਹੋਸਟ ਕੀਤੇ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ) ਜਾਂ ਫਾਈਲ I/O ਓਪਰੇਸ਼ਨ ਕਰਨਾ। ਜਦੋਂ ਕੋਈ ਥਰੈੱਡ I/O ਓਪਰੇਸ਼ਨ ਨਾਲ ਟਕਰਾਉਂਦਾ ਹੈ, ਇਹ GIL ਨੂੰ ਛੱਡ ਸਕਦਾ ਹੈ, ਜੋ I/O ਦੇ ਪੂਰਾ ਹੋਣ ਦੀ ਉਡੀਕ ਕਰਦੇ ਹੋਏ ਦੂਜੇ ਥਰੈੱਡਾਂ ਨੂੰ ਚੱਲਣ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦਾ ਹੈ।

ਦੂਜੇ ਪਾਸੇ, CPU-ਬਾਊਂਡ ਕਾਰਜਾਂ ਲਈ, ਜਦਿਆਂ ਕਿ ਗਹਿਰੀ ਗਣਨਾ ਜਾਂ ਲੰਬੇ-ਸਮੇਂ ਦੀ AI ਵਰਕਰ ਪ੍ਰੋਸੈਸਿੰਗ, GIL ਥਰੈੱਡ-ਆਧਾਰਿਤ ਸਮਾਨੰਤਰਤਾ ਦੇ ਸੰਭਾਵੀ ਪ੍ਰਦਰਸ਼ਨ ਲਾਭਾਂ ਨੂੰ ਸੀਮਿਤ ਕਰ ਸਕਦਾ ਹੈ। ਕਿਉਂਕਿ ਇੱਕ ਸਮੇਂ 'ਤੇ ਸਿਰਫ਼ ਇੱਕ ਥਰੈੱਡ Ruby ਕੋਡ ਚਲਾ ਸਕਦਾ ਹੈ, ਸਮੁੱਚਾ ਕਾਰਜਕ੍ਰਮ ਸਮਾਂ ਕ੍ਰਮਵਾਰ ਪ੍ਰੋਸੈਸਿੰਗ ਦੇ ਮੁਕਾਬਲੇ ਮਹੱਤਵਪੂਰਨ ਤੌਰ 'ਤੇ ਘੱਟ ਨਹੀਂ ਹੋ ਸਕਦਾ।

Ruby ਵਿੱਚ CPU-ਬਾਊਂਡ ਕਾਰਜਾਂ ਲਈ ਸੱਚੀ ਸਮਾਨੰਤਰ ਕਾਰਜਕ੍ਰਮ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ, ਤੁਹਾਨੂੰ ਵਿਕਲਪਿਕ ਪਹੁੰਚਾਂ ਦੀ ਪੜਚੋਲ ਕਰਨ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ, ਜਦਿਆਂ ਕਿ:

- ਵੱਖ-ਵੱਖ CPU ਕੋਰ 'ਤੇ ਚੱਲ ਰਹੀਆਂ ਕਈ Ruby ਪ੍ਰਕਰਿਆਵਾਂ ਨਾਲ ਪ੍ਰਕਰਿਆ-ਆਧਾਰਿਤ ਸਮਾਨੰਤਰਤਾ ਦੀ ਵਰਤੋਂ ਕਰਨਾ।
- ਬਾਹਰੀ ਲਾਇਬ੍ਰੇਰੀਆਂ ਜਾਂ ਫਰੇਮਵਰਕਾਂ ਦਾ ਲਾਭ ਲੈਣਾ ਜੋ ਨੇਟਿਵ ਐਕਸਟੈਨਸ਼ਨਾਂ ਜਾਂ GIL ਤੋਂ ਬਨਿਾਂ ਭਾਸ਼ਾਵਾਂ ਲਈ ਇੰਟਰਫੇਸ ਪ੍ਰਦਾਨ ਕਰਦੀਆਂ ਹਨ, ਜਦਿਆਂ ਕਿ C ਜਾਂ Rust।,
- ਕਈ ਮਸ਼ੀਨਾਂ ਜਾਂ ਪ੍ਰਕਰਿਆਵਾਂ ਵਿੱਚ ਕਾਰਜਾਂ ਨੂੰ ਵੰਡਣ ਲਈ ਵੰਡੀ ਹੋਈ ਕੰਪਿਊਟਿੰਗ ਫਰੇਮਵਰਕਾਂ ਜਾਂ ਸੁਨੇਹਾ ਕਤਾਰਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨਾ।

Ruby ਵਿੱਚ ਅਸਿੰਕ੍ਰੋਨਸ ਪ੍ਰੋਸੈਸਿੰਗ ਨੂੰ ਡਿਜ਼ਾਈਨ ਅਤੇ ਲਾਗੂ ਕਰਦੇ ਸਮੇਂ ਆਪਣੇ ਕਾਰਜਾਂ ਦੀ ਪ੍ਰਕਰਿਤੀ ਅਤੇ GIL ਦੁਆਰਾ ਲਗਾਈਆਂ ਸੀਮਾਵਾਂ 'ਤੇ ਵਿਚਾਰ ਕਰਨਾ ਬਹੁਤ ਜ਼ਰੂਰੀ ਹੈ। ਜਦੋਂ ਕਿ ਥਰੈੱਡ-ਆਧਾਰਿਤ ਅਸਿੰਕ੍ਰੋਨਸ ਪ੍ਰੋਸੈਸਿੰਗ I/O-ਬਾਊਂਡ ਕਾਰਜਾਂ ਲਈ ਲਾਭ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੀ ਹੈ, GIL ਦੀਆਂ ਸੀਮਾਵਾਂ ਕਾਰਨ ਇਹ CPU-ਬਾਊਂਡ ਕਾਰਜਾਂ ਲਈ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਦਰਸ਼ਨ ਸੁਧਾਰ ਪ੍ਰਦਾਨ ਨਹੀਂ ਕਰ ਸਕਦੀ।

ਬਹਿਤਰ ਸ਼ੁੱਧਤਾ ਲਈ ਐਨਸੈਂਬਲ ਤਕਨੀਕਾਂ

ਐਨਸੈਂਬਲ ਤਕਨੀਕਾਂ ਵਿੱਚ ਸਿਸਟਮ ਦੀ ਸਮੁੱਚੀ ਸ਼ੁੱਧਤਾ ਜਾਂ ਮਜ਼ਬੂਤੀ ਨੂੰ ਸੁਧਾਰਨ ਲਈ ਕਈ AI ਵਰਕਰਾਂ ਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਜੋੜਨਾ ਸ਼ਾਮਲ ਹੈ। ਇੱਕ ਇਕੱਲੇ AI ਵਰਕਰ 'ਤੇ ਭਰੋਸਾ ਕਰਨ ਦੀ ਬਜਾਏ, ਐਨਸੈਂਬਲ ਤਕਨੀਕਾਂ

ਵਧੇਰੇ ਜਾਣਕਾਰੀ ਭਰਪੂਰ ਫੈਸਲੇ ਲੈਣ ਲਈ ਕਈ ਵਰਕਰਾਂ ਦੀ ਸਮੂਹਿਕ ਬੁੱਧੀ ਦਾ ਲਾਭ ਲੈਂਦੀਆਂ ਹਨ।

ਐਨਸੈਂਬਲ ਖਾਸ ਤੌਰ 'ਤੇ ਮਹੱਤਵਪੂਰਨ ਹੁੰਦੇ ਹਨ ਜੇਕਰ ਤੁਹਾਡੇ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਦੇ ਵੱਖ-ਵੱਖ ਹੱਸੇ ਵੱਖ-ਵੱਖ ਏ.ਆਈ. ਮਾਡਲਾਂ ਨਾਲ ਬਹਿਤਰ ਕੰਮ ਕਰਦੇ ਹਨ, ਜੋ ਕਿ ਤੁਹਾਡੇ ਸੋਚਣ ਨਾਲੋਂ ਵੱਧ ਆਮ ਹੈ। GPT-4 ਵਰਗੇ ਸ਼ਕਤੀਸ਼ਾਲੀ ਮਾਡਲ ਘੱਟ ਸਮਰੱਥਾ ਵਾਲੇ ਓਪਨ ਸੋਰਸ ਵਿਕਲਪਾਂ ਦੇ ਮੁਕਾਬਲੇ ਬਹੁਤ ਮਹੰਗੇ ਹਨ, ਅਤੇ ਸ਼ਾਇਦ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਹਰ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਕਦਮ ਲਈ ਜ਼ਰੂਰੀ ਨਹੀਂ ਹਨ।

ਇੱਕ ਆਮ ਐਨਸੈਂਬਲ ਤਕਨੀਕ ਬਹੁਮਤ ਵੋਟਿੰਗ ਹੈ, ਜੱਥੇ ਕਈ ਏ.ਆਈ. ਵਰਕਰ ਸੁਤੰਤਰ ਤੌਰ 'ਤੇ ਇੱਕੋ ਇਨਪੁੱਟ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਦੇ ਹਨ, ਅਤੇ ਅੰਤਮਿ ਆਉਟਪੁੱਟ ਬਹੁਮਤ ਸਹਮਿਤੀ ਦੁਆਰਾ ਨਰਿਧਾਰਤਿ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਪਹੁੰਚ ਵਅਿਕਤੀਗਤ ਵਰਕਰ ਗਲਤੀਆਂ ਦੇ ਪ੍ਰਭਾਵ ਨੂੰ ਘਟਾਉਣ ਅਤੇ ਸਸਿਟਮ ਦੀ ਸਮੁੱਚੀ ਭਰੋਸੇਯੋਗਤਾ ਨੂੰ ਬਹਿਤਰ ਬਣਾਉਣ ਵੱਚਿ ਮਦਦ ਕਰ ਸਕਦੀ ਹੈ।

ਆਓ ਇੱਕ ਉਦਾਹਰਣ 'ਤੇ ਵਚਿਾਰ ਕਰੀਏ ਜੱਥੇ ਸਾਡੇ ਕੋਲ ਭਾਵਨਾ ਵਸ਼ਿਲੇਸ਼ਣ ਲਈ ਤੰਨ ਿਏ.ਆਈ. ਵਰਕਰ ਹਨ, ਹਰੇਕ ਵੱਖਰੇ ਮਾਡਲ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ ਜਾਂ ਵੱਖਰੇ ਸੰਦਰਭਾਂ ਨਾਲ ਪ੍ਰਦਾਨ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਅਸੀਂ ਅੰਤਮਿ ਭਾਵਨਾ ਭਵਖਿਬਾਣੀ ਨੂੰ ਨਰਿਧਾਰਤਿ ਕਰਨ ਲਈ ਬਹੁਮਤ ਵੋਟਿੰਗ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਉਹਨਾਂ ਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਜੋੜ ਸਕਦੇ ਹਾਂ।

```
class SentimentAnalysisEnsemble
  def initialize(text)
    @text = text
  end

  def analyze
    predictions = [
      SentimentAnalysisWorker1.new(@text).analyze,
      SentimentAnalysisWorker2.new(@text).analyze,
      SentimentAnalysisWorker3.new(@text).analyze
    ]

    predictions
      .group_by { |sentiment| sentiment }
      .max_by { |_, votes| votes.size }
      .first

  end
end
```

ਇਸ ਉਦਾਹਰਨ ਵੱਚਿ, `SentimentAnalysisEnsemble` ਕਲਾਸ ਟੈਕਸਟ ਨਾਲ ਸ਼ੁਰੂ ਹੁੰਦੀ ਹੈ ਅਤੇ ਤੰਨਿ ਵੱਖ-ਵੱਖ ਭਾਵਨਾ ਵਸਿ਼ਲੇਸ਼ਣ ਏ.ਆਈ. ਵਰਕਰਾਂ ਨੂੰ ਬੁਲਾਉਂਦੀ ਹੈ। `analyze` ਮੈਥਡ ਹਰ ਵਰਕਰ ਤੋਂ ਭਵੱਖਬਾਣੀਆਂ ਇਕੱਠੀਆਂ ਕਰਦੀ ਹੈ ਅਤੇ `group_by` ਅਤੇ `max_by` ਮੈਥਡਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਬਹੁਮਤ ਭਾਵਨਾ ਨਰਿਧਾਰਤ ਕਰਦੀ ਹੈ। ਅੰਤਮਿ ਆਉਟਪੁੱਟ ਉਹ ਭਾਵਨਾ ਹੈ ਜੋ ਵਰਕਰਾਂ ਦੇ ਸਮੂਹ ਤੋਂ ਸਭ ਤੋਂ ਵੱਧ ਵੋਟਾਂ ਪ੍ਰਾਪਤ ਕਰਦੀ ਹੈ

ਸਮੂਹ ਸਪੱਸ਼ਟ ਤੌਰ 'ਤੇ ਅਜਹਿਾ ਕੇਸ ਹਨ ਜੱਥੇ ਸਮਾਨੰਤਰਤਾ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਨਾ ਤੁਹਾਡੇ ਸਮੇਂ ਦੇ ਯੋਗ ਹੋ ਸਕਦਾ ਹੈ।

ਏ.ਆਈ. ਵਰਕਰਾਂ ਦੀ ਗਤੀਸ਼ੀਲ ਚੋਣ ਅਤੇ ਬੁਲਾਹਟ

ਕੁਝ ਜੇ ਨਾ ਜ਼ਿਆਦਾਤਰ ਮਾਮਲਿਆਂ ਵੱਚਿ, ਬੁਲਾਏ ਜਾਣ ਵਾਲੇ ਖਾਸ ਏ.ਆਈ. ਵਰਕਰ ਰਨਟਾਈਮ ਹਾਲਾਤਾਂ ਜਾਂ ਯੂਜ਼ਰ ਇਨਪੁੱਟ 'ਤੇ ਨਰਿਭਰ ਕਰ ਸਕਦੇ ਹਨ। ਏ.ਆਈ. ਵਰਕਰਾਂ ਦੀ ਗਤੀਸ਼ੀਲ ਚੋਣ ਅਤੇ ਬੁਲਾਹਟ ਸਸਿਟਮ ਵੱਚਿ ਲਚਕਤਾ ਅਤੇ ਅਨੁਕੂਲਤਾ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦੀ ਹੈ।

ਤੁਸੀਂ ਆਪਣੇ ਆਪ ਨੂੰ ਇੱਕ ਏ.ਆਈ. ਵਰਕਰ ਵੱਚਿ ਬਹੁਤ ਸਾਰੀ ਕਾਰਜਸ਼ੀਲਤਾ ਫੱਟਿ ਕਰਨ ਦੀ ਕੋਸ਼ਸ਼ਿ ਕਰਦੇ ਹੋਏ ਪਾ ਸਕਦੇ ਹੋ, ਇਸ ਨੂੰ ਕਈ ਫੰਕਸ਼ਨ ਅਤੇ ਇੱਕ ਵੱਡਾ ਗੁੰਝਲਦਾਰ ਪ੍ਰੌਮਪਟ ਦੇ ਕੇ ਜੋ ਇਹਨਾਂ ਨੂੰ ਕਵਿੇਂ ਬੁਲਾਉਣਾ ਹੈ ਦੱਸਦਾ ਹੈ। ਇਸ ਲਲਚਾਵੇ ਦਾ ਵਰਿੋਧ ਕਰੋ, ਮੇਰਾ ਯਕੀਨ ਕਰੋ। ਇਸ ਅਧਆਿਇ ਵੱਚਿ ਜਸਿ ਪਹੁੰਚ ਬਾਰੇ ਅਸੀਂ ਚਰਚਾ ਕਰ ਰਹੇ ਹਾਂ ਉਸਨੂੰ "ਕਰਮਚਾਰੀਆਂ ਦੀ ਬਹੁਤਾਤ" ਕਹਿਾ ਜਾਂਦਾ ਹੈ, ਇਹ ਸਾਨੂੰ ਯਾਦ ਕਰਵਾਉਣ ਲਈ ਕ ਿਬਹੁਤ ਸਾਰੇ ਵਸ਼ਿਸ਼ ਵਰਕਰ ਹੋਣਾ ਵਾਂਛਨੀਯ ਹੈ, ਹਰ ਇੱਕ ਵੱਡੇ ਉਦੇਸ਼ ਦੀ ਸੇਵਾ ਵੱਚਿ ਆਪਣਾ ਛੋਟਾ ਕੰਮ ਕਰ ਰਹਿਾ ਹੈ।

ਉਦਾਹਰਨ ਲਈ, ਇੱਕ ਚੈਟਬੌਟ ਐਪਲੀਕੇਸ਼ਨ 'ਤੇ ਵਚਿਾਰ ਕਰੋ ਜੱਥੇ ਵੱਖ-ਵੱਖ ਏ.ਆਈ. ਵਰਕਰ ਵੱਖ-ਵੱਖ ਕਸਿਮਾਂ ਦੇ ਯੂਜ਼ਰ ਸਵਾਲਾਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਜ਼ਮਿੇਵਾਰ ਹਨ। ਯੂਜ਼ਰ ਦੇ ਇਨਪੁੱਟ ਦੇ ਆਧਾਰ 'ਤੇ, ਐਪਲੀਕੇਸ਼ਨ ਗਤੀਸ਼ੀਲ ਤੌਰ 'ਤੇ ਸਵਾਲ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਨ ਲਈ ਢੁਕਵੇਂ ਏ.ਆਈ. ਵਰਕਰ ਦੀ ਚੋਣ ਕਰਦੀ ਹੈ।

```
class ChatbotController < ApplicationController
  def process_query
    query = params[:query]
    query_type = QueryClassifierWorker.new(query).classify

    case query_type
    when 'greeting'
      response = GreetingWorker.new(query).generate_response
    when 'product_inquiry'
      response = ProductInquiryWorker.new(query).generate_response
    when 'order_status'
      response = OrderStatusWorker.new(query).generate_response
    else
      response = DefaultResponseWorker.new(query).generate_response
    end

    render json: { response: response }
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ, `ChatbotController` ਯੂਜ਼ਰ ਦੀ ਪੁੱਛਗਿੱਛ ਨੂੰ `process_query` ਐਕਸ਼ਨ ਰਾਹੀਂ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ। ਇਹ ਪਹਿਲਾਂ `QueryClassifierWorker` ਦੀ ਵਰਤੋਂ ਪੁੱਛਗਿੱਛ ਦੀ ਕਿਸਮ ਨਿਰਧਾਰਤ ਕਰਨ ਲਈ ਕਰਦਾ ਹੈ। ਵਰਗੀਕ੍ਰਤਿ ਪੁੱਛਗਿੱਛ ਦੀ ਕਿਸਮ ਦੇ ਆਧਾਰ 'ਤੇ, ਕੰਟਰੋਲਰ ਗਤੀਸ਼ੀਲ ਤੌਰ 'ਤੇ ਢੁਕਵੇਂ AI ਵਰਕਰ ਨੂੰ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਲਈ ਚੁਣਦਾ ਹੈ। ਇਹ ਗਤੀਸ਼ੀਲ ਚੋਣ ਚੈਟਬੌਟ ਨੂੰ ਵੱਖ-ਵੱਖ ਕਿਸਮ ਦੀਆਂ ਪੁੱਛਗਿੱਛਾਂ ਨੂੰ ਸੰਭਾਲਣ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਢੁਕਵੇਂ AI ਵਰਕਰਾਂ ਵੱਲ ਭੇਜਣ ਦੀ ਸਮਰੱਥਾ ਦਿੰਦੀ ਹੈ।

ਕਿਉਂਕਿ `QueryClassifierWorker` ਦਾ ਕੰਮ ਤੁਲਨਾਤਮਕ ਤੌਰ 'ਤੇ ਸਧਾਰਨ ਹੈ ਅਤੇ ਇਸ ਨੂੰ ਬਹੁਤ ਜ਼ਿਆਦਾ ਸੰਦਰਭ ਜਾਂ ਫੰਕਸ਼ਨ ਪਰਿਭਾਸ਼ਾਵਾਂ ਦੀ ਲੋੜ ਨਹੀਂ ਹੈ, ਤੁਸੀਂ ਸ਼ਾਇਦ ਇਸਨੂੰ ਅਲਟਰਾ-ਫਾਸਟ ਛੋਟੇ LLM ਜਿੱਥੇ ਕਿ `mistralai/mixtral-8x7b-instruct:nitro` ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਲਾਗੂ ਕਰ ਸਕਦੇ ਹੋ। ਇਸ ਵਿੱਚ ਕਈ ਕਾਰਜਾਂ 'ਤੇ GPT-4 ਪੱਧਰ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਹਨ ਅਤੇ ਜਦੋਂ ਮੈਂ ਇਹ ਲਿਖ ਰਿਹਾ ਹਾਂ, Groq ਇਸਨੂੰ 444 ਟੋਕਨ/ਸਕਿੰਟ ਦੀ ਤੇਜ਼ ਗਤੀ ਨਾਲ ਪੇਸ਼ ਕਰ ਸਕਦਾ ਹੈ।

ਪਰੰਪਰਾਗਤ NLP ਨੂੰ LLMs ਨਾਲ ਜੋੜਨਾ

ਭਾਵੇਂ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ (LLMs) ਨੇ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਪ੍ਰੋਸੈਸਿੰਗ (NLP) ਦੇ ਖੇਤਰ ਵਿੱਚ ਕ੍ਰਾਂਤੀ ਲਿਆਂਦੀ ਹੈ, ਜੋ ਕਿ ਵੱਖ-ਵੱਖ ਕਾਰਜਾਂ ਲਈ ਬੇਮਿਸਾਲ ਬਹੁਮੁੱਖਤਾ ਅਤੇ ਪ੍ਰਦਰਸ਼ਨ ਪੇਸ਼ ਕਰਦੇ ਹਨ, ਪਰ ਇਹ ਹਰ ਸਮੱਸਿਆ ਲਈ ਸਭ ਤੋਂ ਕੁਸ਼ਲ ਜਾਂ ਲਾਗਤ-ਪ੍ਰਭਾਵੀ ਹੱਲ ਨਹੀਂ ਹਨ। ਕਈ ਮਾਮਲਿਆਂ ਵਿੱਚ, ਪਰੰਪਰਾਗਤ NLP ਤਕਨੀਕਾਂ ਨੂੰ LLMs ਨਾਲ ਜੋੜਨ ਨਾਲ ਵਿਸ਼ੇਸ਼ NLP ਚੁਣੌਤੀਆਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਵਧੇਰੇ ਅਨੁਕੂਲਤਾ, ਲਕਸ਼ਿਤ, ਅਤੇ ਕਫਿਾਇਤੀ ਪਹੁੰਚਾਂ ਪ੍ਰਾਪਤ ਹੋ ਸਕਦੀਆਂ ਹਨ।

LLMs ਨੂੰ NLP ਦੇ ਸਵਿਸ ਆਰਮੀ ਚਾਕੂ ਵਜੋਂ ਸੋਚੋ - ਬੇਹੱਦ ਬਹੁਮੁੱਖੀ ਅਤੇ ਸ਼ਕਤੀਸ਼ਾਲੀ, ਪਰ ਜ਼ਰੂਰੀ ਨਹੀਂ ਕਿ ਹਰ ਕੰਮ ਲਈ ਸਭ ਤੋਂ ਵਧੀਆ ਟੂਲ ਹੋਵੇ। ਕਈ ਵਾਰ, ਵਾਈਨ ਦੀ ਬੋਤਲ ਖੋਲ੍ਹਣ ਵਾਲਾ ਜਾਂ ਡੱਬਾ ਖੋਲ੍ਹਣ ਵਾਲਾ ਵਰਗਾ ਸਮਰਪਿਤ ਟੂਲ ਕਿਸੇ ਖਾਸ ਕੰਮ ਲਈ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਅਤੇ ਕੁਸ਼ਲ ਹੋ ਸਕਦਾ ਹੈ। ਇਸੇ ਤਰ੍ਹਾਂ, ਪਰੰਪਰਾਗਤ NLP ਤਕਨੀਕਾਂ, ਜਿਵੇਂ ਕਿ ਦਸਤਾਵੇਜ਼ ਕਲੱਸਟਰਿੰਗ, ਵਿਸ਼ਾ ਪਛਾਣ, ਅਤੇ ਵਰਗੀਕਰਨ, ਅਕਸਰ ਤੁਹਾਡੀ NLP ਪਾਈਪਲਾਈਨ ਦੇ ਕੁਝ ਪਹਲੂਆਂ ਲਈ ਵਧੇਰੇ ਲਕਸ਼ਿਤ ਅਤੇ ਲਾਗਤ-ਪ੍ਰਭਾਵੀ ਹੱਲ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੀਆਂ ਹਨ।

ਪਰੰਪਰਾਗਤ NLP ਤਕਨੀਕਾਂ ਦਾ ਇੱਕ ਮੁੱਖ ਫਾਇਦਾ ਉਨ੍ਹਾਂ ਦੀ ਕੰਪਿਊਟੇਸ਼ਨਲ ਕੁਸ਼ਲਤਾ ਹੈ। ਇਹ ਵਧਿੀਆਂ, ਜੋ ਅਕਸਰ ਸਧਾਰਨ ਅੰਕੜਾ ਮਾਡਲਾਂ ਜਾਂ ਨਿਯਮ-ਆਧਾਰਿਤ ਪਹੁੰਚਾਂ 'ਤੇ ਨਿਰਭਰ ਕਰਦੀਆਂ ਹਨ, LLMs ਦੇ ਮੁਕਾਬਲੇ ਘੱਟ ਕੰਪਿਊਟੇਸ਼ਨਲ ਓਵਰਹੈੱਡ ਨਾਲ ਵੱਡੀ ਮਾਤਰਾ ਵਿੱਚ ਟੈਕਸਟ ਡੇਟਾ ਨੂੰ ਬਹੁਤ ਤੇਜ਼ੀ ਨਾਲ ਪ੍ਰੋਸੈੱਸ ਕਰ ਸਕਦੀਆਂ ਹਨ। ਇਹ ਉਨ੍ਹਾਂ ਨੂੰ ਖਾਸ ਤੌਰ 'ਤੇ ਦਸਤਾਵੇਜ਼ਾਂ ਦੇ ਵੱਡੇ ਸੰਗ੍ਰਹਿ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਅਤੇ ਸੰਗਠਨ ਕਰਨ ਵਾਲੇ ਕਾਰਜਾਂ ਲਈ ਢੁਕਵਾਂ ਬਣਾਉਂਦਾ ਹੈ, ਜਿਵੇਂ ਕਿ ਸਮਾਨ ਲੇਖਾਂ ਨੂੰ ਕਲੱਸਟਰ ਕਰਨਾ ਜਾਂ ਟੈਕਸਟ ਦੇ ਸੰਗ੍ਰਹਿ ਵਿੱਚ ਮੁੱਖ ਵਿਸ਼ਿਆਂ ਦੀ ਪਛਾਣ ਕਰਨਾ।

ਇਸ ਤੋਂ ਇਲਾਵਾ, ਪਰੰਪਰਾਗਤ ਐੱਨ.ਐੱਲ.ਪੀ. ਤਕਨੀਕਾਂ ਅਕਸਰ ਖਾਸ ਕਾਰਜਾਂ ਲਈ ਉੱਚ ਸ਼ੁੱਧਤਾ ਅਤੇ ਸਟੀਕਤਾ ਪ੍ਰਾਪਤ ਕਰ ਸਕਦੀਆਂ ਹਨ, ਖਾਸ ਕਰਕੇ ਜਦੋਂ ਖੇਤਰ-ਵਿਸ਼ਿਸ਼ ਡੇਟਾਸੈੱਟਾਂ 'ਤੇ ਸਿਖਲਾਈ ਦਿੱਤੀ ਜਾਂਦੀ ਹੈ। ਉਦਾਹਰਨ ਲਈ, Support Vector Machines (SVM) ਜਾਂ Naive Bayes ਵਰਗੇ ਪਰੰਪਰਾਗਤ ਮਸ਼ੀਨ ਲਰਨਿੰਗ ਐਲਗੋਰਿਦਮਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇੱਕ ਚੰਗੀ ਤਰ੍ਹਾਂ ਟਿਊਨ ਕੀਤਾ ਦਸਤਾਵੇਜ਼ ਵਰਗੀਕਰਣ ਘੱਟੋ-ਘੱਟ ਕੰਪਿਊਟੇਸ਼ਨਲ ਲਾਗਤ ਨਾਲ ਦਸਤਾਵੇਜ਼ਾਂ ਨੂੰ ਪਹਲਿਾਂ ਤੋਂ ਨਿਰਧਾਰਤ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਸਹੀ ਢੰਗ ਨਾਲ ਵਰਗੀਕ੍ਰਿਤ ਕਰ ਸਕਦਾ ਹੈ।

ਹਾਲਾਂਕਿ, ਐੱਲ.ਐੱਲ.ਐੱਮ. ਉਹਨਾਂ ਕਾਰਜਾਂ ਵਿੱਚ ਸੱਚਮੁੱਚ ਚਮਕਦੇ ਹਨ ਜਿੱਥੇ ਭਾਸ਼ਾ, ਸੰਦਰਭ, ਅਤੇ ਤਰਕ ਦੀ ਡੂੰਘੀ ਸਮਝ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਸੁਸੰਗਤ ਅਤੇ ਸੰਦਰਭਕ ਤੌਰ 'ਤੇ ਢੁਕਵਾਂ ਟੈਕਸਟ ਤਿਆਰ ਕਰਨ, ਸਵਾਲਾਂ ਦੇ ਜਵਾਬ ਦੇਣ,

ਅਤੇ ਲੰਬੇ ਪੈਰ੍ਹਿਆਂ ਨੂੰ ਸੰਖੇਪ ਕਰਨ ਦੀ ਉਨ੍ਹਾਂ ਦੀ ਯੋਗਤਾ ਪਰੰਪਰਾਗਤ ਐਨ.ਐਲ.ਪੀ. ਵਿਧੀਆਂ ਦੁਆਰਾ ਬੇਜੋੜ ਹੈ। ਐਲ.ਐਲ.ਐਮ. ਗੁੰਝਲਦਾਰ ਭਾਸ਼ਾਈ ਵਿਰਤਾਰਿਆਂ, ਜਿਵੇਂ ਕਿ ਅਸਪੱਸ਼ਟਤਾ, ਸਹਿ-ਹਵਾਲਾ, ਅਤੇ ਮੁਹਾਵਰੇਦਾਰ ਅਭਿਵਿਅਕਤੀਆਂ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਸੰਭਾਲ ਸਕਦੇ ਹਨ, ਜੋ ਉਨ੍ਹਾਂ ਨੂੰ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਉਤਪਾਦਨ ਜਾਂ ਸਮਝ ਦੀ ਲੋੜ ਵਾਲੇ ਕਾਰਜਾਂ ਲਈ ਅਮੁੱਲ ਬਣਾਉਂਦੇ ਹਨ।

ਅਸਲ ਤਾਕਤ ਦੋਵਾਂ ਦੀਆਂ ਤਾਕਤਾਂ ਦਾ ਲਾਭ ਲੈਣ ਲਈ ਪਰੰਪਰਾਗਤ ਐਨ.ਐਲ.ਪੀ. ਤਕਨੀਕਾਂ ਨੂੰ ਐਲ.ਐਲ.ਐਮ. ਨਾਲ ਜੋੜ ਕੇ ਹਾਈਬ੍ਰਿਡ ਪਹੁੰਚਾਂ ਬਣਾਉਣ ਵਿੱਚ ਹੈ। ਦਸਤਾਵੇਜ਼ ਪ੍ਰੀਪ੍ਰੋਸੈਸਿੰਗ, ਕਲੱਸਟਰਿੰਗ, ਅਤੇ ਵਿਸ਼ਾ ਕੱਢਣ ਵਰਗੇ ਕਾਰਜਾਂ ਲਈ ਪਰੰਪਰਾਗਤ ਐਨ.ਐਲ.ਪੀ. ਵਿਧੀਆਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਤੁਸੀਂ ਆਪਣੇ ਟੈਕਸਟ ਡੇਟਾ ਨੂੰ ਕੁਸ਼ਲਤਾ ਨਾਲ ਸੰਗਠਿਤ ਅਤੇ ਢਾਂਚਾਗਤ ਕਰ ਸਕਦੇ ਹੋ। ਇਹ ਢਾਂਚਾਗਤ ਜਾਣਕਾਰੀ ਫਿਰ ਸੰਖੇਪ ਤਿਆਰ ਕਰਨ, ਸਵਾਲਾਂ ਦੇ ਜਵਾਬ ਦੇਣ, ਜਾਂ ਵਿਆਪਕ ਰਿਪੋਰਟਾਂ ਬਣਾਉਣ ਵਰਗੇ ਵਧੇਰੇ ਉੱਨਤ ਕਾਰਜਾਂ ਲਈ ਐਲ.ਐਲ.ਐਮ. ਵਿੱਚ ਪਾਈ ਜਾ ਸਕਦੀ ਹੈ।

ਉਦਾਹਰਨ ਲਈ, ਆਓ ਇੱਕ ਅਜਿਹੀ ਵਿਰਤੋਂ ਦੀ ਉਦਾਹਰਨ 'ਤੇ ਵਿਚਾਰ ਕਰੀਏ ਜਿੱਥੇ ਤੁਸੀਂ ਵਿਅਕਤੀਗਤ ਰੁਝਾਨ ਦਸਤਾਵੇਜ਼ਾਂ ਦੇ ਇੱਕ ਵੱਡੇ ਸੰਗ੍ਰਹਿ ਦੇ ਆਧਾਰ 'ਤੇ ਕਿਸੇ ਖਾਸ ਖੇਤਰ ਲਈ ਰੁਝਾਨ ਰਿਪੋਰਟ ਤਿਆਰ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹੋ। ਸਿਰਫ਼ ਐਲ.ਐਲ.ਐਮ. 'ਤੇ ਨਿਰਭਰ ਕਰਨ ਦੀ ਬਜਾਏ, ਜੋ ਵੱਡੀ ਮਾਤਰਾ ਵਿੱਚ ਟੈਕਸਟ ਦੀ ਪ੍ਰੋਸੈਸਿੰਗ ਲਈ ਕੰਪਿਊਟੇਸ਼ਨਲ ਤੌਰ 'ਤੇ ਮਹਿੰਗਾ ਅਤੇ ਸਮਾਂ ਲੈਣ ਵਾਲਾ ਹੋ ਸਕਦਾ ਹੈ, ਤੁਸੀਂ ਇੱਕ ਹਾਈਬ੍ਰਿਡ ਪਹੁੰਚ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹੋ:

1. ਵਿਸ਼ਾ ਮਾਡਲਿੰਗ (ਜਿਵੇਂ ਕਿ Latent Dirichlet Allocation) ਜਾਂ ਕਲੱਸਟਰਿੰਗ ਐਲਗੋਰਿਦਮ (ਜਿਵੇਂ ਕਿ K-means) ਵਰਗੀਆਂ ਪਰੰਪਰਾਗਤ ਐਨ.ਐਲ.ਪੀ. ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰੋ, ਤਾਂ ਜੋ ਸਮਾਨ ਰੁਝਾਨ ਦਸਤਾਵੇਜ਼ਾਂ ਨੂੰ ਇਕੱਠਾ ਕੀਤਾ ਜਾ ਸਕੇ ਅਤੇ ਸੰਗ੍ਰਹਿ ਵਿੱਚ ਮੁੱਖ ਥੀਮਾਂ ਅਤੇ ਵਿਸ਼ਿਆਂ ਦੀ ਪਛਾਣ ਕੀਤੀ ਜਾ ਸਕੇ।
2. ਕਲੱਸਟਰ ਕੀਤੇ ਦਸਤਾਵੇਜ਼ਾਂ ਅਤੇ ਪਛਾਣੇ ਗਏ ਵਿਸ਼ਿਆਂ ਨੂੰ ਐਲ.ਐਲ.ਐਮ. ਵਿੱਚ ਫੀਡ ਕਰੋ, ਹਰੇਕ ਕਲੱਸਟਰ ਜਾਂ ਵਿਸ਼ੇ ਲਈ ਸੁਸੰਗਤ ਅਤੇ ਜਾਣਕਾਰੀ ਭਰਪੂਰ ਸੰਖੇਪ ਬਣਾਉਣ ਲਈ ਇਸਦੀ ਬਹਿਤਰ ਭਾਸ਼ਾ ਸਮਝ ਅਤੇ ਉਤਪਾਦਨ ਸਮਰੱਥਾਵਾਂ ਦਾ ਲਾਭ ਲੈਂਦੇ ਹੋਏ।
3. ਅੰਤ ਵਿੱਚ, ਵਿਅਕਤੀਗਤ ਸੰਖੇਪਾਂ ਨੂੰ ਜੋੜ ਕੇ, ਸਭ ਤੋਂ ਮਹੱਤਵਪੂਰਨ ਰੁਝਾਨਾਂ ਨੂੰ ਉਜਾਗਰ ਕਰਦੇ ਹੋਏ, ਅਤੇ ਇਕੱਤਰ ਕੀਤੀ ਜਾਣਕਾਰੀ ਦੇ ਆਧਾਰ 'ਤੇ ਅੰਤਰਦ੍ਰਸ਼ਿਟੀਆਂ ਅਤੇ ਸਿਫਾਰਸ਼ਾਂ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹੋਏ ਇੱਕ ਵਿਆਪਕ ਰੁਝਾਨ ਰਿਪੋਰਟ ਤਿਆਰ ਕਰਨ ਲਈ ਐਲ.ਐਲ.ਐਮ. ਦੀ ਵਰਤੋਂ ਕਰੋ।

ਇਸ ਤਰ੍ਹਾਂ ਪਰੰਪਰਾਗਤ ਐਨ.ਐਲ.ਪੀ. ਤਕਨੀਕਾਂ ਨੂੰ ਐਲ.ਐਲ.ਐਮ. ਨਾਲ ਜੋੜ ਕੇ, ਤੁਸੀਂ ਵੱਡੀ ਮਾਤਰਾ ਵਿੱਚ ਟੈਕਸਟ ਡੇਟਾ ਨੂੰ ਕੁਸ਼ਲਤਾ ਨਾਲ ਪ੍ਰੋਸੈਸ ਕਰ ਸਕਦੇ ਹੋ, ਅਰਥਪੂਰਨ ਅੰਤਰਦ੍ਰਸ਼ਿਟੀਆਂ ਕੱਢ ਸਕਦੇ ਹੋ, ਅਤੇ

ਕੰਪਿਊਟੇਸ਼ਨਲ ਸਰੋਤਾਂ ਅਤੇ ਲਾਗਤਾਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਂਦੇ ਹੋਏ ਉੱਚ-ਗੁਣਵੱਤਾ ਵਾਲੀਆਂ ਰਪਿਰਟਾਂ ਤਿਆਰ ਕਰ ਸਕਦੇ ਹੋ।

ਜਦੋਂ ਤੁਸੀਂ ਆਪਣੇ ਐੱਨ.ਐੱਲ.ਪੀ. ਪ੍ਰੋਜੈਕਟਾਂ 'ਤੇ ਕੰਮ ਸ਼ੁਰੂ ਕਰਦੇ ਹੋ, ਇਹ ਜ਼ਰੂਰੀ ਹੈ ਕਿ ਹਰ ਕਾਰਜ ਦੀਆਂ ਵਸ਼ਿਸ਼ ਲੋੜਾਂ ਅਤੇ ਸੀਮਾਵਾਂ ਦਾ ਧਿਆਨ ਨਾਲ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇ ਅਤੇ ਇਹ ਵਿਚਾਰਿਆ ਜਾਵੇ ਕਿ ਸਭ ਤੋਂ ਵਧੀਆ ਨਤੀਜੇ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਰਵਾਇਤੀ ਐੱਨ.ਐੱਲ.ਪੀ. ਵਿਧੀਆਂ ਅਤੇ ਐੱਲ.ਐੱਲ.ਐੱਮ. ਨੂੰ ਇਕੱਠੇ ਕਿਵੇਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਰਵਾਇਤੀ ਤਕਨੀਕਾਂ ਦੀ ਕੁਸ਼ਲਤਾ ਅਤੇ ਸੁੱਧਤਾ ਨੂੰ ਐੱਲ.ਐੱਲ.ਐੱਮ. ਦੀ ਬਹੁਮੁੱਖਤਾ ਅਤੇ ਸ਼ਕਤੀ ਨਾਲ ਜੋੜ ਕੇ, ਤੁਸੀਂ ਅਜਿਹੇ ਬੇਹੱਦ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਅਤੇ ਕਿਫਾਇਤੀ ਐੱਨ.ਐੱਲ.ਪੀ. ਹੱਲ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਤੁਹਾਡੇ ਉਪਭੋਗਤਾਵਾਂ ਅਤੇ ਹਿੱਸੇਦਾਰਾਂ ਨੂੰ ਮੁੱਲ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

ਟੂਲ ਵਰਤੋਂ

ਏਆਈ-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨ ਵਕਿਾਸ ਦੇ ਖੇਤਰ ਵਚਿ, "ਟੂਲ ਵਰਤੋਂ" ਜਾਂ "ਫੰਕਸ਼ਨ ਕਾਲਿੰਗ" ਦੀ ਧਾਰਨਾ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਤਕਨੀਕ ਵਜੋਂ ਉਭਰੀ ਹੈ ਜੋ ਤੁਹਾਡੇ LLM ਨੂੰ ਬਾਹਰੀ ਟੂਲਜ਼, APIs, ਫੰਕਸ਼ਨਜ਼, ਡਾਟਾਬੇਸਾਂ, ਅਤੇ ਹੋਰ ਸਰੋਤਾਂ ਨਾਲ ਜੋੜਨ ਦੀ ਸਮਰੱਥਾ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ। ਇਹ ਪਹੁੰਚ ਸਰਿਫ਼ ਟੈਕਸਟ ਆਉਟਪੁੱਟ ਕਰਨ ਨਾਲੋਂ ਵਧੇਰੇ ਵਵਿਹਾਰਾਂ ਦੀ ਇੱਕ ਵਸ਼ਿਾਲ ਸ਼੍ਰੇਣੀ ਦੀ ਆਗਿਆ ਦੇਂਦੀ ਹੈ, ਅਤੇ ਤੁਹਾਡੇ ਏਆਈ ਕੰਪੋਨੈਂਟਸ ਅਤੇ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਪਾਰਸਿਥਤਿੀ ਤੰਤਰ ਦੇ ਬਾਕੀ ਹੰਸਿਆਂ ਵਚਿਕਾਰ ਵਧੇਰੇ ਗਤੀਸ਼ੀਲ ਅੰਤਰਕਰਿਆਵਾਂ ਨੂੰ ਸੰਭਵ ਬਣਾਉਂਦੀ ਹੈ। ਜਵਿੇਂ ਕ ਿਅਸੀ ਇਸ ਅਧਆਇ ਵਚਿ ਵੇਖਾਂਗੇ, ਟੂਲ ਵਰਤੋਂ ਤੁਹਾਨੂੰ ਆਪਣੇ ਏਆਈ ਮਾਡਲ ਨੂੰ ਸੰਰਚਤਿ ਤਰੀਕਆਿਂ ਨਾਲ ਡੇਟਾ ਤਆਿਰ ਕਰਨ ਦਾ ਵਕਿਲਪ ਵੀ ਦੇਂਦੀ ਹੈ।

ਟੂਲ ਵਰਤੋਂ ਕੀ ਹੈ?

ਟੂਲ ਵਰਤੋਂ, ਜਸਿਨੂੰ ਫੰਕਸ਼ਨ ਕਾਲਿੰਗ ਵੀ ਕਹਿਾ ਜਾਂਦਾ ਹੈ, ਇੱਕ ਤਕਨੀਕ ਹੈ ਜੋ ਡਵੈਲਪਰਾਂ ਨੂੰ ਫੰਕਸ਼ਨਾਂ ਦੀ ਸੂਚੀ ਨਰਿਧਾਰਤ ਕਰਨ ਦੀ ਆਗਿਆ ਦੇਂਦੀ ਹੈ ਜਨ੍ਹਿਾਂ ਨਾਲ LLM ਜਨਰੇਸ਼ਨ ਪ੍ਰਕਰਿਆ ਦੌਰਾਨ ਅੰਤਰਕਰਿਆ ਕਰ

ਸਕਦਾ ਹੈ। ਇਹ ਟੂਲਜ਼ ਸਧਾਰਨ ਯੂਟਿਲਿਟੀ ਫੰਕਸ਼ਨਾਂ ਤੋਂ ਲੈ ਕੇ ਗੁੰਝਲਦਾਰ APIs ਜਾਂ ਡਾਟਾਬੇਸ ਕੁਐਰੀਆਂ ਤੱਕ ਹੋ ਸਕਦੇ ਹਨ। LLM ਨੂੰ ਇਨ੍ਹਾਂ ਟੂਲਜ਼ ਤੱਕ ਪਹੁੰਚ ਪ੍ਰਦਾਨ ਕਰਕੇ, ਡਵੈਲਪਰ ਮਾਡਲ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਵਧਾ ਸਕਦੇ ਹਨ ਅਤੇ ਇਸਨੂੰ ਅਜਿਹੇ ਕਾਰਜ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾ ਸਕਦੇ ਹਨ ਜਨ੍ਹਿਾਂ ਲਈ ਬਾਹਰੀ ਗਿਆਨ ਜਾਂ ਕਾਰਵਾਈਆਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

ਚੰਤਿਰ 7. ਦਸਤਾਵੇਜ਼ਾਂ ਦਾ ਵਿਸ਼੍ਲੇਸ਼ਣ ਕਰਨ ਵਾਲੇ ਏਆਈ ਵਰਕਰ ਲਈ ਫੰਕਸ਼ਨ ਪਰਿਭਾਸ਼ਾ ਦੀ ਉਦਾਹਰਣ

```
FUNCTION = {
  name: "save_analysis",
  description: "Save analysis data for document",
  parameters: {
    type: "object",
    properties: {
      title: {
        type: "string",
        maxLength: 140
      },
      summary: {
        type: "string",
        description: "comprehensive multi-paragraph summary with
                      overview and list of sections (if applicable)"
      },
      tags: {
        type: "array",
        items: {
          type: "string",
          description: "lowercase tags representing main themes
                        of the document"
        }
      }
    },
    "required": %w[title summary tags]
  }
}.freeze
```

ਟੂਲ ਦੀ ਵਰਤੋਂ ਦੇ ਪੀਛੇ ਮੁੱਖ ਵਿਚਾਰ ਐੱਲ.ਐੱਲ.ਐੱਮ. ਨੂੰ ਉਪਭੋਗਤਾ ਦੇ ਇਨਪੁੱਟ ਜਾਂ ਕਾਰਜ ਦੇ ਆਧਾਰ 'ਤੇ ਢੁਕਵੇਂ ਟੂਲਾਂ ਦੀ ਚੋਣ ਅਤੇ ਕਾਰਜਾਂਵਿਤ ਕਰਨ ਦੀ ਯੋਗਤਾ ਦੇਣਾ ਹੈ। ਮਾਡਲ ਦੇ ਪਹਲਿਾਂ ਤੋਂ ਸਖਿਲਾਈ ਪ੍ਰਾਪਤ ਗਿਆਨ 'ਤੇ ਨਰਿਭਰ ਕਰਨ ਦੀ ਬਜਾਏ, ਟੂਲ ਦੀ ਵਰਤੋਂ ਐੱਲ.ਐੱਲ.ਐੱਮ. ਨੂੰ ਵਧੇਰੇ ਸਟੀਕ, ਢੁਕਵੇਂ, ਅਤੇ ਕਾਰਜਸ਼ੀਲ ਜਵਾਬ

ਤਿਆਰ ਕਰਨ ਲਈ ਬਾਹਰੀ ਸਰੋਤਾਂ ਦਾ ਲਾਭ ਲੈਣ ਦੀ ਆਗਿਆ ਦੇਂਦੀ ਹੈ। ਟੂਲ ਦੀ ਵਰਤੋਂ ਆਰ.ਏ.ਜੀ. (ਪੁਨਰ-ਪ੍ਰਾਪਤੀ ਵਧਾਇਆ ਉਤਪਾਦਨ) ਵਰਗੀਆਂ ਤਕਨੀਕਾਂ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਬਹੁਤ ਸੌਖਾ ਬਣਾ ਦੇਂਦੀ ਹੈ।

> ਧਿਆਨ ਦਿਓ ਕਿ ਜਦੋਂ ਤੱਕ ਹੋਰ ਨਹੀਂ ਦੱਸਿਆ ਜਾਂਦਾ, ਇਹ ਕਿਤਾਬ ਮੰਨਦੀ ਹੈ ਕਿ ਤੁਹਾਡੇ ਏ.ਆਈ. ਮਾਡਲ ਕੋਲ ਕਿਸੇ ਵੀ ਬਿਲਟ-ਇਨ ਸਰਵਰ-ਸਾਈਡ ਟੂਲ ਤੱਕ ਪਹੁੰਚ ਨਹੀਂ ਹੈ। ਕੋਈ ਵੀ ਟੂਲ ਜੋ ਤੁਸੀਂ ਆਪਣੇ ਏ.ਆਈ. ਲਈ ਉਪਲਬਧ ਕਰਵਾਉਣਾ ਚਾਹੁੰਦੇ ਹੋ, ਉਸਨੂੰ ਹਰੇਕ ਏ.ਪੀ.ਆਈ. ਬੇਨਤੀ ਵਿੱਚ ਤੁਹਾਡੇ ਦੁਆਰਾ ਸਪਸ਼ਟ ਤੌਰ 'ਤੇ ਘੋਸ਼ਿਤ ਕੀਤਾ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ, ਜੇਕਰ ਅਤੇ ਜਦੋਂ ਤੁਹਾਡਾ ਏ.ਆਈ. ਤੁਹਾਨੂੰ ਦੱਸਦਾ ਹੈ ਕਿ ਉਹ ਆਪਣੇ ਜਵਾਬ ਵਿੱਚ ਉਸ ਟੂਲ ਦੀ ਵਰਤੋਂ ਕਰਨਾ ਚਾਹੇਗਾ।

ਟੂਲ ਵਰਤੋਂ ਦੀ ਸੰਭਾਵਨਾ

ਟੂਲ ਦੀ ਵਰਤੋਂ ਏ.ਆਈ.-ਸੰਚਾਲਿਤ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਵਿਆਪਕ ਸੰਭਾਵਨਾਵਾਂ ਖੋਲ੍ਹਦੀ ਹੈ। ਇੱਥੇ ਕੁਝ ਉਦਾਹਰਣਾਂ ਹਨ ਜੋ ਟੂਲ ਵਰਤੋਂ ਨਾਲ ਪ੍ਰਾਪਤ ਕੀਤੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ:

1. **ਚੈਟਬੋਟ ਅਤੇ ਵਰਚੁਅਲ ਸਹਾਇਕ**: ਐੱਲ.ਐੱਲ.ਐੱਮ. ਨੂੰ ਬਾਹਰੀ ਟੂਲਾਂ ਨਾਲ ਜੋੜ ਕੇ, ਚੈਟਬੋਟ ਅਤੇ ਵਰਚੁਅਲ ਸਹਾਇਕ ਵਧੇਰੇ ਗੁੰਝਲਦਾਰ ਕੰਮ ਕਰ ਸਕਦੇ ਹਨ, ਜਿਵੇਂ ਕਿ ਡਾਟਾਬੇਸਾਂ ਤੋਂ ਜਾਣਕਾਰੀ ਪ੍ਰਾਪਤ ਕਰਨਾ, ਏ.ਪੀ.ਆਈ. ਕਾਲਾਂ ਚਲਾਉਣਾ, ਜਾਂ ਹੋਰ ਸਿਸਟਮਾਂ ਨਾਲ ਅੰਤਰਕਰਿਆ ਕਰਨਾ। ਉਦਾਹਰਣ ਲਈ, ਇੱਕ ਚੈਟਬੋਟ ਉਪਭੋਗਤਾ ਦੀ ਬੇਨਤੀ ਦੇ ਆਧਾਰ 'ਤੇ ਸੀ.ਆਰ.ਐੱਮ. ਟੂਲ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਕਿਸੇ ਡੀਲ ਦੀ ਸਥਿਤੀ ਬਦਲ ਸਕਦਾ ਹੈ।
2. **ਡਾਟਾ ਵਿਸ਼ਲੇਸ਼ਣ ਅਤੇ ਅੰਤਰਦ੍ਰਿਸ਼ਟੀ**: ਐੱਲ.ਐੱਲ.ਐੱਮ. ਨੂੰ ਉੱਨਤ ਡਾਟਾ ਪ੍ਰੋਸੈਸਿੰਗ ਕਾਰਜ ਕਰਨ ਲਈ ਡਾਟਾ ਵਿਸ਼ਲੇਸ਼ਣ ਟੂਲਾਂ ਜਾਂ ਲਾਇਬ੍ਰੇਰੀਆਂ ਨਾਲ ਜੋੜਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਅੰਤਰਦ੍ਰਿਸ਼ਟੀ ਤਿਆਰ ਕਰਨ, ਤੁਲਨਾਤਮਕ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਨ, ਜਾਂ ਉਪਭੋਗਤਾ ਪੁੱਛਗਿੱਛਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਡਾਟਾ-ਆਧਾਰਿਤ ਸਿਫਾਰਸ਼ਾਂ ਪ੍ਰਦਾਨ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ।
3. **ਖੋਜ ਅਤੇ ਜਾਣਕਾਰੀ ਪੁਨਰ-ਪ੍ਰਾਪਤੀ**: ਟੂਲ ਦੀ ਵਰਤੋਂ ਐੱਲ.ਐੱਲ.ਐੱਮ. ਨੂੰ ਖੋਜ ਇੰਜਣਾਂ, ਵੈਕਟਰ ਡਾਟਾਬੇਸਾਂ, ਜਾਂ ਹੋਰ ਜਾਣਕਾਰੀ ਪੁਨਰ-ਪ੍ਰਾਪਤੀ ਸਿਸਟਮਾਂ ਨਾਲ ਅੰਤਰਕਰਿਆ ਕਰਨ ਦੀ ਆਗਿਆ ਦੇਂਦੀ ਹੈ। ਉਪਭੋਗਤਾ ਪੁੱਛਗਿੱਛਾਂ ਨੂੰ ਖੋਜ ਪੁੱਛਗਿੱਛਾਂ ਵਿੱਚ ਬਦਲ ਕੇ, ਐੱਲ.ਐੱਲ.ਐੱਮ. ਕਈ ਸਰੋਤਾਂ ਤੋਂ ਢੁਕਵੀਂ

ਜਾਣਕਾਰੀ ਪ੍ਰਾਪਤ ਕਰ ਸਕਦਾ ਹੈ ਅਤੇ ਉਪਭੋਗਤਾ ਦੇ ਸਵਾਲਾਂ ਦੇ ਵਆਿਪਕ ਜਵਾਬ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦਾ ਹੈ।

4. **ਬਾਹਰੀ ਸੇਵਾਵਾਂ ਨਾਲ ਏਕੀਕਰਨ**: ਟੂਲ ਦੀ ਵਰਤੋਂ ਏ.ਆਈ.-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਅਤੇ ਬਾਹਰੀ ਸੇਵਾਵਾਂ ਜਾਂ ਏ.ਪੀ.ਆਈ. ਵਚਿਕਾਰ ਨਰਿਵਘਿਨ ਏਕੀਕਰਨ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦੀ ਹੈ। ਉਦਾਹਰਣ ਲਈ, ਇੱਕ ਐੱਲ.ਐੱਲ.ਐੱਮ. ਰੀਅਲ-ਟਾਈਮ ਮੌਸਮ ਅੱਪਡੇਟ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਮੌਸਮ ਏ.ਪੀ.ਆਈ. ਨਾਲ ਜਾਂ ਬਹੁ-ਭਾਸ਼ਾਈ ਜਵਾਬ ਤਆਿਰ ਕਰਨ ਲਈ ਅਨੁਵਾਦ ਏ.ਪੀ.ਆਈ. ਨਾਲ ਅੰਤਰਕਰਿਆ ਕਰ ਸਕਦਾ ਹੈ।

ਟੂਲ ਵਰਤੋਂ ਵਰਕਫਲੋ

ਟੂਲ ਵਰਤੋਂ ਵਰਕਫਲੋ ਵੱਚਿ ਆਮ ਤੌਰ 'ਤੇ ਚਾਰ ਮੁੱਖ ਕਦਮ ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ:

1. ਆਪਣੇ ਬੇਨਤੀ ਸੰਦਰਭ ਵੱਚਿ ਫੰਕਸ਼ਨ ਪਰਭਿਾਸ਼ਾਵਾਂ ਸ਼ਾਮਲ ਕਰੋ
2. ਡਾਇਨਾਮਕਿ (ਜਾਂ ਸਪੱਸ਼ਟ) ਟੂਲ ਚੋਣ
3. ਫੰਕਸ਼ਨ(ਨਾਂ) ਦਾ ਕ੍ਰਆਿਨਵਯਨ
4. ਮੂਲ ਪ੍ਰੌਮਪਟ ਦੀ ਵਕਿਲਪਕਿ ਨਰਿੰਤਰਤਾ

ਆਓ ਇਨ੍ਹਾਂ ਕਦਮਾਂ ਦੀ ਵਸਿਥਾਰ ਨਾਲ ਸਮੀਖਆਿ ਕਰੀਏ।

ਆਪਣੇ ਬੇਨਤੀ ਸੰਦਰਭ ਵੱਚਿ ਫੰਕਸ਼ਨ ਪਰਭਿਾਸ਼ਾਵਾਂ ਸ਼ਾਮਲ ਕਰੋ

AI ਨੂੰ ਪਤਾ ਹੁੰਦਾ ਹੈ ਕਿ ਇਸ ਕੋਲ ਕਹਿੜੇ ਟੂਲ ਉਪਲਬਧ ਹਨ ਕਿਉਂਕਿ ਤੁਸੀਂ ਇਸਨੂੰ ਆਪਣੀ ਪੂਰਤੀ ਬੇਨਤੀ ਦੇ ਹੱਸਿੇ ਵਜੋਂ ਇੱਕ ਸੂਚੀ ਦੱਦਿੇ ਹੋ (ਆਮ ਤੌਰ 'ਤੇ JSON ਸਕੀਮਾ ਦੇ ਇੱਕ ਰੂਪ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਫੰਕਸ਼ਨਾਂ ਵਜੋਂ ਪਰਭਿਾਸ਼ਤਿ)।

ਟੂਲ ਪਰਭਿਾਸ਼ਾ ਦਾ ਸਹੀ ਸੰਟੈਕਸ ਮਾਡਲ-ਵਸ਼ੇਸ਼ ਹੁੰਦਾ ਹੈ।

Claude 3 ਵੱਚਿ `get_weather` ਫੰਕਸ਼ਨ ਨੂੰ ਇਸ ਤਰ੍ਹਾਂ ਪਰਭਿਾਸ਼ਤਿ ਕੀਤਾ ਜਾਂਦਾ ਹੈ:

```
{
    "name": "get_weather",
    "description": "Get the current weather in a given location",
    "input_schema": {
        "type": "object",
        "properties": {
            "location": {
                "type": "string",
                "description": "The city and state, e.g. San Francisco, CA"
            },
            "unit": {
                "type": "string",
                "enum": ["celsius", "fahrenheit"],
                "description": "The unit of temperature"
        }
    },
    "required": ["location"]
    }
}
```

ਅਤੇ ਇਸੇ ਤਰ੍ਹਾਂ ਤੁਸੀ GPT-4 ਲਈ ਉਹੀ ਫੰਕਸ਼ਨ ਪਰਭਾਸ਼ਿਤ ਕਰੋਗੇ, ਇਸਨੂੰ `tools` ਪੈਰਾਮੀਟਰ ਦੇ ਮੁੱਲ ਵਜੋਂ ਪਾਸ ਕਰਦੇ ਹੋਏ:

```
{
    "name": "get_current_weather",
    "description": "Get the current weather in a given location",
    "parameters": {
        "type": "object",
        "properties": {
            "location": {
                "type": "string",
                "description": "The city and state, e.g. San Francisco, CA",
            },
            "unit": {
                "type": "string",
                "enum": ["celsius", "fahrenheit"],
                "description": "The unit of temperature"
            },
        },
        "required": ["location"],
```

```
    },
}
```

> ਲਗਭਗ ਇੱਕੋ ਜਹਿਾ, ਪਰ ਬਨਿਾਂ ਕਸਿੇ ਸਪੱਸਟ਼ ਕਾਰਨ ਦੇ ਵੱਖਰਾ! ਕੰਨਿਾ ਪਰੇਸ਼ਾਨ ਕਰਨ ਵਾਲਾ ਹੈ।

ਫੰਕਸ਼ਨ ਪਰਭਾਸ਼ਾਵਾਂ ਨਾਮ, ਵੇਰਵਾ, ਅਤੇ ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰ ਨਰਿਧਾਰਤ ਕਰਦੀਆਂ ਹਨ। ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰਾਂ ਨੂੰ ਈਨਮਜ਼ ਵਰਗੀਆਂ ਵਸ਼ਿੇਸ਼ਤਾਵਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਹੋਰ ਪਰਭਾਸ਼ਤਿ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਸਵੀਕਾਰਯੋਗ ਮੁੱਲਾਂ ਨੂੰ ਸੀਮਤ ਕਰਦੀਆਂ ਹਨ, ਅਤੇ ਇਹ ਨਰਿਧਾਰਤ ਕਰਦੀਆਂ ਹਨ ਕ ਿਕੀ ਕੋਈ ਪੈਰਾਮੀਟਰ ਲੋੜੀਦਾ ਹੈ ਜਾਂ ਨਹੀਂ।

ਅਸਲ ਫੰਕਸ਼ਨ ਪਰਭਾਸ਼ਾਵਾਂ ਤੋਂ ਇਲਾਵਾ, ਤੁਸੀਂ ਸਸਿਟਮ ਨਰਿਦੇਸ਼ ਵਚਿ ਫੰਕਸ਼ਨ ਦੀ ਵਰਤੋਂ ਕਉਿ ਅਤੇ ਕਵਿੇਂ ਕਰਨੀ ਹੈ, ਬਾਰੇ ਹਦਾਇਤਾਂ ਜਾਂ ਸੰਦਰਭ ਵੀ ਸ਼ਾਮਲ ਕਰ ਸਕਦੇ ਹੋ।

ਉਦਾਹਰਨ ਲਈ, Olympia ਵਚਿ ਮੇਰਾ ਵੈੱਬ ਖੋਜ ਟੂਲ ਇਹ ਸਸਿਟਮ ਨਰਿਦੇਸ਼ ਸ਼ਾਮਲ ਕਰਦਾ ਹੈ, ਜੋ ਏਆਈ ਨੂੰ ਯਾਦ ਕਰਵਾਉਂਦਾ ਹੈ ਕ ਿਇਸ ਕੋਲ ਦੱਸੇ ਗਏ ਟੂਲ ਉਪਲਬਧ ਹਨ:

```
The `google_search` and `realtime_search` functions let you do research
on behalf of the user. In contrast to Google, realtime search is powered
by Perplexity and provides real-time information to curated current events
databases and news sources. Make sure to include URLs in your response so
user can do followup research.
```

ਵਸਿਤਾਰਪੂਰਵਕ ਵੇਰਵੇ ਪ੍ਰਦਾਨ ਕਰਨਾ ਟੂਲ ਪ੍ਰਦਰਸ਼ਨ ਵਚਿ ਸਭ ਤੋਂ ਮਹੱਤਵਪੂਰਨ ਕਾਰਕ ਮੰਨਆਿ ਜਾਂਦਾ ਹੈ। ਤੁਹਾਡੇ ਵੇਰਵਆਿਂ ਵਚਿ ਟੂਲ ਬਾਰੇ ਹਰ ਜਾਣਕਾਰੀ ਸ਼ਾਮਲ ਹੋਣੀ ਚਾਹੀਦੀ ਹੈ, ਜਸਿ ਵਚਿ ਸ਼ਾਮਲ ਹਨ:

- ਟੂਲ ਕੀ ਕਰਦਾ ਹੈ
- ਇਸ ਦੀ ਵਰਤੋਂ ਕਦੋਂ ਕੀਤੀ ਜਾਣੀ ਚਾਹੀਦੀ ਹੈ (ਅਤੇ ਕਦੋਂ ਨਹੀਂ)
- ਹਰ ਪੈਰਾਮੀਟਰ ਦਾ ਕੀ ਮਤਲਬ ਹੈ ਅਤੇ ਇਹ ਟੂਲ ਦੇ ਵਵਿਹਾਰ ਨੂੰ ਕਵਿੇਂ ਪ੍ਰਭਾਵਤਿ ਕਰਦਾ ਹੈ
- ਕੋਈ ਵੀ ਮਹੱਤਵਪੂਰਨ ਸਾਵਧਾਨੀਆਂ ਜਾਂ ਸੀਮਾਵਾਂ ਜੋ ਟੂਲ ਦੇ ਲਾਗੂਕਰਨ ’ਤੇ ਲਾਗੂ ਹੁੰਦੀਆਂ ਹਨ

ਜੰਨਿਾ ਵੱਧ ਸੰਦਰਭ ਤੁਸੀਂ AI ਨੂੰ ਆਪਣੇ ਟੂਲਾਂ ਬਾਰੇ ਦੇ ਸਕਦੇ ਹੋ, ਓਨਾ ਹੀ ਬਹਿਤਰ ਇਹ ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਕਦੋਂ ਅਤੇ ਕਵਿੇਂ ਕਰਨੀ ਹੈ, ਇਸ ਬਾਰੇ ਫੈਸਲਾ ਲੈਣ ਵਚਿ ਹੋਵੇਗਾ। ਉਦਾਹਰਣ ਵਜੋਂ, Anthropic ਆਪਣੀ Claude

3 ਸੀਰੀਜ਼ ਲਈ ਹਰ ਟੂਲ ਵੇਰਵੇ ਲਈ ਘੱਟੋ-ਘੱਟ 3-4 ਵਾਕਾਂ ਦੀ ਸਫਿਾਰਸ਼ ਕਰਦਾ ਹੈ, ਜੇ ਟੂਲ ਗੁੰਝਲਦਾਰ ਹੈ ਤਾਂ ਹੋਰ ਵੀ ਜ਼ਿਆਦਾ।

ਇਹ ਜ਼ਰੂਰੀ ਤੌਰ 'ਤੇ ਸਹਜਿ ਨਹੀਂ ਹੈ, ਪਰ ਵੇਰਵਆਿਂ ਨੂੰ ਉਦਾਹਰਣਾਂ ਨਾਲੋਂ ਵੀ ਵੱਧ ਮਹੱਤਵਪੂਰਨ ਮੰਨਆਿ ਜਾਂਦਾ ਹੈ। ਭਾਵੇਂ ਤੁਸੀ ਟੂਲ ਦੀ ਵਰਤੋਂ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਨੂੰ ਇਸਦੇ ਵੇਰਵੇ ਵੰਚਿ ਜਾਂ ਨਾਲ ਦੱਤਿੇ ਪ੍ਰੌਮਪਟ ਵੰਚਿ ਸ਼ਾਮਲ ਕਰ ਸਕਦੇ ਹੋ, ਇਹ ਟੂਲ ਦੇ ਉਦੇਸ਼ ਅਤੇ ਪੈਰਾਮੀਟਰਾਂ ਦੀ ਸਪਸ਼ਟ ਅਤੇ ਵਆਿਪਕ ਵਆਿਖਆਿ ਹੋਣ ਨਾਲੋਂ ਘੱਟ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਵੇਰਵੇ ਨੂੰ ਪੂਰੀ ਤਰ੍ਹਾਂ ਵਕਿਸਤਿ ਕਰਨ ਤੋਂ ਬਾਅਦ ਹੀ ਉਦਾਹਰਣਾਂ ਜੋੜੋ।

ਇੱਥੇ Stripe-ਵਰਗੇ API ਫੰਕਸ਼ਨ ਸਪੈਸੀਫਕਿਸ਼ਨ ਦੀ ਇੱਕ ਉਦਾਹਰਣ ਹੈ:

```
{
  "name": "createPayment",
  "description": "Create a new payment request",
  "parameters": {
    "type": "object",
    "properties": {
      "transaction_amount": {
        "type": "number",
        "description": "The amount to be paid"
      },
      "description": {
        "type": "string",
        "description": "A brief description of the payment"
      },
      "payment_method_id": {
        "type": "string",
        "description": "The payment method to be used"
      },
      "payer": {
        "type": "object",
        "description": "Information about the payer, including their name,
                        email, and identification number",
        "properties": {
          "name": {
            "type": "string",
            "description": "The payer's name"
        },
        "email": {
          "type": "string",
          "description": "The payer's email address"
```

```
        },
        "identification": {
          "type": "object",
          "description": "The payer's identification number",
          "properties": {
            "type": {
              "type": "string",
              "description": "Identification document (e.g. CPF, CNPJ)"
            },
            "number": {
              "type": "string",
              "description": "The identification number"
            }
          },
          "required": [ "type", "number" ]
        }
      },
      "required": [ "name", "email", "identification" ]
    }
  }
}
```

ਅਮਲੀ ਤੌਰ 'ਤੇ, ਕੁਝ ਮਾਡਲਾਂ ਨੂੰ ਨੈਸਟਡ ਫੰਕਸ਼ਨ ਸਪੈਸੀਫਿਕੇਸ਼ਨਾਂ ਅਤੇ ਐਰੇ, ਡਿਕਸ਼ਨਰੀਆਂ ਆਦਿ ਵਰਗੇ ਗੁੰਝਲਦਾਰ ਆਉਟਪੁੱਟ ਡਾਟਾ ਟਾਈਪਾਂ ਨਾਲ ਨਜਿੱਠਣ ਵਿੱਚ ਮੁਸ਼ਕਲ ਆਉਂਦੀ ਹੈ। ਪਰ ਸਧਾਂਤਕ ਤੌਰ 'ਤੇ, ਤੁਸੀ ਕਸਿ ਵੀ ਡੂੰਘਾਈ ਦੀਆਂ JSON ਸਕੀਮਾ ਸਪੈਸੀਫਿਕੇਸ਼ਨਾਂ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹੋ!

ਗਤੀਸ਼ੀਲ ਟੂਲ ਚੋਣ

ਜਦੋਂ ਤੁਸੀ ਟੂਲ ਪਰਿਭਾਸ਼ਾਵਾਂ ਸਮੇਤ ਇੱਕ ਚੈਟ ਪੂਰਨਤਾ ਨੂੰ ਲਾਗੂ ਕਰਦੇ ਹੋ, ਤਾਂ LLM ਗਤੀਸ਼ੀਲ ਤੌਰ 'ਤੇ ਵਰਤਣ ਲਈ ਸਭ ਤੋਂ ਢੁਕਵੇਂ ਟੂਲ(ਸ) ਦੀ ਚੋਣ ਕਰਦਾ ਹੈ ਅਤੇ ਹਰੇਕ ਟੂਲ ਲਈ ਲੋੜੀਂਦੇ ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰ ਤਿਆਰ ਕਰਦਾ ਹੈ।

ਅਮਲੀ ਤੌਰ 'ਤੇ, AI ਦੀ ਬਿਲਕੁਲ ਸਹੀ ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਕਰਨ ਦੀ ਸਮਰੱਥਾ, ਅਤੇ ਬਿਲਕੁਲ ਤੁਹਾਡੀ ਇਨਪੁੱਟਸ ਦੀ ਸਪੈਸੀਫਿਕੇਸ਼ਨ ਦੀ ਪਾਲਣਾ ਕਰਨਾ ਹੱਟਿ ਜਾਂ ਮਸਿ ਹੁੰਦਾ ਹੈ। ਟੈਂਪਰੇਚਰ ਹਾਈਪਰਪੈਰਾਮੀਟਰ ਨੂੰ 0.0 ਤੱਕ ਘਟਾਉਣ

ਨਾਲ ਬਹੁਤ ਮਦਦ ਮਲਿਦੀ ਹੈ, ਪਰ ਮੇਰੇ ਤਜਰਬੇ ਵੱਚਿ ਤੁਹਾਨੂੰ ਫਰਿ ਵੀ ਕਦੇ-ਕਦੇ ਗਲਤੀਆਂ ਮਲਿਣਗੀਆਂ। ਇਹਨਾਂ ਅਸਫਲਤਾਵਾਂ ਵੱਚਿ ਕਲਪਤਿ ਫੰਕਸ਼ਨ ਨਾਮ, ਗਲਤ ਨਾਮ ਵਾਲੇ ਜਾਂ ਬਸ ਗੁੰਮ ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰ ਸ਼ਾਮਲ ਹਨ। ਪੈਰਾਮੀਟਰ JSON ਦੇ ਰੂਪ ਵੱਚਿ ਪਾਸ ਕੀਤੇ ਜਾਂਦੇ ਹਨ, ਜਸਿਦਾ ਮਤਲਬ ਹੈ ਕਿ ਕਈ ਵਾਰ ਤੁਸੀਂ ਕੱਟੇ ਹੋਏ, ਗਲਤ ਕੋਟ ਕੀਤੇ, ਜਾਂ ਹੋਰ ਤਰੀਕਿਆਂ ਨਾਲ ਟੁੱਟੇ JSON ਕਾਰਨ ਗਲਤੀਆਂ ਦੇਖੋਗੇ।

ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲਾ ਡਾਟਾ ਪੈਟਰਨ ਸੰਟੈਕਸ ਗਲਤੀਆਂ ਕਾਰਨ ਟੁੱਟੇ ਫੰਕਸ਼ਨ ਕਾਲਾਂ ਨੂੰ ਆਪਣੇ ਆਪ ਠੀਕ ਕਰਨ ਵੱਚਿ ਮਦਦ ਕਰ ਸਕਦੇ ਹਨ।

ਜ਼ਬਰਦਸਤੀ (ਜਾਂ ਸਪੱਸ਼ਟ) ਟੂਲ ਚੋਣ

ਕੁਝ ਮਾਡਲ ਤੁਹਾਨੂੰ ਬੇਨਤੀ ਵੱਚਿ ਇੱਕ ਪੈਰਾਮੀਟਰ ਵਜੋਂ ਕਸਿੇ ਖਾਸ ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਕਰਨ ਲਈ ਮਜਬੂਰ ਕਰਨ ਦਾ ਵਕਿਲਪ ਦੰਦਿੇ ਹਨ। ਨਹੀਂ ਤਾਂ, ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਕਰਨਾ ਜਾਂ ਨਹੀਂ ਕਰਨਾ ਪੂਰੀ ਤਰ੍ਹਾਂ AI ਦੇ ਵਵਿੇਕ 'ਤੇ ਨਰਿਭਰ ਕਰਦਾ ਹੈ।

ਫੰਕਸ਼ਨ ਕਾਲ ਨੂੰ ਜ਼ਬਰਦਸਤੀ ਕਰਨ ਦੀ ਯੋਗਤਾ ਕੁਝ ਖਾਸ ਸਥਤਿੀਆਂ ਵੱਚਿ ਮਹੱਤਵਪੂਰਨ ਹੁੰਦੀ ਹੈ ਜੱਥਿੇ ਤੁਸੀਂ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਚਾਹੁੰਦੇ ਹੋ ਕਿ ਇੱਕ ਖਾਸ ਟੂਲ ਜਾਂ ਫੰਕਸ਼ਨ ਨੂੰ ਲਾਗੂ ਕੀਤਾ ਜਾਵੇ, AI ਦੀ ਗਤੀਸ਼ੀਲ ਚੋਣ ਪ੍ਰਕਰਿਆ ਦੀ ਪਰਵਾਹ ਕੀਤੇ ਬਨਿਾਂ। ਇਸ ਸਮਰੱਥਾ ਦੇ ਮਹੱਤਵਪੂਰਨ ਹੋਣ ਦੇ ਕਈ ਕਾਰਨ ਹਨ:

1. **ਸਪੱਸ਼ਟ ਨਯਿੰਤਰਣ**: ਤੁਸੀਂ AI ਨੂੰ ਇੱਕ ਵੱਖਰੇ ਕੰਪੋਨੈਂਟ ਵਜੋਂ ਜਾਂ ਇੱਕ ਪਹਲਿਾਂ ਤੋਂ ਨਰਿਧਾਰਤਿ ਵਰਕਫਲੋ ਵੱਚਿ ਵਰਤ ਰਹੇ ਹੋ ਸਕਦੇ ਹੋ ਜਸਿ ਵੱਚਿ ਕਸਿ ਖਾਸ ਸਮੇਂ 'ਤੇ ਇੱਕ ਖਾਸ ਫੰਕਸ਼ਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਕਾਲ ਨੂੰ ਜ਼ਬਰਦਸਤੀ ਕਰਕੇ, ਤੁਸੀਂ ਯਕੀਨੀ ਬਣਾ ਸਕਦੇ ਹੋ ਕਿ AI ਨੂੰ ਇਹ ਕਰਨ ਲਈ ਨਮਿਰਤਾ ਨਾਲ ਪੁੱਛਣ ਦੀ ਬਜਾਏ ਲੋੜੀਂਦਾ ਫੰਕਸ਼ਨ ਕਾਲ ਕੀਤਾ ਜਾਵੇ।
2. **ਡੀਬੱਗਿੰਗ ਅਤੇ ਟੈਸਟਿੰਗ**: AI-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਵਕਿਸਤਿ ਅਤੇ ਟੈਸਟ ਕਰਨ ਵੇਲੇ, ਫੰਕਸ਼ਨ ਕਾਲਾਂ ਨੂੰ ਜ਼ਬਰਦਸਤੀ ਕਰਨ ਦੀ ਯੋਗਤਾ ਡੀਬੱਗਿੰਗ ਦੇ ਉਦੇਸ਼ਾਂ ਲਈ ਬੇਮੁੱਲ ਹੈ। ਖਾਸ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਸਪੱਸ਼ਟ ਤੌਰ 'ਤੇ ਟਰਗਿਰ ਕਰਕੇ, ਤੁਸੀਂ ਆਪਣੀ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਵੱਖ-ਵੱਖ ਹੱਸਿਆਂ ਨੂੰ ਵੱਖ ਕਰ ਸਕਦੇ ਹੋ ਅਤੇ ਟੈਸਟ ਕਰ ਸਕਦੇ ਹੋ। ਇਹ ਤੁਹਾਨੂੰ ਫੰਕਸ਼ਨ ਇੰਪਲੀਮੈਂਟੇਸ਼ਨਾਂ ਦੀ ਸਹੀਤਾ ਦੀ ਜਾਂਚ ਕਰਨ, ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰਾਂ ਨੂੰ ਵੈਲੀਡੇਟ ਕਰਨ, ਅਤੇ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣ ਦੀ ਆਗਿਆ ਦੰਦਿਾ ਹੈ ਕਿ ਉਮੀਦ ਕੀਤੇ ਨਤੀਜੇ ਵਾਪਸ ਕੀਤੇ ਜਾ ਰਹੇ ਹਨ।

3. **ਸੀਮਾ ਕੇਸਾਂ ਨੂੰ ਸੰਭਾਲਣਾ**: ਅਜਹਿ ਸੀਮਾ ਕੇਸ ਜਾਂ ਅਸਧਾਰਨ ਸਥਤਿੀਆਂ ਹੋ ਸਕਦੀਆਂ ਹਨ ਜੰਥੇ AI ਦੀ ਗਤੀਸ਼ੀਲ ਚੋਣ ਪ੍ਰਕਰਿਆ ਕਸਿੇ ਫੰਕਸ਼ਨ ਨੂੰ ਚਲਾਉਣ ਦੀ ਚੋਣ ਨਹੀਂ ਕਰ ਸਕਦੀ ਜਸਿ ਨੂੰ ਇਹ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ, ਅਤੇ ਤੁਸੀਂ ਇਹ ਬਾਹਰੀ ਪ੍ਰਕਰਿਆਵਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਜਾਣਦੇ ਹੋ। ਅਜਹਿੀਆਂ ਸਥਤਿੀਆਂ ਵੰਚਿ, ਫੰਕਸ਼ਨ ਕਾਲ ਨੂੰ ਜ਼ਬਰਦਸਤੀ ਕਰਨ ਦੀ ਯੋਗਤਾ ਤੁਹਾਨੂੰ ਇਨ੍ਹਾਂ ਸਥਤਿੀਆਂ ਨੂੰ ਸਪਸ਼ਟ ਤੌਰ 'ਤੇ ਸੰਭਾਲਣ ਦੀ ਆਗਆਿ ਦੰਦੀ ਹੈ। ਆਪਣੀ ਐਪਲੀਕੇਸ਼ਨ ਲੌਜਕਿ ਵੰਚਿ ਨਯਿਮ ਜਾਂ ਸ਼ਰਤਾਂ ਨਰਿਧਾਰਤ ਕਰੋ ਜੋ ਇਹ ਨਰਿਧਾਰਤ ਕਰਨ ਲਈ ਕਿ AI ਦੇ ਵਵੇਕ ਨੂੰ ਕਦੋਂ ਓਵਰਰਾਈਡ ਕਰਨਾ ਹੈ।
4. **ਇਕਸਾਰਤਾ ਅਤੇ ਦੁਹਰਾਉਣਯੋਗਤਾ**: ਜੇਕਰ ਤੁਹਾਡੇ ਕੋਲ ਫੰਕਸ਼ਨਾਂ ਦਾ ਇੱਕ ਵਸ਼ਿਸ਼ ਕ੍ਰਮ ਹੈ ਜਸਿਨੂੰ ਇੱਕ ਖਾਸ ਕ੍ਰਮ ਵੰਚਿ ਚਲਾਉਣ ਦੀ ਲੋੜ ਹੈ, ਤਾਂ ਕਾਲਾਂ ਨੂੰ ਜ਼ਬਰਦਸਤੀ ਕਰਨ ਨਾਲ ਇਹ ਗਾਰੰਟੀ ਹੁੰਦੀ ਹੈ ਕਿ ਹਰ ਵਾਰ ਉਹੀ ਕ੍ਰਮ ਦੀ ਪਾਲਣਾ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਖਾਸ ਤੌਰ 'ਤੇ ਉਨ੍ਹਾਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵੰਚਿ ਮਹੱਤਵਪੂਰਨ ਹੈ ਜੰਥੇ ਇਕਸਾਰਤਾ ਅਤੇ ਭਵੱਖਿਬਾਣੀਯੋਗ ਵਵਿਹਾਰ ਮਹੱਤਵਪੂਰਨ ਹਨ, ਜਵਿੇਂ ਕਿ ਵੰਤੀ ਪ੍ਰਣਾਲੀਆਂ ਜਾਂ ਵਗਿਆਨਕ ਸਮੂਲੇਸ਼ਨਾਂ ਵੰਚਿ।
5. **ਕਾਰਗੁਜ਼ਾਰੀ ਅਨੁਕੂਲਤਾ**: ਕੁਝ ਮਾਮਲਆਿਂ ਵੰਚਿ, ਫੰਕਸ਼ਨ ਕਾਲ ਨੂੰ ਜ਼ਬਰਦਸਤੀ ਕਰਨ ਨਾਲ ਕਾਰਗੁਜ਼ਾਰੀ ਅਨੁਕੂਲਤਾ ਹੋ ਸਕਦੀ ਹੈ। ਜੇਕਰ ਤੁਸੀਂ ਜਾਣਦੇ ਹੋ ਕਿ ਇੱਕ ਵਸ਼ਿਸ਼ ਕੰਮ ਲਈ ਇੱਕ ਖਾਸ ਫੰਕਸ਼ਨ ਦੀ ਲੋੜ ਹੈ ਅਤੇ AI ਦੀ ਗਤੀਸ਼ੀਲ ਚੋਣ ਪ੍ਰਕਰਿਆ ਗੈਰ-ਜ਼ਰੂਰੀ ਓਵਰਹੈੱਡ ਪੇਸ਼ ਕਰ ਸਕਦੀ ਹੈ, ਤਾਂ ਤੁਸੀਂ ਚੋਣ ਪ੍ਰਕਰਿਆ ਨੂੰ ਬਾਈਪਾਸ ਕਰ ਸਕਦੇ ਹੋ ਅਤੇ ਲੋੜੀਂਦੇ ਫੰਕਸ਼ਨ ਨੂੰ ਸੰਧਾ ਬੁਲਾ ਸਕਦੇ ਹੋ। ਇਹ ਲੇਟੈਂਸੀ ਨੂੰ ਘਟਾਉਣ ਅਤੇ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਸਮੁੱਚੀ ਕੁਸ਼ਲਤਾ ਨੂੰ ਸੁਧਾਰਨ ਵੰਚਿ ਮਦਦ ਕਰ ਸਕਦਾ ਹੈ।

ਸੰਖੇਪ ਵੰਚਿ, AI-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵੰਚਿ ਫੰਕਸ਼ਨ ਕਾਲਾਂ ਨੂੰ ਜ਼ਬਰਦਸਤੀ ਕਰਨ ਦੀ ਯੋਗਤਾ ਸਪਸ਼ਟ ਨਯਿੰਤਰਣ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ, ਡੀਬੱਗੰਗ ਅਤੇ ਟੈਸਟੰਗ ਵੰਚਿ ਸਹਾਇਤਾ ਕਰਦੀ ਹੈ, ਸੀਮਾ ਕੇਸਾਂ ਨੂੰ ਸੰਭਾਲਦੀ ਹੈ, ਇਕਸਾਰਤਾ ਅਤੇ ਦੁਹਰਾਉਣਯੋਗਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਂਦੀ ਹੈ। ਇਹ ਤੁਹਾਡੇ ਸ਼ਸਤਰਾਗਾਰ ਵੰਚਿ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਟੂਲ ਹੈ, ਪਰ ਸਾਨੂੰ ਇਸ ਮਹੱਤਵਪੂਰਨ ਵਸ਼ਿਸ਼ਤਾ ਦੇ ਇੱਕ ਹੋਰ ਪਹਲਿੂ ਬਾਰੇ ਵਚਿਾਰ ਕਰਨ ਦੀ ਲੋੜ ਹੈ।

ਬਹੁਤ ਸਾਰੇ ਫੈਸਲਾ-ਲੈਣ ਦੇ ਮਾਮਲਆਿਂ ਵੰਚਿ, ਅਸੀਂ ਹਮੇਸ਼ਾ ਚਾਹੁੰਦੇ ਹਾਂ ਕਿ ਮਾਡਲ ਇੱਕ ਫੰਕਸ਼ਨ ਕਾਲ ਕਰੇ ਅਤੇ ਸ਼ਾਇਦ ਕਦੇ ਵੀ ਨਹੀਂ ਚਾਹੁੰਦੇ ਕਿ ਮਾਡਲ ਸਰਫ਼ ਆਪਣੇ ਅੰਦਰੂਨੀ ਗਆਿਨ ਨਾਲ ਜਵਾਬ ਦੇਵੇ। ਉਦਾਹਰਣ ਲਈ, ਜੇਕਰ ਤੁਸੀਂ ਵੱਖ-ਵੱਖ ਕੰਮਾਂ ਵੰਚਿ ਮਾਹਰਿ ਕਈ ਮਾਡਲਾਂ ਵਚਿਕਾਰ ਰੂਟੰਗ ਕਰ ਰਹੇ ਹੋ (ਬਹੁਭਾਸ਼ੀ ਇਨਪੁਟ, ਗਣਤਿ, ਆਦਦਿ), ਤਾਂ ਤੁਸੀਂ ਬੇਨਤੀਆਂ ਨੂੰ ਸਹਾਇਕ ਮਾਡਲਾਂ ਵੰਚੋਂ ਇੱਕ ਨੂੰ ਸੌਂਪਣ ਲਈ ਫੰਕਸ਼ਨ-ਕਾਲੰਗ ਮਾਡਲ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹੋ ਅਤੇ ਕਦੇ ਵੀ ਸੁਤੰਤਰ ਤੌਰ 'ਤੇ ਜਵਾਬ ਨਹੀਂ ਦੇ ਸਕਦੇ।

ਟੂਲ ਚੋਣ ਪੈਰਾਮੀਟਰ

GPT-4 ਅਤੇ ਹੋਰ ਭਾਸ਼ਾ ਮਾਡਲ ਜੋ ਫੰਕਸ਼ਨ ਕਾਲਿੰਗ ਦਾ ਸਮਰਥਨ ਕਰਦੇ ਹਨ, ਤੁਹਾਨੂੰ ਪੂਰਤੀ ਦੇ ਹਿੱਸੇ ਵਜੋਂ ਟੂਲ ਦੀ ਵਰਤੋਂ ਦੀ ਲੋੜ ਨੂੰ ਨਿਯੰਤਰਿਤ ਕਰਨ ਲਈ `tool_choice` ਪੈਰਾਮੀਟਰ ਦਿੰਦੇ ਹਨ। ਇਸ ਪੈਰਾਮੀਟਰ ਦੇ ਤਿੰਨ ਸੰਭਵ ਮੁੱਲ ਹਨ:

- `auto` AI ਨੂੰ ਕਿਸੇ ਟੂਲ ਦੀ ਵਰਤੋਂ ਕਰਨ ਜਾਂ ਸਿੱਧਾ ਜਵਾਬ ਦੇਣ ਦਾ ਪੂਰਾ ਵਿਵੇਕ ਦਿੰਦਾ ਹੈ
- `required` AI ਨੂੰ ਦੱਸਦਾ ਹੈ ਕਿ ਇਸਨੂੰ ਜਵਾਬ ਦੇਣ ਦੀ ਬਜਾਏ ਇੱਕ ਟੂਲ ਨੂੰ ਲਾਜ਼ਮੀ ਤੌਰ 'ਤੇ ਕਾਲ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ, ਪਰ ਟੂਲ ਦੀ ਚੋਣ AI 'ਤੇ ਛੱਡ ਦਿੰਦਾ ਹੈ।
- ਤੀਜਾ ਵਿਕਲਪ ਉਸ `name_of_function` ਦਾ ਪੈਰਾਮੀਟਰ ਸੈੱਟ ਕਰਨਾ ਹੈ ਜਿਸਨੂੰ ਤੁਸੀਂ ਜ਼ਬਰਦਸਤੀ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹੋ। ਅਗਲੇ ਸੈਕਸ਼ਨ ਵਿੱਚ ਇਸ ਬਾਰੇ ਹੋਰ ਜਾਣਕਾਰੀ ਦਿੱਤੀ ਗਈ ਹੈ।

ਧਿਆਨ ਦਿਓ ਕਿ ਜੇਕਰ ਤੁਸੀਂ tool choice ਨੂੰ `required` ਤੇ ਸੈੱਟ ਕਰਦੇ ਹੋ, ਤਾਂ ਮਾਡਲ ਨੂੰ ਦਿੱਤੇ ਗਏ ਫੰਕਸ਼ਨਾਂ ਵਿੱਚੋਂ ਸਭ ਤੋਂ ਢੁਕਵਾਂ ਫੰਕਸ਼ਨ ਚੁਣਨਾ ਪਵੇਗਾ, ਭਾਵੇਂ ਕੋਈ ਵੀ ਪ੍ਰੌਮਪਟ ਲਈ ਸਹੀ ਨਾ ਹੋਵੇ। ਪ੍ਰਕਾਸ਼ਨ ਦੇ ਸਮੇਂ, ਮੈਨੂੰ ਕਿਸੇ ਅਜਿਹੇ ਮਾਡਲ ਬਾਰੇ ਨਹੀਂ ਪਤਾ ਜੋ ਖਾਲੀ `tool_calls` ਜਵਾਬ ਦੇਵੇਗਾ, ਜਾਂ ਤੁਹਾਨੂੰ ਕਿਸੇ ਹੋਰ ਤਰੀਕੇ ਨਾਲ ਦੱਸੇਗਾ ਕਿ ਇਸਨੂੰ ਕੋਈ ਢੁਕਵਾਂ ਫੰਕਸ਼ਨ ਕਾਲ ਕਰਨ ਲਈ ਨਹੀਂ ਮਿਲਿਆ।

ਢਾਂਚਾਗਤ ਆਉਟਪੁੱਟ ਲਈ ਫੰਕਸ਼ਨ ਨੂੰ ਮਜਬੂਰ ਕਰਨਾ

ਫੰਕਸ਼ਨ ਕਾਲ ਨੂੰ ਮਜਬੂਰ ਕਰਨ ਦੀ ਯੋਗਤਾ ਤੁਹਾਨੂੰ ਚੈਟ ਕੰਪਲੀਸ਼ਨ ਤੋਂ ਢਾਂਚਾਗਤ ਡਾਟਾ ਪ੍ਰਾਪਤ ਕਰਨ ਦਾ ਤਰੀਕਾ ਦਿੰਦੀ ਹੈ, ਬਜਾਏ ਇਸ ਦੇ ਕਿ ਤੁਹਾਨੂੰ ਇਸਨੂੰ ਪਲੇਨਟੈਕਸਟ ਜਵਾਬ ਵਿੱਚੋਂ ਖੁਦ ਕੱਢਣਾ ਪਵੇ।

> ਢਾਂਚਾਗਤ ਆਉਟਪੁੱਟ ਲਈ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਮਜਬੂਰ ਕਰਨਾ ਇੰਨਾ ਮਹੱਤਵਪੂਰਨ ਕਿਉਂ ਹੈ? ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਕਿਉਂਕਿ LLM ਦੇ ਆਉਟਪੁੱਟ ਤੋਂ ਢਾਂਚਾਗਤ ਡਾਟਾ ਕੱਢਣਾ ਬਹੁਤ ਮੁਸ਼ਕਲ ਹੈ। ਤੁਸੀਂ XML ਵਿੱਚ ਡਾਟਾ ਮੰਗ ਕੇ ਆਪਣੀ ਜ਼ਿੰਦਗੀ ਥੋੜ੍ਹੀ ਆਸਾਨ ਬਣਾ ਸਕਦੇ ਹੋ, ਪਰ ਫਿਰ ਤੁਹਾਨੂੰ XML ਨੂੰ ਪਾਰਸ ਕਰਨਾ ਪੈਂਦਾ ਹੈ। ਅਤੇ

> ਜਦੋਂ ਉਹ XML ਗਾਇਬ ਹੋਵੇ ਕਿਉਂਕਿ ਤੁਹਾਡੀ AI ਨੇ ਜਵਾਬ ਦੱਤਿਆ: "ਮੈਂ ਮਾਫ਼ੀ ਚਾਹੁੰਦਾ/ਦੀ ਹਾਂ, ਪਰ ਮੈਂ ਤੁਹਾਡੀ ਬੇਨਤੀ ਕੀਤੀ ਡਾਟਾ ਤਿਆਰ ਨਹੀਂ ਕਰ ਸਕਦਾ/ਦੀ ਕਿਉਂਕਿ ਬਲਾ, ਬਲਾ, ਬਲਾ..." ਤਾਂ ਤੁਸੀਂ ਕੀ ਕਰਦੇ ਹੋ?

ਜਦੋਂ ਇਸ ਤਰ੍ਹਾਂ ਟੂਲਸ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋ:

- ਤੁਹਾਨੂੰ ਸ਼ਾਇਦ ਆਪਣੀ ਬੇਨਤੀ ਵਿੱਚ ਇੱਕ ਟੂਲ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ
- ਯਾਦ ਰੱਖੋ ਕਿ `tool_choice` ਪੈਰਾਮੀਟਰ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇਸਦੇ ਫੰਕਸ਼ਨ ਦੀ ਵਰਤੋਂ ਨੂੰ ਮਜਬੂਰ ਕਰੋ।
- ਯਾਦ ਰੱਖੋ ਕਿ ਮਾਡਲ ਟੂਲ ਨੂੰ ਇਨਪੁੱਟ ਪਾਸ ਕਰੇਗਾ, ਇਸ ਲਈ ਟੂਲ ਦਾ ਨਾਮ ਅਤੇ ਵੇਰਵਾ ਤੁਹਾਡੇ ਨਜ਼ਰੀਏ ਤੋਂ ਨਹੀਂ, ਬਲਕਿ ਮਾਡਲ ਦੇ ਨਜ਼ਰੀਏ ਤੋਂ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ।

ਇਸ ਆਖਰੀ ਨੁਕਤੇ ਨੂੰ ਸਪੱਸ਼ਟਤਾ ਲਈ ਇੱਕ ਉਦਾਹਰਨ ਦੀ ਲੋੜ ਹੈ। ਮੰਨ ਲਓ ਕਿ ਤੁਸੀਂ AI ਨੂੰ ਯੂਜ਼ਰ ਟੈਕਸਟ ਦਾ ਭਾਵਨਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਨ ਲਈ ਕਹਿ ਰਹੇ ਹੋ। ਫੰਕਸ਼ਨ ਦਾ ਨਾਮ `analyze_sentiment` ਨਹੀਂ ਹੋਵੇਗਾ, ਬਲਕਿ ਇਹ `save_sentiment_analysis` ਵਰਗਾ ਕੁਝ ਹੋਵੇਗਾ। ਭਾਵਨਾ ਵਿਸ਼ਲੇਸ਼ਣ AI ਕਰ ਰਹੀ ਹੈ, ਟੂਲ ਨਹੀਂ। ਟੂਲ ਸਿਰਫ਼ (AI ਦੇ ਨਜ਼ਰੀਏ ਤੋਂ) ਵਿਸ਼ਲੇਸ਼ਣ ਦੇ ਨਤੀਜਿਆਂ ਨੂੰ ਸੇਵ ਕਰ ਰਹਿਆ ਹੈ।

ਇੱਥੇ Claude 3 ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇੱਕ ਚੱਤਿਰ ਦੇ ਸਾਰ ਨੂੰ ਚੰਗੀ ਤਰ੍ਹਾਂ ਢਾਂਚਾਗਤ JSON ਵਿੱਚ ਰਿਕਾਰਡ ਕਰਨ ਦੀ ਉਦਾਹਰਨ ਹੈ, ਇਸ ਵਾਰ ਕਮਾਂਡ ਲਾਈਨ ਤੇ `curl` ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ:

```
curl https://api.anthropic.com/v1/messages \
     --header "content-type: application/json" \
     --header "x-api-key: $ANTHROPIC_API_KEY" \
     --header "anthropic-version: 2023-06-01" \
     --header "anthropic-beta: tools-2024-04-04" \
     --data \
'{
    "model": "claude-3-sonnet-20240229",
    "max_tokens": 1024,
    "tools": [{
        "name": "record_summary",
        "description": "Record summary of image into well-structured JSON.",
        "input_schema": {
            "type": "object",
            "properties": {
                "key_colors": {
```

```
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "r": {
                                "type": "number",
                                "description": "red value [0.0, 1.0]"
                            },
                            "g": {
                                "type": "number",
                                "description": "green value [0.0, 1.0]"
                            },
                            "b": {
                                "type": "number",
                                "description": "blue value [0.0, 1.0]"
                            },
                            "name": {
                                "type": "string",
                                "description": "Human-readable color name
                                                in snake_case, e.g.
                                                \"olive_green\"or
                                                \"turquoise\""
                            }
                        },
                        "required": [ "r", "g", "b", "name" ]
                    },
                    "description": "Key colors in the image. Four or less."
                },
                "description": {
                    "type": "string",
                    "description": "Image description. 1-2 sentences max."
                },
                "estimated_year": {
                    "type": "integer",
                    "description": "Estimated year that the image was taken,
                                    if is it a photo. Only set this if the
                                    image appears to be non-fictional.
                                    Rough estimates are okay!"
                }
            },
            "required": [ "key_colors", "description" ]
        }
```

```
    }],
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "type": "image",
                    "source": {
                        "type": "base64",
                        "media_type": "'$IMAGE_MEDIA_TYPE'",
                        "data": "'$IMAGE_BASE64'"
                    }
                },
                {
                    "type": "text",
                    "text": "Use `record_summary` to describe this image."
                }
            ]
        }
    ]
}'
```

ਦੱਤੀ ਗਈ ਉਦਾਹਰਣ ਵੱਚਿ, ਅਸੀ Anthropic ਦੇ Claude 3 ਮਾਡਲ ਦੀ ਵਰਤੋਂ ਇੱਕ ਚੱਤਿਰ ਦਾ ਢਾਂਚਾਗਤ JSON ਸਾਰ ਤਆਿਰ ਕਰਨ ਲਈ ਕਰ ਰਹੇ ਹਾਂ। ਇਹ ਇਸ ਤਰ੍ਹਾਂ ਕੰਮ ਕਰਦਾ ਹੈ:

1. ਅਸੀ ਬੇਨਤੀ ਪੇਲੋਡ ਦੇ `tools` ਐਰੇ ਵੱਚਿ `record_summary` ਨਾਂ ਦਾ ਇੱਕ ਟੂਲ ਪਰਭਾਸ਼ਤਿ ਕਰਦੇ ਹਾਂ। ਇਹ ਟੂਲ ਚੱਤਿਰ ਦੇ ਸਾਰ ਨੂੰ ਚੰਗੀ ਤਰ੍ਹਾਂ ਢਾਂਚਾਗਤ JSON ਵੱਚਿ ਰਕਿਾਰਡ ਕਰਨ ਲਈ ਜ਼ੰਮਿੇਵਾਰ ਹੈ।
2. `record_summary` ਟੂਲ ਵੱਚਿ ਇੱਕ `input_schema` ਹੈ ਜੋ JSON ਆਉਟਪੁੱਟ ਦੀ ਉਮੀਦਤਿ ਬਣਤਰ ਨੂੰ ਦਰਸਾਉਦਾ ਹੈ। ਇਹ ਤੰਨਿ ਗੁਣਾਂ ਨੂੰ ਪਰਭਾਸ਼ਤਿ ਕਰਦਾ ਹੈ:

    - `key_colors`: ਚੱਤਿਰ ਵੱਚਿ ਮੁੱਖ ਰੰਗਾਂ ਨੂੰ ਦਰਸਾਉਣ ਵਾਲੀਆਂ ਵਸਤੂਆਂ ਦੀ ਐਰੇ। ਹਰ ਰੰਗ ਦੀ ਵਸਤੂ ਵੱਚਿ ਲਾਲ, ਹਰਾ, ਅਤੇ ਨੀਲਾ ਮੁੱਲ (0.0 ਤੋਂ 1.0 ਤੱਕ) ਅਤੇ snake_case ਫਾਰਮੈਟ ਵੱਚਿ ਮਨੁੱਖੀ-ਪੜ੍ਹਨਯੋਗ ਰੰਗ ਦਾ ਨਾਮ ਹੁੰਦਾ ਹੈ।
    - `description`: ਚੱਤਿਰ ਦੇ ਸੰਖੇਪ ਵੇਰਵੇ ਲਈ ਇੱਕ ਸਟ੍ਰਿੰਗ ਗੁਣ, ਜੋ 1-2 ਵਾਕਾਂ ਤੱਕ ਸੀਮਤਿ ਹੈ।

    - `estimated_year`: ਇੱਕ ਵਿਕਲਪਿਕ ਪੂਰਨ ਅੰਕ ਗੁਣ, ਜੇਕਰ ਇਹ ਗੈਰ-ਕਾਲਪਨਿਕ ਫੋਟੋ ਜਾਪਦੀ ਹੈ ਤਾਂ ਇਸ ਦੇ ਖਿੱਚੇ ਜਾਣ ਦਾ ਅੰਦਾਜ਼ਨ ਸਾਲ।

3. `messages` ਐਰੇ ਵਿੱਚ, ਅਸੀਂ ਚਿੱਤਰ ਡੇਟਾ ਨੂੰ base64-ਇਨਕੋਡਡ ਸਟ੍ਰਿੰਗ ਦੇ ਰੂਪ ਵਿੱਚ ਮੀਡੀਆ ਕਿਸਮ ਦੇ ਨਾਲ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਾਂ। ਇਹ ਮਾਡਲ ਨੂੰ ਇਨਪੁੱਟ ਦੇ ਹਿੱਸੇ ਵਜੋਂ ਚਿੱਤਰ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ।
4. ਅਸੀਂ Claude ਨੂੰ ਚਿੱਤਰ ਦਾ ਵਰਣਨ ਕਰਨ ਲਈ `record_summary` ਟੂਲ ਦੀ ਵਰਤੋਂ ਕਰਨ ਲਈ ਵੀ ਪ੍ਰੇਰਿਤ ਕਰਦੇ ਹਾਂ।
5. ਜਦੋਂ ਬੇਨਤੀ Claude 3 ਮਾਡਲ ਨੂੰ ਭੇਜੀ ਜਾਂਦੀ ਹੈ, ਇਹ ਚਿੱਤਰ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਦਾ ਹੈ ਅਤੇ ਦੱਸੇ ਗਏ `input_schema` ਦੇ ਆਧਾਰ 'ਤੇ ਇੱਕ JSON ਸਾਰ ਤਿਆਰ ਕਰਦਾ ਹੈ। ਮਾਡਲ ਮੁੱਖ ਰੰਗਾਂ ਨੂੰ ਕੱਢਦਾ ਹੈ, ਸੰਖੇਪ ਵੇਰਵਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ, ਅਤੇ ਚਿੱਤਰ ਦੇ ਖਿੱਚੇ ਜਾਣ ਦੇ ਸਾਲ ਦਾ ਅੰਦਾਜ਼ਾ ਲਗਾਉਂਦਾ ਹੈ (ਜੇ ਲਾਗੂ ਹੋਵੇ)।
6. ਤਿਆਰ ਕੀਤਾ JSON ਸਾਰ `record_summary` ਟੂਲ ਨੂੰ ਪੈਰਾਮੀਟਰਾਂ ਵਜੋਂ ਪਾਸ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਜੋ ਚਿੱਤਰ ਦੀਆਂ ਮੁੱਖ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦੀ ਇੱਕ ਢਾਂਚਾਗਤ ਪ੍ਰਤੀਨਧਿਤਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।

ਚੰਗੀ ਤਰ੍ਹਾਂ ਪਰਿਭਾਸ਼ਿਤ `input_schema` ਨਾਲ `record_summary` ਟੂਲ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਅਸੀਂ ਸਧਾਰਨ ਟੈਕਸਟ ਕੱਢਣ 'ਤੇ ਨਿਰਭਰ ਕੀਤੇ ਬਿਨਾਂ ਇੱਕ ਚਿੱਤਰ ਦਾ ਢਾਂਚਾਗਤ JSON ਸਾਰ ਪ੍ਰਾਪਤ ਕਰ ਸਕਦੇ ਹਾਂ। ਇਹ ਪਹੁੰਚ ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦੀ ਹੈ ਕਿ ਆਉਟਪੁੱਟ ਇੱਕ ਸਥਿਰ ਫਾਰਮੈਟ ਦੀ ਪਾਲਣਾ ਕਰਦਾ ਹੈ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਡਾਊਨਸਟ੍ਰੀਮ ਭਾਗਾਂ ਦੁਆਰਾ ਆਸਾਨੀ ਨਾਲ ਪਾਰਸ ਅਤੇ ਪ੍ਰੋਸੈਸ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।

AI-ਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਟੂਲ ਦੀ ਵਰਤੋਂ ਵਿੱਚ ਫੰਕਸ਼ਨ ਕਾਲ ਨੂੰ ਮਜਬੂਰ ਕਰਨ ਅਤੇ ਉਮੀਦਤਿ ਆਉਟਪੁੱਟ ਢਾਂਚੇ ਨੂੰ ਨਿਰਧਾਰਤਿ ਕਰਨ ਦੀ ਯੋਗਤਾ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਵਿਸ਼ੇਸ਼ਤਾ ਹੈ। ਇਹ ਡਿਵੈਲਪਰਾਂ ਨੂੰ ਤਿਆਰ ਕੀਤੇ ਆਉਟਪੁੱਟ 'ਤੇ ਵਧੇਰੇ ਨਿਯੰਤਰਣ ਰੱਖਣ ਅਤੇ ਉਨ੍ਹਾਂ ਦੇ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਵਰਕਫਲੋ ਵਿੱਚ AI-ਤਿਆਰ ਕੀਤੇ ਡੇਟਾ ਦੇ ਏਕੀਕਰਣ ਨੂੰ ਸਰਲ ਬਣਾਉਣ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ।

ਫੰਕਸ਼ਨ(ਨਾਂ) ਦੀ ਕਾਰਜਸ਼ੀਲਤਾ

ਤੁਸੀਂ ਫੰਕਸ਼ਨ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤੇ ਹਨ, ਅਤੇ ਆਪਣੇ AI ਨੂੰ ਪ੍ਰੇਰਿਤ ਕੀਤਾ ਹੈ, ਜਿਸਨੇ ਫੈਸਲਾ ਕੀਤਾ ਕਿ ਇਸਨੂੰ ਤੁਹਾਡੇ ਫੰਕਸ਼ਨਾਂ ਵਿੱਚੋਂ ਇੱਕ ਨੂੰ ਕਾਲ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ। ਹੁਣ ਤੁਹਾਡੇ ਐਪਲੀਕੇਸ਼ਨ ਕੋਡ ਜਾਂ ਲਾਇਬ੍ਰੇਰੀ ਦਾ ਸਮਾਂ ਹੈ, ਜੇਕਰ ਤੁਸੀਂ `raix-rails` ਵਰਗੀ ਰੂਬੀ ਜੈਮ ਦੀ ਵਰਤੋਂ ਕਰ ਰਹੇ ਹੋ, ਫੰਕਸ਼ਨ ਕਾਲ ਅਤੇ ਇਸਦੇ ਪੈਰਾਮੀਟਰਾਂ ਨੂੰ ਤੁਹਾਡੇ ਐਪਲੀਕੇਸ਼ਨ ਕੋਡ ਵਿੱਚ ਸੰਬੰਧਿਤ ਕਾਰਜਾਂਵਿਧੀ ਵਿੱਚ ਭੇਜਣ ਲਈ।

ਤੁਹਾਡਾ ਐਪਲੀਕੇਸ਼ਨ ਕੋਡ ਫੈਸਲਾ ਕਰਦਾ ਹੈ ਕਿ ਫੰਕਸ਼ਨ ਦੇ ਨਤੀਜਿਆਂ ਨਾਲ ਕੀ ਕਰਨਾ ਹੈ। ਸ਼ਾਇਦ ਕੀ ਕਰਨਾ ਹੈ ਇਹ ਲੈਬਡਾ ਵੱਚਿ ਇੱਕ ਲਾਈਨ ਕੋਡ ਨੂੰ ਸ਼ਾਮਲ ਕਰਦਾ ਹੈ, ਜਾਂ ਸ਼ਾਇਦ ਇਹ ਬਾਹਰੀ ਏ.ਪੀ.ਆਈ. ਨੂੰ ਕਾਲ ਕਰਨ ਨੂੰ ਸ਼ਾਮਲ ਕਰਦਾ ਹੈ। ਸ਼ਾਇਦ ਇਹ ਕਸਿ ਹੋਰ AI ਕੰਪੋਨੈਂਟ ਨੂੰ ਕਾਲ ਕਰਨ ਨੂੰ ਸ਼ਾਮਲ ਕਰਦਾ ਹੈ, ਜਾਂ ਸ਼ਾਇਦ ਇਹ ਤੁਹਾਡੇ ਸਸਿਟਮ ਦੇ ਬਾਕੀ ਹੱਸਿ ਵੱਚਿ ਸੈਂਕੜੇ ਜਾਂ ਹਜ਼ਾਰਾਂ ਲਾਈਨਾਂ ਕੋਡ ਨੂੰ ਸ਼ਾਮਲ ਕਰਦਾ ਹੈ। ਇਹ ਪੂਰੀ ਤਰ੍ਹਾਂ ਤੁਹਾਡੇ 'ਤੇ ਨਰਿਭਰ ਕਰਦਾ ਹੈ।

ਕਈ ਵਾਰ ਫੰਕਸ਼ਨ ਕਾਲ ਕਾਰਜ ਦਾ ਅੰਤ ਹੁੰਦਾ ਹੈ, ਪਰ ਜੇਕਰ ਨਤੀਜੇ ਵਚਿਾਰਾਂ ਦੀ ਲੜੀ ਵੱਚਿ ਜਾਣਕਾਰੀ ਦਰਸਾਉਂਦੇ ਹਨ ਜੋ AI ਦੁਆਰਾ ਜਾਰੀ ਰੱਖੀ ਜਾਣੀ ਹੈ, ਤਾਂ ਤੁਹਾਡੇ ਐਪਲੀਕੇਸ਼ਨ ਕੋਡ ਨੂੰ ਕਾਰਜਕਾਰੀ ਨਤੀਜਿਆਂ ਨੂੰ ਚੈਟ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਵੱਚਿ ਪਾਉਣ ਅਤੇ AI ਨੂੰ ਪ੍ਰੋਸੈਸਿੰਗ ਜਾਰੀ ਰੱਖਣ ਦੇਣ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

ਉਦਾਹਰਣ ਲਈ, ਇੱਥੇ ਇੱਕ Raix ਫੰਕਸ਼ਨ ਘੋਸ਼ਣਾ ਹੈ ਜੋ Olympia ਦੇ `AccountManager` ਦੁਆਰਾ ਗਾਹਕ ਸੇਵਾ ਲਈ ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ ਦੇ ਹੱਸਿ ਵਜੋਂ ਸਾਡੇ ਗਾਹਕਾਂ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਲਈ ਵਰਤੀ ਜਾਂਦੀ ਹੈ।

```ruby
class AccountManager
  include Raix::ChatCompletion
  include Raix::FunctionDispatch

  # lots of other functions...

  function :notify_account_owner,
    "Don't share UUID. Mention dollars if subscription changed",
    message: { type: "string" } do |arguments|
      account.owner.freeform_notify(
        subject: "Account Change Notification",
        message: arguments[:message]
      )
      "Notified account owner"
    end
```

ਇਹ ਤੁਰੰਤ ਸਪੱਸ਼ਟ ਨਹੀਂ ਹੋ ਸਕਦਾ ਕਿ ਇੱਥੇ ਕੀ ਹੋ ਰਹਿਾ ਹੈ, ਇਸ ਲਈ ਮੈਂ ਇਸਨੂੰ ਵਸਿਥਾਰ ਨਾਲ ਸਮਝਾਵਾਂਗਾ।

1. `AccountManager` ਕਲਾਸ ਖਾਤਾ ਪ੍ਰਬੰਧਨ ਨਾਲ ਸਬੰਧਤ ਕਈ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਪਰਭਾਸ਼ਤ ਕਰਦੀ ਹੈ। ਇਹ ਤੁਹਾਡੀ ਯੋਜਨਾ ਨੂੰ ਬਦਲ ਸਕਦੀ ਹੈ, ਟੀਮ ਮੈਂਬਰਾਂ ਨੂੰ ਜੋੜ ਅਤੇ ਹਟਾ ਸਕਦੀ ਹੈ, ਅਤੇ ਹੋਰ ਬਹੁਤ ਕੁਝ ਕਰ ਸਕਦੀ ਹੈ।

2. ਇਸਦੇ ਉੱਪਰਲੇ-ਪੱਧਰ ਦੇ ਨਿਰਦੇਸ਼ `AccountManager` ਨੂੰ ਦੱਸਦੇ ਹਨ ਕਿ ਇਸਨੂੰ `notify_account_owner` ਫੰਕਸ਼ਨ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਖਾਤਾ ਬਦਲਣ ਦੀ ਬੇਨਤੀ ਦੇ ਨਤੀਜਿਆਂ ਬਾਰੇ ਖਾਤਾ ਮਾਲਕ ਨੂੰ ਸੂਚਤਿ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ।
3. ਫੰਕਸ਼ਨ ਦੀ ਸੰਖੇਪ ਪਰਭਾਸ਼ਾ ਵਿੱਚ ਸ਼ਾਮਲ ਹਨ:

- ਨਾਮ
- ਵੇਰਵਾ
- ਪੈਰਾਮੀਟਰ `message: { type: "string" }`
- ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਕਰਨ 'ਤੇ ਚਲਾਉਣ ਲਈ ਇੱਕ ਬਲਾਕ

ਫੰਕਸ਼ਨ ਬਲਾਕ ਦੇ ਨਤੀਜਿਆਂ ਨਾਲ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਨੂੰ ਅਪਡੇਟ ਕਰਨ ਤੋਂ ਬਾਅਦ, `chat_completion` ਮੈਥਡ ਨੂੰ ਦੁਬਾਰਾ ਕਾਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਮੈਥਡ ਅੱਗੇ ਦੀ ਪ੍ਰੋਸੈਸਿੰਗ ਲਈ AI ਮਾਡਲ ਨੂੰ ਅਪਡੇਟ ਕੀਤੀ ਗੱਲਬਾਤ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਭੇਜਣ ਲਈ ਜ਼ਿੰਮੇਵਾਰ ਹੈ। ਅਸੀਂ ਇਸ ਪ੍ਰਕਰਿਆ ਨੂੰ ਗੱਲਬਾਤ ਲੂਪ ਕਹਿੰਦੇ ਹਾਂ।

ਜਦੋਂ AI ਮਾਡਲ ਨੂੰ ਅਪਡੇਟ ਕੀਤੀ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਨਾਲ ਇੱਕ ਨਵੀਂ ਚੈਟ ਪੂਰਤੀ ਬੇਨਤੀ ਪ੍ਰਾਪਤ ਹੁੰਦੀ ਹੈ, ਇਸਨੂੰ ਪਹਿਲਾਂ ਚਲਾਏ ਗਏ ਫੰਕਸ਼ਨ ਦੇ ਨਤੀਜਿਆਂ ਤੱਕ ਪਹੁੰਚ ਹੁੰਦੀ ਹੈ। ਇਹ ਇਨ੍ਹਾਂ ਨਤੀਜਿਆਂ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰ ਸਕਦਾ ਹੈ, ਉਨ੍ਹਾਂ ਨੂੰ ਆਪਣੀ ਫੈਸਲਾ ਲੈਣ ਦੀ ਪ੍ਰਕਰਿਆ ਵਿੱਚ ਸ਼ਾਮਲ ਕਰ ਸਕਦਾ ਹੈ, ਅਤੇ ਗੱਲਬਾਤ ਦੇ ਸੰਚਤਿ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਅਗਲਾ ਜਵਾਬ ਜਾਂ ਕਾਰਵਾਈ ਤਿਆਰ ਕਰ ਸਕਦਾ ਹੈ। ਇਹ ਅਪਡੇਟ ਕੀਤੇ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਵਾਧੂ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਚਲਾਉਣ ਦੀ ਚੋਣ ਕਰ ਸਕਦਾ ਹੈ, ਜਾਂ ਜੇਕਰ ਇਹ ਨਿਰਧਾਰਤ ਕਰਦਾ ਹੈ ਕਿ ਕੋਈ ਹੋਰ ਫੰਕਸ਼ਨ ਕਾਲ ਜ਼ਰੂਰੀ ਨਹੀਂ ਹਨ ਤਾਂ ਇਹ ਮੂਲ ਪ੍ਰੌਮਪਟ ਦਾ ਅੰਤਿਮ ਜਵਾਬ ਤਿਆਰ ਕਰ ਸਕਦਾ ਹੈ।

ਮੂਲ ਪ੍ਰੌਮਪਟ ਦੀ ਵਿਕਲਪਿਕ ਨਿਰੰਤਰਤਾ

ਜਦੋਂ ਤੁਸੀਂ ਟੂਲ ਦੇ ਨਤੀਜੇ LLM ਨੂੰ ਵਾਪਸ ਭੇਜਦੇ ਹੋ ਅਤੇ ਮੂਲ ਪ੍ਰੌਮਪਟ ਦੀ ਪ੍ਰੋਸੈਸਿੰਗ ਜਾਰੀ ਰੱਖਦੇ ਹੋ, AI ਉਨ੍ਹਾਂ ਨਤੀਜਿਆਂ ਦੀ ਵਰਤੋਂ ਵਾਧੂ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਕਾਲ ਕਰਨ ਜਾਂ ਅੰਤਿਮ ਸਧਾਰਨ ਟੈਕਸਟ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਲਈ ਕਰਦਾ ਹੈ।

Cohere ਦੇ Command-R ਵਰਗੇ ਕੁਝ ਮਾਡਲ ਆਪਣੇ ਜਵਾਬਾਂ ਵਿੱਚ ਉਨ੍ਹਾਂ ਦੁਆਰਾ ਵਰਤੇ ਗਏ ਵਿਸ਼ਿਸ਼ ਟੂਲਾਂ ਦਾ ਹਵਾਲਾ ਦੇ ਸਕਦੇ ਹਨ, ਜੋ ਵਾਧੂ ਪਾਰਦਰਸ਼ਤਾ ਅਤੇ ਪਤਾ ਲਗਾਉਣ ਦੀ ਯੋਗਤਾ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

ਵਰਤੇ ਜਾ ਰਹੇ ਮਾਡਲ 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹੋਏ, ਫੰਕਸ਼ਨ ਕਾਲ ਦੇ ਨਤੀਜੇ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਸੁਨੇਹਿਆਂ ਵਿੱਚ ਆਪਣੀ ਖਾਸ ਭੂਮਿਕਾ ਨਾਲ ਰਹਿਣਗੇ ਜਾਂ ਕਸਿੇ ਹੋਰ ਸਟਿੈਕਸ ਵਿੱਚ ਝਲਕਣਗੇ। ਪਰ ਮਹੱਤਵਪੂਰਨ ਹਸਿਾ ਇਹ ਹੈ ਕ ਿਉਹ ਡੇਟਾ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਵਿੱਚ ਹੋਵੇ, ਤਾਂ ਜੋ AI ਅੱਗੇ ਕੀ ਕਰਨਾ ਹੈ ਇਸ ਬਾਰੇ ਫੈਸਲਾ ਲੈਣ ਵੇਲੇ ਇਸ 'ਤੇ ਵਚਿਾਰ ਕਰ ਸਕੇ।

ਇੱਕ ਆਮ (ਅਤੇ ਸੰਭਾਵੀ ਤੌਰ 'ਤੇ ਮਹਗਿੀ) ਗਲਤੀ ਦੀ ਸਥਤਿੀ ਹੈ ਚੈਟ ਨੂੰ ਜਾਰੀ ਰੱਖਣ ਤੋਂ ਪਹਲਿਾਂ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਵਿੱਚ ਫੰਕਸ਼ਨ ਦੇ ਨਤੀਜੇ ਜੋੜਨਾ ਭੁੱਲ ਜਾਣਾ। ਨਤੀਜੇ ਵਜੋਂ, AI ਨੂੰ ਲਗਭਗ ਉਸੇ ਤਰ੍ਹਾਂ ਪ੍ਰਰੌਮਪਟ ਕੀਤਾ ਜਾਵੇਗਾ ਜਵਿੇਂ ਇਹ ਪਹਲਿੀ ਵਾਰ ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਕਰਨ ਤੋਂ ਪਹਲਿਾਂ ਸੀ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, AI ਦੇ ਨਜ਼ਰੀਏ ਤੋਂ, ਇਸ ਨੇ ਅਜੇ ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਨਹੀਂ ਕੀਤਾ ਹੈ। ਇਸ ਲਈ ਇਹ ਇਸਨੂੰ ਦੁਬਾਰਾ ਕਾਲ ਕਰਦਾ ਹੈ। ਅਤੇ ਦੁਬਾਰਾ। ਅਤੇ ਦੁਬਾਰਾ, ਜਦੋਂ ਤੱਕ ਤੁਸੀਂ ਇਸਨੂੰ ਰੋਕਦੇ ਨਹੀਂ। ਉਮੀਦ ਹੈ ਕ ਿਤੁਹਾਡਾ ਸੰਦਰਭ ਬਹੁਤ ਵੱਡਾ ਨਹੀਂ ਸੀ, ਅਤੇ ਤੁਹਾਡਾ ਮਾਡਲ ਬਹੁਤ ਮਹਗਿਾ ਨਹੀਂ ਸੀ!

ਟੂਲ ਦੀ ਵਰਤੋਂ ਲਈ ਸਰਵੋਤਮ ਅਭਆਿਸ

ਟੂਲ ਦੀ ਵਰਤੋਂ ਤੋਂ ਵੱਧ ਤੋਂ ਵੱਧ ਲਾਭ ਲੈਣ ਲਈ, ਹੇਠ ਲਖਿ ਸਰਵੋਤਮ ਅਭਆਿਸਾਂ 'ਤੇ ਵਚਿਾਰ ਕਰੋ।

ਵਰਣਨਾਤਮਕ ਪਰਭਿਾਸ਼ਾਵਾਂ

ਹਰੇਕ ਟੂਲ ਅਤੇ ਇਸਦੇ ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰਾਂ ਲਈ ਸਪੱਸ਼ਟ ਅਤੇ ਵਰਣਨਾਤਮਕ ਨਾਮ ਅਤੇ ਵੇਰਵੇ ਪ੍ਰਦਾਨ ਕਰੋ। ਇਹ LLM ਨੂੰ ਹਰੇਕ ਟੂਲ ਦੇ ਉਦੇਸ਼ ਅਤੇ ਸਮਰੱਥਾਵਾਂ ਨੂੰ ਬਹਿਤਰ ਢੰਗ ਨਾਲ ਸਮਝਣ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ।

> ਮੈਂ ਤਜਰਬੇ ਤੋਂ ਤੁਹਾਨੂੰ ਦੱਸ ਸਕਦਾ ਹਾਂ ਕ ਿਆਮ ਸਆਿਣਪ ਜੋ ਕਹਂਦੀ ਹੈ ਕ ਿ"ਨਾਮਕਰਨ ਮੁਸ਼ਕਲ ਹੈ" ਇੱਥੇ ਲਾਗੂ ਹੁੰਦੀ ਹੈ; ਮੈਂ ਸਰਫ਼ ਫੰਕਸ਼ਨਾਂ ਦੇ ਨਾਮ ਜਾਂ ਵੇਰਵਆਿਂ ਦੀ ਸ਼ਬਦਾਵਲੀ ਬਦਲ ਕੇ LLMs ਤੋਂ ਬਹੁਤ ਵੱਖਰੇ ਨਤੀਜੇ ਦੇਖੇ ਹਨ। ਕਈ ਵਾਰ ਵੇਰਵਆਿਂ ਨੂੰ ਹਟਾਉਣ ਨਾਲ ਪ੍ਰਦਰਸ਼ਨ ਸੁਧਰਦਾ ਹੈ।

ਟੂਲ ਨਤੀਜਿਆਂ ਦੀ ਪ੍ਰੋਸੈਸਿੰਗ

ਜਦੋਂ ਟੂਲ ਦੇ ਨਤੀਜਿਆਂ ਨੂੰ LLM ਨੂੰ ਵਾਪਸ ਭੇਜਿਆ ਜਾਂਦਾ ਹੈ, ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਉਹ ਚੰਗੀ ਤਰ੍ਹਾਂ ਢਾਂਚਾਗਤ ਅਤੇ ਵਿਆਪਕ ਹਨ। ਹਰੇਕ ਟੂਲ ਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਅਰਥਪੂਰਨ ਕੀਜ਼ ਅਤੇ ਵੈਲਿਊਜ਼ ਦੀ ਵਰਤੋਂ ਕਰੋ। ਵੱਖ-ਵੱਖ ਫਾਰਮੈਟਾਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰੋ ਅਤੇ ਦੇਖੋ ਕਿ ਕਿਹੜਾ ਸਭ ਤੋਂ ਵਧੀਆ ਕੰਮ ਕਰਦਾ ਹੈ, JSON ਤੋਂ ਲੈ ਕੇ ਸਧਾਰਨ-ਟੈਕਸਟ ਤੱਕ।

ਨਤੀਜਾ ਵਿਆਖਿਆਕਾਰ ਇਸ ਚੁਣੌਤੀ ਨੂੰ AI ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਨਤੀਜਿਆਂ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਨ ਅਤੇ ਮਨੁੱਖ-ਅਨੁਕੂਲ ਵਿਆਖਿਆਵਾਂ, ਸੰਖੇਪ, ਜਾਂ ਮੁੱਖ ਸੰਟੇ ਪ੍ਰਦਾਨ ਕਰਕੇ ਹੱਲ ਕਰਦਾ ਹੈ।

ਗਲਤੀ ਸੰਭਾਲ

ਮਜ਼ਬੂਤ ਗਲਤੀ ਸੰਭਾਲ ਵਿਧੀਆਂ ਲਾਗੂ ਕਰੋ ਜੋ ਉਨ੍ਹਾਂ ਮਾਮਲਿਆਂ ਨੂੰ ਸੰਭਾਲ ਸਕਣ ਜਿੱਥੇ LLM ਟੂਲ ਕਾਲਾਂ ਲਈ ਅਵੈਧ ਜਾਂ ਅਸਮਰਥਿਤ ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰ ਤਿਆਰ ਕਰ ਸਕਦਾ ਹੈ। ਟੂਲ ਚਲਾਉਣ ਦੌਰਾਨ ਹੋਣ ਵਾਲੀਆਂ ਕਿਸੇ ਵੀ ਗਲਤੀਆਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲੋ ਅਤੇ ਉਨ੍ਹਾਂ ਤੋਂ ਬਾਹਰ ਨਿਕਲੋ।

AI ਦੀ ਇੱਕ ਬਹੁਤ ਹੀ ਵਧੀਆ ਗੁਣਵੱਤਾ ਇਹ ਹੈ ਕਿ ਇਹ ਗਲਤੀ ਸੁਨੇਹਿਆਂ ਨੂੰ ਸਮਝਦੀ ਹੈ! ਜਿਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਜੇ ਤੁਸੀਂ ਛੇਤੀ ਅਤੇ ਗੰਦੇ ਮਾਨਸਿਕਤਾ ਨਾਲ ਕੰਮ ਕਰ ਰਹੇ ਹੋ, ਤੁਸੀਂ ਬਸ ਟੂਲ ਦੀ ਲਾਗੂ ਕਰਨ ਵਿੱਚ ਪੈਦਾ ਹੋਈਆਂ ਕਿਸੇ ਵੀ ਐਕਸੈਪਸ਼ਨ ਨੂੰ ਫੜ ਸਕਦੇ ਹੋ, ਅਤੇ ਇਸਨੂੰ AI ਨੂੰ ਵਾਪਸ ਭੇਜ ਸਕਦੇ ਹੋ ਤਾਂ ਜੋ ਇਹ ਜਾਣ ਸਕੇ ਕਿ ਕੀ ਹੋਇਆ!

ਉਦਾਹਰਣ ਲਈ, ਇੱਥੇ Olympia ਵਿੱਚ google ਖੋਜ ਦੀ ਲਾਗੂ ਕਰਨ ਦਾ ਇੱਕ ਛੋਟਾ ਕੀਤਾ ਸੰਸਕਰਣ ਹੈ:

```
def google_search(conversation, params)
  conversation.update_cstatus("Searching Google...")
  query = params[:query]
  search = GoogleSearch.new(query).get_hash

  conversation.update_cstatus("Summarizing results...")
  SummarizeKnowledgeGraph.new.perform(conversation, search.to_json)
rescue StandardError => e
  Honeybadger.notify(e)
  { error: e.message }.inspect
end
```

ਓਲੰਪੀਆ ਵੱਚਿ ਗੂਗਲ ਖੋਜਾਂ ਦੋ-ਪੜਾਵੀ ਪ੍ਰਕਰਿਆ ਹਨ। ਪਹਲਿਾਂ ਤੁਸੀ ਖੋਜ ਕਰਦੇ ਹੋ, ਫਰਿ ਨਤੀਜਆਿਂ ਦਾ ਸਾਰ ਕੱਢਦੇ ਹੋ। ਜੇ ਕੋਈ ਅਸਫਲਤਾ ਆਉਂਦੀ ਹੈ, ਭਾਵੇਂ ਉਹ ਕੁਝ ਵੀ ਹੋਵੇ, ਗਲਤੀ ਦਾ ਸੁਨੇਹਾ ਪੈਕ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਏ.ਆਈ. ਨੂੰ ਵਾਪਸ ਭੇਜ ਦਤਿਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਤਕਨੀਕ ਲਗਭਗ ਸਾਰੇ ਬੁੱਧੀਮਾਨ ਗਲਤੀ ਪ੍ਰਬੰਧਨ ਪੈਟਰਨਾਂ ਦਾ ਆਧਾਰ ਹੈ।

ਉਦਾਹਰਨ ਲਈ, ਮੰਨ ਲਓ ਕਿ `GoogleSearch` ਏ.ਪੀ.ਆਈ. ਕਾਲ 503 ਸਰਵਸਿ ਅਣਉਪਲਬਧ ਗਲਤੀ ਕਾਰਨ ਅਸਫਲ ਹੋ ਜਾਂਦੀ ਹੈ। ਇਹ ਉੱਪਰਲੇ-ਪੱਧਰ ਦੇ ਰੈਸਕਊ ਤੱਕ ਪਹੁੰਚਦੀ ਹੈ, ਅਤੇ ਗਲਤੀ ਦਾ ਵੇਰਵਾ ਫੰਕਸ਼ਨ ਕਾਲ ਦੇ ਨਤੀਜੇ ਵਜੋਂ ਏ.ਆਈ. ਨੂੰ ਵਾਪਸ ਭੇਜ ਦਤਿਾ ਜਾਂਦਾ ਹੈ। ਵਰਤੋਂਕਾਰ ਨੂੰ ਖਾਲੀ ਸਕ੍ਰੀਨ ਜਾਂ ਤਕਨੀਕੀ ਗਲਤੀ ਦੇਣ ਦੀ ਬਜਾਏ, ਏ.ਆਈ. ਕੁਝ ਅਜਹਿਾ ਕਹਿੰਦੀ ਹੈ "ਮੈਨੂੰ ਮਾਫ਼ ਕਰਨਾ, ਪਰ ਮੈਂ ਇਸ ਸਮੇਂ ਆਪਣੀਆਂ ਗੂਗਲ ਖੋਜ ਸਮਰੱਥਾਵਾਂ ਤੱਕ ਪਹੁੰਚ ਨਹੀਂ ਕਰ ਸਕਦੀ। ਜੇ ਤੁਸੀ ਚਾਹੋ ਤਾਂ ਮੈਂ ਬਾਅਦ ਵੱਚਿ ਦੁਬਾਰਾ ਕੋਸ਼ਸ਼ ਕਰ ਸਕਦੀ ਹਾਂ।"

ਇਹ ਸਰਫ਼ ਇੱਕ ਚਲਾਕ ਚਾਲ ਜਾਪ ਸਕਦੀ ਹੈ, ਪਰ ਇੱਕ ਵੱਖਰੀ ਕਸਿਮ ਦੀ ਗਲਤੀ 'ਤੇ ਵਚਿਾਰ ਕਰੋ, ਜੱਥੇ ਏ.ਆਈ. ਬਾਹਰੀ ਏ.ਪੀ.ਆਈ. ਨੂੰ ਕਾਲ ਕਰ ਰਹੀ ਸੀ ਅਤੇ ਏ.ਪੀ.ਆਈ. ਨੂੰ ਭੇਜੇ ਜਾਣ ਵਾਲੇ ਮਾਪਦੰਡਾਂ 'ਤੇ ਸੰਧਾ ਨਯਿੰਤਰਣ ਸੀ। ਹੋ ਸਕਦਾ ਹੈ ਕਿ ਇਸਨੇ ਉਹਨਾਂ ਮਾਪਦੰਡਾਂ ਨੂੰ ਤਆਿਰ ਕਰਨ ਦੇ ਤਰੀਕੇ ਵੱਚਿ ਕੋਈ ਗਲਤੀ ਕੀਤੀ ਹੋਵੇ? ਬਸਰਤੇ ਕਿ ਬਾਹਰੀ ਏ.ਪੀ.ਆਈ. ਤੋਂ ਗਲਤੀ ਦਾ ਸੁਨੇਹਾ ਕਾਫ਼ੀ ਵਸਿਥਾਰਪੂਰਵਕ ਹੋਵੇ, ਗਲਤੀ ਦਾ ਸੁਨੇਹਾ ਕਾਲ ਕਰਨ ਵਾਲੀ ਏ.ਆਈ. ਨੂੰ ਵਾਪਸ ਭੇਜਣ ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਇਹ ਉਹਨਾਂ ਮਾਪਦੰਡਾਂ 'ਤੇ ਮੁੜ ਵਚਿਾਰ ਕਰ ਸਕਦੀ ਹੈ ਅਤੇ ਦੁਬਾਰਾ ਕੋਸ਼ਸ਼ ਕਰ ਸਕਦੀ ਹੈ। ਆਪਣੇ ਆਪ। ਭਾਵੇਂ ਗਲਤੀ ਕੋਈ ਵੀ ਹੋਵੇ।

ਹੁਣ ਸੋਚੋ ਕਿ ਆਮ ਕੋਡ ਵੱਚਿ ਉਸ ਕਸਿਮ ਦੇ ਮਜ਼ਬੂਤ ਗਲਤੀ ਪ੍ਰਬੰਧਨ ਨੂੰ ਦੁਹਰਾਉਣ ਲਈ ਕੀ ਲੋੜੀਂਦਾ ਹੋਵੇਗਾ। ਇਹ ਲਗਭਗ ਅਸੰਭਵ ਹੈ।

ਲਗਾਤਾਰ ਸੁਧਾਰ

ਜੇਕਰ ਐਲ.ਐਲ.ਐਮ. ਢੁਕਵੇਂ ਟੂਲਜ਼ ਦੀ ਸਫਿਾਰਸ਼ ਨਹੀਂ ਕਰ ਰਹਿਾ ਜਾਂ ਘੱਟ-ਵਧੀਆ ਜਵਾਬ ਤਆਿਰ ਕਰ ਰਹਿਾ ਹੈ, ਤਾਂ ਟੂਲ ਪਰਭਾਸ਼ਾਵਾਂ, ਵੇਰਵਆਿਂ, ਅਤੇ ਇਨਪੁੱਟ ਮਾਪਦੰਡਾਂ 'ਤੇ ਕੰਮ ਕਰਦੇ ਰਹੋ। ਦੇਖੇ ਗਏ ਵਵਿਹਾਰ ਅਤੇ ਲੋੜੀਂਦੇ ਨਤੀਜਆਿਂ ਦੇ ਆਧਾਰ 'ਤੇ ਟੂਲ ਸੈੱਟਅੱਪ ਨੂੰ ਲਗਾਤਾਰ ਸੁਧਾਰੋ ਅਤੇ ਬਹਿਤਰ ਬਣਾਓ।

1. ਸਧਾਰਨ ਟੂਲ ਪਰਭਾਸ਼ਾਵਾਂ ਨਾਲ ਸ਼ੁਰੂ ਕਰੋ: ਸਪੱਸ਼ਟ ਅਤੇ ਸੰਖੇਪ ਨਾਮਾਂ, ਵੇਰਵਆਿਂ, ਅਤੇ ਇਨਪੁੱਟ ਮਾਪਦੰਡਾਂ ਨਾਲ ਟੂਲਜ਼ ਨੂੰ ਪਰਭਾਸ਼ਤਿ ਕਰਕੇ ਸ਼ੁਰੂ ਕਰੋ। ਸ਼ੁਰੂ ਵੱਚਿ ਟੂਲ ਸੈੱਟਅੱਪ ਨੂੰ ਗੁੰਝਲਦਾਰ ਬਣਾਉਣ ਤੋਂ ਬਚੋ ਅਤੇ ਮੁੱਖ ਕਾਰਜਸ਼ੀਲਤਾ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰੋ। ਉਦਾਹਰਨ ਲਈ, ਜੇਕਰ ਤੁਸੀਂ ਭਾਵਨਾ ਵਸ਼ਿਲੇਸ਼ਣ ਦੇ ਨਤੀਜਆਿਂ ਨੂੰ ਸੇਵ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹੋ, ਤਾਂ ਇੱਕ ਮੁੱਢਲੀ ਪਰਭਾਸ਼ਾ ਨਾਲ ਸ਼ੁਰੂ ਕਰੋ ਜਵਿੇਂ:

```
{
  "name": "save_sentiment_score",
  "description": "Analyze user-provided text and generate sentiment score",
  "parameters": {
    "type": "object",
    "properties": {
      "score": {
        "type": "float",
        "description": "sentiment score from -1 (negative) to 1 (positive)"
      }
    },
    "required": ["score"]
  }
}
```

2. ਟੈਸਟ ਅਤੇ ਨਰੀਖਣ ਕਰੋ: ਜਦੋਂ ਤੁਹਾਡੇ ਕੋਲ ਸ਼ੁਰੂਆਤੀ ਟੂਲ ਪਰਭਾਸ਼ਾਵਾਂ ਤਆਿਰ ਹੋਣ, ਵੱਖ-ਵੱਖ ਪ੍ਰੌਮਪਟਸ ਨਾਲ ਉਨ੍ਹਾਂ ਦੀ ਜਾਂਚ ਕਰੋ ਅਤੇ ਦੇਖੋ ਕਿ LLM ਟੂਲ ਨਾਲ ਕਵਿੇਂ ਅੰਤਰਕਰਿਆ ਕਰਦਾ ਹੈ। ਉਤਪੰਨ ਕੀਤੇ ਜਵਾਬਾਂ ਦੀ ਗੁਣਵੱਤਾ ਅਤੇ ਢੁਕਵੇਂਪਣ ਵੱਲ ਧਆਿਨ ਦਓਿ। ਜੇ LLM ਘੱਟ-ਵਧੀਆ ਜਵਾਬ ਤਆਿਰ ਕਰ ਰਹਿਾ ਹੈ, ਤਾਂ ਟੂਲ ਪਰਭਾਸ਼ਾਵਾਂ ਨੂੰ ਸੁਧਾਰਨ ਦਾ ਸਮਾਂ ਆ ਗਆਿ ਹੈ।
3. ਵੇਰਵਆਿਂ ਨੂੰ ਸੁਧਾਰੋ: ਜੇ LLM ਕਸਿੇ ਟੂਲ ਦੇ ਉਦੇਸ਼ ਨੂੰ ਗਲਤ ਸਮਝ ਰਹਿਾ ਹੈ, ਤਾਂ ਟੂਲ ਦੇ ਵੇਰਵੇ ਨੂੰ ਸੁਧਾਰਨ ਦੀ ਕੋਸ਼ਸ਼ ਕਰੋ। LLM ਨੂੰ ਟੂਲ ਦੀ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਵਰਤੋਂ ਵੱਚਿ ਮਾਰਗਦਰਸ਼ਨ ਕਰਨ ਲਈ ਵਧੇਰੇ ਸੰਦਰਭ, ਉਦਾਹਰਣਾਂ, ਜਾਂ ਸਪੱਸ਼ਟੀਕਰਨ ਪ੍ਰਦਾਨ ਕਰੋ। ਉਦਾਹਰਣ ਲਈ, ਤੁਸੀਂ ਭਾਵਨਾਤਮਕ ਵਸ਼ਿਲੇਸ਼ਣ

ਟੂਲ ਦੇ ਵੇਰਵੇ ਨੂੰ ਵਸਿ਼ਲੇਸ਼ਣ ਕੀਤੇ ਜਾ ਰਹੇ ਟੈਕਸਟ ਦੀ ਭਾਵਨਾਤਮਕ ਸੁਰ ਨੂੰ ਵਧੇਰੇ ਖਾਸ ਤੌਰ 'ਤੇ ਸੰਬੋਧਤਿ ਕਰਨ ਲਈ ਅੱਪਡੇਟ ਕਰ ਸਕਦੇ ਹੋ:

```
{
  "name": "save_sentiment_score",
  "description": "Determine the overall emotional tone of a piece of text,
   such as customer reviews, social media posts, or feedback comments.",
  ...
}
```

4. ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰ ਵਵਿਸਥਤਿ ਕਰੋ: ਜੇਕਰ ਐੱਲਐੱਲਐੱਮ ਕਸਿੇ ਟੂਲ ਲਈ ਅਯੋਗ ਜਾਂ ਅਪ੍ਰਸੰਗਕਿ ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰ ਤਆਿਰ ਕਰ ਰਹਿਾ ਹੈ, ਤਾਂ ਪੈਰਾਮੀਟਰ ਪਰਭਾਸ਼ਾਵਾਂ ਨੂੰ ਵਵਿਸਥਤਿ ਕਰਨ 'ਤੇ ਵਚਿਾਰ ਕਰੋ। ਅਪੇਖਤਿ ਇਨਪੁੱਟ ਫਾਰਮੈਟ ਨੂੰ ਸਪੱਸ਼ਟ ਕਰਨ ਲਈ ਵਧੇਰੇ ਵਸਿ਼ਸ਼ ਸੀਮਾਵਾਂ, ਪ੍ਰਮਾਣੀਕਰਨ ਨਯਿਮ, ਜਾਂ ਉਦਾਹਰਣਾਂ ਸ਼ਾਮਲ ਕਰੋ।
5. ਫੀਡਬੈਕ ਦੇ ਆਧਾਰ 'ਤੇ ਸੁਧਾਰ ਕਰੋ: ਆਪਣੇ ਟੂਲਜ਼ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ 'ਤੇ ਲਗਾਤਾਰ ਨਜ਼ਰ ਰੱਖੋ ਅਤੇ ਉਪਭੋਗਤਾਵਾਂ ਜਾਂ ਹੱਸਿੇਦਾਰਾਂ ਤੋਂ ਫੀਡਬੈਕ ਇਕੱਠਾ ਕਰੋ। ਇਸ ਫੀਡਬੈਕ ਦੀ ਵਰਤੋਂ ਸੁਧਾਰ ਦੇ ਖੇਤਰਾਂ ਦੀ ਪਛਾਣ ਕਰਨ ਅਤੇ ਟੂਲ ਪਰਭਾਸ਼ਾਵਾਂ ਵਚਿ ਲਗਾਤਾਰ ਸੁਧਾਰ ਕਰਨ ਲਈ ਕਰੋ। ਉਦਾਹਰਣ ਲਈ, ਜੇਕਰ ਉਪਭੋਗਤਾ ਰਪਿੋਰਟ ਕਰਦੇ ਹਨ ਕ ਿਵਸਿ਼ਲੇਸ਼ਣ ਵਅਿੰਗ ਨੂੰ ਚੰਗੀ ਤਰ੍ਹਾਂ ਨਹੀ ਸੰਭਾਲ ਰਹਿਾ ਹੈ, ਤਾਂ ਤੁਸੀ ਵੇਰਵੇ ਵਚਿ ਇੱਕ ਨੋਟ ਜੋੜ ਸਕਦੇ ਹੋ:

```
{
  "name": "save_sentiment_score",
  "description": "Analyze the sentiment of a given text and return a sentiment
   score between -1 (negative) and 1 (positive). Note: Sarcasm should be
   considered negative.",
  ...
}
```

ਆਪਣੀਆਂ ਟੂਲ ਪਰਭਾਸ਼ਾਵਾਂ ਨੂੰ ਦੇਖੇ ਗਏ ਵਵਿਹਾਰ ਅਤੇ ਫੀਡਬੈਕ ਦੇ ਆਧਾਰ 'ਤੇ ਲਗਾਤਾਰ ਸੁਧਾਰ ਕਰਕੇ, ਤੁਸੀ ਆਪਣੀ AI-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ੀਲਤਾ ਨੂੰ ਹੌਲੀ-ਹੌਲੀ ਬਹਿਤਰ ਬਣਾ ਸਕਦੇ ਹੋ। ਟੂਲ ਪਰਭਾਸ਼ਾਵਾਂ ਨੂੰ ਸਪੱਸ਼ਟ, ਸੰਖੇਪ, ਅਤੇ ਖਾਸ ਕੰਮ 'ਤੇ ਕੇਂਦਰਤਿ ਰੱਖਣਾ ਯਾਦ ਰੱਖੋ। ਨਯਿਮਤਿ ਤੌਰ 'ਤੇ ਟੂਲ ਇੰਟਰੈਕਸ਼ਨਾਂ ਦੀ ਜਾਂਚ ਅਤੇ ਪੁਸ਼ਟੀ ਕਰੋ ਤਾਂ ਜੋ ਇਹ ਯਕੀਨੀ ਬਣਾਇਆ ਜਾ ਸਕੇ ਕ ਿਉਹ ਤੁਹਾਡੇ ਇੱਛਤ ਨਤੀਜਆਿਂ ਨਾਲ ਮੇਲ ਖਾਂਦੇ ਹਨ।

ਟੂਲਜ਼ ਦੀ ਰਚਨਾ ਅਤੇ ਲੜੀਬੱਧਤਾ

ਟੂਲ ਵਰਤੋਂ ਦਾ ਸਭ ਤੋਂ ਸ਼ਕਤੀਸ਼ਾਲੀ ਪਹਲੂ, ਜਸਿ ਬਾਰੇ ਹੁਣ ਤੱਕ ਸਰਿਫ਼ ਸੰਕੇਤ ਕੀਤਾ ਗਆਿ ਹੈ, ਉਹ ਹੈ ਗੁੰਝਲਦਾਰ ਕੰਮਾਂ ਨੂੰ ਪੂਰਾ ਕਰਨ ਲਈ ਕਈ ਟੂਲਾਂ ਨੂੰ ਇਕੱਠੇ ਜੋੜਨ ਅਤੇ ਲੜੀਬੱਧ ਕਰਨ ਦੀ ਯੋਗਤਾ। ਆਪਣੀਆਂ ਟੂਲ ਪਰਭਿਾਸ਼ਾਵਾਂ ਅਤੇ ਉਨ੍ਹਾਂ ਦੇ ਇਨਪੁੱਟ/ਆਉਟਪੁੱਟ ਫਾਰਮੈਟਾਂ ਨੂੰ ਧਆਿਨ ਨਾਲ ਡਜ਼ਾਈਨ ਕਰਕੇ, ਤੁਸੀਂ ਮੁੜ-ਵਰਤੋਂ ਯੋਗ ਬਲਿਡਿੰਗ ਬਲਾਕ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਵੱਖ-ਵੱਖ ਤਰੀਕਆਿਂ ਨਾਲ ਜੋੜੇ ਜਾ ਸਕਦੇ ਹਨ।

ਆਓ ਇੱਕ ਉਦਾਹਰਣ 'ਤੇ ਵਚਾਰ ਕਰੀਏ ਜੱਥੇ ਤੁਸੀਂ ਆਪਣੀ AI-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨ ਲਈ ਇੱਕ ਡਾਟਾ ਵਸ਼ਿਲੇਸ਼ਣ ਪਾਈਪਲਾਈਨ ਬਣਾ ਰਹੇ ਹੋ। ਤੁਹਾਡੇ ਕੋਲ ਹੇਠ ਲਖਿ ਟੂਲ ਹੋ ਸਕਦੇ ਹਨ:

1. **`DataRetrieval`:** ਇੱਕ ਟੂਲ ਜੋ ਨਰਿਧਾਰਤ ਮਾਪਦੰਡਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਡਾਟਾਬੇਸ ਜਾਂ API ਤੋਂ ਡਾਟਾ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ।
2. **`DataProcessing`:** ਇੱਕ ਟੂਲ ਜੋ ਪ੍ਰਾਪਤ ਕੀਤੇ ਡਾਟਾ 'ਤੇ ਗਣਨਾਵਾਂ, ਰੂਪਾਂਤਰਣ, ਜਾਂ ਇਕੱਤਰੀਕਰਨ ਕਰਦਾ ਹੈ।
3. **`DataVisualization`:** ਇੱਕ ਟੂਲ ਜੋ ਪ੍ਰੋਸੈਸ ਕੀਤੇ ਡਾਟਾ ਨੂੰ ਵਰਤੋਂਕਾਰ-ਅਨੁਕੂਲ ਫਾਰਮੈਟ ਵੱਚਿ ਪੇਸ਼ ਕਰਦਾ ਹੈ, ਜਵਿੇਂ ਕ ਿਚਾਰਟ ਜਾਂ ਗਰਾਫ।

ਇਨ੍ਹਾਂ ਟੂਲਾਂ ਨੂੰ ਇੱਕ ਲੜੀ ਵੱਚਿ ਜੋੜ ਕੇ, ਤੁਸੀਂ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਪ੍ਰਸੰਗਕਿ ਡਾਟਾ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ, ਇਸ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਦਾ ਹੈ, ਅਤੇ ਨਤੀਜਆਿਂ ਨੂੰ ਅਰਥਪੂਰਨ ਢੰਗ ਨਾਲ ਪੇਸ਼ ਕਰਦਾ ਹੈ। ਇੱਥੇ ਟੂਲ ਵਰਤੋਂ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਕਵਿੇਂ ਦਖਿਾਈ ਦੇ ਸਕਦਾ ਹੈ:

1. LLM ਇੱਕ ਖਾਸ ਉਤਪਾਦ ਸ਼੍ਰੇਣੀ ਲਈ ਵਕਿਰੀ ਡਾਟਾ 'ਤੇ ਅੰਤਰਦ੍ਰਸ਼ਿਟੀ ਦੀ ਮੰਗ ਕਰਦੇ ਵਰਤੋਂਕਾਰ ਦੇ ਸਵਾਲ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ।
2. LLM `DataRetrieval` ਟੂਲ ਦੀ ਚੋਣ ਕਰਦਾ ਹੈ ਅਤੇ ਡਾਟਾਬੇਸ ਤੋਂ ਪ੍ਰਸੰਗਕਿ ਵਕਿਰੀ ਡਾਟਾ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਢੁਕਵੇਂ ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰ ਤਆਿਰ ਕਰਦਾ ਹੈ।
3. ਪ੍ਰਾਪਤ ਕੀਤਾ ਡਾਟਾ `DataProcessing` ਟੂਲ ਨੂੰ "ਪਾਸ" ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਜੋ ਕੁੱਲ ਮਾਲੀਆ, ਔਸਤ ਵਕਿਰੀ ਕੀਮਤ, ਅਤੇ ਵਕਿਾਸ ਦਰ ਵਰਗੇ ਮੈਟ੍ਰਕਿਸ ਦੀ ਗਣਨਾ ਕਰਦਾ ਹੈ।
4. ਫਰਿ ਪ੍ਰੋਸੈਸ ਕੀਤਾ ਡਾਟਾ `DataVisualization` ਟੂਲ ਦੁਆਰਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਜੋ ਅੰਤਰਦ੍ਰਸ਼ਿਟੀਆਂ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਇੱਕ ਆਕਰਸ਼ਕ ਚਾਰਟ ਜਾਂ ਗਰਾਫ ਬਣਾਉਂਦਾ ਹੈ, ਅਤੇ ਚਾਰਟ ਦਾ URL LLM ਨੂੰ ਵਾਪਸ ਭੇਜਦਾ ਹੈ।

5. ਅੰਤ ਵੱਚਿ, LLM ਮਾਰਕਡਾਊਨ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਵਰਤੋਂਕਾਰ ਦੇ ਸਵਾਲ ਦਾ ਇੱਕ ਫਾਰਮੈਟਡ ਜਵਾਬ ਤਆਿਰ ਕਰਦਾ ਹੈ, ਜਸਿ ਵੱਚਿ ਵਜ਼ੁਅਲਾਈਜ਼ ਕੀਤਾ ਡਾਟਾ ਅਤੇ ਮੁੱਖ ਨਤੀਜਆਿਂ ਦਾ ਸਾਰ ਸ਼ਾਮਲ ਹੁੰਦਾ ਹੈ।

ਇਨ੍ਹਾਂ ਟੂਲਾਂ ਨੂੰ ਇਕੱਠੇ ਜੋੜ ਕੇ, ਤੁਸੀ ਇੱਕ ਨਰਿਵਘਿਨ ਡਾਟਾ ਵਸਿ਼ਲੇਸ਼ਣ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਵੱਚਿ ਆਸਾਨੀ ਨਾਲ ਏਕੀਕ੍ਰਤਿ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਸ ਪਹੁੰਚ ਦੀ ਖੂਬਸੂਰਤੀ ਇਹ ਹੈ ਕਿ ਹਰੇਕ ਟੂਲ ਨੂੰ ਸੁਤੰਤਰ ਤੌਰ ’ਤੇ ਵਕਿਸਤਿ ਅਤੇ ਟੈਸਟ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਅਤੇ ਫਰਿ ਵੱਖ-ਵੱਖ ਸਮੱਸਆਿਵਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਵੱਖ-ਵੱਖ ਤਰੀਕਆਿਂ ਨਾਲ ਜੋੜਆਿ ਜਾ ਸਕਦਾ ਹੈ।

ਟੂਲਜ਼ ਦੀ ਸੁਚਾਰੂ ਕੰਪੋਜ਼ੀਸ਼ਨ ਅਤੇ ਚੇਨੰਗਿ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਣ ਲਈ, ਹਰ ਟੂਲ ਲਈ ਸਪੱਸ਼ਟ ਇਨਪੁੱਟ ਅਤੇ ਆਉਟਪੁੱਟ ਫਾਰਮੈਟਸ ਨੂੰ ਪਰਭਾਸ਼ਤਿ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ।

ਉਦਾਹਰਣ ਵਜੋਂ, `DataRetrieval` ਟੂਲ ਡੇਟਾਬੇਸ ਕਨੈਕਸ਼ਨ ਵੇਰਵੇ, ਟੇਬਲ ਨਾਮ, ਅਤੇ ਕਵੇਰੀ ਸ਼ਰਤਾਂ ਵਰਗੇ ਪੈਰਾਮੀਟਰ ਸਵੀਕਾਰ ਕਰ ਸਕਦਾ ਹੈ, ਅਤੇ ਨਤੀਜੇ ਨੂੰ ਇੱਕ ਸੰਰਚਤਿ JSON ਆਬਜੈਕਟ ਵਜੋਂ ਵਾਪਸ ਕਰ ਸਕਦਾ ਹੈ। `DataProcessing` ਟੂਲ ਫਰਿ ਇਸ JSON ਆਬਜੈਕਟ ਨੂੰ ਇਨਪੁੱਟ ਵਜੋਂ ਲੈ ਸਕਦਾ ਹੈ ਅਤੇ ਇੱਕ ਬਦਲਆਿ ਹੋਇਆ JSON ਆਬਜੈਕਟ ਆਉਟਪੁੱਟ ਵਜੋਂ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ। ਟੂਲਜ਼ ਵਚਿਕਾਰ ਡਾਟਾ ਪ੍ਰਵਾਹ ਨੂੰ ਮਆਿਰੀ ਬਣਾ ਕੇ, ਤੁਸੀ ਅਨੁਕੂਲਤਾ ਅਤੇ ਮੁੜ-ਵਰਤੋਂ ਨੂੰ ਯਕੀਨੀ ਬਣਾ ਸਕਦੇ ਹੋ।

ਜਦੋਂ ਤੁਸੀ ਆਪਣੇ ਟੂਲ ਈਕੋਸਸਿਟਮ ਦੀ ਡਜ਼ਾਈਨ ਕਰਦੇ ਹੋ, ਇਸ ਬਾਰੇ ਸੋਚੋ ਕਿ ਵੱਖ-ਵੱਖ ਟੂਲਜ਼ ਨੂੰ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਵੱਚਿ ਆਮ ਵਰਤੋਂ ਦੇ ਕੇਸਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਕਵਿੇਂ ਜੋੜਆਿ ਜਾ ਸਕਦਾ ਹੈ। ਉੱਚ-ਪੱਧਰੀ ਟੂਲਜ਼ ਬਣਾਉਣ ਬਾਰੇ ਵਚਿਾਰ ਕਰੋ ਜੋ ਆਮ ਵਰਕਫਲੋਜ਼ ਜਾਂ ਬਜ਼ਿਨਸ ਲੌਜਕਿ ਨੂੰ ਸ਼ਾਮਲ ਕਰਦੇ ਹਨ, ਜਸਿ ਨਾਲ LLM ਲਈ ਉਨ੍ਹਾਂ ਨੂੰ ਚੁਣਨਾ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਵਰਤਣਾ ਆਸਾਨ ਹੋ ਜਾਂਦਾ ਹੈ।

ਯਾਦ ਰੱਖੋ, ਟੂਲ ਵਰਤੋਂ ਦੀ ਸ਼ਕਤੀ ਇਸ ਦੀ ਲਚਕਤਾ ਅਤੇ ਮੋਡੂਲਰਤਾ ਵੱਚਿ ਹੈ। ਗੁੰਝਲਦਾਰ ਕਾਰਜਾਂ ਨੂੰ ਛੋਟੇ, ਮੁੜ-ਵਰਤੋਂ ਯੋਗ ਟੂਲਜ਼ ਵੱਚਿ ਵੰਡ ਕੇ, ਤੁਸੀ ਇੱਕ ਮਜ਼ਬੂਤ ਅਤੇ ਅਨੁਕੂਲ AI-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਵੱਖ-ਵੱਖ ਚੁਣੌਤੀਆਂ ਦਾ ਸਾਹਮਣਾ ਕਰ ਸਕਦੀ ਹੈ।

ਭਵੱਖਿ ਦੀਆਂ ਦਸ਼ਾਵਾਂ

ਜਵਿੇਂ-ਜਵਿੇਂ AI-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨ ਵਕਿਾਸ ਦਾ ਖੇਤਰ ਵਕਿਸਤ ਹੁੰਦਾ ਹੈ, ਅਸੀ ਟੂਲ ਵਰਤੋਂ ਸਮਰੱਥਾਵਾਂ ਵੱਚਿ ਹੋਰ ਤਰੱਕੀ ਦੀ ਉਮੀਦ ਕਰ ਸਕਦੇ ਹਾਂ। ਕੁਝ ਸੰਭਾਵਤਿ ਭਵੱਖਿ ਦੀਆਂ ਦਸ਼ਾਵਾਂ ਵੱਚਿ ਸ਼ਾਮਲ ਹਨ:

1. **ਮਲਟੀ-ਹੌਪ ਟੂਲ ਵਰਤੋਂ**: LLMs ਇਹ ਫੈਸਲਾ ਕਰਨ ਦੇ ਯੋਗ ਹੋ ਸਕਦੇ ਹਨ ਕਿ ਉਨ੍ਹਾਂ ਨੂੰ ਸੰਤੋਸ਼ਜਨਕ ਜਵਾਬ ਤਿਆਰ ਕਰਨ ਲਈ ਕਿੰਨੀ ਵਾਰ ਟੂਲਜ਼ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦੀ ਲੋੜ ਹੈ। ਇਸ ਵਿੱਚ ਵਿਚਿਕਾਰਲੇ ਨਤੀਜਿਆਂ ਦੇ ਆਧਾਰ 'ਤੇ ਟੂਲ ਚੋਣ ਅਤੇ ਕਾਰਜਾਂਵਿਤੀ ਦੇ ਕਈ ਗੇੜ ਸ਼ਾਮਲ ਹੋ ਸਕਦੇ ਹਨ।
2. **ਪਹਲਿਾਂ-ਪਰਭਾਸ਼ਿਤ ਟੂਲਜ਼**: AI ਪਲੇਟਫਾਰਮ ਪਹਲਿਾਂ-ਪਰਭਾਸ਼ਿਤ ਟੂਲਜ਼ ਦਾ ਸੈੱਟ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ ਜਨ੍ਹਿਾਂ ਨੂੰ ਡਵਿੈਲਪਰ ਆਉਟ-ਆਫ-ਦ-ਬਾਕਸ ਵਰਤ ਸਕਦੇ ਹਨ, ਜਵਿੇਂ ਕਿ Python ਇੰਟਰਪ੍ਰੇਟਰ, ਵੈੱਬ ਖੋਜ ਟੂਲਜ਼, ਜਾਂ ਆਮ ਯੂਟਲਿਟੀ ਫੰਕਸ਼ਨ।
3. **ਨਰਿਵਘਿਨ ਏਕੀਕਰਣ**: ਜਵਿੇਂ-ਜਵਿੇਂ ਟੂਲ ਵਰਤੋਂ ਵਧੇਰੇ ਪ੍ਰਚਲਤਿ ਹੁੰਦੀ ਹੈ, ਅਸੀਂ AI ਪਲੇਟਫਾਰਮਾਂ ਅਤੇ ਪ੍ਰਸਧਿ ਵਕਿਾਸ ਫਰੇਮਵਰਕਾਂ ਵਚਿਕਾਰ ਬਹਿਤਰ ਏਕੀਕਰਣ ਦੀ ਉਮੀਦ ਕਰ ਸਕਦੇ ਹਾਂ, ਜੋ ਡਵਿੈਲਪਰਾਂ ਲਈ ਆਪਣੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਟੂਲ ਵਰਤੋਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨਾ ਆਸਾਨ ਬਣਾਉਂਦਾ ਹੈ।

ਟੂਲ ਵਰਤੋਂ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਤਕਨੀਕ ਹੈ ਜੋ ਡਵਿੈਲਪਰਾਂ ਨੂੰ AI-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ LLMs ਦੀ ਪੂਰੀ ਸਮਰੱਥਾ ਦਾ ਲਾਭ ਲੈਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦੀ ਹੈ। LLMs ਨੂੰ ਬਾਹਰੀ ਟੂਲਜ਼ ਅਤੇ ਸਰੋਤਾਂ ਨਾਲ ਜੋੜ ਕੇ, ਤੁਸੀਂ ਵਧੇਰੇ ਗਤੀਸ਼ੀਲ, ਬੁੱਧੀਮਾਨ, ਅਤੇ ਸੰਦਰਭ-ਜਾਗਰੂਕ ਸਸਿਟਮ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਉਪਭੋਗਤਾ ਦੀਆਂ ਲੋੜਾਂ ਦੇ ਅਨੁਕੂਲ ਹੋ ਸਕਦੇ ਹਨ ਅਤੇ ਮੁੱਲਵਾਨ ਅੰਤਰਦ੍ਰਸ਼ਿਟੀ ਅਤੇ ਕਾਰਵਾਈਆਂ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ।

ਭਾਵੇਂ ਟੂਲ ਵਰਤੋਂ ਵਸ਼ਿਾਲ ਸੰਭਾਵਨਾਵਾਂ ਪੇਸ਼ ਕਰਦੀ ਹੈ, ਪਰ ਸੰਭਾਵੀ ਚੁਣੌਤੀਆਂ ਅਤੇ ਵਚਿਾਰਾਂ ਤੋਂ ਜਾਣੂ ਹੋਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਇੱਕ ਮੁੱਖ ਪਹਲਿੂ ਟੂਲ ਇੰਟਰੈਕਸ਼ਨਾਂ ਦੀ ਗੁੰਝਲਤਾ ਦਾ ਪ੍ਰਬੰਧਨ ਕਰਨਾ ਅਤੇ ਸਮੁੱਚੇ ਸਸਿਟਮ ਦੀ ਸਥਰਿਤਾ ਅਤੇ ਵਸ਼ਿਵਾਸਯੋਗਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣਾ ਹੈ। ਤੁਹਾਨੂੰ ਉਨ੍ਹਾਂ ਸਥਤਿੀਆਂ ਨੂੰ ਸੰਭਾਲਣਾ ਪਵੇਗਾ ਜੱਥੇ ਟੂਲ ਕਾਲਾਂ ਅਸਫਲ ਹੋ ਸਕਦੀਆਂ ਹਨ, ਅਣਚਤਿਵੇ ਨਤੀਜੇ ਵਾਪਸ ਕਰ ਸਕਦੀਆਂ ਹਨ, ਜਾਂ ਪ੍ਰਦਰਸ਼ਨ 'ਤੇ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦੀਆਂ ਹਨ। ਇਸ ਤੋਂ ਇਲਾਵਾ, ਤੁਹਾਨੂੰ ਟੂਲਜ਼ ਦੀ ਅਣਅਧਕਿਾਰਤ ਜਾਂ ਦੁਰਭਾਵਨਾਪੂਰਨ ਵਰਤੋਂ ਨੂੰ ਰੋਕਣ ਲਈ ਸੁਰੱਖਿਆ ਅਤੇ ਪਹੁੰਚ ਨਯਿੰਤਰਣ ਉਪਾਵਾਂ 'ਤੇ ਵਚਿਾਰ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ। ਤੁਹਾਡੀ AI-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਅਖੰਡਤਾ ਅਤੇ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਬਣਾਈ ਰੱਖਣ ਲਈ ਉਚਤਿ ਗਲਤੀ ਸੰਭਾਲ, ਲੌਗਿੰਗ, ਅਤੇ ਨਗਿਰਾਨੀ ਵਧਿੀਆਂ ਮਹੱਤਵਪੂਰਨ ਹਨ।

ਜਵਿੇਂ ਤੁਸੀਂ ਆਪਣੇ ਪ੍ਰੋਜੈਕਟਾਂ ਵਿੱਚ ਸੰਦ ਦੀ ਵਰਤੋਂ ਦੀਆਂ ਸੰਭਾਵਨਾਵਾਂ ਦੀ ਖੋਜ ਕਰਦੇ ਹੋ, ਯਾਦ ਰੱਖੋ ਕਿ ਸਪੱਸ਼ਟ ਉਦੇਸ਼ਾਂ ਨਾਲ ਸ਼ੁਰੂ ਕਰੋ, ਚੰਗੀ ਤਰ੍ਹਾਂ ਢਾਂਚਾਗਤ ਸੰਦ ਪਰਭਾਸ਼ਾਵਾਂ ਤਿਆਰ ਕਰੋ, ਅਤੇ ਫੀਡਬੈਕ ਅਤੇ ਨਤੀਜਿਆਂ ਦੇ ਆਧਾਰ 'ਤੇ ਸੁਧਾਰ ਕਰਦੇ ਰਹੋ। ਸਹੀ ਪਹੁੰਚ ਅਤੇ ਮਾਨਸਕਿਤਾ ਨਾਲ, ਸੰਦ ਦੀ ਵਰਤੋਂ ਤੁਹਾਡੀਆਂ ਏ.ਆਈ.-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਨਵੀਨਤਾ ਅਤੇ ਮੁੱਲ ਦੇ ਨਵੇਂ ਪੱਧਰਾਂ ਨੂੰ ਖੋਲ੍ਹ ਸਕਦੀ ਹੈ

ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ

HTTP ਉੱਤੇ ਸਟ੍ਰੀਮਿੰਗ ਡਾਟਾ, ਜਸਿਨੂੰ ਸਰਵਰ-ਭੇਜੀਆਂ ਘਟਨਾਵਾਂ (SSE) ਵੀ ਕਹਿਾ ਜਾਂਦਾ ਹੈ, ਇੱਕ ਅਜਹਿਾ ਤਰੀਕਾ ਹੈ ਜਿੱਥੇ ਸਰਵਰ ਲਗਾਤਾਰ ਕਲਾਇੰਟ ਨੂੰ ਡਾਟਾ ਭੇਜਦਾ ਹੈ ਜਵਿੇਂ ਹੀ ਇਹ ਉਪਲਬਧ ਹੁੰਦਾ ਹੈ, ਬਨਿਾਂ ਕਲਾਇੰਟ ਦੁਆਰਾ ਸਪੱਸ਼ਟ ਤੌਰ 'ਤੇ ਬੇਨਤੀ ਕੀਤੇ। ਜਵਿੇਂ-ਜਵਿੇਂ AI ਦਾ ਜਵਾਬ ਕ੍ਰਮਵਾਰ ਤਆਿਰ ਹੁੰਦਾ ਹੈ, AI ਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਤਆਿਰ ਹੁੰਦੇ ਹੀ ਦਖਿਾ ਕੇ ਇੱਕ ਪ੍ਰਤੀਕਰਿਆਸ਼ੀਲ ਯੂਜ਼ਰ ਅਨੁਭਵ ਪ੍ਰਦਾਨ ਕਰਨਾ ਸਮਝਦਾਰੀ ਵਾਲੀ ਗੱਲ ਹੈ। ਅਤੇ ਅਸਲ ਵਚਿ ਮੈਨੂੰ ਜਾਣਕਾਰੀ ਵਾਲੇ ਸਾਰੇ AI ਪ੍ਰਦਾਤਾ APIs ਆਪਣੇ ਪੂਰਤੀ ਐਂਡਪੁਆਇੰਟਸ ਵਚਿ ਇੱਕ ਵਕਿਲਪ ਦੇ ਤੌਰ 'ਤੇ ਸਟ੍ਰੀਮਿੰਗ ਜਵਾਬ ਪੇਸ਼ ਕਰਦੇ ਹਨ।

ਇਸ ਕਤਿਾਬ ਵਚਿ ਇਹ ਅਧਆਿਇ ਟੂਲਜ਼ ਦੀ ਵਰਤੋਂ ਤੋਂ ਤੁਰੰਤ ਬਾਅਦ ਇੱਥੇ ਆਉਣ ਦਾ ਕਾਰਨ ਇਹ ਹੈ ਕ ਿਯੂਜ਼ਰਾਂ ਨੂੰ ਲਾਈਵ AI ਜਵਾਬਾਂ ਨਾਲ ਟੂਲਜ਼ ਦੀ ਵਰਤੋਂ ਨੂੰ ਜੋੜਨਾ ਕਿੰਨਾ ਸ਼ਕਤੀਸ਼ਾਲੀ ਹੋ ਸਕਦਾ ਹੈ। ਅਜਹਿਾ ਕਰਨ ਨਾਲ ਗਤੀਸ਼ੀਲ ਅਤੇ ਇੰਟਰਐਕਟਵਿ ਅਨੁਭਵਾਂ ਦੀ ਆਗਆਿ ਮਲਿਦੀ ਹੈ ਜਿੱਥੇ AI ਯੂਜ਼ਰ ਇਨਪੁੱਟ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰ ਸਕਦਾ ਹੈ, ਆਪਣੀ ਮਰਜ਼ੀ ਨਾਲ ਵੱਖ-ਵੱਖ ਟੂਲਜ਼ ਅਤੇ ਫੰਕਸ਼ਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦਾ ਹੈ, ਅਤੇ ਫਰਿ ਰੀਅਲ-ਟਾਈਮ ਜਵਾਬ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦਾ ਹੈ।

ਇਸ ਨਰਿਵਘਿਨ ਇੰਟਰੈਕਸ਼ਨ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ, ਤੁਹਾਨੂੰ ਸਟ੍ਰੀਮ ਹੈਂਡਲਰ ਲਖਿਣੇ ਪੈਣਗੇ ਜੋ AI-ਚਲਾਏ

ਟੂਲ ਫੰਕਸ਼ਨ ਕਾਲਾਂ ਦੇ ਨਾਲ-ਨਾਲ ਸਧਾਰਨ ਟੈਕਸਟ ਆਉਟਪੁੱਟ ਨੂੰ ਅੰਤਮਿ ਯੂਜ਼ਰ ਨੂੰ ਭੇਜ ਸਕਣ। ਟੂਲ ਫੰਕਸ਼ਨ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਨ ਤੋਂ ਬਾਅਦ ਲੂਪ ਕਰਨ ਦੀ ਜ਼ਰੂਰਤ ਕੰਮ ਵੱਚਿ ਇੱਕ ਦਲਿਚਸਪ ਚੁਣੌਤੀ ਜੋੜਦੀ ਹੈ।

ReplyStream ਨੂੰ ਲਾਗੂ ਕਰਨਾ

ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ ਨੂੰ ਕਵਿੇਂ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਇਹ ਦਖਿਾਉਣ ਲਈ, ਇਹ ਅਧਆਿਇ Olympia ਵੱਚਿ ਵਰਤੀ ਜਾਂਦੀ `ReplyStream` ਕਲਾਸ ਦੇ ਇੱਕ ਸਰਲ ਸੰਸਕਰਣ ਵੱਚਿ ਡੂੰਘਾ ਉਤਰੇਗਾ। ਇਸ ਕਲਾਸ ਦੇ ਇੰਸਟੈਂਸ ruby-openai ਅਤੇ openrouter ਵਰਗੀਆਂ AI ਕਲਾਇੰਟ ਲਾਇਬ੍ਰੇਰੀਆਂ ਵੱਚਿ `stream` ਪੈਰਾਮੀਟਰ ਵਜੋਂ ਪਾਸ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ।

ਇੱਥੇ ਦੱਸਆਿ ਗਆਿ ਹੈ ਕ ਿਮੈਂ Olympia ਦੇ `PromptSubscriber` ਵੱਚਿ `ReplyStream` ਦੀ ਵਰਤੋਂ ਕਵਿੇਂ ਕਰਦਾ ਹਾਂ, ਜੋ Wisper ਰਾਹੀਂ ਨਵੇਂ ਯੂਜ਼ਰ ਸੁਨੇਹਆਿਂ ਦੀ ਸਰਜਿਣਾ ਲਈ ਸੁਣਦਾ ਹੈ।

```ruby
class PromptSubscriber
  include Raix::ChatCompletion
  include Raix::PromptDeclarations

  # many other declarations omitted...

  prompt text: -> { user_message.content },
         stream: -> { ReplyStream.new(self) },
         until: -> { bot_message.complete? }

  def message_created(message) # invoked by Wisper
    return unless message.role.user? && message.content?

    # rest of the implementation omitted...
```

`context` ਰੈਫਰੈਂਸ ਤੋਂ ਇਲਾਵਾ ਜੋ ਪ੍ਰੌਂਪਟ ਸਬਸਕ੍ਰਾਈਬਰ ਦੁਆਰਾ ਬਣਾਇਆ ਗਆਿ ਹੈ, `ReplyStream` ਕਲਾਸ ਵੱਚਿ ਪ੍ਰਾਪਤ ਡਾਟਾ ਦਾ ਬਫਰ ਸਟੋਰ ਕਰਨ ਲਈ ਇੰਸਟੈਂਸ ਵੇਰੀਏਬਲ, ਅਤੇ ਸਟਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ ਦੌਰਾਨ ਫੰਕਸ਼ਨ ਨਾਂਵਾਂ ਅਤੇ ਆਰਗੂਮੈਂਟਸ ਦਾ ਟਰੈਕ ਰੱਖਣ ਲਈ ਐਰੇ ਵੀ ਹਨ।

```
class ReplyStream
  attr_accessor :buffer, :f_name, :f_arguments, :context

  delegate :bot_message, :dispatch, to: :context

  def initialize(context)
    self.context = context
    self.buffer = []
    self.f_name = []
    self.f_arguments = []
  end

  def call(chunk, bytesize = nil)
    # ...
  end

  # ...
end
```

`initialize` ਵਿਧੀ `ReplyStream` ਇੰਸਟੈਂਸ ਦੀ ਸ਼ੁਰੂਆਤੀ ਸਥਿਤੀ ਨੂੰ ਸਥਾਪਤਿ ਕਰਦੀ ਹੈ, ਬਫਰ, ਸੰਦਰਭ, ਅਤੇ ਹੋਰ ਵੇਰੀਏਬਲਾਂ ਨੂੰ ਸ਼ੁਰੂ ਕਰਦੀ ਹੈ।

`call` ਵਿਧੀ ਸਟ੍ਰੀਮਿੰਗ ਡਾਟਾ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਨ ਲਈ ਮੁੱਖ ਐਂਟਰੀ ਪੁਆਇੰਟ ਹੈ। ਇਹ ਡਾਟਾ ਦੇ `chunk` (ਜੋ ਇੱਕ ਹੈਸ਼ ਦੇ ਰੂਪ ਵਿੱਚ ਦਰਸਾਇਆ ਗਿਆ ਹੈ) ਅਤੇ ਇੱਕ ਵਿਕਲਪਿਕ `bytesize` ਪੈਰਾਮੀਟਰ ਲੈਂਦੀ ਹੈ, ਜੋ ਸਾਡੀ ਉਦਾਹਰਣ ਵਿੱਚ ਵਰਤਿਆ ਨਹੀਂ ਗਿਆ। ਇਸ ਵਿਧੀ ਦੇ ਅੰਦਰ, ਕਲਾਸ ਪ੍ਰਾਪਤ ਕੀਤੇ ਚੰਕ ਦੀ ਬਣਤਰ ਦੇ ਆਧਾਰ 'ਤੇ ਵੱਖ-ਵੱਖ ਸਥਿਤੀਆਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਪੈਟਰਨ ਮੈਚਿੰਗ ਦੀ ਵਰਤੋਂ ਕਰਦੀ ਹੈ।

ਚੰਕ 'ਤੇ `deep_symbolize_keys` ਨੂੰ ਕਾਲ ਕਰਨਾ ਪੈਟਰਨ ਮੈਚਿੰਗ ਨੂੰ ਵਧੇਰੇ ਸੁੰਦਰ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ, ਕਿਉਂਕਿ ਇਹ ਸਾਨੂੰ ਸਟ੍ਰਿੰਗਾਂ ਦੀ ਬਜਾਏ ਸਿੰਬਲਾਂ 'ਤੇ ਕੰਮ ਕਰਨ ਦਿੰਦਾ ਹੈ।

```ruby
def call(chunk, _bytesize)
  case chunk.deep_symbolize_keys

  in { # match function name
    choices: [
      {
        delta: {
          tool_calls: [
            { index: index, function: {name: name} }
          ]
        }
      }
    ] }

    f_name[index] = name
```

ਪਹਲਿਾ ਪੈਟਰਨ ਜਸਿਨ੍ਹੂੰ ਅਸੀਂ ਮੈਚ ਕਰ ਰਹੇ ਹਾਂ ਉਹ ਹੈ ਇੱਕ ਟੂਲ ਕਾਲ ਅਤੇ ਇਸਦੇ ਨਾਲ ਜੁੜਆਿ ਫੰਕਸ਼ਨ ਨਾਮ। ਜੇ ਅਸੀਂ ਇਸਨ੍ਹੂੰ ਖੋਜਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਇਸਨ੍ਹੂੰ `f_name` ਐਰੇ ਵਚਿ ਸਟੋਰ ਕਰ ਦਦਿੇ ਹਾਂ। ਅਸੀਂ ਫੰਕਸ਼ਨ ਨਾਮਾਂ ਨ੍ਹੂੰ ਇੱਕ ਇੰਡੈਕਸਡ ਐਰੇ ਵਚਿ ਸਟੋਰ ਕਰਦੇ ਹਾਂ, ਕਉਿਂਕ ਿਮਾਡਲ ਸਮਾਨਾਂਤਰ ਫੰਕਸ਼ਨ ਕਾਲਾਂਿਗ ਕਰਨ ਦੇ ਯੋਗ ਹੈ, ਜੋ ਇੱਕੋ ਸਮੇਂ 'ਤੇ ਕਈ ਫੰਕਸ਼ਨਾਂ ਨ੍ਹੂੰ ਚਲਾਉਣ ਲਈ ਭੇਜ ਸਕਦਾ ਹੈ।

> ਸਮਾਨਾਂਤਰ ਫੰਕਸ਼ਨ ਕਾਲਾਂਿਗ ਇੱਕ AI ਮਾਡਲ ਦੀ ਉਹ ਯੋਗਤਾ ਹੈ ਜੋ ਕਈ ਫੰਕਸ਼ਨ ਕਾਲਾਂ ਨ੍ਹੂੰ ਇਕੱਠੇ ਕਰ ਸਕਦੀ ਹੈ, ਜਸਿ ਨਾਲ ਇਹਨਾਂ ਫੰਕਸ਼ਨ ਕਾਲਾਂ ਦੇ ਪ੍ਰਭਾਵ ਅਤੇ ਨਤੀਜੇ ਸਮਾਨਾਂਤਰ ਰੂਪ ਵਚਿ ਹੱਲ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਇਹ ਖਾਸ ਤੌਰ 'ਤੇ ਉਦੋਂ ਲਾਭਦਾਇਕ ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਫੰਕਸ਼ਨਾਂ ਨ੍ਹੂੰ ਲੰਬਾ ਸਮਾਂ ਲੱਗਦਾ ਹੈ, ਅਤੇ ਇਹ API ਨਾਲ ਰਾਊਂਡ ਟ੍ਰਪਿਸ ਨ੍ਹੂੰ ਘਟਾਉਂਦਾ ਹੈ, ਜੋ ਬਦਲੇ ਵਚਿ ਟੋਕਨ ਖਰਚ ਦੀ ਮਹੱਤਵਪੂਰਨ ਮਾਤਰਾ ਨ੍ਹੂੰ ਬਚਾ ਸਕਦਾ ਹੈ।

ਅਗਲਾ ਸਾਨ੍ਹੂੰ ਫੰਕਸ਼ਨ ਕਾਲਾਂ ਦੇ ਅਨੁਸਾਰੀ ਆਰਗੂਮੈਂਟਸ ਲਈ ਮੈਚ ਕਰਨਾ ਪਵੇਗਾ।

```
in { # match arguments
  choices: [
    {
      delta: {
        tool_calls: [
          {
            index: index, function: {arguments: argument }
          }
        ]
      }
    }
  ]}

  f_arguments[index] ||= "" # initialize if not already
  f_arguments[index] << argument
```

ਜਦੋਂ ਅਸੀਂ ਫੰਕਸ਼ਨ ਨਾਵਾਂ ਨੂੰ ਸੰਭਾਲਿਆ ਸੀ, ਅਸੀਂ ਆਰਗੂਮੈਂਟਸ ਨੂੰ ਇੱਕ ਇੰਡੈਕਸਡ ਐਰੇ ਵਿੱਚ ਸਟੋਰ ਕਰਦੇ ਹਾਂ।

ਅੱਗੇ, ਅਸੀਂ ਆਮ ਯੂਜ਼ਰ-ਫੇਸਿੰਗ ਸੁਨੇਹਿਆਂ ਦੀ ਭਾਲ ਕਰਦੇ ਹਾਂ, ਜੋ ਸਰਵਰ ਤੋਂ ਇੱਕ ਟੋਕਨ ਦੇ ਹਿਸਾਬ ਨਾਲ ਆਉਣਗੇ ਅਤੇ `new_content` ਵੇਰੀਏਬਲ ਨੂੰ ਅਸਾਈਨ ਕੀਤੇ ਜਾਣਗੇ। ਸਾਨੂੰ `finish_reason` 'ਤੇ ਵੀ ਨਜ਼ਰ ਰੱਖਣੀ ਪਵੇਗੀ। ਇਹ ਆਉਟਪੁੱਟ ਸੀਕੁਐਂਸ ਦੇ ਆਖਰੀ ਹਿੱਸੇ ਤੱਕ `nil` ਰਹੇਗਾ।

```
in {
  choices: [
    { delta: {content: new_content}, finish_reason: finish_reason }
  ]}

  # you could transmit every chunk to the user here...
  buffer << new_content.to_s

  if finish_reason.present?
    finalize
  elsif new_content.to_s.match?(/\n\n/)
    send_to_client # ...or buffer and transmit once per paragraph
  end
```

ਮਹੱਤਵਪੂਰਨ ਤੌਰ 'ਤੇ, ਅਸੀਂ ਏ.ਆਈ. ਮਾਡਲ ਪ੍ਰਦਾਤਾ ਦੁਆਰਾ ਭੇਜੇ ਗਏ ਗਲਤੀ ਸੁਨੇਹਿਆਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਇੱਕ ਪੈਟਰਨ ਮੈਚ ਐਕਸਪ੍ਰੈਸ਼ਨ ਜੋੜਦੇ ਹਾਂ। ਸਥਾਨਕ ਵਿਕਾਸ ਵਾਤਾਵਰਣਾਂ ਵਿੱਚ, ਅਸੀਂ ਇੱਕ ਅਪਵਾਦ ਉਠਾਉਂਦੇ ਹਾਂ, ਪਰ ਪ੍ਰੋਡਕਸ਼ਨ ਵਿੱਚ, ਅਸੀਂ ਗਲਤੀ ਨੂੰ ਲੌਗ ਕਰਦੇ ਹਾਂ ਅਤੇ ਅੰਤਮ ਕਰਦੇ ਹਾਂ।

```
  in { error: { message: } }
    if Rails.env.local?
      raise message
    else
      Honeybadger.notify("AI Error: #{message}")
      finalize
    end
```

ਕੇਸ ਦਾ ਆਖਰੀ ਹੋਰ ਕਲਾਜ਼ ਉਦੋਂ ਚੱਲੇਗਾ ਜਦੋਂ ਪਛਿਲੇ ਕੋਈ ਵੀ ਪੈਟਰਨ ਮੇਲ ਨਹੀਂ ਖਾਂਦੇ। ਇਹ ਸਿਰਫ਼ ਇੱਕ ਸੁਰੱਖਿਆ ਉਪਾਅ ਹੈ ਤਾਂ ਜੋ ਜੇਕਰ ਏ.ਆਈ. ਮਾਡਲ ਸਾਨੂੰ ਅਣਪਛਾਤੇ ਟੁਕੜੇ ਭੇਜਣਾ ਸ਼ੁਰੂ ਕਰ ਦੇਵੇ ਤਾਂ ਸਾਨੂੰ ਇਸ ਬਾਰੇ ਪਤਾ ਲੱਗ ਜਾਵੇ।

```
  else
    Honeybadger.notify("Unrecognized Chunk: #{chunk}")
  end
end
```

`send_to_client` ਮੈਥਡ ਬਫ਼ਰਡ ਸਮੱਗਰੀ ਨੂੰ ਕਲਾਇੰਟ ਨੂੰ ਭੇਜਣ ਲਈ ਜ਼ਿੰਮੇਵਾਰ ਹੈ। ਇਹ ਜਾਂਚ ਕਰਦਾ ਹੈ ਕਿ ਬਫ਼ਰ ਖਾਲੀ ਨਹੀਂ ਹੈ, ਬੋਟ ਸੁਨੇਹੇ ਦੀ ਸਮੱਗਰੀ ਨੂੰ ਅੱਪਡੇਟ ਕਰਦਾ ਹੈ, ਬੋਟ ਸੁਨੇਹੇ ਨੂੰ ਰੈਂਡਰ ਕਰਦਾ ਹੈ, ਅਤੇ ਡਾਟਾ ਸਥਿਰਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਸਮੱਗਰੀ ਨੂੰ ਡਾਟਾਬੇਸ ਵਿੱਚ ਸੰਭਾਲਦਾ ਹੈ।

```
def send_to_client
  # no need to process pure whitespace
  return if buffer.join.squish.blank?

  # set the buffer content on the bot message
  content = buffer.join
  bot_message.content = content

  # save to database so that we never lose data
  # even if the stream doesn't terminate correctly
  bot_message.update_column(:content, content)

  # update content via websocket
  ConversationRenderer.update(bot_message)
end
```

`finalize` ਮੈਥਡ ਨੂੰ ਉਦੋਂ ਬੁਲਾਇਆ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ ਪੂਰੀ ਹੋ ਜਾਂਦੀ ਹੈ। ਇਹ ਫੰਕਸ਼ਨ ਕਾਲਜ਼ ਨੂੰ ਡਿਸਪੈਚ ਕਰਦਾ ਹੈ ਜੇਕਰ ਸਟ੍ਰੀਮ ਦੌਰਾਨ ਕੋਈ ਪ੍ਰਾਪਤ ਹੋਏ ਸਨ, ਬੋਟ ਸੁਨੇਹੇ ਨੂੰ ਅੰਤਮਿ ਸਮੱਗਰੀ ਅਤੇ ਹੋਰ ਸੰਬੰਧਤਿ ਜਾਣਕਾਰੀ ਨਾਲ ਅੱਪਡੇਟ ਕਰਦਾ ਹੈ, ਅਤੇ ਫੰਕਸ਼ਨ ਕਾਲ ਇਤਹਿਾਸ ਨੂੰ ਰੀਸੈੱਟ ਕਰਦਾ ਹੈ।

```
def finalize
  if f_name.any?
    f_name.each_with_index do |name, index|
      # takes care of calling the function wherever it's implemented
      dispatch(name:, arguments: JSON.parse(f_arguments[index]))
    end

    # reset the function call history
    f_name.clear
    f_arguments.clear
  else
    content = buffer.join.presence
    bot_message.update!(content:, complete: true)
    ConversationRenderer.update(bot_message)
  end
end
```

ਜੇਕਰ ਮਾਡਲ ਕੋਈ ਫੰਕਸ਼ਨ ਕਾਲ ਕਰਨ ਦਾ ਫੈਸਲਾ ਕਰਦਾ ਹੈ, ਤਾਂ ਤੁਹਾਨੂੰ ਉਸ ਫੰਕਸ਼ਨ ਕਾਲ (ਨਾਮ ਅਤੇ ਆਰਗੁਮੈਂਟਸ) ਨੂੰ ਇਸ ਤਰ੍ਹਾਂ "ਡਿਸਪੈਚ" ਕਰਨ ਦੀ ਲੋੜ ਹੈ ਕਿ ਇਹ ਚੱਲ ਜਾਵੇ ਅਤੇ `function_call` ਅਤੇ `function_result` ਸੁਨੇਹੇ ਗੱਲਬਾਤ ਦੀ ਲਖਿਤ ਵੱਚਿ ਜੋੜੇ ਜਾਣ।

ਮੇਰੇ ਤਜਰਬੇ ਵੱਚਿ, ਟੂਲ ਇੰਪਲੀਮੈਂਟੇਸ਼ਨਾਂ 'ਤੇ ਭਰੋਸਾ ਕਰਨ ਦੀ ਬਜਾਏ ਆਪਣੇ ਕੋਡਬੇਸ ਵੱਚਿ ਇੱਕ ਥਾਂ 'ਤੇ ਫੰਕਸ਼ਨ ਸੁਨੇਹਆਿਂ ਦੀ ਰਚਨਾ ਨੂੰ ਸੰਭਾਲਣਾ ਬਹਿਤਰ ਹੈ। ਇਹ ਸਾਫ਼ ਸੁਥਰਾ ਹੈ, ਪਰ ਇੱਕ ਬਹੁਤ ਮਹੱਤਵਪੂਰਨ ਵਹਿਾਰਕ ਕਾਰਨ ਵੀ ਹੈ: ਜੇਕਰ ਏਆਈ ਮਾਡਲ ਕੋਈ ਫੰਕਸ਼ਨ ਕਾਲ ਕਰਦਾ ਹੈ, ਅਤੇ ਜਦੋਂ ਤੁਸੀਂ ਲੂਪ ਕਰਦੇ ਹੋ ਤਾਂ ਲਖਿਤ ਵੱਚਿ ਨਤੀਜੇ ਵਾਲੇ ਕਾਲ ਅਤੇ ਨਤੀਜੇ ਦੇ ਸੁਨੇਹੇ ਨਹੀਂ ਦੇਖਦਾ, ਇਹ ਉਸੇ ਫੰਕਸ਼ਨ ਨੂੰ ਦੁਬਾਰਾ ਕਾਲ ਕਰੇਗਾ। ਸੰਭਾਵੀ ਤੌਰ 'ਤੇ ਹਮੇਸ਼ਾ ਲਈ। ਯਾਦ ਰੱਖੋ ਕਿ ਏਆਈ ਪੂਰੀ ਤਰ੍ਹਾਂ ਸਟੇਟਲੈੱਸ ਹੈ, ਇਸ ਲਈ ਜਦੋਂ ਤੱਕ ਤੁਸੀਂ ਉਹਨਾਂ ਫੰਕਸ਼ਨ ਕਾਲਾਂ ਨੂੰ ਵਾਪਸ ਇਸ ਨੂੰ ਨਹੀਂ ਦਖਿਾਉਂਦੇ, ਉਹ ਹੋਈਆਂ ਹੀ ਨਹੀਂ।

```
# PromptSubscriber#dispatch

def dispatch(name:, arguments:)
  # adds a function_call message to the conversation transcript
  # plus dispatches to tool and returns result
  conversation.function_call!(name, arguments).then do |result|
    # add function result message to the transcript
    conversation.function_result!(name, result)
  end
end
```

ਫੰਕਸ਼ਨ ਕਾਲ ਹਿਸਟਰੀ ਨੂੰ ਡਿਸਪੈਚ ਕਰਨ ਤੋਂ ਬਾਅਦ ਸਾਫ਼ ਕਰਨਾ ਓਨਾ ਹੀ ਮਹੱਤਵਪੂਰਨ ਹੈ ਜਿੰਨਾ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਕਿ ਕਾਲ ਅਤੇ ਨਤੀਜੇ ਤੁਹਾਡੇ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਵਿੱਚ ਸ਼ਾਮਲ ਹੋਣ, ਤਾਂ ਜੋ ਤੁਸੀਂ ਹਰ ਵਾਰ ਲੂਪ ਕਰਨ 'ਤੇ ਉਹੀ ਫੰਕਸ਼ਨ ਵਾਰ-ਵਾਰ ਨਾ ਕਾਲ ਕਰਦੇ ਰਹੋ।

"ਗੱਲਬਾਤ ਲੂਪ"

ਮੈਂ ਲੂਪਿੰਗ ਦਾ ਜ਼ਿਕਰ ਕਰਦਾ ਰਹਿੰਦਾ ਹਾਂ, ਪਰ ਜੇ ਤੁਸੀਂ ਫੰਕਸ਼ਨ ਕਾਲਿੰਗ ਵਿੱਚ ਨਵੇਂ ਹੋ, ਤਾਂ ਇਹ ਸਪੱਸ਼ਟ ਨਹੀਂ ਹੋ ਸਕਦਾ ਕਿ ਸਾਨੂੰ ਲੂਪ ਦੀ ਲੋੜ ਕਿਉਂ ਹੈ। ਕਾਰਨ ਇਹ ਹੈ ਕਿ ਜਦੋਂ AI ਤੁਹਾਨੂੰ ਆਪਣੀ ਤਰਫ਼ੋਂ ਟੂਲ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ "ਪੁੱਛਦੀ" ਹੈ, ਤਾਂ ਇਹ ਜਵਾਬ ਦੇਣਾ ਬੰਦ ਕਰ ਦੇਵੇਗੀ। ਇਹ ਤੁਹਾਡੀ ਜ਼ਿੰਮੇਵਾਰੀ ਹੈ ਕਿ ਤੁਸੀਂ ਉਹਨਾਂ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਚਲਾਓ, ਨਤੀਜੇ ਇਕੱਠੇ ਕਰੋ, ਨਤੀਜਿਆਂ ਨੂੰ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਵਿੱਚ ਜੋੜੋ, ਅਤੇ ਫਿਰ ਨਵੇਂ ਫੰਕਸ਼ਨ ਕਾਲਾਂ ਜਾਂ ਯੂਜ਼ਰ-ਫੇਸਿੰਗ ਨਤੀਜੇ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਮੂਲ ਪ੍ਰੌਮਪਟ ਨੂੰ ਦੁਬਾਰਾ ਸਬਮਿਟ ਕਰੋ।

`PromptSubscriber` ਕਲਾਸ ਵਿੱਚ, ਅਸੀਂ `PromptDeclarations` ਮੌਡਿਊਲ ਤੋਂ `prompt` ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਗੱਲਬਾਤ ਲੂਪ ਦੇ ਵਵਿਹਾਰ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ ਕਰਦੇ ਹਾਂ। `until` ਪੈਰਾਮੀਟਰ ਨੂੰ `-> { bot_message.complete? }` 'ਤੇ ਸੈੱਟ ਕੀਤਾ ਗਿਆ ਹੈ, ਜਿਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਲੂਪ ਉਦੋਂ ਤੱਕ ਜਾਰੀ ਰਹੇਗਾ ਜਦੋਂ ਤੱਕ `bot_message` ਨੂੰ ਪੂਰਾ ਹੋਇਆ ਮਾਰਕ ਨਹੀਂ ਕੀਤਾ ਜਾਂਦਾ।

```
prompt text: -> { user_message.content },
       stream: -> { ReplyStream.new(self) },
       until: -> { bot_message.complete? }
```

ਪਰ `bot_message` ਨੂੰ ਪੂਰਾ ਕਦੋਂ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ? ਜੇ ਤੁਸੀਂ ਭੁੱਲ ਗਏ ਹੋ, ਤਾਂ `finalize` ਮੈਥਡ ਦੀ ਲਾਈਨ 13 ਵੱਲ ਮੁੜ ਕੇ ਦੇਖੋ।

ਆਓ ਪੂਰੀ ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ ਲੌਜਿਕ ਦੀ ਸਮੀਖਿਆ ਕਰੀਏ।

1. `PromptSubscriber` ਨੂੰ `message_created` ਮੈਥਡ ਰਾਹੀਂ ਇੱਕ ਨਵਾਂ ਯੂਜ਼ਰ ਮੈਸੇਜ ਪ੍ਰਾਪਤ ਹੁੰਦਾ ਹੈ, ਜੋ ਕਿ Wisper pub/sub ਸਿਸਟਮ ਦੁਆਰਾ ਹਰ ਵਾਰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਅੰਤਿਮ ਯੂਜ਼ਰ ਇੱਕ ਨਵਾਂ ਪ੍ਰੌਮਪਟ ਬਣਾਉਂਦਾ ਹੈ।
2. `prompt` ਕਲਾਸ ਮੈਥਡ `PromptSubscriber` ਲਈ ਚੈਟ ਕੰਪਲੀਸ਼ਨ ਲੌਜਿਕ ਦੇ ਵਵਿਹਾਰ ਨੂੰ ਘੋਸ਼ਿਤ ਰੂਪ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ। ਏ.ਆਈ. ਮਾਡਲ ਯੂਜ਼ਰ ਦੇ ਮੈਸੇਜ ਕੰਟੈਂਟ ਨਾਲ ਇੱਕ ਚੈਟ ਕੰਪਲੀਸ਼ਨ ਕਰੇਗਾ, `ReplyStream` ਦੀ ਇੱਕ ਨਵੀਂ ਇੰਸਟੈਂਸ ਨੂੰ ਸਟ੍ਰੀਮ ਪੈਰਾਮੀਟਰ ਵਜੋਂ, ਅਤੇ ਨਿਰਧਾਰਿਤ ਲੂਪ ਸ਼ਰਤ ਨਾਲ।
3. ਏ.ਆਈ. ਮਾਡਲ ਪ੍ਰੌਮਪਟ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਦਾ ਹੈ ਅਤੇ ਜਵਾਬ ਤਿਆਰ ਕਰਨਾ ਸ਼ੁਰੂ ਕਰਦਾ ਹੈ। ਜਿਵੇਂ-ਜਿਵੇਂ ਜਵਾਬ ਸਟ੍ਰੀਮ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, `ReplyStream` ਇੰਸਟੈਂਸ ਦਾ `call` ਮੈਥਡ ਹਰੇਕ ਡੇਟਾ ਚੰਕ ਲਈ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ।
4. ਜੇ ਏ.ਆਈ. ਮਾਡਲ ਕਿਸੇ ਟੂਲ ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਕਰਨ ਦਾ ਫੈਸਲਾ ਕਰਦਾ ਹੈ, ਤਾਂ ਫੰਕਸ਼ਨ ਨਾਮ ਅਤੇ ਆਰਗੂਮੈਂਟਸ ਨੂੰ ਚੰਕ ਵਿੱਚੋਂ ਕੱਢ ਕੇ ਕ੍ਰਮਵਾਰ `f_name` ਅਤੇ `f_arguments` ਐਰੇ ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
5. ਜੇ ਏ.ਆਈ. ਮਾਡਲ ਯੂਜ਼ਰ-ਫੇਸਿੰਗ ਕੰਟੈਂਟ ਤਿਆਰ ਕਰਦਾ ਹੈ, ਤਾਂ ਇਸਨੂੰ ਬਫ਼ਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ `send_to_client` ਮੈਥਡ ਰਾਹੀਂ ਕਲਾਇੰਟ ਨੂੰ ਭੇਜਿਆ ਜਾਂਦਾ ਹੈ।
6. ਜਦੋਂ ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ ਪੂਰੀ ਹੋ ਜਾਂਦੀ ਹੈ, `finalize` ਮੈਥਡ ਨੂੰ ਕਾਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਜੇ ਸਟ੍ਰੀਮ ਦੌਰਾਨ ਕੋਈ ਟੂਲ ਫੰਕਸ਼ਨ ਕਾਲ ਕੀਤੇ ਗਏ ਸਨ, ਤਾਂ ਉਹਨਾਂ ਨੂੰ `PromptSubscriber` ਦੇ `dispatch` ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਡਿਸਪੈਚ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
7. `dispatch` ਮੈਥਡ ਗੱਲਬਾਤ ਦੀ ਲਿਖਿਤ ਵਿੱਚ ਇੱਕ `function_call` ਮੈਸੇਜ ਜੋੜਦਾ ਹੈ, ਸੰਬੰਧਿਤ ਟੂਲ ਫੰਕਸ਼ਨ ਨੂੰ ਚਲਾਉਂਦਾ ਹੈ, ਅਤੇ ਫੰਕਸ਼ਨ ਕਾਲ ਦੇ ਨਤੀਜੇ ਨਾਲ ਲਿਖਿਤ ਵਿੱਚ ਇੱਕ `function_-result` ਮੈਸੇਜ ਜੋੜਦਾ ਹੈ।

8. ਟੂਲ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਡਸਿਪੈਚ ਕਰਨ ਤੋਂ ਬਾਅਦ, ਬਾਅਦ ਦੇ ਲੂਪਾਂ ਵਿੱਚ ਡੁਪਲੀਕੇਟ ਫੰਕਸ਼ਨ ਕਾਲਾਂ ਨੂੰ ਰੋਕਣ ਲਈ ਫੰਕਸ਼ਨ ਕਾਲ ਹਿਸਟਰੀ ਨੂੰ ਸਾਫ਼ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
9. ਜੇ ਕੋਈ ਟੂਲ ਫੰਕਸ਼ਨ ਨਹੀਂ ਚਲਾਏ ਗਏ ਸਨ, ਤਾਂ `finalize` ਮੈਥਡ `bot_message` ਨੂੰ ਅੰਤਿਮ ਕੰਟੈਂਟ ਨਾਲ ਅੱਪਡੇਟ ਕਰਦਾ ਹੈ, ਇਸਨੂੰ ਪੂਰਾ ਹੋਇਆ ਮਾਰਕ ਕਰਦਾ ਹੈ, ਅਤੇ ਅੱਪਡੇਟ ਕੀਤਾ ਮੈਸੇਜ ਕਲਾਇੰਟ ਨੂੰ ਭੇਜਦਾ ਹੈ।
10. ਲੂਪ ਸ਼ਰਤ `-> { bot_message.complete? }` ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਜੇ `bot_message` ਨੂੰ ਪੂਰਾ ਹੋਇਆ ਮਾਰਕ ਨਹੀਂ ਕੀਤਾ ਗਿਆ ਹੈ, ਤਾਂ ਲੂਪ ਜਾਰੀ ਰਹਿੰਦਾ ਹੈ, ਅਤੇ ਮੂਲ ਪ੍ਰੌਮਪਟ ਨੂੰ ਅੱਪਡੇਟ ਕੀਤੀ ਗੱਲਬਾਤ ਦੀ ਲਖਿਤ ਨਾਲ ਦੁਬਾਰਾ ਸਬਮਿਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
11. ਕਦਮ 3-10 ਨੂੰ ਉਦੋਂ ਤੱਕ ਦੁਹਰਾਇਆ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਤੱਕ `bot_message` ਨੂੰ ਮੁਕੰਮਲ ਵਜੋਂ ਚਿੰਨ੍ਹਿਤ ਨਹੀਂ ਕੀਤਾ ਜਾਂਦਾ, ਜੋ ਦਰਸਾਉਂਦਾ ਹੈ ਕਿ AI ਮਾਡਲ ਨੇ ਆਪਣਾ ਜਵਾਬ ਤਿਆਰ ਕਰ ਲਿਆ ਹੈ ਅਤੇ ਹੋਰ ਟੂਲ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਚਲਾਉਣ ਦੀ ਲੋੜ ਨਹੀਂ ਹੈ।

ਇਸ ਗੱਲਬਾਤ ਲੂਪ ਨੂੰ ਲਾਗੂ ਕਰਕੇ, ਤੁਸੀਂ AI ਮਾਡਲ ਨੂੰ ਐਪਲੀਕੇਸ਼ਨ ਨਾਲ ਦੋ-ਤਰਫ਼ਾ ਗੱਲਬਾਤ ਵਿੱਚ ਸ਼ਾਮਲ ਹੋਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦੇ ਹੋ, ਜਿੱਥੇ ਲੋੜ ਅਨੁਸਾਰ ਟੂਲ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਗੱਲਬਾਤ ਦੇ ਕੁਦਰਤੀ ਸਿੱਟੇ ਤੱਕ ਪਹੁੰਚਣ ਤੱਕ ਯੂਜ਼ਰ-ਫੇਸਿੰਗ ਜਵਾਬ ਤਿਆਰ ਕੀਤੇ ਜਾਂਦੇ ਹਨ।

ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ ਅਤੇ ਗੱਲਬਾਤ ਲੂਪ ਦਾ ਮੇਲ ਗਤੀਸ਼ੀਲ ਅਤੇ ਇੰਟਰਐਕਟਿਵ AI-ਸੰਚਾਲਿਤ ਤਜਰਬਿਆਂ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ, ਜਿੱਥੇ AI ਮਾਡਲ ਯੂਜ਼ਰ ਇਨਪੁੱਟ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰ ਸਕਦਾ ਹੈ, ਵੱਖ-ਵੱਖ ਟੂਲਜ਼ ਅਤੇ ਫੰਕਸ਼ਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦਾ ਹੈ, ਅਤੇ ਵਿਕਸਿਤ ਹੋ ਰਹੇ ਗੱਲਬਾਤ ਦੇ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਰੀਅਲ-ਟਾਈਮ ਜਵਾਬ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦਾ ਹੈ।

ਆਟੋ ਕੰਟੀਨਊਏਸ਼ਨ

ਇਹ AI ਆਉਟਪੁੱਟ ਦੀਆਂ ਸੀਮਾਵਾਂ ਬਾਰੇ ਜਾਣਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਜ਼ਿਆਦਾਤਰ ਮਾਡਲਾਂ ਕੋਲ ਇੱਕ ਸਿੰਗਲ ਜਵਾਬ ਵਿੱਚ ਵੱਧ ਤੋਂ ਵੱਧ ਟੋਕਨਾਂ ਦੀ ਸੰਖਿਆ ਹੁੰਦੀ ਹੈ, ਜੋ `max_tokens` ਪੈਰਾਮੀਟਰ ਦੁਆਰਾ ਨਿਰਧਾਰਤ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜੇਕਰ AI ਮਾਡਲ ਜਵਾਬ ਤਿਆਰ ਕਰਦੇ ਸਮੇਂ ਇਸ ਸੀਮਾ ਤੱਕ ਪਹੁੰਚ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਇਹ ਅਚਾਨਕ ਰੁਕ ਜਾਵੇਗਾ ਅਤੇ ਦਰਸਾਏਗਾ ਕਿ ਆਉਟਪੁੱਟ ਕੱਟਿਆ ਗਿਆ ਸੀ।

AI ਪਲੇਟਫਾਰਮ API ਤੋਂ ਸਟ੍ਰੀਮਿੰਗ ਜਵਾਬ ਵਿੱਚ, ਤੁਸੀਂ ਚੰਕ ਵਿੱਚ `finish_reason` ਵੇਰੀਏਬਲ ਦੀ ਜਾਂਚ ਕਰਕੇ ਇਸ ਸਥਿਤੀ ਦਾ ਪਤਾ ਲਗਾ ਸਕਦੇ ਹੋ। ਜੇਕਰ `finish_reason` ਨੂੰ `"length"` 'ਤੇ ਸੈੱਟ ਕੀਤਾ ਗਿਆ

ਹੈ (ਜਾਂ ਮਾਡਲ ਲਈ ਕੋਈ ਹੋਰ ਖਾਸ ਕੀ ਵੈਲਯੂ), ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਮਾਡਲ ਜਨਰੇਸ਼ਨ ਦੌਰਾਨ ਆਪਣੀ ਵੱਧ ਤੋਂ ਵੱਧ ਟੋਕਨ ਸੀਮਾ ਤੱਕ ਪਹੁੰਚ ਗਿਆ ਹੈ ਅਤੇ ਆਉਟਪੁੱਟ ਛੋਟਾ ਕਰ ਦਿੱਤਾ ਗਿਆ ਹੈ।

ਇਸ ਸਥਿਤੀ ਨੂੰ ਸੁਚਾਰੂ ਢੰਗ ਨਾਲ ਸੰਭਾਲਣ ਅਤੇ ਇੱਕ ਨਿਰਵਘਿਨ ਯੂਜ਼ਰ ਤਜਰਬਾ ਪ੍ਰਦਾਨ ਕਰਨ ਦਾ ਇੱਕ ਤਰੀਕਾ ਹੈ ਆਪਣੀ ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ ਲੌਜਿਕ ਵਿੱਚ ਇੱਕ ਆਟੋ-ਕੰਟੀਨਊਏਸ਼ਨ ਮੈਕੇਨਿਜ਼ਮ ਨੂੰ ਲਾਗੂ ਕਰਨਾ। ਲੰਬਾਈ-ਸੰਬੰਧਿਤ ਸਮਾਪਤੀ ਕਾਰਨਾਂ ਲਈ ਇੱਕ ਪੈਟਰਨ ਮੈਚ ਜੋੜ ਕੇ, ਤੁਸੀਂ ਲੂਪ ਕਰਨ ਅਤੇ ਜਿੱਥੇ ਇਹ ਛੱਡਿਆ ਸੀ ਉੱਥੋਂ ਆਉਟਪੁੱਟ ਨੂੰ ਆਟੋਮੈਟਿਕ ਰੂਪ ਵਿੱਚ ਜਾਰੀ ਰੱਖਣ ਦੀ ਚੋਣ ਕਰ ਸਕਦੇ ਹੋ।

ਇੱਥੇ `ReplyStream` ਕਲਾਸ ਵਿੱਚ `call` ਮੈਥਡ ਨੂੰ ਆਟੋ ਕੰਟੀਨਊਏਸ਼ਨ ਦਾ ਸਮਰਥਨ ਕਰਨ ਲਈ ਕਵਿੰ ਸੰਸ਼ੋਧਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਇਸਦਾ ਇੱਕ ਜਾਣਬੁੱਝ ਕੇ ਸਰਲ ਉਦਾਹਰਣ ਹੈ:

```
LENGTH_STOPS = %w[length MAX_TOKENS]

def call(chunk, _bytesize)
  case chunk.deep_symbolize_keys
    # ...

    in {
      choices: [
        { delta: {content: new_content},
          finish_reason: finish_reason } ] }

      buffer << new_content.to_s

      if finish_reason.blank?
        send_to_client if new_content.to_s.match?(/\n\n/)
      elsif LENGTH_STOPS.include?(finish_reason)
        continue_cutoff
      else
        finalize
      end

    # ...
  end
end

private

def continue_cutoff
  conversation.bot_message!(buffer.join, visible: false)
```

```
  conversation.user_message!("please continue", visible: false)
  bot_message.update_column(:created_at, Time.current)
end
```

ਇਸ ਸੋਧੇ ਹੋਏ ਵਰਜਨ ਵਿੱਚ, ਜਦੋਂ `finish_reason` ਕੱਟੇ ਹੋਏ ਆਉਟਪੁੱਟ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ, ਸਟ੍ਰੀਮ ਨੂੰ ਅੰਤਮਿ ਕਰਨ ਦੀ ਬਜਾਏ, ਅਸੀਂ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਵਿੱਚ ਸੁਨੇਹਿਆਂ ਦੀ ਇੱਕ ਜੋੜੀ ਨੂੰ ਅੰਤਮਿ ਕੀਤੇ ਬਨਿਾਂ ਜੋੜਦੇ ਹਾਂ, ਮੂਲ ਯੂਜ਼ਰ-ਫੇਸਿੰਗ ਜਵਾਬੀ ਸੁਨੇਹੇ ਨੂੰ ਇਸਦੀ `created_at` ਵਸ਼ਿਸ਼ਤਾ ਨੂੰ ਅਪਡੇਟ ਕਰਕੇ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਦੇ "ਹੇਠਾਂ" ਭੇਜਦੇ ਹਾਂ, ਅਤੇ ਫਰਿ ਲੂਪ ਨੂੰ ਚੱਲਣ ਦਿੰਦੇ ਹਾਂ, ਤਾਂ ਜੋ ਏ.ਆਈ. ਜਥਿੇ ਛੱਡਿਆ ਸੀ ਉੱਥੋਂ ਜਾਰੀ ਰੱਖ ਸਕੇ।

ਯਾਦ ਰੱਖੋ ਕ ਿਏ.ਆਈ. ਕੰਪਲੀਸ਼ਨ ਐਂਡਪੁਆਇੰਟ ਸਟੇਟਲੈੱਸ ਹੈ। ਇਹ ਸਰਫ਼ ਉਹੀ "ਜਾਣਦਾ" ਹੈ ਜੋ ਤੁਸੀਂ ਇਸਨੂੰ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਰਾਹੀਂ ਦੱਸਦੇ ਹੋ। ਇਸ ਮਾਮਲੇ ਵਿੱਚ, ਜਸਿ ਤਰੀਕੇ ਨਾਲ ਅਸੀਂ ਏ.ਆਈ. ਨੂੰ ਦੱਸਦੇ ਹਾਂ ਕ ਿਇਹ ਕੱਟਿਆ ਗਿਆ ਸੀ, ਉਹ ਹੈ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਵਿੱਚ "ਅਦੱਖ" (ਅੰਤਮਿ ਯੂਜ਼ਰ ਲਈ) ਸੁਨੇਹੇ ਜੋੜ ਕੇ। ਪਰ ਯਾਦ ਰੱਖੋ, ਕ ਿਇਹ ਜਾਣਬੂੱਝ ਕੇ ਸਰਲ ਬਣਾਇਆ ਗਿਆ ਉਦਾਹਰਣ ਹੈ। ਇੱਕ ਅਸਲ ਲਾਗੂਕਰਨ ਨੂੰ ਹੋਰ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਪ੍ਰਬੰਧਨ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇਗੀ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਕ ਿਅਸੀਂ ਟੋਕਨ ਬਰਬਾਦ ਨਾ ਕਰੀਏ ਅਤੇ/ਜਾਂ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਵਿੱਚ ਦੁਹਰਾਏ ਗਏ ਸਹਾਇਕ ਸੁਨੇਹਿਆਂ ਨਾਲ ਏ.ਆਈ. ਨੂੰ ਭਰਮਤਿ ਨਾ ਕਰੀਏ।

ਆਟੋ-ਕੰਟੀਨਊਿਏਸ਼ਨ ਦਾ ਇੱਕ ਅਸਲ ਲਾਗੂਕਰਨ ਵਿੱਚ ਤਥਾਕਥਤਿ "ਸਰਕਟ ਬਰੇਕਰ ਲੌਜਕਿ" ਵੀ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਜੋ ਬੇਕਾਬੂ ਲੂਪਿੰਗ ਨੂੰ ਰੋਕਣ ਲਈ ਹੈ। ਇਸਦਾ ਕਾਰਨ ਇਹ ਹੈ ਕ,ਿ ਕੁਝ ਖਾਸ ਕਸਿਮ ਦੇ ਯੂਜ਼ਰ ਪ੍ਰੌਮਪਟਸ ਅਤੇ ਘੱਟ `max_tokens` ਸੈਟਿੰਗਾਂ ਦੇ ਨਾਲ, ਏ.ਆਈ. ਯੂਜ਼ਰ-ਫੇਸਿੰਗ ਆਉਟਪੁੱਟ ਨੂੰ ਲਗਾਤਾਰ ਲੂਪ ਕਰ ਸਕਦਾ ਹੈ।

> ਧਆਿਨ ਰੱਖੋ ਕ ਿਹਰ ਲੂਪ ਨੂੰ ਇੱਕ ਵੱਖਰੀ ਬੇਨਤੀ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਅਤੇ ਹਰ ਬੇਨਤੀ ਤੁਹਾਡੀ ਪੂਰੀ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਨੂੰ ਦੁਬਾਰਾ ਵਰਤਦੀ ਹੈ। ਤੁਹਾਨੂੰ ਆਪਣੀ ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ ਆਟੋ ਕੰਟੀਨਊਿਏਸ਼ਨ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਫੈਸਲਾ ਕਰਦੇ ਸਮੇਂ ਯੂਜ਼ਰ ਅਨੁਭਵ ਅਤੇ ਏ.ਪੀ.ਆਈ. ਵਰਤੋਂ ਵਚਿਕਾਰ ਟ੍ਰੇਡ-ਆਫ਼ਸ ’ਤੇ ਜਰੂਰ ਵਚਿਾਰ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ। ਆਟੋ-ਕੰਟੀਨਊਿਏਸ਼ਨ ਖਾਸ ਤੌਰ ’ਤੇ ਖ਼ਤਰਨਾਕ ਢੰਗ ਨਾਲ ਮਹਿੰਗਾ ਹੋ ਸਕਦਾ ਹੈ, ਖਾਸਕਰ ਜਦੋਂ ਪ੍ਰੀਮੀਅਮ ਵਪਾਰਕ ਮਾਡਲਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ।

ਸਾਂਟਾ

ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ ਏ.ਆਈ.-ਸੰਚਾਲਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦੇ ਨਰਿਮਾਣ ਦਾ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਪਹਲਿੂ ਹੈ ਜੋ ਟੂਲ ਵਰਤੋਂ ਨੂੰ ਲਾਈਵ ਏ.ਆਈ. ਜਵਾਬਾਂ ਨਾਲ ਜੋੜਦਾ ਹੈ। ਏ.ਆਈ. ਪਲੇਟਫਾਰਮ ਏ.ਪੀ.ਆਈ.ਜ਼ ਤੋਂ ਸਟ੍ਰੀਮਿੰਗ ਡੇਟਾ ਨੂੰ ਕੁਸ਼ਲਤਾ ਨਾਲ ਸੰਭਾਲ ਕੇ, ਤੁਸੀਂ ਇੱਕ ਨਰਿਵਘਿਨ ਅਤੇ ਇੰਟਰਐਕਟਵਿ ਯੂਜ਼ਰ ਅਨੁਭਵ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹੋ, ਵੱਡੇ ਜਵਾਬਾਂ ਨੂੰ ਸੰਭਾਲ ਸਕਦੇ ਹੋ, ਸਰੋਤਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦੇ ਹੋ, ਅਤੇ ਗਲਤੀਆਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲ ਸਕਦੇ ਹੋ।

ਪ੍ਰਦਾਨ ਕੀਤੀ `Conversation::ReplyStream` ਕਲਾਸ ਦਰਸਾਉਂਦੀ ਹੈ ਕਿ ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ ਨੂੰ Ruby ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ ਪੈਟਰਨ ਮੈਚਿੰਗ ਅਤੇ ਈਵੈਂਟ-ਆਧਾਰਤਿ ਆਰਕੀਟੈਕਚਰ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਕਵਿੇਂ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ ਤਕਨੀਕਾਂ ਨੂੰ ਸਮਝ ਕੇ ਅਤੇ ਉਨ੍ਹਾਂ ਦਾ ਲਾਭ ਉਠਾ ਕੇ, ਤੁਸੀਂ ਆਪਣੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਏ.ਆਈ. ਏਕੀਕਰਣ ਦੀ ਪੂਰੀ ਸਮਰੱਥਾ ਨੂੰ ਅਨਲੌਕ ਕਰ ਸਕਦੇ ਹੋ ਅਤੇ ਸ਼ਕਤੀਸ਼ਾਲੀ ਅਤੇ ਰੁਝੇਵੇਂ ਵਾਲੇ ਯੂਜ਼ਰ ਅਨੁਭਵ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹੋ।

ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲਾ ਡਾਟਾ

ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲਾ ਡਾਟਾ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਡਾਟਾ ਦੀ ਅਖੰਡਤਾ, ਇਕਸਾਰਤਾ, ਅਤੇ ਗੁਣਵੱਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਪਹੁੰਚ ਹੈ, ਜੋ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ (ਐੱਲ.ਐੱਲ.ਐੱਮ.) ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਦਾ ਲਾਭ ਲੈਂਦੀ ਹੈ। ਪੈਟਰਨਾਂ ਦੀ ਇਹ ਸ਼੍ਰੇਣੀ ਏ.ਆਈ. ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਡਾਟਾ ਅਸਧਾਰਨਤਾਵਾਂ, ਅਸੰਗਤੀਆਂ, ਜਾਂ ਗਲਤੀਆਂ ਨੂੰ ਆਪਣੇ ਆਪ ਪਤਾ ਲਗਾਉਣ, ਨਿਦਾਨ ਕਰਨ, ਅਤੇ ਸੁਧਾਰਨ ਦੇ ਵਿਚਾਰ 'ਤੇ ਕੇਂਦਰਿਤ ਹੈ, ਜਿਸ ਨਾਲ ਡਿਵੈਲਪਰਾਂ 'ਤੇ ਬੋਝ ਘੱਟ ਜਾਂਦਾ ਹੈ ਅਤੇ ਡਾਟਾ ਦੀ ਵਿਸ਼ਵਾਸਯੋਗਤਾ ਦਾ ਉੱਚ ਪੱਧਰ ਬਣਿਆ ਰਹਿੰਦਾ ਹੈ।

ਆਪਣੇ ਮੂਲ ਰੂਪ ਵਿੱਚ, ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਪੈਟਰਨ ਇਹ ਪਛਾਣਦੇ ਹਨ ਕਿ ਡਾਟਾ ਕਿਸੇ ਵੀ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਜਾਨ ਹੁੰਦਾ ਹੈ, ਅਤੇ ਇਸਦੀ ਸ਼ੁੱਧਤਾ ਅਤੇ ਅਖੰਡਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣਾ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਸਹੀ ਕੰਮਕਾਜ ਅਤੇ ਯੂਜ਼ਰ ਅਨੁਭਵ ਲਈ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਹਾਲਾਂਕਿ, ਡਾਟਾ ਗੁਣਵੱਤਾ ਦਾ ਪ੍ਰਬੰਧਨ ਅਤੇ ਰੱਖ-ਰਖਾਅ ਇੱਕ ਜਟਿਲ ਅਤੇ ਸਮਾਂ ਲੈਣ ਵਾਲਾ ਕੰਮ ਹੋ ਸਕਦਾ ਹੈ, ਖਾਸ ਕਰਕੇ ਜਦੋਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਆਕਾਰ ਅਤੇ ਜਟਿਲਤਾ ਵਿੱਚ ਵਧਦੀਆਂ ਹਨ। ਇੱਥੇ ਏ.ਆਈ. ਦੀ ਸ਼ਕਤੀ ਕੰਮ ਆਉਂਦੀ ਹੈ।

ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਪੈਟਰਨਾਂ ਵਿੱਚ, ਏ.ਆਈ. ਵਰਕਰਾਂ ਨੂੰ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਡਾਟਾ ਦੀ ਲਗਾਤਾਰ ਨਿਗਰਾਨੀ ਅਤੇ ਵਿਸ਼ਲੇਸ਼ਣ ਲਈ ਨਿਯੁਕਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਮਾਡਲ ਡਾਟਾ ਦੇ ਅੰਦਰ ਪੈਟਰਨ, ਸਬੰਧ, ਅਤੇ

ਅਸਧਾਰਨਤਾਵਾਂ ਨੂੰ ਸਮਝਣ ਅਤੇ ਵਆਿਖਆਿ ਕਰਨ ਦੀ ਯੋਗਤਾ ਰੱਖਦੇ ਹਨ। ਉਹਨਾਂ ਦੀਆਂ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਪ੍ਰੋਸੈਸਿੰਗ ਅਤੇ ਸਮਝ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਦਾ ਲਾਭ ਲੈ ਕੇ, ਉਹ ਡਾਟਾ ਵਿੱਚ ਸੰਭਾਵੀ ਸਮੱਸਆਿਵਾਂ ਜਾਂ ਅਸੰਗਤੀਆਂ ਦੀ ਪਛਾਣ ਕਰ ਸਕਦੇ ਹਨ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਠੀਕ ਕਰਨ ਲਈ ਢੁਕਵੀਆਂ ਕਾਰਵਾਈਆਂ ਕਰ ਸਕਦੇ ਹਨ।

ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਦੀ ਪ੍ਰਕਰਿਆ ਵਿੱਚ ਆਮ ਤੌਰ 'ਤੇ ਕਈ ਮੁੱਖ ਕਦਮ ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ:

1. **ਡਾਟਾ ਨਗਿਰਾਨੀ:** ਏ.ਆਈ. ਵਰਕਰ ਲਗਾਤਾਰ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਡਾਟਾ ਸਟ੍ਰੀਮਾਂ, ਡਾਟਾਬੇਸਾਂ, ਜਾਂ ਸਟੋਰੇਜ ਸਸਿਟਮਾਂ 'ਤੇ ਨਜ਼ਰ ਰੱਖਦੇ ਹਨ, ਕਸਿੇ ਵੀ ਅਸਧਾਰਨਤਾਵਾਂ, ਅਸੰਗਤੀਆਂ, ਜਾਂ ਗਲਤੀਆਂ ਦੇ ਸੰਕੇਤਾਂ ਦੀ ਭਾਲ ਕਰਦੇ ਹਨ। ਵਕਿਲਪਕ ਤੌਰ 'ਤੇ, ਤੁਸੀਂ ਕਸਿੇ ਅਪਵਾਦ ਦੀ ਪ੍ਰਤੀਕਰਿਆ ਵਿੱਚ ਏ.ਆਈ. ਕੰਪੋਨੈਂਟ ਨੂੰ ਸਕਰਿਆ ਕਰ ਸਕਦੇ ਹੋ।
2. **ਅਸਧਾਰਨਤਾ ਦੀ ਪਛਾਣ:** ਜਦੋਂ ਕੋਈ ਸਮੱਸਆਿ ਦਾ ਪਤਾ ਲੱਗਦਾ ਹੈ, ਏ.ਆਈ. ਵਰਕਰ ਸਮੱਸਆਿ ਦੀ ਵਸ਼ਿੇਸ਼ ਪ੍ਰਕਰਿਤੀ ਅਤੇ ਦਾਇਰੇ ਦੀ ਪਛਾਣ ਕਰਨ ਲਈ ਡਾਟਾ ਦਾ ਵਸਿਥਾਰ ਨਾਲ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਦਾ ਹੈ। ਇਸ ਵਿੱਚ ਗੁੰਮ ਮੁੱਲਾਂ, ਅਸੰਗਤ ਫਾਰਮੈਟਾਂ, ਜਾਂ ਪਹਲਿਾਂ ਤੋਂ ਨਰਿਧਾਰਤ ਨਯਿਮਾਂ ਜਾਂ ਪਾਬੰਦੀਆਂ ਦੀ ਉਲੰਘਣਾ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਦਾ ਪਤਾ ਲਗਾਉਣਾ ਸ਼ਾਮਲ ਹੋ ਸਕਦਾ ਹੈ।
3. **ਨਦਿਾਨ ਅਤੇ ਸੁਧਾਰ:** ਇੱਕ ਵਾਰ ਸਮੱਸਆਿ ਦੀ ਪਛਾਣ ਹੋ ਜਾਣ 'ਤੇ, ਏ.ਆਈ. ਵਰਕਰ ਢੁਕਵੀਂ ਕਾਰਵਾਈ ਨਰਿਧਾਰਤ ਕਰਨ ਲਈ ਡਾਟਾ ਡੋਮੇਨ ਦੇ ਆਪਣੇ ਗਆਿਨ ਅਤੇ ਸਮਝ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ। ਇਸ ਵਿੱਚ ਆਪਣੇ ਆਪ ਡਾਟਾ ਨੂੰ ਠੀਕ ਕਰਨਾ, ਗੁੰਮ ਮੁੱਲਾਂ ਨੂੰ ਭਰਨਾ, ਜਾਂ ਜੇ ਜ਼ਰੂਰੀ ਹੋਵੇ ਤਾਂ ਮਨੁੱਖੀ ਦਖਲਅੰਦਾਜ਼ੀ ਲਈ ਸਮੱਸਆਿ ਨੂੰ ਫਲੈਗ ਕਰਨਾ ਸ਼ਾਮਲ ਹੋ ਸਕਦਾ ਹੈ।
4. **ਨਰਿੰਤਰ ਸਖਿਣ (ਵਕਿਲਪਕ, ਵਰਤੋਂ ਕੇਸ 'ਤੇ ਨਰਿਭਰ ਕਰਦਾ ਹੈ):** ਜਵਿੇਂ-ਜਵਿੇਂ ਤੁਹਾਡਾ ਏ.ਆਈ. ਵਰਕਰ ਵੱਖ-ਵੱਖ ਡਾਟਾ ਸਮੱਸਆਿਵਾਂ ਦਾ ਸਾਹਮਣਾ ਕਰਦਾ ਹੈ ਅਤੇ ਹੱਲ ਕਰਦਾ ਹੈ, ਇਹ ਇਸ ਬਾਰੇ ਦੱਸਦਾ ਹੈ ਕਿ ਕੀ ਹੋਇਆ ਅਤੇ ਇਸ ਨੇ ਕਵਿੇਂ ਜਵਾਬ ਦੱਤਿਾ। ਇਹ ਮੈਟਾਡਾਟਾ ਸਖਿਣ ਦੀਆਂ ਪ੍ਰਕਰਿਆਵਾਂ ਵਿੱਚ ਫੀਡ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਤੁਹਾਨੂੰ (ਅਤੇ ਸ਼ਾਇਦ ਅੰਡਰਲਾਈਂਗ ਮਾਡਲ ਨੂੰ, ਫਾਈਨ-ਟਊਿਨਿੰਗ ਰਾਹੀਂ) ਡਾਟਾ ਅਸਧਾਰਨਤਾਵਾਂ ਦੀ ਪਛਾਣ ਕਰਨ ਅਤੇ ਹੱਲ ਕਰਨ ਵਿੱਚ ਸਮੇਂ ਦੇ ਨਾਲ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਅਤੇ ਕੁਸ਼ਲ ਬਣਾਉਂਦਾ ਹੈ।

ਸਵੈਚਲਤਿ ਤੌਰ 'ਤੇ ਡਾਟਾ ਸਮੱਸਆਿਵਾਂ ਦੀ ਪਛਾਣ ਅਤੇ ਸੁਧਾਰ ਕਰਕੇ, ਤੁਸੀਂ ਯਕੀਨੀ ਬਣਾ ਸਕਦੇ ਹੋ ਕਿ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਉੱਚ-ਗੁਣਵੱਤਾ, ਭਰੋਸੇਯੋਗ ਡਾਟਾ 'ਤੇ ਕੰਮ ਕਰਦੀ ਹੈ। ਇਹ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਕਾਰਜਸ਼ੀਲਤਾ ਜਾਂ ਯੂਜ਼ਰ ਅਨੁਭਵ ਨੂੰ ਪ੍ਰਭਾਵਤਿ ਕਰਨ ਵਾਲੀਆਂ ਗਲਤੀਆਂ, ਅਸੰਗਤੀਆਂ, ਜਾਂ ਡਾਟਾ-ਸੰਬੰਧਤਿ ਬੱਗਾਂ ਦੇ ਜੋਖਮ ਨੂੰ ਘਟਾਉਂਦਾ ਹੈ।

ਜਦੋਂ ਤੁਹਾਡੇ ਕੋਲ ਡਾਟਾ ਨਗਿਰਾਨੀ ਅਤੇ ਸੁਧਾਰ ਦੇ ਕੰਮ ਨੂੰ ਸੰਭਾਲਣ ਲਈ AI ਵਰਕਰ ਹੁੰਦੇ ਹਨ, ਤਾਂ ਤੁਸੀਂ ਆਪਣੀਆਂ ਕੋਸ਼ਸ਼ਿਾਂ ਨੂੰ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਹੋਰ ਮਹੱਤਵਪੂਰਨ ਪਹਲੂਆਂ 'ਤੇ ਕੇਂਦਰਤਿ ਕਰ ਸਕਦੇ ਹੋ। ਇਹ ਉਸ ਸਮੇਂ ਅਤੇ ਸਰੋਤਾਂ

ਨੂੰ ਬਚਾਉਂਦਾ ਹੈ ਜੋ ਹੋਰ ਤਰ੍ਹਾਂ ਮੈਨੂਅਲ ਡਾਟਾ ਸਫਾਈ ਅਤੇ ਰੱਖ-ਰਖਾਅ 'ਤੇ ਖਰਚ ਹੋਣੇ ਸਨ। ਅਸਲ ਵਿੱਚ, ਜਿਵੇਂ-ਜਿਵੇਂ ਤੁਹਾਡੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਆਕਾਰ ਅਤੇ ਗੁੰਝਲਤਾ ਵਿੱਚ ਵਧਦੀਆਂ ਹਨ, ਮੈਨੂਅਲ ਤੌਰ 'ਤੇ ਡਾਟਾ ਗੁਣਵੱਤਾ ਦਾ ਪ੍ਰਬੰਧਨ ਕਰਨਾ ਵਧੇਰੇ ਚੁਣੌਤੀਪੂਰਨ ਹੁੰਦਾ ਜਾਂਦਾ ਹੈ। "ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲਾ ਡਾਟਾ" ਪੈਟਰਨ AI ਦੀ ਸ਼ਕਤੀ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਵੱਡੀ ਮਾਤਰਾ ਵਿੱਚ ਡਾਟਾ ਨੂੰ ਸੰਭਾਲਣ ਅਤੇ ਰੀਅਲ-ਟਾਈਮ ਵਿੱਚ ਸਮੱਸਿਆਵਾਂ ਦਾ ਪਤਾ ਲਗਾਉਣ ਲਈ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਸਕੇਲ ਕਰਦਾ ਹੈ।

> ਆਪਣੀ ਪ੍ਰਕਰਿਤੀ ਦੇ ਕਾਰਨ, AI ਮਾਡਲ ਬਿਨਾਂ ਕਿਸੇ ਨਿਗਰਾਨੀ ਜਾਂ ਬਹੁਤ ਘੱਟ ਨਿਗਰਾਨੀ ਦੇ ਸਮੇਂ ਦੇ ਨਾਲ ਬਦਲਦੇ ਡਾਟਾ ਪੈਟਰਨਾਂ, ਸਕੀਮਾਂ, ਜਾਂ ਲੋੜਾਂ ਦੇ ਅਨੁਕੂਲ ਹੋ ਸਕਦੇ ਹਨ। ਜਿੰਨਾ ਚਰਿ ਉਹਨਾਂ ਦੇ ਨਿਰਦੇਸ਼ ਢੁਕਵੀਂ ਅਗਵਾਈ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ, ਖਾਸ ਕਰਕੇ ਇੱਛਿਤ ਨਤੀਜਿਆਂ ਦੇ ਸੰਬੰਧ ਵਿੱਚ, ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਵਿਆਪਕ ਮੈਨੂਅਲ ਦਖਲਅੰਦਾਜ਼ੀ ਜਾਂ ਕੋਡ ਤਬਦੀਲੀਆਂ ਦੀ ਲੋੜ ਤੋਂ ਬਿਨਾਂ ਵਿਕਸਿਤ ਹੋ ਸਕਦੀ ਹੈ ਅਤੇ ਨਵੇਂ ਡਾਟਾ ਸਨਾਰੀਓ ਨੂੰ ਸੰਭਾਲ ਸਕਦੀ ਹੈ।

ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਪੈਟਰਨ ਹੋਰ ਸ੍ਰੇਣੀਆਂ ਦੇ ਪੈਟਰਨਾਂ ਨਾਲ ਚੰਗੀ ਤਰ੍ਹਾਂ ਮੇਲ ਖਾਂਦੇ ਹਨ, ਜਿਵੇਂ ਕਿ "ਵਰਕਰਾਂ ਦੀ ਬਹੁਤਾਤ"। ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਦੀ ਸਮਰੱਥਾ ਨੂੰ ਇੱਕ ਵਿਸ਼ੇਸ਼ ਕਿਸਮ ਦੇ ਵਰਕਰ ਵਜੋਂ ਵੇਖਿਆ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਖਾਸ ਤੌਰ 'ਤੇ ਡਾਟਾ ਗੁਣਵੱਤਾ ਅਤੇ ਅਖੰਡਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ 'ਤੇ ਧਿਆਨ ਕੇਂਦਰਿਤ ਕਰਦਾ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਦਾ ਵਰਕਰ ਹੋਰ AI ਵਰਕਰਾਂ ਦੇ ਨਾਲ ਕੰਮ ਕਰਦਾ ਹੈ, ਹਰ ਇੱਕ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਕਾਰਜਸ਼ੀਲਤਾ ਦੇ ਵੱਖ-ਵੱਖ ਪਹਲੂਆਂ ਵਿੱਚ ਯੋਗਦਾਨ ਪਾਉਂਦਾ ਹੈ।

ਅਮਲ ਵਿੱਚ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਪੈਟਰਨਾਂ ਨੂੰ ਲਾਗੂ ਕਰਨ ਲਈ ਐਪਲੀਕੇਸ਼ਨ ਆਰਕੀਟੈਕਚਰ ਵਿੱਚ AI ਮਾਡਲਾਂ ਦੇ ਧਿਆਨਪੂਰਵਕ ਡਿਜ਼ਾਈਨ ਅਤੇ ਏਕੀਕਰਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਡਾਟਾ ਦੇ ਨੁਕਸਾਨ ਅਤੇ ਖਰਾਬੀ ਦੇ ਜੋਖਮਾਂ ਕਾਰਨ, ਤੁਹਾਨੂੰ ਸਪਸ਼ਟ ਦਿਸ਼ਾ-ਨਿਰਦੇਸ਼ ਨਿਰਧਾਰਤ ਕਰਨੇ ਚਾਹੀਦੇ ਹਨ ਕਿ ਤੁਸੀਂ ਇਸ ਤਕਨੀਕ ਦੀ ਵਰਤੋਂ ਕਿਵੇਂ ਕਰੋਗੇ। ਤੁਹਾਨੂੰ ਪ੍ਰਦਰਸ਼ਨ, ਸਕੇਲੇਬਲਿਟੀ, ਅਤੇ ਡਾਟਾ ਸੁਰੱਖਿਆ ਵਰਗੇ ਕਾਰਕਾਂ 'ਤੇ ਵੀ ਵਿਚਾਰ ਕਰਨਾ ਚਾਹੀਦਾ ਹੈ।

ਵਿਵਹਾਰਕ ਕੇਸ ਸਟੱਡੀ: ਖਰਾਬ JSON ਨੂੰ ਠੀਕ ਕਰਨਾ

ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਦਾ ਲਾਭ ਲੈਣ ਦੇ ਸਭ ਤੋਂ ਵਿਵਹਾਰਕ ਅਤੇ ਸੁਵਧਿਾਜਨਕ ਤਰੀਕਿਆਂ ਵਿੱਚੋਂ ਇੱਕ ਸਮਝਾਉਣ ਲਈ ਵੀ ਬਹੁਤ ਸਰਲ ਹੈ: ਖਰਾਬ JSON ਨੂੰ ਠੀਕ ਕਰਨਾ।

ਇਹ ਤਕਨੀਕ LLMs ਦੁਆਰਾ ਤਿਆਰ ਕੀਤੇ ਗਏ ਅਧੂਰੇ ਜਾਂ ਅਸੰਗਤ ਡਾਟਾ, ਜਵਿੇਂ ਕਿ ਖਰਾਬ JSON, ਨਾਲ ਨਜਿੰਠਣ ਦੀ ਆਮ ਚੁਣੌਤੀ 'ਤੇ ਲਾਗੂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ, ਅਤੇ ਇਹਨਾਂ ਸਮੱਸਿਆਵਾਂ ਨੂੰ ਸਵੈਚਲਤਿ ਤੌਰ 'ਤੇ ਪਛਾਣਨ ਅਤੇ ਠੀਕ ਕਰਨ ਲਈ ਇੱਕ ਪਹੁੰਚ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ।

Olympia ਵਖਿ ਮੈਂ ਨਯਿਮਤਿ ਤੌਰ 'ਤੇ ਅਜਹਿੀਆਂ ਸਥਤਿੀਆਂ ਦਾ ਸਾਹਮਣਾ ਕਰਦਾ ਹਾਂ ਜਥਿੇ LLMs ਅਜਹਿਾ JSON ਡਾਟਾ ਤਿਆਰ ਕਰਦੇ ਹਨ ਜੋ ਪੂਰੀ ਤਰ੍ਹਾਂ ਵੈਧ ਨਹੀਂ ਹੁੰਦਾ। ਇਹ ਵੱਖ-ਵੱਖ ਕਾਰਨਾਂ ਕਰਕੇ ਹੋ ਸਕਦਾ ਹੈ, ਜਵਿੇਂ ਕਿ LLM ਦੁਆਰਾ ਅਸਲ JSON ਕੋਡ ਤੋਂ ਪਹਲਿਾਂ ਜਾਂ ਬਾਅਦ ਵਚਿ ਟਪਿੱਣੀਆਂ ਜੋੜਨਾ, ਜਾਂ ਗੁੰਮ ਕਾਮਿਆਂ ਜਾਂ ਅਣ-ਐਸਕੇਪਡ ਡਬਲ ਕੋਟਸ ਵਰਗੀਆਂ ਸਟਿੈਕਸ ਗਲਤੀਆਂ ਪੇਸ਼ ਕਰਨਾ। ਇਹ ਸਮੱਸਿਆਵਾਂ ਪਾਰਸਿੰਗ ਗਲਤੀਆਂ ਦਾ ਕਾਰਨ ਬਣ ਸਕਦੀਆਂ ਹਨ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਕਾਰਜਸ਼ੀਲਤਾ ਵਚਿ ਵਘਿਨ ਪਾ ਸਕਦੀਆਂ ਹਨ।

ਇਸ ਸਮੱਸਿਆ ਦੇ ਹੱਲ ਲਈ, ਮੈਂ JsonFixer ਕਲਾਸ ਦੇ ਰੂਪ ਵਚਿ ਇੱਕ ਵਵਿਹਾਰਕ ਹੱਲ ਲਾਗੂ ਕੀਤਾ ਹੈ। ਇਹ ਕਲਾਸ "ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲਾ ਡਾਟਾ" pattern ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ, ਜੋ ਟੁੱਟੇ ਹੋਏ JSON ਨੂੰ ਇਨਪੁੱਟ ਵਜੋਂ ਲੈਂਦੀ ਹੈ ਅਤੇ LLM ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇਸਨੂੰ ਠੀਕ ਕਰਦੀ ਹੈ, ਜਸਿ ਵਚਿ ਜਨ੍ਹਿੀ ਵੱਧ ਤੋਂ ਵੱਧ ਜਾਣਕਾਰੀ ਅਤੇ ਇਰਾਦੇ ਨੂੰ ਸੰਭਵ ਹੋ ਸਕੇ ਬਰਕਰਾਰ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ।

```ruby
class JsonFixer
  include Raix::ChatCompletion

  def call(bad_json, error_message)
    raise "No data provided" if bad_json.blank? || error_message.blank?

    transcript << {
        system: "Consider user-provided JSON that generated a parse
                 exception. Do your best to fix it while preserving the
                 original content and intent as much as possible." }
    transcript << { user: bad_json }
    transcript << { assistant: "What is the error message?"}
    transcript << { user: error_message }
    transcript << { assistant: "Here is the corrected JSON\n```json\n" }

    self.stop = ["```"]

    chat_completion(json: true)
  end

  def model
```

```
    "mistralai/mixtral-8x7b-instruct:nitro"
  end
end
```

ਧਿਆਨ ਦਿਓ ਕਿ `JsonFixer` Ventriloquist ਦੀ ਵਰਤੋਂ ਏ.ਆਈ. ਦੇ ਜਵਾਬਾਂ ਨੂੰ ਨਿਰਦੇਸ਼ਿਤ ਕਰਨ ਲਈ ਕਰਦਾ ਹੈ।

JSON ਡਾਟਾ ਦੀ ਸਵੈ-ਠੀਕ ਕਰਨ ਦੀ ਪ੍ਰਕਿਰਿਆ ਇਸ ਤਰ੍ਹਾਂ ਕੰਮ ਕਰਦੀ ਹੈ:

1. **JSON ਜਨਰੇਸ਼ਨ:** ਕੁਝ ਪ੍ਰੌਮਪਟਸ ਜਾਂ ਲੋੜਾਂ ਦੇ ਆਧਾਰ 'ਤੇ JSON ਡਾਟਾ ਤਿਆਰ ਕਰਨ ਲਈ LLM ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਹਾਲਾਂਕਿ, LLMs ਦੀ ਪ੍ਰਕਿਰਤੀ ਕਾਰਨ, ਤਿਆਰ ਕੀਤਾ JSON ਹਮੇਸ਼ਾ ਪੂਰੀ ਤਰ੍ਹਾਂ ਵੈਧ ਨਹੀਂ ਹੋ ਸਕਦਾ। ਜੇਕਰ ਤੁਸੀਂ ਅਵੈਧ JSON ਦਿੰਦੇ ਹੋ ਤਾਂ JSON ਪਾਰਸਰ ਬੇਸ਼ੱਕ `ParserError` ਉਠਾਏਗਾ।

```
begin
  JSON.parse(llm_generated_json)
rescue JSON::ParserError => e
  JsonFixer.new.call(llm_generated_json, e.message)
end
```

ਧਿਆਨ ਰੱਖੋ ਕਿ ਐਕਸੈਪਸ਼ਨ ਸੁਨੇਹਾ ਵੀ `JSONFixer` ਕਾਲ ਨੂੰ ਪਾਸ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਜੋ ਇਸਨੂੰ ਡਾਟਾ ਨਾਲ ਕੀ ਗਲਤ ਹੈ ਬਾਰੇ ਪੂਰੀ ਤਰ੍ਹਾਂ ਅੰਦਾਜ਼ਾ ਨਾ ਲਗਾਉਣਾ ਪਵੇ, ਖਾਸ ਕਰਕੇ ਜਦੋਂ ਪਾਰਸਰ ਅਕਸਰ ਤੁਹਾਨੂੰ ਬਿਲਕੁਲ ਦੱਸ ਦਿੰਦਾ ਹੈ ਕਿ ਕੀ ਗਲਤ ਹੈ।

2. **ਐੱਲਐੱਲਐੱਮ-ਆਧਾਰਿਤ ਸੁਧਾਰ:** `JSONFixer` ਕਲਾਸ ਟੁੱਟੇ ਹੋਏ JSON ਨੂੰ ਐੱਲਐੱਲਐੱਮ ਨੂੰ ਭੇਜਦੀ ਹੈ, ਨਾਲ ਹੀ JSON ਨੂੰ ਠੀਕ ਕਰਨ ਲਈ ਇੱਕ ਖਾਸ ਪ੍ਰੌਮਪਟ ਜਾਂ ਨਿਰਦੇਸ਼ ਦਿੰਦੀ ਹੈ, ਜਿਸ ਵਿੱਚ ਮੂਲ ਜਾਣਕਾਰੀ ਅਤੇ ਇਰਾਦੇ ਨੂੰ ਜਿੰਨਾ ਸੰਭਵ ਹੋ ਸਕੇ ਬਰਕਰਾਰ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ। ਐੱਲਐੱਲਐੱਮ, ਜੋ ਵੱਡੀ ਮਾਤਰਾ ਵਿੱਚ ਡਾਟਾ 'ਤੇ ਸਿਖਲਾਈ ਪ੍ਰਾਪਤ ਹੈ ਅਤੇ JSON ਸਿੰਟੈਕਸ ਨੂੰ ਸਮਝਦਾ ਹੈ, ਗਲਤੀਆਂ ਨੂੰ ਠੀਕ ਕਰਨ ਅਤੇ ਇੱਕ ਵੈਧ JSON ਸਤਰ ਬਣਾਉਣ ਦੀ ਕੋਸ਼ਿਸ਼ ਕਰਦਾ ਹੈ। ਜਵਾਬ ਸੀਮਾਬੰਦੀ ਦੀ ਵਰਤੋਂ ਐੱਲਐੱਲਐੱਮ ਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਸੀਮਤ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਅਤੇ ਅਸੀਂ Mixtral 8x7B ਨੂੰ AI ਮਾਡਲ ਵਜੋਂ ਚੁਣਦੇ ਹਾਂ, ਕਿਉਂਕਿ ਇਹ ਇਸ ਤਰ੍ਹਾਂ ਦੇ ਕੰਮ ਲਈ ਖਾਸ ਤੌਰ 'ਤੇ ਚੰਗਾ ਹੈ।

3. **ਪ੍ਰਮਾਣੀਕਰਨ ਅਤੇ ਏਕੀਕਰਨ:** ਐਲਐਲਐਮ ਦੁਆਰਾ ਵਾਪਸ ਕੀਤੀ ਗਈ ਠੀਕ ਕੀਤੀ JSON ਸਤਰ ਨੂੰ `JSONFixer` ਕਲਾਸ ਦੁਆਰਾ ਖੁਦ ਪਾਰਸ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਕਿਉਂਕਿ ਇਸਨੇ `chat_completion(json: true)` ਨੂੰ ਕਾਲ ਕੀਤਾ। ਜੇਕਰ ਠੀਕ ਕੀਤਾ JSON ਪ੍ਰਮਾਣੀਕਰਨ ਪਾਸ ਕਰ ਲੈਂਦਾ ਹੈ, ਤਾਂ ਇਸਨੂੰ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਵਰਕਫਲੋ ਵਿੱਚ ਏਕੀਕ੍ਰਤਿ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਜੋ ਐਪਲੀਕੇਸ਼ਨ ਨੂੰ ਨਰਿਵਘਿਨ ਢੰਗ ਨਾਲ ਡਾਟਾ ਦੀ ਪ੍ਰੋਸੈਸਿੰਗ ਜਾਰੀ ਰੱਖਣ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦਾ ਹੈ। ਖਰਾਬ JSON "ਠੀਕ" ਹੋ ਗਿਆ ਹੈ।

ਭਾਵੇਂ ਮੈਂ ਆਪਣੀ `JSONFixer` ਇੰਪਲੀਮੈਂਟੇਸ਼ਨ ਨੂੰ ਕਈ ਵਾਰ ਲਖਿਆ ਅਤੇ ਦੁਬਾਰਾ ਲਖਿਆ ਹੈ, ਮੈਨੂੰ ਸ਼ੱਕ ਹੈ ਕਿ ਉਨ੍ਹਾਂ ਸਾਰੇ ਵਰਜਨਾਂ ਵਿੱਚ ਲਗਾਇਆ ਕੁੱਲ ਸਮਾਂ ਇੱਕ ਜਾਂ ਦੋ ਘੰਟਿਆਂ ਤੋਂ ਵੱਧ ਹੈ।

ਧਿਆਨ ਰੱਖੋ ਕਿ ਇਰਾਦੇ ਦੀ ਸੁਰੱਖਿਆ ਕਿਸੇ ਵੀ ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਪੈਟਰਨ ਦਾ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਤੱਤ ਹੈ। ਐਲਐਲਐਮ-ਆਧਾਰਤਿ ਸੁਧਾਰ ਪ੍ਰਕਰਿਆ ਦਾ ਉਦੇਸ਼ ਜੰਨਿਾ ਸੰਭਵ ਹੋ ਸਕੇ ਉਤਪੰਨ ਕੀਤੇ JSON ਦੀ ਮੂਲ ਜਾਣਕਾਰੀ ਅਤੇ ਇਰਾਦੇ ਨੂੰ ਬਰਕਰਾਰ ਰੱਖਣਾ ਹੈ। ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕਿ ਠੀਕ ਕੀਤਾ JSON ਆਪਣਾ ਸ਼ਬਦੀ ਅਰਥ ਬਰਕਰਾਰ ਰੱਖਦਾ ਹੈ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਸੰਦਰਭ ਵਿੱਚ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।

Olympia ਵਿੱਚ "ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ" ਪਹੁੰਚ ਦੀ ਇਹ ਵਵਿਹਾਰਕ ਇੰਪਲੀਮੈਂਟੇਸ਼ਨ ਸਪੱਸ਼ਟ ਤੌਰ 'ਤੇ ਦਰਸਾਉਂਦੀ ਹੈ ਕਿ AI, ਖਾਸ ਕਰਕੇ ਐਲਐਲਐਮ, ਨੂੰ ਅਸਲ ਦੁਨੀਆਂ ਦੀਆਂ ਡਾਟਾ ਚੁਣੌਤੀਆਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਕਵਿੇਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਮਜ਼ਬੂਤ ਅਤੇ ਕੁਸ਼ਲ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣ ਲਈ ਪਰੰਪਰਾਗਤ ਪ੍ਰੋਗਰਾਮਿੰਗ ਤਕਨੀਕਾਂ ਨੂੰ AI ਸਮਰੱਥਾਵਾਂ ਨਾਲ ਜੋੜਨ ਦੀ ਸ਼ਕਤੀ ਨੂੰ ਪ੍ਰਦਰਸ਼ਤਿ ਕਰਦੀ ਹੈ।

ਪੋਸਟਲ ਦਾ ਨਯਿਮ ਅਤੇ "ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ" ਪੈਟਰਨ

"ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲਾ ਡਾਟਾ," ਜਵਿੇਂ ਕਿ JSONFixer ਕਲਾਸ ਦੁਆਰਾ ਦਰਸਾਇਆ ਗਿਆ ਹੈ, ਪੋਸਟਲ ਦੇ ਨਯਿਮ ਨਾਲ ਚੰਗੀ ਤਰ੍ਹਾਂ ਮੇਲ ਖਾਂਦਾ ਹੈ, ਜਸਿਨੂੰ ਮਜ਼ਬੂਤੀ ਦਾ ਸਧਿਾਂਤ ਵੀ ਕਹਿਾ ਜਾਂਦਾ ਹੈ। ਪੋਸਟਲ ਦਾ ਨਯਿਮ ਕਹਿੰਦਾ ਹੈ:

"ਜੋ ਤੁਸੀਂ ਕਰਦੇ ਹੋ ਉਸ ਵਿੱਚ ਰੂੜ੍ਹੀਵਾਦੀ ਬਣੋ, ਦੂਜਿਆਂ ਤੋਂ ਜੋ ਸਵੀਕਾਰ ਕਰਦੇ ਹੋ ਉਸ ਵਿੱਚ ਉਦਾਰ ਬਣੋ।"

ਇਹ ਸਧਿਾਂਤ, ਜੋ ਮੂਲ ਰੂਪ ਵਿੱਚ ਸੁਰੂਆਤੀ ਇੰਟਰਨੈੱਟ ਦੇ ਪਾਇਨੀਅਰ ਜੌਨ ਪੋਸਟਲ ਦੁਆਰਾ ਦੱਸਿਆ ਗਿਆ

ਸੀ, ਅਜਹਿੀਆਂ ਪ੍ਰਣਾਲੀਆਂ ਬਣਾਉਣ ਦੇ ਮਹੱਤਵ 'ਤੇ ਜੋਰ ਦੰਦਿਾ ਹੈ ਜੋ ਆਉਟਪੁੱਟ ਭੇਜਣ ਸਮੇਂ ਨਰਿਧਾਰਤ ਪ੍ਰੋਟੋਕੋਲਾਂ ਦੀ ਸਖ਼ਤ ਪਾਲਣਾ ਬਣਾਈ ਰੱਖਦੇ ਹੋਏ ਵੱਖ-ਵੱਖ ਜਾਂ ਥੋੜ੍ਹੇ ਗਲਤ ਇਨਪੁੱਟਾਂ ਨੂੰ ਸਹਣਿ ਕਰਨ ਦੇ ਯੋਗ ਹਨ।

"ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ" ਦੇ ਸੰਦਰਭ ਵੰਚਿ, JSONFixer ਕਲਾਸ ਪੋਸਟੇਲ ਦੇ ਨਯਿਮ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ, ਜੋ LLMs ਦੁਆਰਾ ਤਆਿਰ ਕੀਤੇ ਟੁੱਟੇ ਜਾਂ ਅਧੂਰੇ JSON ਡਾਟਾ ਨੂੰ ਸਵੀਕਾਰ ਕਰਨ ਵੰਚਿ ਉਦਾਰ ਹੈ। ਇਹ ਉਦੋਂ ਤੁਰੰਤ ਅਸਵੀਕਾਰ ਜਾਂ ਅਸਫਲ ਨਹੀਂ ਹੁੰਦੀ ਜਦੋਂ ਅਜਹਿੀ JSON ਦਾ ਸਾਹਮਣਾ ਕਰਦੀ ਹੈ ਜੋ ਸਖ਼ਤੀ ਨਾਲ ਉਮੀਦ ਕੀਤੇ ਫਾਰਮੈਟ ਦੀ ਪਾਲਣਾ ਨਹੀਂ ਕਰਦਾ। ਇਸ ਦੀ ਬਜਾਏ, ਇਹ LLMs ਦੀ ਸ਼ਕਤੀ ਦੀ ਵਰਤੋਂ ਕਰਕੇ JSON ਨੂੰ ਠੀਕ ਕਰਨ ਦੀ ਕੋਸ਼ਸ਼ਿ ਕਰਦੀ ਹੈ।

ਅਧੂਰੇ JSON ਨੂੰ ਸਵੀਕਾਰ ਕਰਨ ਵੰਚਿ ਉਦਾਰ ਹੋਣ ਦੁਆਰਾ, JSONFixer ਕਲਾਸ ਮਜ਼ਬੂਤੀ ਅਤੇ ਲਚਕਤਾ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ। ਇਹ ਮੰਨਦੀ ਹੈ ਕ ਿਅਸਲ ਦੁਨੀਆ ਵੰਚਿ ਡਾਟਾ ਅਕਸਰ ਵੱਖ-ਵੱਖ ਰੂਪਾਂ ਵੰਚਿ ਆਉਂਦਾ ਹੈ ਅਤੇ ਹਮੇਸ਼ਾ ਸਖ਼ਤ ਵਸ਼ਿੇਸ਼ਤਾਵਾਂ ਦੇ ਅਨੁਕੂਲ ਨਹੀਂ ਹੋ ਸਕਦਾ। ਇਹਨਾਂ ਵਚਿਲਨਾਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲ ਕੇ ਅਤੇ ਸੁਧਾਰ ਕੇ, ਕਲਾਸ ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦੀ ਹੈ ਕ ਿਐਪਲੀਕੇਸ਼ਨ ਅਧੂਰੇ ਡਾਟਾ ਦੀ ਮੌਜੂਦਗੀ ਵੰਚਿ ਵੀ ਨਰਿਵਘਿਨ ਕੰਮ ਕਰਦੀ ਰਹੇ।

ਦੂਜੇ ਪਾਸੇ, JSONFixer ਕਲਾਸ ਆਉਟਪੁੱਟ ਦੇ ਮਾਮਲੇ ਵੰਚਿ ਪੋਸਟੇਲ ਦੇ ਨਯਿਮ ਦੇ ਰੂੜ੍ਹੀਵਾਦੀ ਪਹਲਿੂ ਦੀ ਵੀ ਪਾਲਣਾ ਕਰਦੀ ਹੈ। LLMs ਦੀ ਵਰਤੋਂ ਕਰਕੇ JSON ਨੂੰ ਠੀਕ ਕਰਨ ਤੋਂ ਬਾਅਦ, ਕਲਾਸ ਸੁਧਾਰੇ ਗਏ JSON ਦੀ ਪੁਸ਼ਟੀ ਕਰਦੀ ਹੈ ਤਾਂ ਜੋ ਇਹ ਯਕੀਨੀ ਬਣਾਇਆ ਜਾ ਸਕੇ ਕ ਿਇਹ ਸਖ਼ਤੀ ਨਾਲ ਉਮੀਦ ਕੀਤੇ ਫਾਰਮੈਟ ਦੇ ਅਨੁਕੂਲ ਹੈ। ਇਹ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਹੋਰ ਹੰਸਿਆਂ ਨੂੰ ਭੇਜਣ ਤੋਂ ਪਹਲਿਾਂ ਡਾਟਾ ਦੀ ਅਖੰਡਤਾ ਅਤੇ ਸ਼ੁੱਧਤਾ ਨੂੰ ਬਣਾਈ ਰੱਖਦੀ ਹੈ। ਇਹ ਰੂੜ੍ਹੀਵਾਦੀ ਪਹੁੰਚ ਗਾਰੰਟੀ ਦੰਦਿੀ ਹੈ ਕ ਿJSONFixer ਕਲਾਸ ਦਾ ਆਉਟਪੁੱਟ ਭਰੋਸੇਯੋਗ ਅਤੇ ਇਕਸਾਰ ਹੈ, ਜੋ ਅੰਤਰ-ਸੰਚਾਲਨ ਯੋਗਤਾ ਨੂੰ ਵਧਾਉਂਦਾ ਹੈ ਅਤੇ ਗਲਤੀਆਂ ਦੇ ਫੈਲਣ ਨੂੰ ਰੋਕਦਾ ਹੈ।

ਜੌਨ ਪੋਸਟੇਲ ਬਾਰੇ ਦਲਿਚਸਪ ਤੱਥ:

- ਜੌਨ ਪੋਸਟੇਲ (1943-1998) ਇੱਕ ਅਮਰੀਕੀ ਕੰਪਊਿਟਰ ਵਗਿਆਨੀ ਸਨ ਜਨ੍ਹਿਾਂ ਨੇ ਇੰਟਰਨੈੱਟ ਦੇ ਵਕਿਾਸ ਵੰਚਿ ਮਹੱਤਵਪੂਰਨ ਭੂਮਕਿਾ ਨਭਾਈ। ਉਹਨਾਂ ਨੂੰ ਬੁਨਆਿਦੀ ਪ੍ਰੋਟੋਕੋਲ ਅਤੇ ਮਆਿਰਾਂ ਵੰਚਿ ਉਹਨਾਂ ਦੇ ਮਹੱਤਵਪੂਰਨ ਯੋਗਦਾਨ ਲਈ "ਇੰਟਰਨੈੱਟ ਦੇ ਭਗਵਾਨ" ਵਜੋਂ ਜਾਣਆਿ ਜਾਂਦਾ ਸੀ।
- ਪੋਸਟੇਲ Request for Comments (RFC) ਦਸਤਾਵੇਜ਼ ਲੜੀ ਦੇ ਸੰਪਾਦਕ ਸਨ, ਜੋ ਇੰਟਰਨੈੱਟ ਬਾਰੇ ਤਕਨੀਕੀ ਅਤੇ ਸੰਗਠਨਾਤਮਕ ਨੋਟਾਂ ਦੀ ਇੱਕ ਲੜੀ ਹੈ। ਉਹਨਾਂ ਨੇ TCP, IP, ਅਤੇ SMTP ਵਰਗੇ ਬੁਨਆਿਦੀ ਪ੍ਰੋਟੋਕੋਲਾਂ ਸਮੇਤ 200 ਤੋਂ ਵੱਧ RFCs ਦੀ ਰਚਨਾ ਜਾਂ ਸਹ-ਿਰਚਨਾ ਕੀਤੀ।

- ਆਪਣੇ ਤਕਨੀਕੀ ਯੋਗਦਾਨਾਂ ਤੋਂ ਇਲਾਵਾ, ਪੋਸਟੇਲ ਆਪਣੇ ਨਮਿਰ ਅਤੇ ਸਹਯੋਗੀ ਪਹੁੰਚ ਲਈ ਜਾਣੇ ਜਾਂਦੇ ਸਨ। ਉਹ ਸਹਮਿਤੀ ਪ੍ਰਾਪਤ ਕਰਨ ਅਤੇ ਇੱਕ ਮਜ਼ਬੂਤ ਅਤੇ ਅੰਤਰ-ਸੰਚਾਲਨਯੋਗ ਨੈੱਟਵਰਕ ਬਣਾਉਣ ਲਈ ਇਕੱਠੇ ਕੰਮ ਕਰਨ ਦੀ ਮਹੱਤਤਾ ਵੱਚਿ ਵਸ਼ਿਵਾਸ ਰੱਖਦੇ ਸਨ।
- ਪੋਸਟੇਲ ਨੇ 1977 ਤੋਂ ਆਪਣੀ ਅਕਾਲ ਮੌਤ 1998 ਤੱਕ University of Southern California (USC) ਦੇ Information Sciences Institute (ISI) ਵੱਚਿ ਕੰਪਊਿਟਰ ਨੈੱਟਵਰਕਸ ਡਵਿੀਜ਼ਨ ਦੇ ਡਾਇਰੈਕਟਰ ਵਜੋਂ ਸੇਵਾ ਕੀਤੀ।
- ਉਹਨਾਂ ਦੇ ਵਸ਼ਿਾਲ ਯੋਗਦਾਨਾਂ ਦੀ ਮਾਨਤਾ ਵੱਚਿ, ਪੋਸਟੇਲ ਨੂੰ 1998 ਵੱਚਿ ਮਰਣ ਉਪਰੰਤ ਪ੍ਰਤਸ਼ਿਠਤਿ ਟਊਿਰੰਗਿ ਪੁਰਸਕਾਰ ਨਾਲ ਸਨਮਾਨਤਿ ਕੀਤਾ ਗਆਿ, ਜਸਿਨੂੰ ਅਕਸਰ "ਕੰਪਊਿਟੰਗਿ ਦਾ ਨੋਬਲ ਪੁਰਸਕਾਰ" ਕਹਿਾ ਜਾਂਦਾ ਹੈ।

`JSONFixer` ਕਲਾਸ ਮਜ਼ਬੂਤੀ, ਲਚਕਤਾ, ਅਤੇ ਅੰਤਰ-ਸੰਚਾਲਨ ਯੋਗਤਾ ਨੂੰ ਵਧਾਵਾ ਦੰਦੀ ਹੈ, ਜੋ ਉਹ ਮੁੱਖ ਕਦਰਾਂ-ਕੀਮਤਾਂ ਸਨ ਜਨ਼ਿਹਾਂ ਨੂੰ ਪੋਸਟੇਲ ਨੇ ਆਪਣੇ ਪੂਰੇ ਕੈਰੀਅਰ ਦੌਰਾਨ ਕਾਇਮ ਰੱਖਆਿ। ਅਜਹਿੀਆਂ ਪ੍ਰਣਾਲੀਆਂ ਬਣਾ ਕੇ ਜੋ ਅਪੂਰਨਤਾਵਾਂ ਪ੍ਰਤੀ ਸਹਣਿਸ਼ੀਲ ਹਨ ਜਦੋਂ ਕ ਿਪ੍ਰੋਟੋਕੋਲਾਂ ਦੀ ਸਖ਼ਤ ਪਾਲਣਾ ਕਰਦੀਆਂ ਹਨ, ਅਸੀ ਅਜਹਿੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾ ਸਕਦੇ ਹਾਂ ਜੋ ਅਸਲ ਦੁਨੀਆ ਦੀਆਂ ਚੁਣੌਤੀਆਂ ਦਾ ਸਾਹਮਣਾ ਕਰਨ ਵੱਚਿ ਵਧੇਰੇ ਲਚਕਦਾਰ ਅਤੇ ਅਨੁਕੂਲ ਹਨ।

ਵਚਿਾਰ ਅਤੇ ਵਰਿੋਧੀ ਸੰਕੇਤ

ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡੇਟਾ ਪਹੁੰਚਾਂ ਦੀ ਲਾਗੂ ਕਰਨ ਯੋਗਤਾ ਪੂਰੀ ਤਰ੍ਹਾਂ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਦੁਆਰਾ ਸੰਭਾਲੇ ਜਾਣ ਵਾਲੇ ਡੇਟਾ ਦੀ ਕਸਿਮ 'ਤੇ ਨਰਿਭਰ ਕਰਦੀ ਹੈ। ਇੱਕ ਕਾਰਨ ਹੈ ਕ ਿਤੁਸੀ ਆਪਣੀ ਐਪਲੀਕੇਸ਼ਨ ਵੱਚਿ ਸਾਰੀਆਂ JSON ਪਾਰਸੰਗਿ ਗਲਤੀਆਂ ਨੂੰ ਆਪਣੇ ਆਪ ਠੀਕ ਕਰਨ ਲਈ `JSON.parse` ਨੂੰ ਸੱਧਿਾ monkeypatch ਨਹੀ ਕਰਨਾ ਚਾਹੋਗੇ: ਸਾਰੀਆਂ ਗਲਤੀਆਂ ਨੂੰ ਆਪਣੇ ਆਪ ਠੀਕ ਨਹੀ ਕੀਤਾ ਜਾ ਸਕਦਾ ਜਾਂ ਕਰਨਾ ਨਹੀ ਚਾਹੀਦਾ।

ਸਵੈ-ਠੀਕ ਹੋਣਾ ਖਾਸ ਤੌਰ 'ਤੇ ਜੋਖਮ ਭਰਆਿ ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਇਹ ਡੇਟਾ ਹੈਂਡਲੰਗਿ ਅਤੇ ਪ੍ਰੋਸੈਸੰਗਿ ਨਾਲ ਸੰਬੰਧਤਿ ਨਯਿਾਮਕ ਜਾਂ ਪਾਲਣਾ ਜ਼ਰੂਰਤਾਂ ਨਾਲ ਜੁੜਆਿ ਹੁੰਦਾ ਹੈ। ਕੁਝ ਉਦਯੋਗਾਂ, ਜਵਿੇਂ ਕ ਿਸਹਿਤ ਸੰਭਾਲ ਅਤੇ ਵੱਤਿ ਵੱਚਿ, ਡੇਟਾ ਸੰਪੂਰਨਤਾ ਅਤੇ ਲੇਖਾ-ਜੋਖਾ ਬਾਰੇ ਇੰਨੇ ਸਖਤ ਨਯਿਮ ਹਨ ਕ ਿਉਚਤਿ ਨਗਿਰਾਨੀ ਜਾਂ ਲੌਗੰਗਿ ਤੋਂ ਬਨਿਾਂ ਕਸਿੇ ਵੀ ਤਰ੍ਹਾਂ ਦੀ "ਕਾਲਾ ਬਕਸਾ" ਡੇਟਾ ਸੁਧਾਰ ਕਰਨਾ ਇਹਨਾਂ ਨਯਿਮਾਂ ਦੀ ਉਲੰਘਣਾ ਕਰ ਸਕਦਾ ਹੈ। ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਬਹੁਤ ਜ਼ਰੂਰੀ ਹੈ ਕ ਿਤੁਹਾਡੇ ਦੁਆਰਾ ਵਕਿਸਤਿ ਕੀਤੀਆਂ ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੀਆਂ ਡੇਟਾ ਤਕਨੀਕਾਂ ਲਾਗੂ ਕਾਨੂੰਨੀ ਅਤੇ ਨਯਿਾਮਕ ਢਾਂਚਆਿਂ ਦੇ ਅਨੁਕੂਲ ਹੋਣ।

ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡੇਟਾ ਤਕਨੀਕਾਂ ਨੂੰ ਲਾਗੂ ਕਰਨਾ, ਖਾਸ ਕਰਕੇ ਏ.ਆਈ. ਮਾਡਲਾਂ ਨਾਲ ਸੰਬੰਧਤਿ, ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਅਤੇ ਸਰੋਤ ਵਰਤੋਂ 'ਤੇ ਵੀ ਵੱਡਾ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦਾ ਹੈ। ਗਲਤੀ ਦੀ ਪਛਾਣ ਅਤੇ ਸੁਧਾਰ ਲਈ ਏ.ਆਈ. ਮਾਡਲਾਂ ਰਾਹੀ ਵੱਡੀ ਮਾਤਰਾ ਵਚਿ ਡੇਟਾ ਦੀ ਪ੍ਰੋਸੈਸਿੰਗ ਕੰਪਊਿਟਰ ਸਰੋਤਾਂ ਦੀ ਗਹਨਿ ਵਰਤੋਂ ਕਰ ਸਕਦੀ ਹੈ। ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡੇਟਾ ਦੇ ਫਾਇਦਆਿਂ ਅਤੇ ਸੰਬੰਧਤਿ ਕਾਰਗੁਜ਼ਾਰੀ ਅਤੇ ਸਰੋਤ ਲਾਗਤਾਂ ਵਚਿਕਾਰ ਸੰਤੁਲਨ ਦਾ ਮੁਲਾਂਕਣ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ।

ਇਹ ਕਹਾਂ ਕੇ, ਆਓ ਇਸ ਸ਼ਕਤੀਸ਼ਾਲੀ ਪਹੁੰਚ ਨੂੰ ਕਦੋਂ ਅਤੇ ਕਿੱਥੇ ਲਾਗੂ ਕਰਨਾ ਹੈ, ਇਸ ਬਾਰੇ ਫੈਸਲਾ ਕਰਨ ਵਚਿ ਸ਼ਾਮਲ ਕਾਰਕਾਂ ਵਚਿ ਡੂੰਘਾਈ ਨਾਲ ਜਾਈਏ।

ਡੇਟਾ ਮਹੱਤਤਾ

ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡੇਟਾ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ 'ਤੇ ਵਚਿਾਰ ਕਰਦੇ ਸਮੇਂ, ਪ੍ਰੋਸੈਸ ਕੀਤੇ ਜਾ ਰਹੇ ਡੇਟਾ ਦੀ ਮਹੱਤਤਾ ਦਾ ਮੁਲਾਂਕਣ ਕਰਨਾ ਬਹੁਤ ਜ਼ਰੂਰੀ ਹੈ। ਮਹੱਤਤਾ ਦਾ ਪੱਧਰ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਅਤੇ ਇਸਦੇ ਵਪਾਰਕ ਖੇਤਰ ਦੇ ਸੰਦਰਭ ਵਚਿ ਡੇਟਾ ਦੀ ਮਹੱਤਤਾ ਅਤੇ ਸੰਵੇਦਨਸ਼ੀਲਤਾ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

ਕੁਝ ਮਾਮਲਆਿਂ ਵਚਿ, ਡੇਟਾ ਗਲਤੀਆਂ ਨੂੰ ਆਪਣੇ ਆਪ ਠੀਕ ਕਰਨਾ ਢੁਕਵਾਂ ਨਹੀਂ ਹੋ ਸਕਦਾ, ਖਾਸ ਕਰਕੇ ਜੇਕਰ ਡੇਟਾ ਬਹੁਤ ਸੰਵੇਦਨਸ਼ੀਲ ਹੈ ਜਾਂ ਇਸਦੇ ਕਾਨੂੰਨੀ ਪ੍ਰਭਾਵ ਹਨ। ਉਦਾਹਰਣ ਲਈ, ਹੇਠ ਲਖਿੀਆਂ ਸਥਤਿੀਆਂ 'ਤੇ ਵਚਿਾਰ ਕਰੋ:

1. **ਵੱਤਿੀ ਲੈਣ-ਦੇਣ:** ਵੱਤਿੀ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਚਿ, ਜਵਿੇਂ ਕਿ ਬੈਂਕਿੰਗ ਸਸਿਟਮ ਜਾਂ ਟ੍ਰੇਡਿੰਗ ਪਲੇਟਫਾਰਮ, ਡੇਟਾ ਦੀ ਸ਼ੁੱਧਤਾ ਸਭ ਤੋਂ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਵੱਤਿੀ ਡੇਟਾ ਵਚਿ ਛੋਟੀਆਂ ਗਲਤੀਆਂ ਵੀ ਮਹੱਤਵਪੂਰਨ ਨਤੀਜੇ ਦੇ ਸਕਦੀਆਂ ਹਨ, ਜਵਿੇਂ ਕਿ ਗਲਤ ਖਾਤਾ ਬਕਾਇਆ, ਗਲਤ ਦਸਿ਼ਾ ਵਚਿ ਭੇਜੇ ਗਏ ਫੰਡ, ਜਾਂ ਗਲਤ ਟ੍ਰੇਡਿੰਗ ਫੈਸਲੇ। ਇਹਨਾਂ ਮਾਮਲਆਿਂ ਵਚਿ, ਪੂਰੀ ਜਾਂਚ ਅਤੇ ਲੇਖਾ-ਜੋਖਾ ਤੋਂ ਬਨਿਾਂ ਆਟੋਮੈਟਕਿ ਸੁਧਾਰ ਅਸਵੀਕਾਰਯੋਗ ਜੋਖਮਾਂ ਨੂੰ ਪੇਸ਼ ਕਰ ਸਕਦੇ ਹਨ।
2. **ਮੈਡੀਕਲ ਰਕਿਾਰਡ:** ਸਹਤਿ ਸੰਭਾਲ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਹੁਤ ਸੰਵੇਦਨਸ਼ੀਲ ਅਤੇ ਗੁਪਤ ਮਰੀਜ਼ ਡੇਟਾ ਨਾਲ ਨਜਾਂਠਿਦੀਆਂ ਹਨ। ਮੈਡੀਕਲ ਰਕਿਾਰਡਾਂ ਵਚਿ ਅਸ਼ੁੱਧੀਆਂ ਮਰੀਜ਼ ਦੀ ਸੁਰੱਖਆਿ ਅਤੇ ਇਲਾਜ ਦੇ ਫੈਸਲਆਿਂ 'ਤੇ ਗੰਭੀਰ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦੀਆਂ ਹਨ। ਯੋਗ ਸਹਤਿ ਸੰਭਾਲ ਪੇਸ਼ੇਵਰਾਂ ਦੁਆਰਾ ਉਚਤਿ ਨਗਿਰਾਨੀ ਅਤੇ ਪ੍ਰਮਾਣੀਕਰਨ ਤੋਂ ਬਨਿਾਂ ਮੈਡੀਕਲ ਡੇਟਾ ਨੂੰ ਆਪਣੇ ਆਪ ਸੋਧਣਾ ਨਯਿਾਮਕ ਜ਼ਰੂਰਤਾਂ ਦੀ ਉਲੰਘਣਾ ਕਰ ਸਕਦਾ ਹੈ ਅਤੇ ਮਰੀਜ਼ ਦੀ ਭਲਾਈ ਨੂੰ ਜੋਖਮ ਵਚਿ ਪਾ ਸਕਦਾ ਹੈ।
3. **ਕਾਨੂੰਨੀ ਦਸਤਾਵੇਜ਼:** ਕਾਨੂੰਨੀ ਦਸਤਾਵੇਜ਼ਾਂ, ਜਵਿੇਂ ਕਿ ਇਕਰਾਰਨਾਮੇ, ਸਮਝੌਤੇ, ਜਾਂ ਅਦਾਲਤੀ ਫਾਈਲਿੰਗ ਨੂੰ ਸੰਭਾਲਣ ਵਾਲੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਸਖ਼ਤ ਸ਼ੁੱਧਤਾ ਅਤੇ ਅਖੰਡਤਾ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਕਾਨੂੰਨੀ ਡੇਟਾ ਵਚਿ

ਛੋਟੀਆਂ ਗਲਤੀਆਂ ਵੀ ਮਹੱਤਵਪੂਰਨ ਕਾਨੂੰਨੀ ਨਤੀਜੇ ਦੇ ਸਕਦੀਆਂ ਹਨ। ਇਸ ਖੇਤਰ ਵਿੱਚ ਆਟੋਮੈਟਿਡ ਸੁਧਾਰ ਢੁਕਵੇਂ ਨਹੀਂ ਹੋ ਸਕਦੇ, ਕਿਉਂਕਿ ਡੇਟਾ ਨੂੰ ਅਕਸਰ ਇਸਦੀ ਵੈਧਤਾ ਅਤੇ ਲਾਗੂ ਕਰਨ ਯੋਗਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਕਾਨੂੰਨੀ ਮਾਹਰਿਾਂ ਦੁਆਰਾ ਮੈਨੂਅਲ ਸਮੀਖਿਆ ਅਤੇ ਪ੍ਰਮਾਣੀਕਰਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

ਇਹਨਾਂ ਮਹੱਤਵਪੂਰਨ ਡਾਟਾ ਸਬਤਿੀਆਂ ਵਿੱਚ, ਸਵੈਚਾਲਤਿ ਸੁਧਾਰਾਂ ਨਾਲ ਜੁੜੇ ਜੋਖਮ ਅਕਸਰ ਸੰਭਾਵੀ ਲਾਭਾਂ ਤੋਂ ਵੱਧ ਹੁੰਦੇ ਹਨ। ਗਲਤੀਆਂ ਦੀ ਸ਼ੁਰੂਆਤ ਜਾਂ ਡਾਟਾ ਨੂੰ ਗਲਤ ਢੰਗ ਨਾਲ ਸੋਧਣ ਦੇ ਨਤੀਜੇ ਗੰਭੀਰ ਹੋ ਸਕਦੇ ਹਨ, ਜਸਿ ਨਾਲ ਵਤਿੀ ਨੁਕਸਾਨ, ਕਾਨੂੰਨੀ ਜ਼ੰਮਿਵਾਰੀਆਂ, ਜਾਂ ਇੱਥੋਂ ਤੱਕ ਕਿ ਵਅਿਕਤੀਆਂ ਨੂੰ ਨੁਕਸਾਨ ਹੋ ਸਕਦਾ ਹੈ।

ਬਹੁਤ ਮਹੱਤਵਪੂਰਨ ਡਾਟਾ ਨਾਲ ਨਜਂਠਿਦੇ ਸਮੇਂ, ਮੈਨੂਅਲ ਤਸਦੀਕ ਅਤੇ ਪ੍ਰਮਾਣੀਕਰਨ ਪ੍ਰਕਰਿਆਵਾਂ ਨੂੰ ਤਰਜੀਹ ਦੇਣਾ ਜ਼ਰੂਰੀ ਹੈ। ਡਾਟਾ ਦੀ ਸ਼ੁੱਧਤਾ ਅਤੇ ਅਖੰਡਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਵਿੱਚ ਮਨੁੱਖੀ ਨਗਿਰਾਨੀ ਅਤੇ ਮੁਹਾਰਤ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਸੰਭਾਵੀ ਗਲਤੀਆਂ ਜਾਂ ਅਸੰਗਤੀਆਂ ਨੂੰ ਚੰਨ੍ਹਿਤਿ ਕਰਨ ਲਈ ਸਵੈਚਾਲਤਿ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ, ਪਰ ਸੁਧਾਰਾਂ ਬਾਰੇ ਅੰਤਮਿ ਫੈਸਲੇ ਵਿੱਚ ਮਨੁੱਖੀ ਫੈਸਲੇ ਅਤੇ ਮਨਜ਼ੂਰੀ ਸ਼ਾਮਲ ਹੋਣੀ ਚਾਹੀਦੀ ਹੈ।

ਹਾਲਾਂਕਿ, ਇਹ ਨੋਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਕਸਿੇ ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ ਸਾਰਾ ਡਾਟਾ ਇੱਕੋ ਜਹਿੀ ਮਹੱਤਤਾ ਦਾ ਨਹੀਂ ਹੋ ਸਕਦਾ। ਇੱਕੋ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਅੰਦਰ, ਡਾਟਾ ਦੇ ਅਜਹਿੇ ਉਪ-ਸਮੂਹ ਹੋ ਸਕਦੇ ਹਨ ਜੋ ਘੱਟ ਸੰਵੇਦਨਸ਼ੀਲ ਹਨ ਜਾਂ ਗਲਤੀਆਂ ਹੋਣ 'ਤੇ ਘੱਟ ਪ੍ਰਭਾਵ ਪੈਦਾ ਹੈ। ਅਜਹਿੇ ਮਾਮਲਿਆਂ ਵਿੱਚ, ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਨੂੰ ਚੁਣਵੇਂ ਤੌਰ 'ਤੇ ਉਹਨਾਂ ਵਸ਼ਿਸ਼ ਡਾਟਾ ਉਪ-ਸਮੂਹਾਂ 'ਤੇ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਜਦੋਂ ਕਿ ਮਹੱਤਵਪੂਰਨ ਡਾਟਾ ਮੈਨੂਅਲ ਤਸਦੀਕ ਦੇ ਅਧੀਨ ਰਹਂਦਿਾ ਹੈ।

ਮੁੱਖ ਗੱਲ ਇਹ ਹੈ ਕਿ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ ਹਰੇਕ ਡਾਟਾ ਸ਼੍ਰੇਣੀ ਦੀ ਮਹੱਤਤਾ ਦਾ ਧਿਆਨ ਨਾਲ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇ ਅਤੇ ਸੰਬੰਧਤਿ ਜੋਖਮਾਂ ਅਤੇ ਪ੍ਰਭਾਵਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਸੁਧਾਰਾਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਸਪਸ਼ਟ ਦਸ਼ਿਾ-ਨਰਿਦੇਸ਼ ਅਤੇ ਪ੍ਰਕਰਿਆਵਾਂ ਨਰਿਧਾਰਤ ਕੀਤੀਆਂ ਜਾਣ। ਮਹੱਤਵਪੂਰਨ (ਯਾਨੀ ਲੇਖਾ-ਬਹੀਆਂ, ਮੈਡੀਕਲ ਰਕਿਾਰਡ) ਅਤੇ ਗੈਰ-ਮਹੱਤਵਪੂਰਨ ਡਾਟਾ (ਯਾਨੀ ਡਾਕ ਪਤੇ, ਸਰੋਤ ਚੇਤਾਵਨੀਆਂ) ਵਿੱਚ ਅੰਤਰ ਕਰਕੇ, ਤੁਸੀਂ ਢੁਕਵੇਂ ਸਥਾਨਾਂ 'ਤੇ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਦੇ ਲਾਭਾਂ ਦਾ ਫਾਇਦਾ ਲੈਣ ਅਤੇ ਜਂਥੇ ਜ਼ਰੂਰੀ ਹੋਵੇ ਸਖ਼ਤ ਨਯਿੰਤਰਣ ਅਤੇ ਨਗਿਰਾਨੀ ਬਣਾਈ ਰੱਖਣ ਵਿੱਚ ਸੰਤੁਲਨ ਬਣਾ ਸਕਦੇ ਹੋ।

ਅੰਤ ਵਿੱਚ, ਮਹੱਤਵਪੂਰਨ ਡਾਟਾ 'ਤੇ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਫੈਸਲਾ ਖੇਤਰ ਦੇ ਮਾਹਰਿਾਂ, ਕਾਨੂੰਨੀ ਸਲਾਹਕਾਰਾਂ, ਅਤੇ ਹੋਰ ਸੰਬੰਧਤਿ ਹਸ਼ਿੇਦਾਰਾਂ ਨਾਲ ਸਲਾਹ-ਮਸ਼ਵਰੇ ਨਾਲ ਲਿਆ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ। ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਡਾਟਾ ਨਾਲ ਜੁੜੀਆਂ ਵਸ਼ਿਸ਼ ਲੋੜਾਂ, ਨਯਿਮਾਂ, ਅਤੇ ਜੋਖਮਾਂ 'ਤੇ ਵਚਿਾਰ ਕਰਨਾ ਅਤੇ ਡਾਟਾ ਸੁਧਾਰ ਰਣਨੀਤੀਆਂ ਨੂੰ ਉਸ ਅਨੁਸਾਰ ਅਨੁਕੂਲ ਕਰਨਾ ਜ਼ਰੂਰੀ ਹੈ।

ਗਲਤੀ ਦੀ ਗੰਭੀਰਤਾ

ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਨੂੰ ਲਾਗੂ ਕਰਦੇ ਸਮੇਂ, ਡਾਟਾ ਗਲਤੀਆਂ ਦੀ ਗੰਭੀਰਤਾ ਅਤੇ ਪ੍ਰਭਾਵ ਦਾ ਮੁਲਾਂਕਣ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਸਾਰੀਆਂ ਗਲਤੀਆਂ ਬਰਾਬਰ ਨਹੀਂ ਹੁੰਦੀਆਂ, ਅਤੇ ਢੁਕਵੀਂ ਕਾਰਵਾਈ ਮੁੱਦੇ ਦੀ ਗੰਭੀਰਤਾ 'ਤੇ ਨਿਰਭਰ ਕਰਦੇ ਹੋਏ ਵੱਖ-ਵੱਖ ਹੋ ਸਕਦੀ ਹੈ।

ਮਾਮੂਲੀ ਅਸੰਗਤੀਆਂ ਜਾਂ ਫਾਰਮੈਟਿੰਗ ਮੁੱਦੇ ਸਵੈਚਾਲਤਿ ਸੁਧਾਰ ਲਈ ਢੁਕਵੇਂ ਹੋ ਸਕਦੇ ਹਨ। ਉਦਾਹਰਨ ਲਈ, ਟੁੱਟੇ ਹੋਏ JSON ਨੂੰ ਠੀਕ ਕਰਨ ਲਈ ਨਯਿੁਕਤ ਕੀਤਾ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲਾ ਡਾਟਾ ਵਰਕਰ ਗੁੰਮ ਕਾਮਿਆਂ ਜਾਂ ਅਣ-ਐਸਕੇਪਡ ਡਬਲ ਕੋਟਸ ਨੂੰ ਡਾਟਾ ਦੇ ਅਰਥ ਜਾਂ ਢਾਂਚੇ ਨੂੰ ਮਹੱਤਵਪੂਰਨ ਤੌਰ 'ਤੇ ਬਦਲੇ ਬਨਿਾਂ ਸੰਭਾਲ ਸਕਦਾ ਹੈ। ਇਸ ਕਸਿਮ ਦੀਆਂ ਗਲਤੀਆਂ ਨੂੰ ਅਕਸਰ ਸਧਿਾ ਠੀਕ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਅਤੇ ਸਮੁੱਚੀ ਡਾਟਾ ਅਖੰਡਤਾ 'ਤੇ ਘੱਟੋ-ਘੱਟ ਪ੍ਰਭਾਵ ਪੈਂਦਾ ਹੈ।

ਹਾਲਾਂਕਿ, ਵਧੇਰੇ ਗੰਭੀਰ ਗਲਤੀਆਂ ਜੋ ਮੂਲ ਰੂਪ ਵਚਿ ਡਾਟਾ ਦੇ ਅਰਥ ਜਾਂ ਅਖੰਡਤਾ ਨੂੰ ਬਦਲਦੀਆਂ ਹਨ, ਉਹਨਾਂ ਲਈ ਵੱਖਰੇ ਪਹੁੰਚ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ। ਅਜਹਿੇ ਮਾਮਲਆਿਂ ਵਚਿ, ਸਵੈਚਾਲਤਿ ਸੁਧਾਰ ਕਾਫ਼ੀ ਨਹੀਂ ਹੋ ਸਕਦੇ, ਅਤੇ ਡਾਟਾ ਦੀ ਸ਼ੁੱਧਤਾ ਅਤੇ ਵੈਧਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਮਨੁੱਖੀ ਦਖਲਅੰਦਾਜ਼ੀ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ।

ਇੱਥੇ ਗਲਤੀ ਦੀ ਗੰਭੀਰਤਾ ਨਰਿਧਾਰਤਿ ਕਰਨ ਲਈ ਏ.ਆਈ. ਦੀ ਵਰਤੋਂ ਦੀ ਧਾਰਨਾ ਸਾਹਮਣੇ ਆਉਂਦੀ ਹੈ। ਏ.ਆਈ. ਮਾਡਲਾਂ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਦਾ ਲਾਭ ਲੈਂਦੇ ਹੋਏ, ਅਸੀਂ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਵਰਕਰ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਾਂ ਜੋ ਨਾ ਸਰਫ਼ਿ ਗਲਤੀਆਂ ਨੂੰ ਠੀਕ ਕਰਦੇ ਹਨ, ਬਲਕਿ ਉਨ੍ਹਾਂ ਗਲਤੀਆਂ ਦੀ ਗੰਭੀਰਤਾ ਦਾ ਮੁਲਾਂਕਣ ਵੀ ਕਰਦੇ ਹਨ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਸੰਭਾਲਣ ਦੇ ਤਰੀਕੇ ਬਾਰੇ ਸੂਝਵਾਨ ਫੈਸ਼ਲੇ ਲੈਂਦੇ ਹਨ।

ਉਦਾਹਰਣ ਵਜੋਂ, ਆਓ ਇੱਕ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਵਰਕਰ 'ਤੇ ਵਚਿਾਰ ਕਰੀਏ ਜੋ ਗਾਹਕ ਡਾਟਾਬੇਸ ਵਚਿ ਆ ਰਹੇ ਡਾਟਾ ਵਚਿ ਅਸੰਗਤੀਆਂ ਨੂੰ ਠੀਕ ਕਰਨ ਲਈ ਜ਼ਮਿੇਵਾਰ ਹੈ। ਵਰਕਰ ਨੂੰ ਡਾਟਾ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਨ ਅਤੇ ਸੰਭਾਵੀ ਗਲਤੀਆਂ ਦੀ ਪਛਾਣ ਕਰਨ ਲਈ ਤਆਿਰ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਜਵਿੇਂ ਕ ਿਗੁੰਮ ਜਾਂ ਵਰਿੋਧੀ ਜਾਣਕਾਰੀ। ਹਾਲਾਂਕਿ, ਸਾਰੀਆਂ ਗਲਤੀਆਂ ਨੂੰ ਆਪਣੇ ਆਪ ਠੀਕ ਕਰਨ ਦੀ ਬਜਾਏ, ਵਰਕਰ ਨੂੰ ਵਾਧੂ ਟੂਲ ਕਾਲਾਂ ਨਾਲ ਲੈਸ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਇਸਨੂੰ ਮਨੁੱਖੀ ਸਮੀਖਆਿ ਲਈ ਗੰਭੀਰ ਗਲਤੀਆਂ ਨੂੰ ਫਲੈਗ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦੀਆਂ ਹਨ।

ਇੱਥੇ ਇੱਕ ਉਦਾਹਰਣ ਹੈ ਕ ਿਇਸਨੂੰ ਕਵਿੇਂ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ:

```ruby
class CustomerDataReviewer
  include Raix::ChatCompletion
  include Raix::FunctionDeclarations

  attr_accessor :customer

  function :flag_for_review, reason: { type: "string" } do |params|
    AdminNotifier.review_request(customer, params[:reason])
  end

  def initialize(customer)
    self.customer = customer
  end

  def call(customer_data)
    transcript << {
      system: "You are a customer data reviewer. Your task is to identify
        and correct inconsistencies in customer data.

        < additional instructions here... >

        If you encounter severe errors that require human review, use the
        `flag_for_review` tool to flag the data for manual intervention." }

    transcript << { user: customer.to_json }
    transcript << { assistant: "Reviewed/corrected data:\n```json\n" }

    self.stop = ["```"]

    chat_completion(json: true).then do |result|
      return if result.blank?

      customer.update(result)
    end
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵੱਚਿ, `CustomerDataHealer` ਵਰਕਰ ਨੂੰ ਗਾਹਕ ਡਾਟਾ ਵੱਚਿ ਅਸੰਗਤੀਆਂ ਦੀ ਪਛਾਣ ਅਤੇ ਸੁਧਾਰ ਕਰਨ ਲਈ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ। ਇੱਕ ਵਾਰ ਫਿਰ, ਅਸੀਂ ਢਾਂਚਾਗਤ ਆਉਟਪੁੱਟ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ Response Fencing ਅਤੇ Ventriloquist ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਖਾਸ ਤੌਰ 'ਤੇ, ਵਰਕਰ ਦੇ ਸਸਿਟਮ

ਨਿਰਦੇਸ਼ਾਂ ਵਿੱਚ ਗੰਭੀਰ ਗਲਤੀਆਂ ਮਿਲਣ 'ਤੇ `flag_for_review` ਫੰਕਸ਼ਨ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦੀਆਂ ਹਦਾਇਤਾਂ ਸ਼ਾਮਲ ਹਨ।

ਜਦੋਂ ਵਰਕਰ ਗਾਹਕ ਡਾਟਾ ਨੂੰ ਪ੍ਰੋਸੈਸ ਕਰਦਾ ਹੈ, ਇਹ ਡਾਟਾ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਦਾ ਹੈ ਅਤੇ ਕਿਸੇ ਵੀ ਅਸੰਗਤੀਆਂ ਨੂੰ ਠੀਕ ਕਰਨ ਦੀ ਕੋਸ਼ਿਸ਼ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਵਰਕਰ ਨਿਰਧਾਰਤ ਕਰਦਾ ਹੈ ਕਿ ਗਲਤੀਆਂ ਗੰਭੀਰ ਹਨ ਅਤੇ ਮਨੁੱਖੀ ਦਖਲ ਦੀ ਲੋੜ ਹੈ, ਤਾਂ ਇਹ ਡਾਟਾ ਨੂੰ ਫਲੈਗ ਕਰਨ ਅਤੇ ਫਲੈਗਿੰਗ ਦਾ ਕਾਰਨ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ `flag_-for_review` ਟੂਲ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦਾ ਹੈ।

`chat_completion` ਮੈਥਡ ਨੂੰ `json: true` ਨਾਲ ਕਾਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਜੋ ਸੁਧਾਰੇ ਗਏ ਗਾਹਕ ਡਾਟਾ ਨੂੰ JSON ਵਜੋਂ ਪਾਰਸ ਕੀਤਾ ਜਾ ਸਕੇ। ਫੰਕਸ਼ਨ ਕਾਲ ਤੋਂ ਬਾਅਦ ਲੂਪਿੰਗ ਲਈ ਕੋਈ ਪ੍ਰਬੰਧ ਨਹੀਂ ਹੈ, ਇਸ ਲਈ ਜੇਕਰ `flag_for_review` ਨੂੰ ਬੁਲਾਇਆ ਗਿਆ ਸੀ ਤਾਂ ਨਤੀਜਾ ਖਾਲੀ ਹੋਵੇਗਾ। ਨਹੀਂ ਤਾਂ, ਗਾਹਕ ਨੂੰ ਸਮੀਖਿਆ ਕੀਤੇ ਅਤੇ ਸੰਭਾਵਿਤ ਤੌਰ 'ਤੇ ਸੁਧਾਰੇ ਗਏ ਡਾਟਾ ਨਾਲ ਅੱਪਡੇਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਗਲਤੀ ਦੀ ਗੰਭੀਰਤਾ ਦੇ ਮੁਲਾਂਕਣ ਅਤੇ ਮਨੁੱਖੀ ਸਮੀਖਿਆ ਲਈ ਡਾਟਾ ਨੂੰ ਫਲੈਗ ਕਰਨ ਦੇ ਵਿਕਲਪ ਨੂੰ ਸ਼ਾਮਲ ਕਰਕੇ, ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲਾ ਡਾਟਾ ਵਰਕਰ ਵਧੇਰੇ ਬੁੱਧੀਮਾਨ ਅਤੇ ਅਨੁਕੂਲ ਬਣ ਜਾਂਦਾ ਹੈ। ਇਹ ਛੋਟੀਆਂ ਗਲਤੀਆਂ ਨੂੰ ਆਪਣੇ ਆਪ ਸੰਭਾਲ ਸਕਦਾ ਹੈ ਜਦੋਂ ਕਿ ਗੰਭੀਰ ਗਲਤੀਆਂ ਨੂੰ ਮੈਨੂਅਲ ਦਖਲ ਲਈ ਮਨੁੱਖੀ ਮਾਹਰਾਂ ਕੋਲ ਭੇਜਿਆ ਜਾ ਸਕਦਾ ਹੈ।

ਗਲਤੀ ਦੀ ਗੰਭੀਰਤਾ ਨਿਰਧਾਰਤ ਕਰਨ ਲਈ ਵਿਸ਼ਿਸ਼ ਮਾਪਦੰਡ ਡੋਮੇਨ ਗਿਆਨ ਅਤੇ ਵਪਾਰਕ ਲੋੜਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਵਰਕਰ ਦੇ ਨਿਰਦੇਸ਼ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਗੰਭੀਰਤਾ ਦਾ ਮੁਲਾਂਕਣ ਕਰਦੇ ਸਮੇਂ ਡਾਟਾ ਸੰਪੂਰਨਤਾ 'ਤੇ ਪ੍ਰਭਾਵ, ਡਾਟਾ ਦੇ ਨੁਕਸਾਨ ਜਾਂ ਖਰਾਬੀ ਦੀ ਸੰਭਾਵਨਾ, ਅਤੇ ਗਲਤ ਡਾਟਾ ਦੇ ਨਤੀਜਿਆਂ ਵਰਗੇ ਕਾਰਕਾਂ 'ਤੇ ਵਿਚਾਰ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।

ਗਲਤੀ ਦੀ ਗੰਭੀਰਤਾ ਦਾ ਮੁਲਾਂਕਣ ਕਰਨ ਲਈ AI ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਅਤੇ ਮਨੁੱਖੀ ਦਖਲ ਲਈ ਵਿਕਲਪ ਪ੍ਰਦਾਨ ਕਰਕੇ, ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਸਵੈਚਾਲਨ ਅਤੇ ਡਾਟਾ ਸ਼ੁੱਧਤਾ ਬਣਾਈ ਰੱਖਣ ਵਿਚਕਾਰ ਸੰਤੁਲਨ ਬਣਾ ਸਕਦੀਆਂ ਹਨ। ਇਹ ਪਹੁੰਚ ਯਕੀਨੀ ਬਣਾਉਂਦੀ ਹੈ ਕਿ ਛੋਟੀਆਂ ਗਲਤੀਆਂ ਨੂੰ ਕੁਸ਼ਲਤਾ ਨਾਲ ਠੀਕ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਕਿ ਗੰਭੀਰ ਗਲਤੀਆਂ ਨੂੰ ਮਨੁੱਖੀ ਸਮੀਖਿਅਕਾਂ ਤੋਂ ਜ਼ਰੂਰੀ ਧਿਆਨ ਅਤੇ ਮੁਹਾਰਤ ਮਿਲਦੀ ਹੈ।

ਖੇਤਰ ਦੀ ਜਟਿਲਤਾ

ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ 'ਤੇ ਵਿਚਾਰ ਕਰਦੇ ਸਮੇਂ, ਡਾਟਾ ਖੇਤਰ ਦੀ ਜਟਿਲਤਾ ਅਤੇ ਇਸਦੇ ਢਾਂਚੇ ਅਤੇ ਸੰਬੰਧਾਂ ਨੂੰ ਨਿਯੰਤਰਿਤ ਕਰਨ ਵਾਲੇ ਨਿਯਮਾਂ ਦਾ ਮੁਲਾਂਕਣ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਖੇਤਰ ਦੀ

ਜਟਲਿਤਾ ਸਵੈਚਲਤਿ ਡਾਟਾ ਸੁਧਾਰ ਪਹੁੰਚਾਂ ਦੀ ਪ੍ਰਭਾਵਸ਼ੀਲਤਾ ਅਤੇ ਵਵਿਹਾਰਕਤਾ 'ਤੇ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦੀ ਹੈ।

ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਉਦੋਂ ਚੰਗੀ ਤਰ੍ਹਾਂ ਕੰਮ ਕਰਦੀਆਂ ਹਨ ਜਦੋਂ ਡਾਟਾ ਚੰਗੀ ਤਰ੍ਹਾਂ ਪਰਭਿਾਸ਼ਤਿ ਪੈਟਰਨਾਂ ਅਤੇ ਪਾਬੰਦੀਆਂ ਦੀ ਪਾਲਣਾ ਕਰਦਾ ਹੈ। ਉਹਨਾਂ ਖੇਤਰਾਂ ਵੱਚਿ ਜੱਥੇ ਡਾਟਾ ਢਾਂਚਾ ਤੁਲਨਾਤਮਕ ਤੌਰ 'ਤੇ ਸਧਾਰਨ ਹੈ ਅਤੇ ਡਾਟਾ ਤੱਤਾਂ ਵਚਿਕਾਰ ਸੰਬੰਧ ਸਾਂਧੇ ਹਨ, ਸਵੈਚਲਤਿ ਸੁਧਾਰ ਉੱਚ ਵਸ਼ਿਵਾਸ ਨਾਲ ਲਾਗੂ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਉਦਾਹਰਣ ਲਈ, ਫਾਰਮੈਟਿੰਗ ਮੁੱਦਆਿਂ ਨੂੰ ਠੀਕ ਕਰਨਾ ਜਾਂ ਬੁਨਆਿਦੀ ਡਾਟਾ ਕਸਿਮ ਦੀਆਂ ਪਾਬੰਦੀਆਂ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਅਕਸਰ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਵਰਕਰਾਂ ਦੁਆਰਾ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਸੰਭਾਲਆਿ ਜਾ ਸਕਦਾ ਹੈ।

ਹਾਲਾਂਕ, ਜਵਿੇਂ-ਜਵਿੇਂ ਡਾਟਾ ਖੇਤਰ ਦੀ ਜਟਲਿਤਾ ਵਧਦੀ ਹੈ, ਸਵੈਚਲਤਿ ਡਾਟਾ ਸੁਧਾਰ ਨਾਲ ਜੁੜੀਆਂ ਚੁਣੌਤੀਆਂ ਵੀ ਵਧਦੀਆਂ ਹਨ। ਜਟਲਿ ਵਪਾਰਕ ਤਰਕ, ਡਾਟਾ ਇਕਾਈਆਂ ਵਚਿਕਾਰ ਗੁੰਝਲਦਾਰ ਸਬੰਧਾਂ, ਜਾਂ ਖੇਤਰ-ਵਸ਼ਿਸ਼ ਨਯਿਮਾਂ ਅਤੇ ਅਪਵਾਦਾਂ ਵਾਲੇ ਖੇਤਰਾਂ ਵੱਚਿ, ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਹਮੇਸ਼ਾ ਬਾਰੀਕੀਆਂ ਨੂੰ ਨਹੀਂ ਸਮਝ ਸਕਦੀਆਂ ਅਤੇ ਅਣਚਾਹੇ ਨਤੀਜੇ ਪੈਦਾ ਕਰ ਸਕਦੀਆਂ ਹਨ।

ਆਓ ਇੱਕ ਜਟਲਿ ਖੇਤਰ ਦੀ ਉਦਾਹਰਣ ਲਈਏ: ਇੱਕ ਵੱਤੀ ਵਪਾਰ ਪ੍ਰਣਾਲੀ। ਇਸ ਖੇਤਰ ਵੱਚਿ, ਡਾਟਾ ਵੱਚਿ ਵੱਖ-ਵੱਖ ਵੱਤੀ ਸਾਧਨ, ਮਾਰਕੀਟ ਡਾਟਾ, ਵਪਾਰਕ ਨਯਿਮ, ਅਤੇ ਨਯਿਾਮਕ ਲੋੜਾਂ ਸ਼ਾਮਲ ਹੁੰਦੀਆਂ ਹਨ। ਵੱਖ-ਵੱਖ ਡਾਟਾ ਤੱਤਾਂ ਵਚਿਕਾਰ ਸਬੰਧ ਗੁੰਝਲਦਾਰ ਹੋ ਸਕਦੇ ਹਨ, ਅਤੇ ਡਾਟਾ ਦੀ ਵੈਧਤਾ ਅਤੇ ਸਥਰਿਤਾ ਨੂੰ ਨਯਿੰਤਰਤਿ ਕਰਨ ਵਾਲੇ ਨਯਿਮ ਖੇਤਰ-ਵਸ਼ਿਸ਼ ਹੋ ਸਕਦੇ ਹਨ।

ਅਜਹਿੇ ਜਟਲਿ ਖੇਤਰ ਵੱਚਿ, ਵਪਾਰ ਡਾਟਾ ਵੱਚਿ ਅਸੰਗਤੀਆਂ ਨੂੰ ਠੀਕ ਕਰਨ ਲਈ ਨਯੁਕਤ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਵਰਕਰ ਨੂੰ ਖੇਤਰ-ਵਸ਼ਿਸ਼ ਨਯਿਮਾਂ ਅਤੇ ਸੀਮਾਵਾਂ ਦੀ ਗਹਰਿੀ ਸਮਝ ਦੀ ਲੋੜ ਹੋਵੇਗੀ। ਇਸ ਨੂੰ ਮਾਰਕੀਟ ਨਯਿਮਾਂ, ਵਪਾਰਕ ਸੀਮਾਵਾਂ, ਜੋਖਮ ਗਣਨਾਵਾਂ, ਅਤੇ ਨਪਿਟਾਰਾ ਪ੍ਰਕਰਿਆਿਵਾਂ ਵਰਗੇ ਕਾਰਕਾਂ 'ਤੇ ਵਚਿਾਰ ਕਰਨਾ ਹੋਵੇਗਾ। ਇਸ ਸੰਦਰਭ ਵੱਚਿ ਸਵੈਚਲਤਿ ਸੁਧਾਰ ਹਮੇਸ਼ਾ ਖੇਤਰ ਦੀ ਪੂਰੀ ਜਟਲਿਤਾ ਨੂੰ ਨਹੀਂ ਸਮਝ ਸਕਦੇ ਅਤੇ ਅਚਾਨਕ ਗਲਤੀਆਂ ਜਾਂ ਖੇਤਰ-ਵਸ਼ਿਸ਼ ਨਯਿਮਾਂ ਦੀ ਉਲੰਘਣਾ ਕਰ ਸਕਦੇ ਹਨ।

ਖੇਤਰ ਦੀ ਜਟਲਿਤਾ ਦੀਆਂ ਚੁਣੌਤੀਆਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ, ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਨੂੰ ਏ.ਆਈ. ਮਾਡਲਾਂ ਅਤੇ ਵਰਕਰਾਂ ਵੱਚਿ ਖੇਤਰ-ਵਸ਼ਿਸ਼ ਗਆਿਨ ਅਤੇ ਨਯਿਮਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਕੇ ਵਧਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਹੇਠ ਲਖਿੀਆਂ ਤਕਨੀਕਾਂ ਰਾਹੀਂ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ:

1. **ਖੇਤਰ-ਵਸ਼ਿਸ਼ ਸਖਿਲਾਈ:** ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਏ.ਆਈ. ਮਾਡਲਾਂ ਨੂੰ ਖੇਤਰ-ਵਸ਼ਿਸ਼ ਡਾਟਾਸੈੱਟਾਂ 'ਤੇ ਨਰਿਦੇਸ਼ਤਿ ਜਾਂ ਫਾਈਨ-ਟਊਿਨ ਵੀ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਵਸ਼ਿਸ਼ ਖੇਤਰ ਦੀਆਂ

ਬਾਰੀਕੀਆਂ ਅਤੇ ਨਯਿਮਾਂ ਨੂੰ ਸਮਝਦੇ ਹਨ। ਮਾਡਲਾਂ ਨੂੰ ਪ੍ਰਤੀਨਧਿੀ ਡਾਟਾ ਅਤੇ ਸਥਤਿੀਆਂ ਦੇ ਸੰਪਰਕ ਵੱਚਿ ਲਆਿ ਕੇ, ਉਹ ਖੇਤਰ-ਵਸਿ਼ਸ਼ ਪੈਟਰਨ, ਸੀਮਾਵਾਂ, ਅਤੇ ਅਪਵਾਦਾਂ ਨੂੰ ਸੱਖਿ ਸਕਦੇ ਹਨ।

2. **ਨਯਿਮ-ਆਧਾਰਤਿ ਸੀਮਾਵਾਂ:** ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਵਰਕਰਾਂ ਨੂੰ ਸਪੱਸ਼ਟ ਨਯਿਮ-ਆਧਾਰਤਿ ਸੀਮਾਵਾਂ ਨਾਲ ਵਧਾਇਆ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਖੇਤਰ-ਵਸਿ਼ਸ਼ ਗਆਿਨ ਨੂੰ ਕੋਡ ਕਰਦੀਆਂ ਹਨ। ਇਹ ਨਯਿਮ ਖੇਤਰ ਦੇ ਮਾਹਰਾਂ ਦੁਆਰਾ ਪਰਭਾਸ਼ਤਿ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ ਅਤੇ ਡਾਟਾ ਸੁਧਾਰ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਏਕੀਕ੍ਰਤਿ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਏ.ਆਈ. ਮਾਡਲ ਫਰਿ ਇਹਨਾਂ ਨਯਿਮਾਂ ਦੀ ਵਰਤੋਂ ਆਪਣੇ ਫੈਸਲਆਿਂ ਨੂੰ ਨਰਿਦੇਸ਼ਤਿ ਕਰਨ ਅਤੇ ਖੇਤਰ-ਵਸਿ਼ਸ਼ ਲੋੜਾਂ ਦੀ ਪਾਲਣਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਕਰ ਸਕਦੇ ਹਨ।
3. **ਖੇਤਰ ਮਾਹਰਾਂ ਨਾਲ ਸਹਯਿੋਗ:** ਜਟਲਿ ਖੇਤਰਾਂ ਵੱਚਿ, ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਦੀ ਡਜ਼ਾਇੀਨ ਅਤੇ ਵਕਿਾਸ ਵੱਚਿ ਖੇਤਰ ਮਾਹਰਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਖੇਤਰ ਮਾਹਰ ਡਾਟਾ ਦੀਆਂ ਬਾਰੀਕੀਆਂ, ਵਪਾਰਕ ਨਯਿਮਾਂ, ਅਤੇ ਸੰਭਾਵੀ ਸੀਮਾ ਮਾਮਲਆਿਂ ਬਾਰੇ ਮੁੱਲਵਾਨ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ। ਉਹਨਾਂ ਦੇ ਗਆਿਨ ਨੂੰ ਮਨੁੱਖੀ-ਸੰਚਾਲਤਿ ਪ੍ਰਕਰਿਆ ਪੈਟਰਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਏ.ਆਈ. ਮਾਡਲਾਂ ਅਤੇ ਵਰਕਰਾਂ ਵੱਚਿ ਸ਼ਾਮਲ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਤਾਂ ਜੋ ਸਵੈਚਲਤਿ ਡਾਟਾ ਸੁਧਾਰਾਂ ਦੀ ਸੁੱਧਤਾ ਅਤੇ ਭਰੋਸੇਯੋਗਤਾ ਨੂੰ ਵਧਾਇਆ ਜਾ ਸਕੇ।
4. **ਕ੍ਰਮਵਾਰ ਅਤੇ ਦੁਹਰਾਉਣ ਵਾਲਾ ਪਹੁੰਚ:** ਜਟਲਿ ਖੇਤਰਾਂ ਨਾਲ ਨਜੱਠਿਦੇ ਸਮੇਂ, ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਲਈ ਕ੍ਰਮਵਾਰ ਅਤੇ ਦੁਹਰਾਉਣ ਵਾਲਾ ਪਹੁੰਚ ਅਪਣਾਉਣਾ ਅਕਸਰ ਲਾਭਦਾਇਕ ਹੁੰਦਾ ਹੈ। ਪੂਰੇ ਖੇਤਰ ਲਈ ਇੱਕੋ ਵਾਰ ਸੁਧਾਰਾਂ ਨੂੰ ਸਵੈਚਲਤਿ ਕਰਨ ਦੀ ਕੋਸ਼ਸ਼ਿ ਕਰਨ ਦੀ ਬਜਾਏ, ਵਸਿ਼ਸ਼ ਉਪ-ਖੇਤਰਾਂ ਜਾਂ ਡਾਟਾ ਸ਼੍ਰੇਣੀਆਂ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰੋ ਜੱਥੇ ਨਯਿਮ ਅਤੇ ਸੀਮਾਵਾਂ ਚੰਗੀ ਤਰ੍ਹਾਂ ਸਮਝੀਆਂ ਜਾਂਦੀਆਂ ਹਨ। ਜਵਿੇਂ-ਜਵਿੇਂ ਖੇਤਰ ਦੀ ਸਮਝ ਵਧਦੀ ਹੈ ਅਤੇ ਤਕਨੀਕਾਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਸਾਬਤ ਹੁੰਦੀਆਂ ਹਨ, ਹੌਲੀ-ਹੌਲੀ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਤਕਨੀਕਾਂ ਦੇ ਦਾਇਰੇ ਨੂੰ ਵਧਾਓ।

ਡਾਟਾ ਖੇਤਰ ਦੀ ਜਟਲਿਤਾ ਨੂੰ ਧਆਿਨ ਵੱਚਿ ਰੱਖਦੇ ਹੋਏ ਅਤੇ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਵੱਚਿ ਖੇਤਰ-ਵਸਿ਼ਸ਼ ਗਆਿਨ ਨੂੰ ਸ਼ਾਮਲ ਕਰਕੇ, ਤੁਸੀਂ ਸਵੈਚਾਲਨ ਅਤੇ ਸੁੱਧਤਾ ਵਚਿਕਾਰ ਸੰਤੁਲਨ ਬਣਾ ਸਕਦੇ ਹੋ। ਇਹ ਸਮਝਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲਾ ਡਾਟਾ ਇੱਕ ਸਰਬ-ਅਨੁਕੂਲ ਹੱਲ ਨਹੀਂ ਹੈ ਅਤੇ ਇਸ ਪਹੁੰਚ ਨੂੰ ਹਰ ਖੇਤਰ ਦੀਆਂ ਵਸਿ਼ਸ਼ ਲੋੜਾਂ ਅਤੇ ਚੁਣੌਤੀਆਂ ਦੇ ਅਨੁਸਾਰ ਢਾਲਆਿ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ।

ਜਟਲਿ ਖੇਤਰਾਂ ਵੱਚਿ, ਇੱਕ ਮਸ਼ਿਰਤਿ ਪਹੁੰਚ ਜੋ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਨੂੰ ਮਨੁੱਖੀ ਮੁਹਾਰਤ ਅਤੇ ਨਗਿਰਾਨੀ ਨਾਲ ਜੋੜਦੀ ਹੈ, ਸਭ ਤੋਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਹੋ ਸਕਦੀ ਹੈ। ਸਵੈਚਲਤਿ ਸੁਧਾਰ ਨਯਿਮਤਿ ਅਤੇ ਚੰਗੀ ਤਰ੍ਹਾਂ ਪਰਭਾਸ਼ਤਿ ਮਾਮਲਆਿਂ ਨੂੰ ਸੰਭਾਲ ਸਕਦੇ ਹਨ, ਜਦੋਂ ਕਿ ਜਟਲਿ ਸਥਤਿੀਆਂ ਜਾਂ ਅਪਵਾਦਾਂ ਨੂੰ ਮਨੁੱਖੀ ਸਮੀਖਆਿ ਅਤੇ ਦਖਲਅੰਦਾਜ਼ੀ ਲਈ ਚੰਨ੍ਹਿਤਿ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਸਹਯਿੋਗੀ ਪਹੁੰਚ ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦੀ ਹੈ ਕਿ ਜਟਲਿ ਡਾਟਾ ਖੇਤਰਾਂ ਵੱਚਿ ਜ਼ਰੂਰੀ ਨਯਿੰਤਰਣ ਅਤੇ ਸੁੱਧਤਾ ਨੂੰ ਬਣਾਈ ਰੱਖਦੇ ਹੋਏ ਸਵੈਚਾਲਨ ਦੇ ਲਾਭ ਪ੍ਰਾਪਤ

ਕੀਤੇ ਜਾਂਦੇ ਹਨ।

ਵਿਆਖਿਆਯੋਗਤਾ ਅਤੇ ਪਾਰਦਰਸ਼ਤਾ

ਵਿਆਖਿਆਯੋਗਤਾ ਏ.ਆਈ. ਮਾਡਲਾਂ ਦੁਆਰਾ ਲਏ ਗਏ ਫੈਸਲਿਆਂ ਦੇ ਪੱਛੇ ਦੇ ਤਰਕ ਨੂੰ ਸਮਝਣ ਅਤੇ ਵਿਆਖਿਆ ਕਰਨ ਦੀ ਯੋਗਤਾ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ, ਜਦੋਂ ਕਿ ਪਾਰਦਰਸ਼ਤਾ ਵੱਚਿ ਡਾਟਾ ਸੁਧਾਰ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਸਪੱਸ਼ਟ ਦ੍ਰਿਸ਼ਟੀ ਪ੍ਰਦਾਨ ਕਰਨਾ ਸ਼ਾਮਲ ਹੈ।

ਕਈ ਸੰਦਰਭਾਂ ਵੱਚਿ, ਡਾਟਾ ਸੰਸੋਧਨਾਂ ਦੀ ਲੇਖਾ-ਪ੍ਰੀਖਿਆ ਅਤੇ ਔਚੱਤਿ ਹੋਣਾ ਜਰੂਰੀ ਹੈ। ਹੱਸੇਦਾਰ, ਜਨ੍ਹਿਾਂ ਵੱਚਿ ਵਪਾਰਕ ਉਪਭੋਗਤਾ, ਲੇਖਾ ਪ੍ਰੀਖਕ, ਅਤੇ ਨਯਿਾਮਕ ਸੰਸਥਾਵਾਂ ਸ਼ਾਮਲ ਹਨ, ਨੂੰ ਇਹ ਵਿਆਖਿਆਵਾਂ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ ਕਿ ਕੁਝ ਖਾਸ ਡਾਟਾ ਸੁਧਾਰ ਕਿਉਂ ਕੀਤੇ ਗਏ ਅਤੇ ਏ.ਆਈ. ਮਾਡਲ ਉਨ੍ਹਾਂ ਫੈਸਲਿਆਂ 'ਤੇ ਕਿਵੇਂ ਪਹੁੰਚੇ। ਇਹ ਖਾਸ ਤੌਰ 'ਤੇ ਉਨ੍ਹਾਂ ਖੇਤਰਾਂ ਵੱਚਿ ਮਹੱਤਵਪੂਰਨ ਹੈ ਜੱਥੇ ਡਾਟਾ ਸ਼ੁੱਧਤਾ ਅਤੇ ਸੰਪੂਰਨਤਾ ਦੇ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਹੁੰਦੇ ਹਨ, ਜਵਿੇਂ ਕਿ ਵਿੱਤਿ, ਸਹਿਤ ਸੰਭਾਲ, ਅਤੇ ਕਾਨੂੰਨੀ ਮਾਮਲੇ।

ਵਿਆਖਿਆਯੋਗਤਾ ਅਤੇ ਪਾਰਦਰਸ਼ਤਾ ਦੀ ਲੋੜ ਨੂੰ ਪੂਰਾ ਕਰਨ ਲਈ, ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਵੱਚਿ ਅਜਹਿੇ ਤੰਤਰ ਸ਼ਾਮਲ ਹੋਣੇ ਚਾਹੀਦੇ ਹਨ ਜੋ ਏ.ਆਈ. ਮਾਡਲਾਂ ਦੀ ਫੈਸਲਾ ਲੈਣ ਦੀ ਪ੍ਰਕਰਿਆ ਬਾਰੇ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ। ਇਹ ਵੱਖ-ਵੱਖ ਪਹੁੰਚਾਂ ਰਾਹੀ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ:

1. **ਵਚਿਾਰਾਂ ਦੀ ਲੜੀ:** ਡਾਟਾ ਵੱਚਿ ਤਬਦੀਲੀਆਂ ਲਾਗੂ ਕਰਨ ਤੋਂ ਪਹਲਿਾਂ ਮਾਡਲ ਨੂੰ ਆਪਣੀ ਸੋਚ ਨੂੰ "ਉੱਚੀ ਆਵਾਜ਼ ਵੱਚਿ" ਸਮਝਾਉਣ ਲਈ ਕਹਣਿਾ ਫੈਸਲਾ ਲੈਣ ਦੀ ਪ੍ਰਕਰਿਆ ਨੂੰ ਸਮਝਣ ਵੱਚਿ ਆਸਾਨੀ ਕਰ ਸਕਦਾ ਹੈ ਅਤੇ ਕੀਤੇ ਗਏ ਸੁਧਾਰਾਂ ਲਈ ਮਨੁੱਖੀ-ਪੜ੍ਹਨਯੋਗ ਵਿਆਖਿਆਵਾਂ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ। ਇਸਦਾ ਨੁਕਸਾਨ ਵਿਆਖਿਆ ਨੂੰ ਸੰਰਚਤਿ ਡਾਟਾ ਆਉਟਪੁੱਟ ਤੋਂ ਵੱਖ ਕਰਨ ਵੱਚਿ ਥੋੜ੍ਹੀ ਜਹਿੀ ਵਧੇਰੇ ਜਟਲਿਤਾ ਹੈ, ਜਸਿ ਨੂੰ ਹੱਲ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ...
2. **ਵਿਆਖਿਆ ਨਰਿਮਾਣ:** ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਵਰਕਰਾਂ ਨੂੰ ਉਨ੍ਹਾਂ ਦੁਆਰਾ ਕੀਤੇ ਗਏ ਸੁਧਾਰਾਂ ਲਈ ਮਨੁੱਖੀ-ਪੜ੍ਹਨਯੋਗ ਵਿਆਖਿਆਵਾਂ ਤਆਿਰ ਕਰਨ ਦੀ ਯੋਗਤਾ ਨਾਲ ਲੈਸ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਮਾਡਲ ਨੂੰ ਆਪਣੀ ਫੈਸਲਾ ਲੈਣ ਦੀ ਪ੍ਰਕਰਿਆ ਨੂੰ ਡਾਟਾ ਵੱਚਿ ਹੀ ਏਕੀਕ੍ਰਤਿ ਆਸਾਨੀ ਨਾਲ ਸਮਝਣ ਯੋਗ ਵਿਆਖਿਆਵਾਂ ਵਜੋਂ ਆਉਟਪੁੱਟ ਕਰਨ ਲਈ ਕਹਿ ਕੇ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ, ਇੱਕ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲਾ ਡਾਟਾ ਵਰਕਰ ਇੱਕ ਰਪਿੋਰਟ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ ਜੋ ਉਸ ਦੁਆਰਾ ਪਛਾਣੀਆਂ ਗਈਆਂ ਵਸਿ਼ਸ਼ ਡਾਟਾ ਅਸੰਗਤੀਆਂ, ਲਾਗੂ ਕੀਤੇ ਗਏ ਸੁਧਾਰਾਂ, ਅਤੇ ਉਨ੍ਹਾਂ ਸੁਧਾਰਾਂ ਦੇ ਪੱਛੇ ਦੇ ਤਰਕ ਨੂੰ ਉਜਾਗਰ ਕਰਦੀ ਹੈ।

3. **ਵਸ਼ਿਸ਼ਤਾ ਮਹੱਤਤਾ:** ਏ.ਆਈ. ਮਾਡਲਾਂ ਨੂੰ ਉਨ੍ਹਾਂ ਦੇ ਨਰਿਦੇਸ਼ਾਂ ਦੇ ਹੰਸੇ ਵਜੋਂ ਡਾਟਾ ਸੁਧਾਰ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਵੱਖ-ਵੱਖ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਜਾਂ ਗੁਣਾਂ ਦੀ ਮਹੱਤਤਾ ਬਾਰੇ ਜਾਣਕਾਰੀ ਨਾਲ ਨਰਿਦੇਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਨਰਿਦੇਸ਼, ਬਦਲੇ ਵੱਚਿ, ਮਨੁੱਖੀ ਹੰਸੇਦਾਰਾਂ ਨੂੰ ਦਖਿਾਏ ਜਾ ਸਕਦੇ ਹਨ। ਮਾਡਲ ਦੇ ਫੈਸਲਆਿਂ ਨੂੰ ਪ੍ਰਭਾਵਤਿ ਕਰਨ ਵਾਲੇ ਮੁੱਖ ਕਾਰਕਾਂ ਦੀ ਪਛਾਣ ਕਰਕੇ, ਹੰਸੇਦਾਰ ਸੁਧਾਰਾਂ ਦੇ ਪੰਛੇ ਦੇ ਤਰਕ ਬਾਰੇ ਜਾਣਕਾਰੀ ਪ੍ਰਾਪਤ ਕਰ ਸਕਦੇ ਹਨ ਅਤੇ ਉਨ੍ਹਾਂ ਦੀ ਵੈਧਤਾ ਦਾ ਮੁਲਾਂਕਣ ਕਰ ਸਕਦੇ ਹਨ।
4. **ਲੌਗੰਗਿ ਅਤੇ ਲੇਖਾ-ਜੋਖਾ:** ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਪਾਰਦਰਸ਼ਤਾ ਬਣਾਈ ਰੱਖਣ ਲਈ ਵਆਿਪਕ ਲੌਗੰਗਿ ਅਤੇ ਲੇਖਾ-ਜੋਖਾ ਵਧਿੀਆਂ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਬਹੁਤ ਜਰੂਰੀ ਹੈ। ਏਆਈ ਮਾਡਲਾਂ ਦੁਆਰਾ ਕੀਤੀ ਗਈ ਹਰ ਡਾਟਾ ਸੁਧਾਰ ਨੂੰ ਲੌਗ ਕੀਤਾ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ, ਜਸਿ ਵੱਚਿ ਮੂਲ ਡਾਟਾ, ਸੁਧਾਰਆਿ ਗਆਿ ਡਾਟਾ, ਅਤੇ ਕੀਤੀਆਂ ਗਈਆਂ ਵਸ਼ਿਸ਼ ਕਾਰਵਾਈਆਂ ਸ਼ਾਮਲ ਹਨ। ਇਹ ਲੇਖਾ-ਪੜਤਾਲ ਰਕਿਾਰਡ ਪਛਿਲੇਰੇ ਵਸ਼ਿਲੇਸ਼ਣ ਦੀ ਆਗਆਿ ਦੰਦਿਾ ਹੈ ਅਤੇ ਡਾਟਾ ਵੱਚਿ ਕੀਤੀਆਂ ਗਈਆਂ ਤਬਦੀਲੀਆਂ ਦਾ ਸਪਸ਼ਟ ਰਕਿਾਰਡ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।
5. **ਮਨੁੱਖੀ-ਸ਼ਮੂਲੀਅਤ ਪਹੁੰਚ:** ਮਨੁੱਖੀ-ਸ਼ਮੂਲੀਅਤ ਪਹੁੰਚ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨਾ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਤਕਨੀਕਾਂ ਦੀ ਵਆਿਖਆਿਯੋਗਤਾ ਅਤੇ ਪਾਰਦਰਸ਼ਤਾ ਨੂੰ ਵਧਾ ਸਕਦਾ ਹੈ। ਏਆਈ-ਜਨਰੇਟਡ ਸੁਧਾਰਾਂ ਦੀ ਸਮੀਖਆਿ ਅਤੇ ਪ੍ਰਮਾਣਕਿਤਾ ਵੱਚਿ ਮਾਹਰਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਕੇ, ਸੰਸਥਾਵਾਂ ਇਹ ਯਕੀਨੀ ਬਣਾ ਸਕਦੀਆਂ ਹਨ ਕ ਿਸੁਧਾਰ ਡੋਮੇਨ ਗਆਿਨ ਅਤੇ ਵਪਾਰਕ ਲੋੜਾਂ ਦੇ ਅਨੁਕੂਲ ਹਨ। ਮਨੁੱਖੀ ਨਗਿਰਾਨੀ ਜਵਾਬਦੇਹੀ ਦੀ ਇੱਕ ਵਾਧੂ ਪਰਤ ਜੋੜਦੀ ਹੈ ਅਤੇ ਏਆਈ ਮਾਡਲਾਂ ਵੱਚਿ ਕਸਿ ਵੀ ਸੰਭਾਵੀ ਪੱਖਪਾਤ ਜਾਂ ਗਲਤੀਆਂ ਦੀ ਪਛਾਣ ਕਰਨ ਦੀ ਆਗਆਿ ਦੰਦੀ ਹੈ।
6. **ਨਰਿੰਤਰ ਨਗਿਰਾਨੀ ਅਤੇ ਮੁਲਾਂਕਣ:** ਪਾਰਦਰਸ਼ਤਾ ਅਤੇ ਵਸ਼ਿਵਾਸ ਬਣਾਈ ਰੱਖਣ ਲਈ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਤਕਨੀਕਾਂ ਦੇ ਪ੍ਰਦਰਸ਼ਨ ਦੀ ਨਯਿਮਤਿ ਨਗਿਰਾਨੀ ਅਤੇ ਮੁਲਾਂਕਣ ਕਰਨਾ ਜਰੂਰੀ ਹੈ। ਸਮੇਂ ਦੇ ਨਾਲ ਏਆਈ ਮਾਡਲਾਂ ਦੀ ਸੁੱਧਤਾ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ੀਲਤਾ ਦਾ ਮੁਲਾਂਕਣ ਕਰਕੇ, ਸੰਸਥਾਵਾਂ ਕਸਿ ਵੀ ਵਚਿਲਨ ਜਾਂ ਅਸਧਾਰਨਤਾਵਾਂ ਦੀ ਪਛਾਣ ਕਰ ਸਕਦੀਆਂ ਹਨ ਅਤੇ ਸੁਧਾਰਾਤਮਕ ਕਾਰਵਾਈਆਂ ਕਰ ਸਕਦੀਆਂ ਹਨ। ਨਰਿੰਤਰ ਨਗਿਰਾਨੀ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣ ਵੱਚਿ ਮਦਦ ਕਰਦੀ ਹੈ ਕ ਿਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀ ਡਾਟਾ ਪ੍ਰਕਰਿਆ ਭਰੋਸੇਯੋਗ ਰਹੇ ਅਤੇ ਇੱਛਤ ਨਤੀਜਆਿਂ ਦੇ ਅਨੁਕੂਲ ਹੋਵੇ।

ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਤਕਨੀਕਾਂ ਨੂੰ ਲਾਗੂ ਕਰਦੇ ਸਮੇਂ ਵਆਿਖਆਿਯੋਗਤਾ ਅਤੇ ਪਾਰਦਰਸ਼ਤਾ ਮਹੱਤਵਪੂਰਨ ਵਚਿਾਰ ਹਨ। ਡਾਟਾ ਸੁਧਾਰਾਂ ਲਈ ਸਪੱਸ਼ਟ ਵਆਿਖਆਿਵਾਂ ਪ੍ਰਦਾਨ ਕਰਕੇ, ਵਆਿਪਕ ਲੇਖਾ-ਪੜਤਾਲ ਰਕਿਾਰਡ ਬਣਾਈ ਰੱਖ ਕੇ, ਅਤੇ ਮਨੁੱਖੀ ਨਗਿਰਾਨੀ ਨੂੰ ਸ਼ਾਮਲ ਕਰਕੇ, ਸੰਸਥਾਵਾਂ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀ ਡਾਟਾ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਵਸ਼ਿਵਾਸ ਬਣਾ ਸਕਦੀਆਂ ਹਨ ਅਤੇ ਇਹ ਯਕੀਨੀ ਬਣਾ ਸਕਦੀਆਂ ਹਨ ਕ ਿਡਾਟਾ ਵੱਚਿ ਕੀਤੀਆਂ ਗਈਆਂ ਤਬਦੀਲੀਆਂ ਜਾਇਜ਼ ਹਨ ਅਤੇ ਵਪਾਰਕ ਉਦੇਸ਼ਾਂ ਦੇ ਅਨੁਕੂਲ ਹਨ।

ਸਵੈਚਾਲਨ ਦੇ ਲਾਭਾਂ ਅਤੇ ਪਾਰਦਰਸ਼ਤਾ ਦੀ ਲੋੜ ਵਿਚਕਾਰ ਸੰਤੁਲਨ ਬਣਾਉਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਜਦੋਂ ਕਿ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਤਕਨੀਕਾਂ ਡਾਟਾ ਦੀ ਗੁਣਵੱਤਾ ਅਤੇ ਕੁਸ਼ਲਤਾ ਨੂੰ ਕਾਫ਼ੀ ਸੁਧਾਰ ਸਕਦੀਆਂ ਹਨ, ਇਹ ਡਾਟਾ ਸੁਧਾਰ ਪ੍ਰਕਰਿਆ 'ਤੇ ਦ੍ਰਸ਼ਿਟੀ ਅਤੇ ਨਿਯੰਤਰਣ ਗੁਆਉਣ ਦੀ ਕੀਮਤ 'ਤੇ ਨਹੀਂ ਹੋਣੀਆਂ ਚਾਹੀਦੀਆਂ। ਵਿਆਖਿਆਯੋਗਤਾ ਅਤੇ ਪਾਰਦਰਸ਼ਤਾ ਨੂੰ ਧਿਆਨ ਵਿੱਚ ਰੱਖਦੇ ਹੋਏ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਵਰਕਰਾਂ ਨੂੰ ਡਿਜ਼ਾਈਨ ਕਰਕੇ, ਸੰਸਥਾਵਾਂ ਏਆਈ ਦੀ ਸ਼ਕਤੀ ਦਾ ਲਾਭ ਲੈ ਸਕਦੀਆਂ ਹਨ ਜਦੋਂ ਕਿ ਡਾਟਾ ਵਿੱਚ ਜ਼ਰੂਰੀ ਜਵਾਬਦੇਹੀ ਅਤੇ ਵਿਸ਼ਵਾਸ ਦਾ ਪੱਧਰ ਬਣਾਈ ਰੱਖ ਸਕਦੀਆਂ ਹਨ।

ਅਣਚਾਹੇ ਨਤੀਜੇ

ਜਦੋਂ ਕਿ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਤਕਨੀਕਾਂ ਦਾ ਉਦੇਸ਼ ਡਾਟਾ ਦੀ ਗੁਣਵੱਤਾ ਅਤੇ ਸਥਰਿਤਾ ਨੂੰ ਸੁਧਾਰਨਾ ਹੈ, ਅਣਚਾਹੇ ਨਤੀਜਿਆਂ ਦੀ ਸੰਭਾਵਨਾ ਬਾਰੇ ਜਾਗਰੂਕ ਹੋਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਸਵੈਚਲਤਿ ਸੁਧਾਰ, ਜੇਕਰ ਧਿਆਨ ਨਾਲ ਡਿਜ਼ਾਈਨ ਅਤੇ ਨਿਗਰਾਨੀ ਨਹੀਂ ਕੀਤੀ ਜਾਂਦੀ, ਅਣਜਾਣੇ ਵਿੱਚ ਡਾਟਾ ਦੇ ਅਰਥ ਜਾਂ ਸੰਦਰਭ ਨੂੰ ਬਦਲ ਸਕਦੇ ਹਨ, ਜਸਿ ਨਾਲ ਅੱਗੇ ਦੀਆਂ ਸਮੱਸਿਆਵਾਂ ਪੈਦਾ ਹੋ ਸਕਦੀਆਂ ਹਨ।

ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਦੇ ਮੁੱਖ ਜੋਖਮਾਂ ਵਿੱਚੋਂ ਇੱਕ ਡਾਟਾ ਸੁਧਾਰ ਪ੍ਰਕਰਿਆ ਵਿੱਚ ਪੱਖਪਾਤ ਜਾਂ ਗਲਤੀਆਂ ਦਾ ਸ਼ਾਮਲ ਹੋਣਾ ਹੈ। ਏਆਈ ਮਾਡਲ, ਕਸਿੇ ਵੀ ਹੋਰ ਸਾਫਟਵੇਅਰ ਸਸਿਟਮ ਵਾਂਗ, ਸਖਿਲਾਈ ਡਾਟਾ ਵਿੱਚ ਮੌਜੂਦ ਪੱਖਪਾਤਾਂ ਜਾਂ ਐਲਗੋਰਦਿਮ ਦੇ ਡਿਜ਼ਾਈਨ ਰਾਹੀਂ ਪੇਸ਼ ਕੀਤੇ ਗਏ ਪੱਖਪਾਤਾਂ ਦੇ ਅਧੀਨ ਹੋ ਸਕਦੇ ਹਨ। ਜੇਕਰ ਇਹਨਾਂ ਪੱਖਪਾਤਾਂ ਦੀ ਪਛਾਣ ਅਤੇ ਘਟਾਇਆ ਨਹੀਂ ਜਾਂਦਾ, ਤਾਂ ਉਹ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀ ਡਾਟਾ ਪ੍ਰਕਰਿਆ ਰਾਹੀਂ ਫੈਲ ਸਕਦੇ ਹਨ ਅਤੇ ਝੁਕੇ ਹੋਏ ਜਾਂ ਗਲਤ ਡਾਟਾ ਸੋਧਾਂ ਦਾ ਕਾਰਨ ਬਣ ਸਕਦੇ ਹਨ।

ਉਦਾਹਰਣ ਦੇ ਤੌਰ 'ਤੇ, ਇੱਕ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਵਰਕਰ 'ਤੇ ਵਿਚਾਰ ਕਰੋ ਜਸਿਨੂੰ ਗਾਹਕ ਜਨਸੰਖਿਅਕ ਡਾਟਾ ਵਿੱਚ ਅਸੰਗਤੀਆਂ ਨੂੰ ਠੀਕ ਕਰਨ ਦਾ ਕੰਮ ਸੌਂਪਿਆ ਗਿਆ ਹੈ। ਜੇਕਰ AI ਮਾਡਲ ਨੇ ਇਤਿਹਾਸਕ ਡਾਟਾ ਤੋਂ ਪੱਖਪਾਤ ਸੀਖਿਆ ਹੈ, ਜਵਿੇਂ ਕਿ ਕੁਝ ਪੇਸ਼ਿਆਂ ਜਾਂ ਆਮਦਨ ਦੇ ਪੱਧਰਾਂ ਨੂੰ ਵਸ਼ਿੇਸ਼ ਲਿੰਗਾਂ ਜਾਂ ਨਸਲਾਂ ਨਾਲ ਜੋੜਨਾ, ਤਾਂ ਇਹ ਗਲਤ ਧਾਰਨਾਵਾਂ ਬਣਾ ਸਕਦਾ ਹੈ ਅਤੇ ਡਾਟਾ ਨੂੰ ਅਜਹਿੇ ਢੰਗ ਨਾਲ ਸੋਧ ਸਕਦਾ ਹੈ ਜੋ ਇਹਨਾਂ ਪੱਖਪਾਤਾਂ ਨੂੰ ਮਜ਼ਬੂਤ ਕਰਦਾ ਹੈ। ਇਹ ਗਲਤ ਗਾਹਕ ਪ੍ਰੋਫਾਈਲਾਂ, ਗੁਮਰਾਹਕੁੰਨ ਵਪਾਰਕ ਫੈਸਲਿਆਂ, ਅਤੇ ਸੰਭਾਵੀ ਭੇਦਭਾਵਪੂਰਨ ਨਤੀਜਿਆਂ ਵੱਲ ਲੈ ਜਾ ਸਕਦਾ ਹੈ।

ਇੱਕ ਹੋਰ ਸੰਭਾਵੀ ਅਣਚਾਹਿਆ ਨਤੀਜਾ ਡਾਟਾ ਸੁਧਾਰ ਪ੍ਰਕਰਿਆ ਦੌਰਾਨ ਕੀਮਤੀ ਜਾਣਕਾਰੀ ਜਾਂ ਸੰਦਰਭ ਦਾ ਨੁਕਸਾਨ ਹੈ। ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਅਕਸਰ ਸਥਰਿਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਡਾਟਾ ਨੂੰ ਮਿਆਰੀ ਅਤੇ ਆਮ ਬਣਾਉਣ 'ਤੇ ਕੇਂਦਰਤਿ ਹੁੰਦੀਆਂ ਹਨ। ਹਾਲਾਂਕਿ, ਕੁਝ ਮਾਮਲਿਆਂ ਵਿੱਚ, ਮੂਲ ਡਾਟਾ ਵਿੱਚ ਬਾਰੀਕੀਆਂ, ਅਪਵਾਦ, ਜਾਂ ਸੰਦਰਭਕ ਜਾਣਕਾਰੀ ਹੋ ਸਕਦੀ ਹੈ ਜੋ ਪੂਰੀ ਤਸਵੀਰ ਨੂੰ ਸਮਝਣ ਲਈ ਮਹੱਤਵਪੂਰਨ

ਹੈ। ਅੰਨ੍ਹੇਵਾਹ ਮਆਿਰੀਕਰਨ ਨੂੰ ਲਾਗੂ ਕਰਨ ਵਾਲੇ ਸਵੈਚਾਲਤਿ ਸੁਧਾਰ ਅਣਜਾਣੇ ਵੱਚਿ ਇਸ ਕੀਮਤੀ ਜਾਣਕਾਰੀ ਨੂੰ ਹਟਾ ਜਾਂ ਧੁੰਦਲਾ ਕਰ ਸਕਦੇ ਹਨ।

ਉਦਾਹਰਣ ਲਈ, ਇੱਕ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਵਰਕਰ ਦੀ ਕਲਪਨਾ ਕਰੋ ਜੋ ਮੈਡੀਕਲ ਰਕਿਾਰਡਾਂ ਵੱਚਿ ਅਸੰਗਤੀਆਂ ਨੂੰ ਠੀਕ ਕਰਨ ਲਈ ਜ਼ੰਮਿੇਵਾਰ ਹੈ। ਜੇਕਰ ਵਰਕਰ ਨੂੰ ਕਸਿੇ ਦੁਰਲੱਭ ਸਥਤਿੀ ਜਾਂ ਅਸਧਾਰਨ ਇਲਾਜ ਯੋਜਨਾ ਵਾਲੇ ਮਰੀਜ਼ ਦੇ ਮੈਡੀਕਲ ਇਤਹਿਾਸ ਦਾ ਸਾਹਮਣਾ ਕਰਨਾ ਪੈਂਦਾ ਹੈ, ਤਾਂ ਇਹ ਡਾਟਾ ਨੂੰ ਇੱਕ ਵਧੇਰੇ ਆਮ ਪੈਟਰਨ ਵੱਚਿ ਫਟਿ ਕਰਨ ਦੀ ਕੋਸ਼ਸ਼ਿ ਕਰ ਸਕਦਾ ਹੈ। ਹਾਲਾਂਕ,ਿ ਅਜਹਿਾ ਕਰਦਆਿਂ, ਇਹ ਉਹਨਾਂ ਵਸ਼ੇਸ਼ ਵੇਰਵਆਿਂ ਅਤੇ ਸੰਦਰਭ ਨੂੰ ਗੁਆ ਸਕਦਾ ਹੈ ਜੋ ਮਰੀਜ਼ ਦੀ ਵਲਿੱਖਣ ਸਥਤਿੀ ਨੂੰ ਸਹੀ ਢੰਗ ਨਾਲ ਦਰਸਾਉਣ ਲਈ ਮਹੱਤਵਪੂਰਨ ਹਨ। ਜਾਣਕਾਰੀ ਦਾ ਇਹ ਨੁਕਸਾਨ ਮਰੀਜ਼ ਦੀ ਦੇਖਭਾਲ ਅਤੇ ਮੈਡੀਕਲ ਫੈਸਲਾ ਲੈਣ ਲਈ ਗੰਭੀਰ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦਾ ਹੈ।

ਅਣਚਾਹੇ ਨਤੀਜਆਿਂ ਦੇ ਜੋਖਮਾਂ ਨੂੰ ਘੱਟ ਕਰਨ ਲਈ, ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਤਕਨੀਕਾਂ ਨੂੰ ਡਜ਼ਾਇੀਨ ਅਤੇ ਲਾਗੂ ਕਰਦੇ ਸਮੇਂ ਸਰਗਰਮ ਪਹੁੰਚ ਅਪਣਾਉਣਾ ਜ਼ਰੂਰੀ ਹੈ:

1. **ਵਆਿਪਕ ਟੈਸਟਂਗ ਅਤੇ ਪ੍ਰਮਾਣੀਕਰਨ:** ਉਤਪਾਦਨ ਵੱਚਿ ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੇ ਡਾਟਾ ਵਰਕਰਾਂ ਨੂੰ ਤੈਨਾਤ ਕਰਨ ਤੋਂ ਪਹਲਿਾਂ, ਵੱਖ-ਵੱਖ ਸਥਤਿੀਆਂ ਦੇ ਵਰੁੱਧ ਉਨ੍ਹਾਂ ਦੇ ਵਵਿਹਾਰ ਦੀ ਵਆਿਪਕ ਜਾਂਚ ਅਤੇ ਪ੍ਰਮਾਣਕਿਤਾ ਕਰਨਾ ਜ਼ਰੂਰੀ ਹੈ। ਇਸ ਵੱਚਿ ਪ੍ਰਤੀਨਧਿੀ ਡੇਟਾਸੇਟਾਂ ਨਾਲ ਟੈਸਟਂਗ ਸ਼ਾਮਲ ਹੈ ਜੋ ਵੱਖ-ਵੱਖ ਐਜ ਕੇਸਾਂ, ਅਪਵਾਦਾਂ, ਅਤੇ ਸੰਭਾਵੀ ਪੱਖਪਾਤਾਂ ਨੂੰ ਕਵਰ ਕਰਦੇ ਹਨ। ਸਖ਼ਤ ਟੈਸਟਂਗ ਅਸਲ ਡਾਟਾ 'ਤੇ ਪ੍ਰਭਾਵ ਪਾਉਣ ਤੋਂ ਪਹਲਿਾਂ ਕਸਿੇ ਵੀ ਅਣਚਾਹੇ ਨਤੀਜਆਿਂ ਦੀ ਪਛਾਣ ਕਰਨ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਵੱਚਿ ਮਦਦ ਕਰਦੀ ਹੈ।
2. **ਲਗਾਤਾਰ ਨਗਿਰਾਨੀ ਅਤੇ ਮੁਲਾਂਕਣ:** ਸਮੇਂ ਦੇ ਨਾਲ ਅਣਚਾਹੇ ਨਤੀਜਆਿਂ ਦਾ ਪਤਾ ਲਗਾਉਣ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਘੱਟ ਕਰਨ ਲਈ ਲਗਾਤਾਰ ਨਗਿਰਾਨੀ ਅਤੇ ਮੁਲਾਂਕਣ ਵਧੀਆਂ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਜ਼ਰੂਰੀ ਹੈ। ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀਆਂ ਡਾਟਾ ਪ੍ਰਕਰਿਆਵਾਂ ਦੇ ਨਤੀਜਆਿਂ ਦੀ ਨਯਿਮਤਿ ਸਮੀਖਆਿ ਕਰਨਾ, ਡਾਊਨਸਟ੍ਰੀਮ ਸਸਿਟਮਾਂ ਅਤੇ ਫੈਸਲਾ ਲੈਣ 'ਤੇ ਪ੍ਰਭਾਵ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਨਾ, ਅਤੇ ਹੱਸਿੇਦਾਰਾਂ ਤੋਂ ਫੀਡਬੈਕ ਇਕੱਠਾ ਕਰਨਾ ਕਸਿੇ ਵੀ ਮਾੜੇ ਪ੍ਰਭਾਵਾਂ ਦੀ ਪਛਾਣ ਕਰਨ ਅਤੇ ਸਮੇਂ ਸਰਿ ਸੁਧਾਰਾਤਮਕ ਕਾਰਵਾਈਆਂ ਨੂੰ ਉਤਸ਼ਾਹਤਿ ਕਰਨ ਵੱਚਿ ਮਦਦ ਕਰ ਸਕਦਾ ਹੈ। ਜੇਕਰ ਤੁਹਾਡੀ ਸੰਸਥਾ ਕੋਲ ਓਪਰੇਸ਼ਨਲ ਡੈਸ਼ਬੋਰਡ ਹਨ, ਤਾਂ ਸਵੈਚਾਲਤਿ ਡਾਟਾ ਤਬਦੀਲੀਆਂ ਨਾਲ ਸਬੰਧਤ ਸਪੱਸ਼ਟ ਤੌਰ 'ਤੇ ਦਖਿਾਈ ਦੇਣ ਵਾਲੇ ਮੈਟ੍ਰਕਿਸ ਜੋੜਨਾ ਸ਼ਾਇਦ ਇੱਕ ਚੰਗਾ ਵਚਿਾਰ ਹੈ। ਸਧਾਰਨ ਡਾਟਾ ਤਬਦੀਲੀ ਗਤੀਵਧਿੀ ਤੋਂ ਵੱਡੇ ਵਚਿਲਨਾਂ ਨਾਲ ਜੁੜੇ ਅਲਾਰਮ ਜੋੜਨਾ ਸ਼ਾਇਦ ਇੱਕ ਹੋਰ ਵੀ ਬਹਿਤਰ ਵਚਿਾਰ ਹੈ!
3. **ਮਨੁੱਖੀ ਨਗਿਰਾਨੀ ਅਤੇ ਦਖਲਅੰਦਾਜ਼ੀ:** ਸਵੈ-ਠੀਕ ਕਰਨ ਵਾਲੀ ਡਾਟਾ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਮਨੁੱਖੀ ਨਗਿਰਾਨੀ ਅਤੇ ਦਖਲਅੰਦਾਜ਼ੀ ਦੀ ਯੋਗਤਾ ਨੂੰ ਬਣਾਈ ਰੱਖਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਭਾਵੇਂ ਸਵੈਚਾਲਨ ਕੁਸ਼ਲਤਾ

ਨੂੰ ਬਹੁਤ ਸੁਧਾਰ ਸਕਦਾ ਹੈ, ਪਰ ਇਹ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਮਨੁੱਖੀ ਮਾਹਰ AI ਮਾਡਲਾਂ ਦੁਆਰਾ ਕੀਤੇ ਗਏ ਸੁਧਾਰਾਂ ਦੀ ਸਮੀਖਿਆ ਅਤੇ ਪ੍ਰਮਾਣਿਕਤਾ ਕਰਨ, ਖਾਸ ਕਰਕੇ ਮਹੱਤਵਪੂਰਨ ਜਾਂ ਸੰਵੇਦਨਸ਼ੀਲ ਖੇਤਰਾਂ ਵਿੱਚ। ਮਨੁੱਖੀ ਫੈਸਲਾ ਅਤੇ ਡੋਮੇਨ ਮੁਹਾਰਤ ਕਿਸੇ ਵੀ ਅਣਚਾਹੇ ਨਤੀਜਿਆਂ ਦੀ ਪਛਾਣ ਕਰਨ ਅਤੇ ਹੱਲ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰ ਸਕਦੇ ਹਨ ਜੋ ਪੈਦਾ ਹੋ ਸਕਦੇ ਹਨ।

4. **ਵਿਆਖਿਆਯੋਗ ਏ.ਆਈ. (ਐਕਸ.ਏ.ਆਈ.) ਅਤੇ ਪਾਰਦਰਸ਼ਤਾ:** ਜਦਿਂ ਕਿ ਪਿਛਲੇ ਉਪ-ਭਾਗ ਵਿੱਚ ਚਰਚਾ ਕੀਤੀ ਗਈ ਹੈ, ਵਿਆਖਿਆਯੋਗ ਏ.ਆਈ. ਤਕਨੀਕਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨਾ ਅਤੇ ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਪ੍ਰਕਰਿਆ ਵਿੱਚ ਪਾਰਦਰਸ਼ਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣਾ ਅਣਚਾਹੇ ਨਤੀਜਿਆਂ ਨੂੰ ਘੱਟ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰ ਸਕਦਾ ਹੈ। ਡਾਟਾ ਸੁਧਾਰਾਂ ਲਈ ਸਪੱਸ਼ਟ ਵਿਆਖਿਆਵਾਂ ਪ੍ਰਦਾਨ ਕਰਕੇ ਅਤੇ ਵਿਆਪਕ ਲੇਖਾ-ਪੜਤਾਲ ਰਿਕਾਰਡ ਬਣਾਈ ਰੱਖ ਕੇ, ਸੰਸਥਾਵਾਂ ਏ.ਆਈ. ਮਾਡਲਾਂ ਦੁਆਰਾ ਕੀਤੇ ਗਏ ਬਦਲਾਵਾਂ ਦੇ ਪੀਛੇ ਦੇ ਤਰਕ ਨੂੰ ਬਹਿਤਰ ਢੰਗ ਨਾਲ ਸਮਝ ਅਤੇ ਟਰੇਸ ਕਰ ਸਕਦੀਆਂ ਹਨ।
5. **ਕ੍ਰਮਵਾਰ ਅਤੇ ਦੁਹਰਾਉਣ ਵਾਲਾ ਪਹੁੰਚ:** ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਲਈ ਕ੍ਰਮਵਾਰ ਅਤੇ ਦੁਹਰਾਉਣ ਵਾਲੇ ਪਹੁੰਚ ਨੂੰ ਅਪਣਾਉਣ ਨਾਲ ਅਣਚਾਹੇ ਨਤੀਜਿਆਂ ਦੇ ਜੋਖਮ ਨੂੰ ਘੱਟ ਕਰਨ ਵਿੱਚ ਮਦਦ ਮਿਲ ਸਕਦੀ ਹੈ। ਪੂਰੇ ਡਾਟਾਸੇਟ 'ਤੇ ਇੱਕੋ ਵਾਰ ਸਵੈਚਲਿਤ ਸੁਧਾਰ ਲਾਗੂ ਕਰਨ ਦੀ ਬਜਾਏ, ਡਾਟਾ ਦੇ ਇੱਕ ਉਪ-ਸਮੂਹ ਨਾਲ ਸੁਰੂ ਕਰੋ ਅਤੇ ਜਦਿਂ-ਜਦਿਂ ਤਕਨੀਕਾਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਅਤੇ ਭਰੋਸੇਯੋਗ ਸਾਬਤ ਹੁੰਦੀਆਂ ਹਨ, ਹੌਲੀ-ਹੌਲੀ ਦਾਇਰੇ ਨੂੰ ਵਧਾਓ। ਇਹ ਰਾਹ ਵਿੱਚ ਧਿਆਨ ਨਾਲ ਨਗਿਰਾਨੀ ਅਤੇ ਵਿਵਿਸਥਾ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦਾ ਹੈ, ਜੋ ਕਿਸੇ ਵੀ ਅਣਚਾਹੇ ਨਤੀਜਿਆਂ ਦੇ ਪ੍ਰਭਾਵ ਨੂੰ ਘਟਾਉਂਦਾ ਹੈ।
6. **ਸਹਯੋਗ ਅਤੇ ਫੀਡਬੈਕ:** ਵੱਖ-ਵੱਖ ਖੇਤਰਾਂ ਦੇ ਹਿੱਸੇਦਾਰਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨਾ ਅਤੇ ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਪ੍ਰਕਰਿਆ ਦੌਰਾਨ ਸਹਯੋਗ ਅਤੇ ਫੀਡਬੈਕ ਨੂੰ ਉਤਸ਼ਾਹਿਤ ਕਰਨਾ ਅਣਚਾਹੇ ਨਤੀਜਿਆਂ ਦੀ ਪਛਾਣ ਅਤੇ ਹੱਲ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰ ਸਕਦਾ ਹੈ। ਖੇਤਰ ਦੇ ਮਾਹਰਿਂ, ਡਾਟਾ ਉਪਭੋਗਤਾਵਾਂ, ਅਤੇ ਅੰਤਮਿ ਵਰਤੋਂਕਾਰਾਂ ਤੋਂ ਨਯਿਮਤਿ ਤੌਰ 'ਤੇ ਇਨਪੁੱਟ ਲੈਣਾ ਡਾਟਾ ਸੁਧਾਰਾਂ ਦੇ ਅਸਲ-ਸੰਸਾਰ ਪ੍ਰਭਾਵ ਬਾਰੇ ਮੁੱਲਵਾਨ ਅੰਤਰਦ੍ਰਸ਼ਿਟੀ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦਾ ਹੈ ਅਤੇ ਕਿਸੇ ਵੀ ਮੁੱਦਿਆਂ ਨੂੰ ਉਜਾਗਰ ਕਰ ਸਕਦਾ ਹੈ ਜੋ ਅਣਦੇਖੇ ਰਹਿ ਗਏ ਹੋ ਸਕਦੇ ਹਨ।

ਅਣਚਾਹੇ ਨਤੀਜਿਆਂ ਦੇ ਜੋਖਮ ਨੂੰ ਸਰਗਰਮੀ ਨਾਲ ਹੱਲ ਕਰਕੇ ਅਤੇ ਢੁਕਵੇਂ ਸੁਰੱਖਿਆ ਉਪਾਅ ਲਾਗੂ ਕਰਕੇ, ਸੰਸਥਾਵਾਂ ਸੰਭਾਵੀ ਮਾੜੇ ਪ੍ਰਭਾਵਾਂ ਨੂੰ ਘੱਟ ਕਰਦੇ ਹੋਏ ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਤਕਨੀਕਾਂ ਦੇ ਲਾਭਾਂ ਦਾ ਫਾਇਦਾ ਉਠਾ ਸਕਦੀਆਂ ਹਨ। ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਨੂੰ ਇੱਕ ਦੁਹਰਾਉਣ ਵਾਲੀ ਅਤੇ ਸਹਯੋਗੀ ਪ੍ਰਕਰਿਆ ਵਜੋਂ ਲੈਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ, ਜੋ ਲਗਾਤਾਰ ਨਗਿਰਾਨੀ, ਮੁਲਾਂਕਣ, ਅਤੇ ਤਕਨੀਕਾਂ ਨੂੰ ਸੁਧਾਰਦੀ ਹੈ ਤਾਂ ਜੋ ਇਹ ਯਕੀਨੀ ਬਣਾਇਆ ਜਾ ਸਕੇ ਕਿ ਉਹ ਲੋੜੀਂਦੇ ਨਤੀਜਿਆਂ ਨਾਲ ਮੇਲ ਖਾਂਦੇ ਹਨ ਅਤੇ ਡਾਟਾ ਦੀ ਅਖੰਡਤਾ ਅਤੇ ਵਿਸ਼ਵਾਸਯੋਗਤਾ ਨੂੰ ਬਣਾਈ

ਰੱਖਦੇ ਹਨ।

ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਪੈਟਰਨਾਂ ਦੀ ਵਰਤੋਂ 'ਤੇ ਵਿਚਾਰ ਕਰਦੇ ਸਮੇਂ, ਇਨ੍ਹਾਂ ਕਾਰਕਾਂ ਦਾ ਧਿਆਨ ਨਾਲ ਮੁਲਾਂਕਣ ਕਰਨਾ ਅਤੇ ਸੰਭਾਵੀ ਜੋਖਮਾਂ ਅਤੇ ਸੀਮਾਵਾਂ ਦੇ ਵਿਰੁੱਧ ਲਾਭਾਂ ਨੂੰ ਤੋਲਣਾ ਜ਼ਰੂਰੀ ਹੈ। ਕੁਝ ਮਾਮਲਿਆਂ ਵਿੱਚ, ਇੱਕ ਹਾਈਬ੍ਰਿਡ ਪਹੁੰਚ ਜੋ ਮਨੁੱਖੀ ਨਿਗਰਾਨੀ ਅਤੇ ਦਖਲਅੰਦਾਜ਼ੀ ਨਾਲ ਸਵੈਚਲਿਤ ਸੁਧਾਰਾਂ ਨੂੰ ਜੋੜਦੀ ਹੈ, ਸਭ ਤੋਂ ਢੁਕਵਾਂ ਹੱਲ ਹੋ ਸਕਦਾ ਹੈ।

ਇਹ ਨੋਟ ਕਰਨਾ ਵੀ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਤਕਨੀਕਾਂ ਨੂੰ ਮਜ਼ਬੂਤ ਡਾਟਾ ਪ੍ਰਮਾਣੀਕਰਨ, ਇਨਪੁੱਟ ਸੈਨੀਟਾਈਜ਼ੇਸ਼ਨ, ਅਤੇ ਗਲਤੀ ਨਿਯੰਤਰਣ ਵਿਧੀਆਂ ਦੇ ਬਦਲ ਵਜੋਂ ਨਹੀਂ ਦੇਖਿਆ ਜਾਣਾ ਚਾਹੀਦਾ। ਇਹ ਬੁਨਿਆਦੀ ਅਭਿਆਸ ਡਾਟਾ ਦੀ ਅਖੰਡਤਾ ਅਤੇ ਸੁਰੱਖਿਆ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਮਹੱਤਵਪੂਰਨ ਰਹਿੰਦੇ ਹਨ। ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਨੂੰ ਇੱਕ ਪੂਰਕ ਪਹੁੰਚ ਵਜੋਂ ਦੇਖਿਆ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ ਜੋ ਇਨ੍ਹਾਂ ਮੌਜੂਦਾ ਉਪਾਵਾਂ ਨੂੰ ਵਧਾ ਅਤੇ ਬਿਹਤਰ ਬਣਾ ਸਕਦਾ ਹੈ।

ਅੰਤ ਵਿੱਚ, ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਪੈਟਰਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦਾ ਫੈਸਲਾ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ ਲੋੜਾਂ, ਸੀਮਾਵਾਂ, ਅਤੇ ਤਰਜੀਹਾਂ 'ਤੇ ਨਿਰਭਰ ਕਰਦਾ ਹੈ। ਉੱਪਰ ਦੱਸੇ ਗਏ ਵਿਚਾਰਾਂ 'ਤੇ ਧਿਆਨ ਨਾਲ ਵਿਚਾਰ ਕਰਕੇ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਟੀਚਿਆਂ ਅਤੇ ਆਰਕੀਟੈਕਚਰ ਨਾਲ ਜੋੜ ਕੇ, ਤੁਸੀਂ ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲੇ ਡਾਟਾ ਤਕਨੀਕਾਂ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਕਦੋਂ ਅਤੇ ਕਿਵੇਂ ਵਰਤਣਾ ਹੈ, ਬਾਰੇ ਸੂਚਿਤ ਫੈਸਲੇ ਲੈ ਸਕਦੇ ਹੋ।

ਪ੍ਰਸੰਗਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ

ਪ੍ਰਸੰਗਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ (LLMs) ਦੀ ਸ਼ਕਤੀ ਦਾ ਲਾਭ ਲੈਂਦੇ ਹਨ ਤਾਂ ਜੋ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਗਤੀਸ਼ੀਲ ਅਤੇ ਪ੍ਰਸੰਗ-ਵਸ਼ਿਸ਼ ਸਮੱਗਰੀ ਤਆਿਰ ਕੀਤੀ ਜਾ ਸਕੇ। ਪੈਟਰਨਾਂ ਦੀ ਇਹ ਸ੍ਰੇਣੀ ਉਪਭੋਗਤਾਵਾਂ ਦੀਆਂ ਵਸ਼ਿਸ਼ ਲੋੜਾਂ, ਤਰਜੀਹਾਂ, ਅਤੇ ਇੱਥੋਂ ਤੱਕ ਕ ਿਐਪਲੀਕੇਸ਼ਨ ਨਾਲ ਉਨ੍ਹਾਂ ਦੀਆਂ ਪਛਿਲੀਆਂ ਅਤੇ ਮੌਜੂਦਾ ਅੰਤਰਕਰਿਆਵਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਨਜ਼ਿੀ ਅਤੇ ਢੁਕਵੀਂ ਸਮੱਗਰੀ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਮਹੱਤਤਾ ਨੂੰ ਪਛਾਣਦੀ ਹੈ।

> ਇਸ ਪਹੁੰਚ ਦੇ ਸੰਦਰਭ ਵਿੱਚ, "ਸਮੱਗਰੀ" ਮੁੱਖ ਸਮੱਗਰੀ (ਜਵਿੇਂ ਬਲੌਗ ਪੋਸਟਾਂ, ਲੇਖ, ਆਦਦ) ਅਤੇ ਮੈਟਾ-ਸਮੱਗਰੀ, ਜਵਿੇਂ ਕ ਿਮੁੱਖ ਸਮੱਗਰੀ ਲਈ ਸਫਿਾਰਸ਼ਾਂ, ਦੋਵਾਂ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ।

ਪ੍ਰਸੰਗਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨ ਤੁਹਾਡੇ ਯੂਜ਼ਰ ਜੁੜਾਅ ਪੱਧਰਾਂ ਨੂੰ ਵਧਾਉਣ, ਵਅਿਕਤੀਗਤ ਤਜਰਬੇ ਪ੍ਰਦਾਨ ਕਰਨ, ਅਤੇ ਤੁਹਾਡੇ ਅਤੇ ਤੁਹਾਡੇ ਯੂਜ਼ਰਾਂ ਲਈ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਕਾਰਜਾਂ ਨੂੰ ਸਵੈਚਾਲਤਿ ਕਰਨ ਵਿੱਚ ਮਹੱਤਵਪੂਰਨ

ਭੂਮਕਿਾ ਨਭਿਾ ਸਕਦੇ ਹਨ। ਇਸ ਅਧਆਿਇ ਵੱਚਿ ਅਸੀ ਜਹਿੜੇ ਪੈਟਰਨਾਂ ਦਾ ਵਰਣਨ ਕਰਦੇ ਹਾਂ, ਉਨ੍ਹਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਤੁਸੀ ਅਜਹਿੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰਦੀਆਂ ਹਨ, ਅਸਲ ਸਮੇਂ ਵੱਚਿ ਪ੍ਰਸੰਗ ਅਤੇ ਇਨਪੁੱਟ ਦੇ ਅਨੁਕੂਲ ਹੁੰਦੀਆਂ ਹਨ।

ਇਹ ਪੈਟਰਨ LLMs ਨੂੰ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਆਉਟਪੁੱਟ ਵੱਚਿ ਏਕੀਕ੍ਰਤਿ ਕਰਕੇ ਕੰਮ ਕਰਦੇ ਹਨ, ਜੋ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ (ਕਈ ਵਾਰ "ਕ੍ਰੋਮ" ਵਜੋਂ ਜਾਣਆਿ ਜਾਂਦਾ ਹੈ) ਤੋਂ ਲੈ ਕੇ ਈਮੇਲਾਂ ਅਤੇ ਸੂਚਨਾਵਾਂ ਦੇ ਹੋਰ ਰੂਪਾਂ, ਅਤੇ ਕਸਿੇ ਵੀ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪਾਈਪਲਾਈਨਾਂ ਤੱਕ ਫੈਲੇ ਹੋਏ ਹਨ।

ਜਦੋਂ ਕੋਈ ਯੂਜ਼ਰ ਐਪਲੀਕੇਸ਼ਨ ਨਾਲ ਅੰਤਰਕਰਿਆਿ ਕਰਦਾ ਹੈ ਜਾਂ ਕਸਿ ਵਸ਼ਿੇਸ਼ ਸਮੱਗਰੀ ਬੇਨਤੀ ਨੂੰ ਟ੍ਰਗਿਰ ਕਰਦਾ ਹੈ, ਐਪਲੀਕੇਸ਼ਨ ਢੁਕਵੇਂ ਪ੍ਰਸੰਗ ਨੂੰ ਕੈਪਚਰ ਕਰਦੀ ਹੈ, ਜਵਿੇਂ ਕ ਿਯੂਜ਼ਰ ਤਰਜੀਹਾਂ, ਪਛਿਲੀਆਂ ਅੰਤਰਕਰਿਆਿਵਾਂ, ਜਾਂ ਵਸ਼ਿੇਸ਼ ਪ੍ਰੌਂਪਟ। ਇਹ ਪ੍ਰਸੰਗਕਿ ਜਾਣਕਾਰੀ ਫਰਿ LLM ਵੱਚਿ ਭੇਜੀ ਜਾਂਦੀ ਹੈ, ਕਸਿ ਵੀ ਜਰੂਰੀ ਟੈਂਪਲੇਟ ਜਾਂ ਦਸ਼ਿਾ-ਨਰਿਦੇਸ਼ਾਂ ਦੇ ਨਾਲ ਅਤੇ ਟੈਕਸਟ ਆਉਟਪੁੱਟ ਤਆਿਰ ਕਰਨ ਲਈ ਵਰਤੀ ਜਾਂਦੀ ਹੈ ਜੋ ਨਹੀਂ ਤਾਂ ਜਾਂ ਤਾਂ ਹਾਰਡਕੋਡ ਕੀਤੀ ਜਾਣੀ ਹੁੰਦੀ, ਡੇਟਾਬੇਸ ਵੱਚਿ ਸਟੋਰ ਕੀਤੀ ਜਾਣੀ ਹੁੰਦੀ, ਜਾਂ ਐਲਗੋਰਦਿਮਕਿ ਢੰਗ ਨਾਲ ਤਆਿਰ ਕੀਤੀ ਜਾਣੀ ਹੁੰਦੀ।

LLM ਦੁਆਰਾ ਤਆਿਰ ਕੀਤੀ ਸਮੱਗਰੀ ਵੱਖ-ਵੱਖ ਰੂਪ ਲੈ ਸਕਦੀ ਹੈ, ਜਵਿੇਂ ਕ ਿਨਜਿੀ ਸਫਿਾਰਸ਼ਾਂ, ਗਤੀਸ਼ੀਲ ਉਤਪਾਦ ਵੇਰਵੇ, ਵਅਿਕਤੀਗਤ ਈਮੇਲ ਜਵਾਬ, ਜਾਂ ਇੱਥੋਂ ਤੱਕ ਕ ਿਪੂਰੇ ਲੇਖ ਜਾਂ ਬਲੌਗ ਪੋਸਟਾਂ। ਇਸ ਸਮੱਗਰੀ ਦੀਆਂ ਸਭ ਤੋਂ ਮੌਲਕਿ ਵਰਤੋਂਆਂ ਵੱਚਿੋਂ ਇੱਕ ਜੋ ਮੈਂ ਇੱਕ ਸਾਲ ਤੋਂ ਵੱਧ ਸਮਾਂ ਪਹਲਿਾਂ ਸ਼ੁਰੂ ਕੀਤੀ ਸੀ, ਉਹ ਹੈ UI ਐਲੀਮੈਂਟਸ ਜਵਿੇਂ ਫਾਰਮ ਲੇਬਲ, ਟੂਲਟਪਿਸ, ਅਤੇ ਹੋਰ ਕਸਿਮਾਂ ਦੇ ਵਅਿਾਖਆਿਤਮਕ ਟੈਕਸਟ ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਤਆਿਰ ਕਰਨਾ।

ਨਜਿੀਕਰਨ

ਪ੍ਰਸੰਗਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨਾਂ ਦੇ ਮੁੱਖ ਲਾਭਾਂ ਵੱਚਿੋਂ ਇੱਕ ਯੂਜ਼ਰਾਂ ਨੂੰ ਬਹੁਤ ਜ਼ਆਿਦਾ ਨਜਿੀ ਤਜਰਬੇ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਯੋਗਤਾ ਹੈ। ਯੂਜ਼ਰ-ਵਸ਼ਿੇਸ਼ ਪ੍ਰਸੰਗ ਦੇ ਆਧਾਰ 'ਤੇ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰਕੇ, ਇਹ ਪੈਟਰਨ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਵਅਿਕਤੀਗਤ ਯੂਜ਼ਰਾਂ ਦੀਆਂ ਰੁਚੀਆਂ, ਤਰਜੀਹਾਂ, ਅਤੇ ਅੰਤਰਕਰਿਆਿਵਾਂ ਦੇ ਅਨੁਸਾਰ ਸਮੱਗਰੀ ਨੂੰ ਢਾਲਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦੇ ਹਨ।

ਨਜਿੀਕਰਨ ਸਰਫ਼ ਆਮ ਸਮੱਗਰੀ ਵੱਚਿ ਯੂਜ਼ਰ ਦਾ ਨਾਮ ਪਾਉਣ ਤੋਂ ਕਤਿੇ ਵੱਧ ਹੈ। ਇਸ ਵੱਚਿ ਹਰ ਯੂਜ਼ਰ ਬਾਰੇ ਉਪਲਬਧ ਵਸਿਤ੍ਰਤਿ ਸੰਦਰਭ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਅਜਹਿੀ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰਨਾ ਸ਼ਾਮਲ ਹੈ ਜੋ ਉਨ੍ਹਾਂ ਦੀਆਂ ਵਸ਼ਿੇਸ਼ ਲੋੜਾਂ ਅਤੇ ਇੱਛਾਵਾਂ ਨਾਲ ਮੇਲ ਖਾਂਦੀ ਹੋਵੇ। ਇਸ ਸੰਦਰਭ ਵੱਚਿ ਕਈ ਕਾਰਕਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਜਵਿੇਂ ਕ:

1. **ਯੂਜ਼ਰ ਪ੍ਰੋਫਾਈਲ ਜਾਣਕਾਰੀ:** ਇਸ ਤਕਨੀਕ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦੇ ਸਭ ਤੋਂ ਆਮ ਪੱਧਰ 'ਤੇ, ਜਨਸੰਖਿਅਕ ਡਾਟਾ, ਰੁਚੀਆਂ, ਤਰਜੀਹਾਂ, ਅਤੇ ਹੋਰ ਪ੍ਰੋਫਾਈਲ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦੀ ਵਰਤੋਂ ਯੂਜ਼ਰ ਦੀ ਪੰਠਿਭੂਮੀ ਅਤੇ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦੇ ਅਨੁਕੂਲ ਸਮੱਗਰੀ ਤਿਆਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।
2. **ਵਵਿਹਾਰਕ ਡਾਟਾ:** ਐਪਲੀਕੇਸ਼ਨ ਨਾਲ ਯੂਜ਼ਰ ਦੀਆਂ ਪਛਿਲੀਆਂ ਅੰਤਰਕਰਿਆਵਾਂ, ਜਵਿੇਂ ਕਿ ਦੇਖੇ ਗਏ ਪੰਨੇ, ਕਲਕਿ ਕੀਤੇ ਲਿੰਕ, ਜਾਂ ਖਰੀਦੇ ਗਏ ਉਤਪਾਦ, ਉਨ੍ਹਾਂ ਦੇ ਵਵਿਹਾਰ ਅਤੇ ਰੁਚੀਆਂ ਬਾਰੇ ਮੁੱਲਵਾਨ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ। ਇਸ ਡਾਟਾ ਦੀ ਵਰਤੋਂ ਉਨ੍ਹਾਂ ਦੇ ਰੁਝੇਵੇਂ ਦੇ ਪੈਟਰਨਾਂ ਨੂੰ ਦਰਸਾਉਣ ਅਤੇ ਉਨ੍ਹਾਂ ਦੀਆਂ ਭਵਖਿੀ ਲੋੜਾਂ ਦੀ ਭਵਖਿਬਾਣੀ ਕਰਨ ਵਾਲੀ ਸਮੱਗਰੀ ਦੇ ਸੁਝਾਅ ਤਿਆਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।
3. **ਸੰਦਰਭਕਿ ਕਾਰਕ:** ਯੂਜ਼ਰ ਦਾ ਮੌਜੂਦਾ ਸੰਦਰਭ, ਜਵਿੇਂ ਕਿ ਉਨ੍ਹਾਂ ਦੀ ਲੋਕੇਸ਼ਨ, ਡਵਿਾਈਸ, ਦਨਿ ਦਾ ਸਮਾਂ, ਜਾਂ ਮੌਸਮ ਵੀ, ਸਮੱਗਰੀ ਨਿਰਮਾਣ ਪ੍ਰਕਰਿਆ ਨੂੰ ਪ੍ਰਭਾਵਤਿ ਕਰ ਸਕਦੇ ਹਨ। ਉਦਾਹਰਨ ਲਈ, ਇੱਕ ਯਾਤਰਾ ਐਪਲੀਕੇਸ਼ਨ ਵਚਿ ਇੱਕ AI ਵਰਕਰ ਹੋ ਸਕਦਾ ਹੈ ਜੋ ਯੂਜ਼ਰ ਦੀ ਮੌਜੂਦਾ ਲੋਕੇਸ਼ਨ ਅਤੇ ਮੌਜੂਦਾ ਮੌਸਮ ਦੀਆਂ ਸਥਤਿੀਆਂ ਦੇ ਆਧਾਰ 'ਤੇ ਨਜਿੀ ਸਫਿਾਰਸ਼ਾਂ ਤਿਆਰ ਕਰਨ ਦੇ ਯੋਗ ਹੈ।

ਇਨ੍ਹਾਂ ਸੰਦਰਭਕਿ ਕਾਰਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ, ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਿਰਮਾਣ ਪੈਟਰਨ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਹਰ ਵਅਿਕਤੀਗਤ ਯੂਜ਼ਰ ਲਈ ਵਿਸ਼ੇਸ਼ ਤੌਰ 'ਤੇ ਤਿਆਰ ਕੀਤੀ ਗਈ ਸਮੱਗਰੀ ਪ੍ਰਦਾਨ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦੇ ਹਨ। ਨਜਿੀਕਰਨ ਦੇ ਇਸ ਪੱਧਰ ਦੇ ਕਈ ਮਹੱਤਵਪੂਰਨ ਲਾਭ ਹਨ:

1. **ਵਧਿਆ ਰੁਝੇਵਾਂ:** ਨਜਿੀਕ੍ਰਤਿ ਸਮੱਗਰੀ ਯੂਜ਼ਰਾਂ ਦਾ ਧਿਆਨ ਖਚਿੰਦੀ ਹੈ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਐਪਲੀਕੇਸ਼ਨ ਨਾਲ ਜੁੜੇ ਰੱਖਦੀ ਹੈ। ਜਦੋਂ ਯੂਜ਼ਰਾਂ ਨੂੰ ਲੱਗਦਾ ਹੈ ਕਿ ਸਮੱਗਰੀ ਪ੍ਰਾਸੰਗਿਕ ਹੈ ਅਤੇ ਸੱਧਿੇ ਤੌਰ 'ਤੇ ਉਨ੍ਹਾਂ ਦੀਆਂ ਲੋੜਾਂ ਦੀ ਗੱਲ ਕਰਦੀ ਹੈ, ਤਾਂ ਉਹ ਐਪਲੀਕੇਸ਼ਨ ਨਾਲ ਅੰਤਰਕਰਿਆ ਕਰਨ ਅਤੇ ਇਸਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦੀ ਪੜਚੋਲ ਕਰਨ 'ਤੇ ਵਧੇਰੇ ਸਮਾਂ ਬਤਿਾਉਣ ਦੀ ਸੰਭਾਵਨਾ ਰੱਖਦੇ ਹਨ।
2. **ਬਹਿਤਰ ਯੂਜ਼ਰ ਸੰਤੁਸ਼ਟੀ:** ਨਜਿੀਕ੍ਰਤਿ ਸਮੱਗਰੀ ਦਰਸਾਉਂਦੀ ਹੈ ਕਿ ਐਪਲੀਕੇਸ਼ਨ ਯੂਜ਼ਰ ਦੀਆਂ ਵਲਿੱਖਣ ਲੋੜਾਂ ਨੂੰ ਸਮਝਦੀ ਹੈ ਅਤੇ ਉਨ੍ਹਾਂ ਦਾ ਖਆਿਲ ਰੱਖਦੀ ਹੈ। ਸਹਾਇਕ, ਜਾਣਕਾਰੀ ਭਰਪੂਰ, ਅਤੇ ਉਨ੍ਹਾਂ ਦੀਆਂ ਰੁਚੀਆਂ ਦੇ ਅਨੁਕੂਲ ਸਮੱਗਰੀ ਪ੍ਰਦਾਨ ਕਰਕੇ, ਐਪਲੀਕੇਸ਼ਨ ਯੂਜ਼ਰ ਸੰਤੁਸ਼ਟੀ ਨੂੰ ਵਧਾ ਸਕਦੀ ਹੈ ਅਤੇ ਆਪਣੇ ਯੂਜ਼ਰਾਂ ਨਾਲ ਮਜ਼ਬੂਤ ਸੰਬੰਧ ਬਣਾ ਸਕਦੀ ਹੈ।
3. **ਉੱਚ ਤਬਦੀਲੀ ਦਰਾਂ:** ਈ-ਕਾਮਰਸ ਜਾਂ ਮਾਰਕੀਟਿੰਗ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦੇ ਸੰਦਰਭ ਵਚਿ, ਨਜਿੀਕ੍ਰਤਿ ਸਮੱਗਰੀ ਤਬਦੀਲੀ ਦਰਾਂ 'ਤੇ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦੀ ਹੈ। ਯੂਜ਼ਰਾਂ ਨੂੰ ਉਨ੍ਹਾਂ ਦੀਆਂ ਤਰਜੀਹਾਂ ਅਤੇ ਵਵਿਹਾਰ ਦੇ ਅਨੁਕੂਲ ਉਤਪਾਦ, ਆਫਰ, ਜਾਂ ਸਫਿਾਰਸ਼ਾਂ ਪੇਸ਼ ਕਰਕੇ, ਐਪਲੀਕੇਸ਼ਨ ਯੂਜ਼ਰਾਂ ਦੇ ਇੱਛਤ ਕਾਰਵਾਈਆਂ ਕਰਨ, ਜਵਿੇਂ ਕਿ ਖਰੀਦਦਾਰੀ ਕਰਨ ਜਾਂ ਕਸਿੇ ਸੇਵਾ ਲਈ ਸਾਈਨ ਅੱਪ ਕਰਨ ਦੀ ਸੰਭਾਵਨਾ ਨੂੰ ਵਧਾ ਸਕਦੀ ਹੈ।

ਉਤਪਾਦਕਤਾ

ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨ ਰਚਨਾਤਮਕ ਪ੍ਰਕਰਿਆਵਾਂ ਵਿੱਚ ਮੈਨੂਅਲ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਅਤੇ ਸੰਪਾਦਨ ਦੀ ਲੋੜ ਨੂੰ ਘਟਾ ਕੇ ਕੁਝ ਕਸਿਮਾਂ ਦੀ ਉਤਪਾਦਕਤਾ ਨੂੰ ਕਾਫ਼ੀ ਵਧਾ ਸਕਦੇ ਹਨ। LLMs ਦੀ ਸ਼ਕਤੀ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਤੁਸੀਂ ਵੱਡੇ ਪੱਧਰ 'ਤੇ ਉੱਚ-ਗੁਣਵੱਤਾ ਵਾਲੀ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹੋ, ਜੋ ਤੁਹਾਡੇ ਸਮੱਗਰੀ ਨਰਿਮਾਤਾਵਾਂ ਅਤੇ ਡਵਿੈਲਪਰਾਂ ਨੂੰ ਉਬਾਊ ਮੈਨੂਅਲ ਕੰਮ ਕਰਨ 'ਤੇ ਖਰਚ ਕਰਨਾ ਪੈਣ ਵਾਲਾ ਸਮਾਂ ਅਤੇ ਮਹਿਨਤ ਬਚਾਉਂਦੀ ਹੈ।

ਰਵਾਇਤੀ ਤੌਰ 'ਤੇ, ਸਮੱਗਰੀ ਨਰਿਮਾਤਾਵਾਂ ਨੂੰ ਖੋਜ, ਲਖਿਣ, ਸੰਪਾਦਨ ਅਤੇ ਸਮੱਗਰੀ ਨੂੰ ਫਾਰਮੈਟ ਕਰਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ ਤਾਂ ਜੋ ਇਹ ਐਪਲੀਕੇਸ਼ਨ ਦੀਆਂ ਜ਼ਰੂਰਤਾਂ ਅਤੇ ਉਪਭੋਗਤਾ ਦੀਆਂ ਉਮੀਦਾਂ ਨੂੰ ਪੂਰਾ ਕਰ ਸਕੇ। ਇਹ ਪ੍ਰਕਰਿਆ ਸਮਾਂ ਲੈਣ ਵਾਲੀ ਅਤੇ ਸਰੋਤ-ਗਹਣਿ ਹੋ ਸਕਦੀ ਹੈ, ਖਾਸ ਕਰਕੇ ਜਦੋਂ ਸਮੱਗਰੀ ਦੀ ਮਾਤਰਾ ਵਧਦੀ ਹੈ।

ਹਾਲਾਂਕਿ, ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨਾਂ ਨਾਲ, ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪ੍ਰਕਰਿਆ ਨੂੰ ਕਾਫ਼ੀ ਹੱਦ ਤੱਕ ਸਵੈਚਾਲਤਿ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ ਦੱਤਿੇ ਗਏ ਪ੍ਰੇਰਕ ਨਰਿਦੇਸ਼ਾਂ ਅਤੇ ਦਸ਼ਿਾ-ਨਰਿਦੇਸ਼ਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਸੁਸੰਗਤ, ਵਆਿਕਰਣਕ ਤੌਰ 'ਤੇ ਸਹੀ, ਅਤੇ ਸੰਦਰਭਕਿ ਤੌਰ 'ਤੇ ਢੁਕਵੀਂ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਸਵੈਚਾਲਨ ਕਈ ਉਤਪਾਦਕਤਾ ਲਾਭ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ:

1. **ਮੈਨੂਅਲ ਮਹਿਨਤ ਵਿੱਚ ਕਮੀ:** ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਦੇ ਕੰਮਾਂ ਨੂੰ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਨੂੰ ਸੌਂਪ ਕੇ, ਸਮੱਗਰੀ ਨਰਿਮਾਤਾ ਸਮੱਗਰੀ ਰਣਨੀਤੀ, ਵਚਿਾਰ ਨਰਿਮਾਣ, ਅਤੇ ਗੁਣਵੱਤਾ ਭਰੋਸੇ ਵਰਗੇ ਉੱਚ-ਪੱਧਰੀ ਕੰਮਾਂ 'ਤੇ ਧਆਿਨ ਕੇਂਦਰਤਿ ਕਰ ਸਕਦੇ ਹਨ। ਉਹ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ ਨੂੰ ਜ਼ਰੂਰੀ ਸੰਦਰਭ, ਖਾਕੇ, ਅਤੇ ਦਸ਼ਿਾ-ਨਰਿਦੇਸ਼ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ ਅਤੇ ਇਸ ਨੂੰ ਅਸਲ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਨੂੰ ਸੰਭਾਲਣ ਦੇ ਸਕਦੇ ਹਨ। ਇਹ ਲਖਿਣ ਅਤੇ ਸੰਪਾਦਨ ਲਈ ਲੋੜੀਂਦੀ ਮੈਨੂਅਲ ਮਹਿਨਤ ਨੂੰ ਘਟਾਉਂਦਾ ਹੈ, ਜਸਿ ਨਾਲ ਸਮੱਗਰੀ ਨਰਿਮਾਤਾ ਵਧੇਰੇ ਉਤਪਾਦਕ ਅਤੇ ਕੁਸ਼ਲ ਹੋ ਸਕਦੇ ਹਨ।
2. **ਤੇਜ਼ ਸਮੱਗਰੀ ਨਰਿਮਾਣ:** ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ ਮਨੁੱਖੀ ਲੇਖਕਾਂ ਨਾਲੋਂ ਬਹੁਤ ਤੇਜ਼ੀ ਨਾਲ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ। ਸਹੀ ਪ੍ਰੇਰਕ ਨਰਿਦੇਸ਼ਾਂ ਅਤੇ ਦਸ਼ਿਾ-ਨਰਿਦੇਸ਼ਾਂ ਨਾਲ, ਇੱਕ ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ ਸਕਟਿਾਂ ਜਾਂ ਮਟਿਾਂ ਵਿੱਚ ਕਈ ਸਮੱਗਰੀ ਦੇ ਟੁਕੜੇ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ। ਇਹ ਗਤੀ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਬਹੁਤ ਤੇਜ਼ੀ ਨਾਲ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦੀ ਹੈ, ਜੋ ਉਪਭੋਗਤਾਵਾਂ ਦੀਆਂ ਮੰਗਾਂ ਅਤੇ ਲਗਾਤਾਰ ਬਦਲ ਰਹੇ ਡਜਿੀਟਲ ਪਰਦ੍ਰਿਸ਼ਿ ਨਾਲ ਤਾਲਮੇਲ ਬਣਾਈ ਰੱਖਦੀ ਹੈ।

> ਕੀ ਤੇਜ਼ ਸਮੱਗਰੀ ਨਰਿਮਾਣ "ਸਾਂਝੇ ਸਰੋਤਾਂ ਦੀ ਤ੍ਰਾਸਦੀ" ਵਾਲੀ ਸਥਤਿੀ ਵੱਲ ਲੈ ਜਾ ਰਹਿਾ ਹੈ ਜਥਿੱਥੇ ਇੰਟਰਨੈੱਟ ਅਜਹਿੀ ਸਮੱਗਰੀ ਨਾਲ ਭਰ ਰਹਿਾ ਹੈ ਜਸਿਨੂੰ ਕੋਈ ਨਹੀ ਪੜ੍ਹਦਾ। ਦੁੱਖ ਦੀ ਗੱਲ ਹੈ, ਮੈਨੂੰ ਲੱਗਦਾ ਹੈ ਕ ਿਜਵਾਬ ਹਾਂ ਹੈ।

3. **ਇਕਸਾਰਤਾ ਅਤੇ ਗੁਣਵੱਤਾ:** ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ ਆਸਾਨੀ ਨਾਲ ਸਮੱਗਰੀ ਨੂੰ ਸੋਧ ਸਕਦੇ ਹਨ ਤਾਂ ਜੋ ਇਹ ਸ਼ੈਲੀ, ਲਹਜ਼ਿ, ਅਤੇ ਗੁਣਵੱਤਾ ਵਚਿੱ ਇਕਸਾਰ ਹੋਵੇ। ਸਪੱਸ਼ਟ ਦਸਿ਼ਾ-ਨਰਿਦੇਸ਼ ਅਤੇ ਉਦਾਹਰਣਾਂ ਦੱਤਿੇ ਜਾਣ 'ਤੇ, ਕੁਝ ਕਸਿਮ ਦੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ (ਜਵਿੇਂ ਨਊਿਜ਼ਰੂਮ, ਪੀਆਰ, ਆਦ।) ਇਹ ਯਕੀਨੀ ਬਣਾ ਸਕਦੀਆਂ ਹਨ ਕ ਿਉਨ੍ਹਾਂ ਦੀ ਮਨੁੱਖ-ਨਰਿਮਤ ਸਮੱਗਰੀ ਉਨ੍ਹਾਂ ਦੀ ਬ੍ਰਾਂਡ ਆਵਾਜ਼ ਨਾਲ ਮੇਲ ਖਾਂਦੀ ਹੈ ਅਤੇ ਲੋੜੀਂਦੇ ਗੁਣਵੱਤਾ ਮਆਿਰਾਂ ਨੂੰ ਪੂਰਾ ਕਰਦੀ ਹੈ। ਇਹ ਇਕਸਾਰਤਾ ਵਆਿਪਕ ਸੰਪਾਦਨ ਅਤੇ ਸੋਧਾਂ ਦੀ ਲੋੜ ਨੂੰ ਘਟਾਉਂਦੀ ਹੈ, ਜੋ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪ੍ਰਕਰਿਆ ਵਚਿੱ ਸਮਾਂ ਅਤੇ ਮਹਨਿਤ ਬਚਾਉਂਦੀ ਹੈ।
4. **ਦੁਹਰਾਓ ਅਤੇ ਅਨੁਕੂਲਤਾ:** ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨ ਸਮੱਗਰੀ ਦੇ ਤੇਜ਼ ਦੁਹਰਾਓ ਅਤੇ ਅਨੁਕੂਲਤਾ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦੇ ਹਨ। ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ ਨੂੰ ਦੱਤਿੇ ਗਏ ਪ੍ਰੇਰਕ ਨਰਿਦੇਸ਼ਾਂ, ਖਾਕਆਿਂ, ਜਾਂ ਦਸਿ਼ਾ-ਨਰਿਦੇਸ਼ਾਂ ਨੂੰ ਵਵਿਸਥਤਿ ਕਰਕੇ, ਤੁਹਾਡੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਤੇਜ਼ੀ ਨਾਲ ਸਮੱਗਰੀ ਦੇ ਵੱਖ-ਵੱਖ ਰੂਪ ਤਆਿਰ ਕਰ ਸਕਦੀਆਂ ਹਨ ਅਤੇ ਵੱਖ-ਵੱਖ ਪਹੁੰਚਾਂ ਦੀ ਜਾਂਚ ਇੱਕ ਸਵੈਚਾਲਤਿ ਤਰੀਕੇ ਨਾਲ ਕਰ ਸਕਦੀਆਂ ਹਨ ਜੋ ਅਤੀਤ ਵਚਿੱ ਕਦੇ ਸੰਭਵ ਨਹੀ ਸੀ। ਇਹ ਦੁਹਰਾਉਣ ਵਾਲੀ ਪ੍ਰਕਰਿਆ ਤੇਜ਼ ਪ੍ਰਯੋਗ ਅਤੇ ਸਮੱਗਰੀ ਰਣਨੀਤੀਆਂ ਦੇ ਸੁਧਾਰ ਦੀ ਆਗਆਿ ਦੱਦਿੀ ਹੈ, ਜੋ ਸਮੇਂ ਦੇ ਨਾਲ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਅਤੇ ਆਕਰਸ਼ਕ ਸਮੱਗਰੀ ਵੱਲ ਲੈ ਜਾਂਦੀ ਹੈ। ਇਹ ਖਾਸ ਤਕਨੀਕ ਈ-ਕਾਮਰਸ ਵਰਗੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਇੱਕ ਪੂਰੀ ਖੇਡ-ਬਦਲਣ ਵਾਲੀ ਹੋ ਸਕਦੀ ਹੈ ਜੋ ਵਾਪਸੀ ਦਰਾਂ ਅਤੇ ਰੁਝੇਵੇਂ ਦੇ ਆਧਾਰ 'ਤੇ ਜੀਉਂਦੀਆਂ ਅਤੇ ਮਰਦੀਆਂ ਹਨ

> ਇਹ ਨੋਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕ ਿਭਾਵੇਂ ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨ ਉਤਪਾਦਕਤਾ ਨੂੰ ਬਹੁਤ ਵਧਾ ਸਕਦੇ ਹਨ, ਪਰ ਇਹ ਮਨੁੱਖੀ ਸ਼ਮੂਲੀਅਤ ਦੀ ਲੋੜ ਨੂੰ ਪੂਰੀ ਤਰ੍ਹਾਂ ਖਤਮ ਨਹੀ ਕਰਦੇ। ਸਮੱਗਰੀ ਨਰਿਮਾਤਾ ਅਤੇ ਸੰਪਾਦਕ ਸਮੁੱਚੀ ਸਮੱਗਰੀ ਰਣਨੀਤੀ ਨੂੰ ਪਰਭਾਸ਼ਤਿ ਕਰਨ, LLM ਨੂੰ ਮਾਰਗਦਰਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰਨ, ਅਤੇ ਤਆਿਰ ਕੀਤੀ ਸਮੱਗਰੀ ਦੀ ਗੁਣਵੱਤਾ ਅਤੇ ਉਚਤਿਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਵਚਿੱ ਮਹੱਤਵਪੂਰਨ ਭੂਮਕਿਾ ਨਭਿਾਉਂਦੇ ਹਨ।

ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਦੇ ਵਧੇਰੇ ਦੁਹਰਾਉਣ ਵਾਲੇ ਅਤੇ ਸਮਾਂ ਲੈਣ ਵਾਲੇ ਪਹਲੂਆਂ ਨੂੰ ਸਵੈਚਾਲਤਿ ਕਰਕੇ, ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨ ਕੀਮਤੀ ਮਨੁੱਖੀ ਸਮਾਂ ਅਤੇ ਸਰੋਤਾਂ ਨੂੰ ਮੁਕਤ ਕਰਦੇ ਹਨ ਜੋ ਉੱਚ-ਮੁੱਲ ਵਾਲੇ ਕੰਮਾਂ ਵੱਲ

ਮੁੜ ਨਰਿਦੇਸ਼ਿਤ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਇਹ ਵਧੀ ਹੋਈ ਉਤਪਾਦਕਤਾ ਤੁਹਾਨੂੰ ਵਰਤੋਂਕਾਰਾਂ ਨੂੰ ਵਧੇਰੇ ਨਜ਼ਿੰਜੀ ਅਤੇ ਆਕਰਸ਼ਕ ਸਮੱਗਰੀ ਪ੍ਰਦਾਨ ਕਰਨ ਦੇ ਨਾਲ-ਨਾਲ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦੀ ਹੈ।

ਤੇਜ਼ ਦੁਹਰਾਓ ਅਤੇ ਪ੍ਰਯੋਗ

ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨ ਤੁਹਾਨੂੰ ਵੱਖ-ਵੱਖ ਸਮੱਗਰੀ ਵੇਰੀਏਸ਼ਨਾਂ ਨਾਲ ਤੇਜ਼ੀ ਨਾਲ ਦੁਹਰਾਉਣ ਅਤੇ ਪ੍ਰਯੋਗ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦੇ ਹਨ, ਜੋ ਤੁਹਾਡੀ ਸਮੱਗਰੀ ਰਣਨੀਤੀ ਦੇ ਤੇਜ਼ ਅਨੁਕੂਲਨ ਅਤੇ ਸੁਧਾਰ ਦੀ ਆਗਿਆ ਦੇਂਦੇ ਹਨ। ਤੁਸੀਂ ਸਿਰਫ਼ ਮਾਡਲ ਨੂੰ ਪ੍ਰਦਾਨ ਕੀਤੇ ਸੰਦਰਭ, ਟੈਂਪਲੇਟ, ਜਾਂ ਦਿਸ਼ਾ-ਨਰਿਦੇਸ਼ਾਂ ਨੂੰ ਵਵਿਸਥਤਿ ਕਰਕੇ ਕੁਝ ਸਕਿੰਟਾਂ ਵੱਚਿ ਸਮੱਗਰੀ ਦੇ ਕਈ ਸੰਸਕਰਣ ਤਿਆਰ ਕਰ ਸਕਦੇ ਹੋ।

ਇਹ ਤੇਜ਼ ਦੁਹਰਾਓ ਸਮਰੱਥਾ ਕਈ ਮੁੱਖ ਲਾਭ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ:

1. **ਟੈਸਟਿੰਗ ਅਤੇ ਅਨੁਕੂਲਨ:** ਸਮੱਗਰੀ ਵੇਰੀਏਸ਼ਨਾਂ ਨੂੰ ਤੇਜ਼ੀ ਨਾਲ ਤਿਆਰ ਕਰਨ ਦੀ ਯੋਗਤਾ ਨਾਲ, ਤੁਸੀਂ ਆਸਾਨੀ ਨਾਲ ਵੱਖ-ਵੱਖ ਪਹੁੰਚਾਂ ਦੀ ਜਾਂਚ ਕਰ ਸਕਦੇ ਹੋ ਅਤੇ ਉਨ੍ਹਾਂ ਦੀ ਪ੍ਰਭਾਵਸ਼ੀਲਤਾ ਨੂੰ ਮਾਪ ਸਕਦੇ ਹੋ। ਉਦਾਹਰਣ ਲਈ, ਤੁਸੀਂ ਇੱਕ ਉਤਪਾਦ ਵੇਰਵੇ ਜਾਂ ਮਾਰਕੀਟਿੰਗ ਸੰਦੇਸ਼ ਦੇ ਕਈ ਸੰਸਕਰਣ ਤਿਆਰ ਕਰ ਸਕਦੇ ਹੋ, ਹਰ ਇੱਕ ਇੱਕ ਵਸ਼ਿੇਸ਼ ਵਰਤੋਂਕਾਰ ਸੈਗਮੈਂਟ ਜਾਂ ਸੰਦਰਭ ਲਈ ਅਨੁਕੂਲ ਕੀਤਾ ਗਿਆ। ਵਰਤੋਂਕਾਰ ਸ਼ਮੂਲੀਅਤ ਮੈਟ੍ਰਕਿਸ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਕੇ, ਜਵਿੇਂ ਕਿ ਕਲਿੱਕ-ਥਰੂ ਦਰਾਂ ਜਾਂ ਰੂਪਾਂਤਰਣ ਦਰਾਂ, ਤੁਸੀਂ ਸਭ ਤੋਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਸਮੱਗਰੀ ਵੇਰੀਏਸ਼ਨਾਂ ਦੀ ਪਛਾਣ ਕਰ ਸਕਦੇ ਹੋ ਅਤੇ ਆਪਣੀ ਸਮੱਗਰੀ ਰਣਨੀਤੀ ਨੂੰ ਉਸ ਅਨੁਸਾਰ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦੇ ਹੋ।

2. **ਏ/ਬੀ ਟੈਸਟਿੰਗ:** ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨ ਸਮੱਗਰੀ ਦੀ ਨਰਿਵਘਿਨ ਏ/ਬੀ ਟੈਸਟਿੰਗ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦੇ ਹਨ। ਤੁਸੀਂ ਸਮੱਗਰੀ ਦੇ ਦੋ ਜਾਂ ਵਧੇਰੇ ਵੇਰੀਏਸ਼ਨ ਤਿਆਰ ਕਰ ਸਕਦੇ ਹੋ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਵਰਤੋਂਕਾਰ ਸਮੂਹਾਂ ਨੂੰ ਬੇਤਰਤੀਬੇ ਢੰਗ ਨਾਲ ਪੇਸ਼ ਕਰ ਸਕਦੇ ਹੋ। ਹਰ ਵੇਰੀਏਸ਼ਨ ਦੇ ਪ੍ਰਦਰਸ਼ਨ ਦੀ ਤੁਲਨਾ ਕਰਕੇ, ਤੁਸੀਂ ਨਰਿਧਾਰਤਿ ਕਰ ਸਕਦੇ ਹੋ ਕਿ ਕਹਿੜੀ ਸਮੱਗਰੀ ਤੁਹਾਡੇ ਟੀਚਾ ਦਰਸ਼ਕਾਂ ਨਾਲ ਸਭ ਤੋਂ ਵਧੀਆ ਢੰਗ ਨਾਲ ਜੁੜਦੀ ਹੈ। ਇਹ ਡਾਟਾ-ਆਧਾਰਤਿ ਪਹੁੰਚ ਤੁਹਾਨੂੰ ਸੂਚਤਿ ਫੈਸਲੇ ਲੈਣ ਅਤੇ ਵਰਤੋਂਕਾਰ ਸ਼ਮੂਲੀਅਤ ਨੂੰ ਵੱਧ ਤੋਂ ਵੱਧ ਕਰਨ ਅਤੇ ਆਪਣੇ ਇੱਛਤਿ ਨਤੀਜਿਆਂ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਆਪਣੀ ਸਮੱਗਰੀ ਨੂੰ ਲਗਾਤਾਰ ਸੁਧਾਰਨ ਦੀ ਆਗਿਆ ਦੇਂਦੀ ਹੈ।
3. **ਨਜ਼ਿੰਜੀਕਰਨ ਪ੍ਰਯੋਗ:** ਤੇਜ਼ ਦੁਹਰਾਓ ਅਤੇ ਪ੍ਰਯੋਗ ਨਜ਼ਿੰਜੀਕਰਨ ਦੇ ਮਾਮਲੇ ਵੱਚਿ ਖਾਸ ਤੌਰ 'ਤੇ ਮੁੱਲਵਾਨ ਹਨ। ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨ ਨਾਲ, ਤੁਸੀਂ ਵੱਖ-ਵੱਖ ਵਰਤੋਂਕਾਰ ਸੈਗਮੈਂਟਾਂ, ਤਰਜੀਹਾਂ, ਜਾਂ

ਵਵਿਹਾਰਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਤੇਜ਼ੀ ਨਾਲ ਨੰਜੀ ਸਮੱਗਰੀ ਵੇਰੀਏਸ਼ਨਾਂ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹੋ। ਵੱਖ-ਵੱਖ ਨੰਜੀਕਰਨ ਰਣਨੀਤੀਆਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਕੇ, ਤੁਸੀਂ ਵਅਿਕਤੀਗਤ ਵਰਤੋਂਕਾਰਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨ ਅਤੇ ਅਨੁਕੂਲਤਿ ਤਜਰਬੇ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਸਭ ਤੋਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪਹੁੰਚਾਂ ਦੀ ਪਛਾਣ ਕਰ ਸਕਦੇ ਹੋ।

4. **ਬਦਲਦੇ ਰੁਝਾਨਾਂ ਨਾਲ ਅਨੁਕੂਲ ਹੋਣਾ:** ਤੇਜ਼ੀ ਨਾਲ ਦੁਹਰਾਉਣ ਅਤੇ ਪ੍ਰਯੋਗ ਕਰਨ ਦੀ ਯੋਗਤਾ ਤੁਹਾਨੂੰ ਚੁਸਤ ਰਹਣਿ ਅਤੇ ਬਦਲਦੇ ਰੁਝਾਨਾਂ ਅਤੇ ਵਰਤੋਂਕਾਰ ਤਰਜੀਹਾਂ ਨਾਲ ਅਨੁਕੂਲ ਹੋਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦੀ ਹੈ। ਜਦੋਂ ਨਵੇਂ ਵਸਿ਼, ਕੀਵਰਡ, ਜਾਂ ਵਰਤੋਂਕਾਰ ਵਵਿਹਾਰ ਉੱਭਰਦੇ ਹਨ, ਤੁਸੀਂ ਤੇਜ਼ੀ ਨਾਲ ਇਨ੍ਹਾਂ ਰੁਝਾਨਾਂ ਨਾਲ ਮੇਲ ਖਾਂਦੀ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹੋ। ਲਗਾਤਾਰ ਪ੍ਰਯੋਗ ਕਰਕੇ ਅਤੇ ਆਪਣੀ ਸਮੱਗਰੀ ਨੂੰ ਸੁਧਾਰ ਕੇ, ਤੁਸੀਂ ਲਗਾਤਾਰ ਵਕਿਸਤ ਹੋ ਰਹੇ ਡਜਿੀਟਲ ਖੇਤਰ ਵੰਚਿ ਪ੍ਰਸੰਗਕਿ ਰਹਾਿ ਸਕਦੇ ਹੋ ਅਤੇ ਮੁਕਾਬਲੇ ਵੰਚਿ ਬੜ੍ਹਤ ਬਣਾਈ ਰੱਖ ਸਕਦੇ ਹੋ।
5. **ਲਾਗਤ-ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰਯੋਗ:** ਰਵਾਇਤੀ ਸਮੱਗਰੀ ਪ੍ਰਯੋਗ ਵੰਚਿ ਆਮ ਤੌਰ 'ਤੇ ਕਾਫ਼ੀ ਸਮਾਂ ਅਤੇ ਸਰੋਤ ਲੱਗਦੇ ਹਨ, ਕਉਿਂਕਾਿ ਸਮੱਗਰੀ ਨਰਿਮਾਤਾਵਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਰੂਪਾਂ ਨੂੰ ਮੈਨੂਅਲ ਤੌਰ 'ਤੇ ਵਕਿਸਤਿ ਅਤੇ ਟੈਸਟ ਕਰਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਹਾਲਾਂਕਾਿ, ਸੰਦਰਭੀ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨਾਂ ਨਾਲ, ਪ੍ਰਯੋਗ ਦੀ ਲਾਗਤ ਬਹੁਤ ਘੱਟ ਹੋ ਜਾਂਦੀ ਹੈ। LLMs ਤੇਜ਼ੀ ਨਾਲ ਅਤੇ ਵੱਡੇ ਪੱਧਰ 'ਤੇ ਸਮੱਗਰੀ ਦੇ ਵੱਖ-ਵੱਖ ਰੂਪ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ, ਜੋ ਤੁਹਾਨੂੰ ਬਨਿਾਂ ਕਸਿੇ ਵੱਡੀ ਲਾਗਤ ਦੇ ਵਚਿਾਰਾਂ ਅਤੇ ਪਹੁੰਚਾਂ ਦੀ ਵਆਿਪਕ ਸ਼੍ਰੇਣੀ ਦੀ ਖੋਜ ਕਰਨ ਦੀ ਆਗਆਿ ਦੰਦੇ ਹਨ।

ਤੇਜ਼ ਦੁਹਰਾਓ ਅਤੇ ਪ੍ਰਯੋਗਾਂ ਦਾ ਵੱਧ ਤੋਂ ਵੱਧ ਲਾਭ ਲੈਣ ਲਈ, ਇੱਕ ਚੰਗੀ ਤਰ੍ਹਾਂ ਪਰਭਾਸ਼ਤਿ ਪ੍ਰਯੋਗਾਤਮਕ ਢਾਂਚਾ ਹੋਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਇਸ ਢਾਂਚੇ ਵੰਚਿ ਸ਼ਾਮਲ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ:

- ਹਰ ਪ੍ਰਯੋਗ ਲਈ ਸਪੱਸ਼ਟ ਉਦੇਸ਼ ਅਤੇ ਪਰਕਿਲਪਨਾਵਾਂ
- ਸਮੱਗਰੀ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਮਾਪਣ ਲਈ ਢੁਕਵੇਂ ਮੈਟ੍ਰਕਿਸ ਅਤੇ ਟਰੈਕਂਗਿ ਵਧਿੀਆਂ
- ਸਹੀ ਯੂਜ਼ਰਾਂ ਨੂੰ ਢੁਕਵੇਂ ਸਮੱਗਰੀ ਦੇ ਰੂਪ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਵਰਗੀਕਰਨ ਅਤੇ ਟਾਰਗੇਟਂਗਿ ਰਣਨੀਤੀਆਂ
- ਪ੍ਰਯੋਗਾਤਮਕ ਡੇਟਾ ਤੋਂ ਅੰਤਰਦ੍ਰਸ਼ਟਿੀ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਵਸ਼ਿਲੇਸ਼ਣ ਅਤੇ ਰਪਿਰਟਂਗਿ ਟੂਲ
- ਤੁਹਾਡੀ ਸਮੱਗਰੀ ਰਣਨੀਤੀ ਵੰਚਿ ਸੰਖਿਣ ਅਤੇ ਅਨੁਕੂਲਤਾ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨ ਦੀ ਪ੍ਰਕਰਿਆ

ਤੇਜ਼ ਦੁਹਰਾਓ ਅਤੇ ਪ੍ਰਯੋਗਾਂ ਨੂੰ ਅਪਣਾ ਕੇ, ਤੁਸੀਂ ਲਗਾਤਾਰ ਆਪਣੀ ਸਮੱਗਰੀ ਨੂੰ ਸੁਧਾਰ ਅਤੇ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦੇ ਹੋ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦੇ ਹੋਏ ਕਾਿ ਇਹ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਟੀਚਆਿਂ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਵੰਚਿ ਰੁਚੀਕਰ, ਪ੍ਰਸੰਗਕਿ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਰਹੇ। ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਲਈ ਇਹ ਚੁਸਤ ਪਹੁੰਚ ਤੁਹਾਨੂੰ ਅੱਗੇ ਰਹਣਿ ਅਤੇ ਸ਼ਾਨਦਾਰ ਯੂਜ਼ਰ ਅਨੁਭਵ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਆਗਆਿ ਦੰਦੀ ਹੈ।

ਸਕੇਲੇਬਲਿਟੀ ਅਤੇ ਕੁਸ਼ਲਤਾ

ਜਵਿੇਂ-ਜਵਿੇਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਧਦੀਆਂ ਹਨ ਅਤੇ ਵਅਿਕਤੀਗਤ ਸਮੱਗਰੀ ਦੀ ਮੰਗ ਵਧਦੀ ਹੈ, ਸੰਦਰਭੀ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਦੇ ਕੁਸ਼ਲ ਸਕੇਲਿੰਗ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦੇ ਹਨ। LLMs ਮਨੁੱਖੀ ਸਰੋਤਾਂ ਵਚਿ ਅਨੁਪਾਤਕ ਵਾਧੇ ਦੀ ਲੋੜ ਤੋਂ ਬਨਿਾਂ, ਇੱਕੋ ਸਮੇਂ ਵੱਡੀ ਗਣਿਤੀ ਵਚਿ ਯੂਜ਼ਰਾਂ ਅਤੇ ਸੰਦਰਭਾਂ ਲਈ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਸਕੇਲੇਬਲਿਟੀ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਆਪਣੀਆਂ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਸਮਰੱਥਾਵਾਂ 'ਤੇ ਦਬਾਅ ਪਾਏ ਬਨਿਾਂ ਵਧਦੇ ਯੂਜ਼ਰ ਆਧਾਰ ਨੂੰ ਵਅਿਕਤੀਗਤ ਅਨੁਭਵ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਆਗਆਿ ਦੇਂਦੀ ਹੈ।

> ਧਆਿਨ ਦਓਿ ਕ ਿਸੰਦਰਭੀ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਦੀ ਵਰਤੋਂ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਨੂੰ "ਤੁਰੰਤ" ਅੰਤਰਰਾਸ਼ਟਰੀਕਰਨ ਕਰਨ ਲਈ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਅਸਲ ਵਚਿ, ਇਹ ਬਲਿਕੁਲ ਉਹੀ ਹੈ ਜੋ ਮੈਂ ਆਪਣੇ Instant18n Gem ਦੀ ਵਰਤੋਂ ਕਰਕੇ Olympia ਨੂੰ ਅੱਧੀ ਦਰਜਨ ਤੋਂ ਵੱਧ ਭਾਸ਼ਾਵਾਂ ਵਚਿ ਪੇਸ਼ ਕਰਨ ਲਈ ਕੀਤਾ, ਭਾਵੇਂ ਕ ਿਅਸੀਂ ਇੱਕ ਸਾਲ ਤੋਂ ਵੀ ਘੱਟ ਪੁਰਾਣੇ ਹਾਂ।

AI ਸੰਚਾਲਤਿ ਸਥਾਨੀਕਰਨ

ਜੇਕਰ ਤੁਸੀਂ ਮੈਨੂੰ ਇੱਕ ਪਲ ਲਈ ਫ਼ਖ਼ਰ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦਓਿ, ਤਾਂ ਮੈਨੂੰ ਲਗਦਾ ਹੈ ਕ ਿRails ਐਪਸ ਲਈ ਮੇਰੀ Instant18n ਲਾਇਬ੍ਰੇਰੀ "ਸੰਦਰਭੀ ਸਮੱਗਰੀ ਨਰਿਮਾਣ" ਪੈਟਰਨ ਦੀ ਇੱਕ ਕ੍ਰਾਂਤੀਕਾਰੀ ਉਦਾਹਰਣ ਹੈ, ਜੋ ਐਪਲੀਕੇਸ਼ਨ ਵਕਿਾਸ ਵਚਿ AI ਦੀ ਰੂਪਾਂਤਰਕ ਸੰਭਾਵਨਾ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ। ਇਹ gem OpenAI ਦੇ GPT ਲਾਰਜ-ਲੈਂਗੁਏਜ ਮਾਡਲ ਦੀ ਸ਼ਕਤੀ ਦਾ ਲਾਭ ਲੈਂਦਾ ਹੈ ਤਾਂ ਜੋ Rails ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਚਿ ਅੰਤਰਰਾਸ਼ਟਰੀਕਰਨ ਅਤੇ ਸਥਾਨੀਕਰਨ ਨੂੰ ਸੰਭਾਲਣ ਦੇ ਤਰੀਕੇ ਨੂੰ ਕ੍ਰਾਂਤੀਕਾਰੀ ਬਣਾਇਆ ਜਾ ਸਕੇ।

ਰਵਾਇਤੀ ਤੌਰ 'ਤੇ, Rails ਐਪਲੀਕੇਸ਼ਨ ਦਾ ਅੰਤਰਰਾਸ਼ਟਰੀਕਰਨ ਕਰਨ ਵਚਿ ਅਨੁਵਾਦ ਕੁੰਜੀਆਂ ਨੂੰ ਮੈਨੂਅਲ ਤੌਰ 'ਤੇ ਪਰਭਾਸ਼ਤਿ ਕਰਨਾ ਅਤੇ ਹਰੇਕ ਸਮਰਥਤਿ ਭਾਸ਼ਾ ਲਈ ਸੰਬੰਧਤਿ ਅਨੁਵਾਦ ਪ੍ਰਦਾਨ ਕਰਨਾ ਸ਼ਾਮਲ ਹੁੰਦਾ ਹੈ। ਇਹ ਪ੍ਰਕਰਿਆ ਸਮਾਂ ਲੈਣ ਵਾਲੀ, ਸਰੋਤ-ਗਹਣਿ, ਅਤੇ ਅਸੰਗਤਤਾਵਾਂ ਦੇ ਅਧੀਨ ਹੋ ਸਕਦੀ ਹੈ। ਹਾਲਾਂਕ,ਿ Instant18n gem ਨਾਲ, ਸਥਾਨੀਕਰਨ ਦਾ ਪੈਰਾਡਾਈਮ ਪੂਰੀ ਤਰ੍ਹਾਂ ਮੁੜ ਪਰਭਾਸ਼ਤਿ ਕੀਤਾ ਗਆਿ ਹੈ।

ਇੱਕ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਕੇ, Instant18n ਜੈਮ ਤੁਹਾਨੂੰ ਟੈਕਸਟ ਦੇ ਸੰਦਰਭ ਅਤੇ ਅਰਥ ਦੇ ਆਧਾਰ 'ਤੇ ਤੁਰੰਤ ਅਨੁਵਾਦ ਤਆਿਰ ਕਰਨ ਦੀ ਸਮਰੱਥਾ ਦੇਂਦਾ ਹੈ। ਪਹਲਿਾਂ ਤੋਂ ਪਰਭਾਸ਼ਤਿ ਅਨੁਵਾਦ ਕੁੰਜੀਆਂ ਅਤੇ

ਸਥਰਿ ਅਨੁਵਾਦਾਂ 'ਤੇ ਨਰਿਭਰ ਕਰਨ ਦੀ ਬਜਾਏ, ਜੈਮ AI ਦੀ ਸ਼ਕਤੀ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਟੈਕਸਟ ਦਾ ਅਨੁਵਾਦ ਕਰਦਾ ਹੈ। ਇਸ ਪਹੁੰਚ ਦੇ ਕਈ ਮੁੱਖ ਫਾਇਦੇ ਹਨ:

1. **ਨਰਿਵਘਿਨ ਸਥਾਨੀਕਰਨ:** Instant18n ਜੈਮ ਨਾਲ, ਡਵਿੈਲਪਰਾਂ ਨੂੰ ਹਰ ਸਮਰਥਤਿ ਭਾਸ਼ਾ ਲਈ ਅਨੁਵਾਦ ਫਾਈਲਾਂ ਨੂੰ ਦਸਤੀ ਤੌਰ 'ਤੇ ਪਰਭਾਸ਼ਤਿ ਅਤੇ ਬਣਾਈ ਰੱਖਣ ਦੀ ਲੋੜ ਨਹੀਂ ਹੈ। ਜੈਮ ਦੱਤਿੇ ਗਏ ਟੈਕਸਟ ਅਤੇ ਇੱਛਤਿ ਟੀਚਾ ਭਾਸ਼ਾ ਦੇ ਆਧਾਰ 'ਤੇ ਆਪਣੇ ਆਪ ਅਨੁਵਾਦ ਤਆਿਰ ਕਰਦਾ ਹੈ, ਜੋ ਸਥਾਨੀਕਰਨ ਪ੍ਰਕਰਿਆ ਨੂੰ ਬਨਿਾਂ ਕਸਿੇ ਮੁਸ਼ਕਲ ਅਤੇ ਨਰਿਵਘਿਨ ਬਣਾਉਂਦਾ ਹੈ।
2. **ਸੰਦਰਭਕਿ ਸ਼ੁੱਧਤਾ:** AI ਨੂੰ ਅਨੁਵਾਦ ਕੀਤੇ ਜਾ ਰਹੇ ਟੈਕਸਟ ਦੀਆਂ ਬਾਰੀਕੀਆਂ ਨੂੰ ਸਮਝਣ ਲਈ ਕਾਫ਼ੀ ਸੰਦਰਭ ਦੱਤਿਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਆਲੇ-ਦੁਆਲੇ ਦੇ ਸੰਦਰਭ, ਮੁਹਾਵਰੇ, ਅਤੇ ਸੱਭਆਿਚਾਰਕ ਹਵਾਲਆਿਂ ਨੂੰ ਧਆਿਨ ਵੱਚਿ ਰੱਖ ਕੇ ਅਜਹਿੇ ਅਨੁਵਾਦ ਤਆਿਰ ਕਰ ਸਕਦਾ ਹੈ ਜੋ ਸਹੀ, ਕੁਦਰਤੀ ਲੱਗਣ ਵਾਲੇ ਅਤੇ ਸੰਦਰਭਕਿ ਤੌਰ 'ਤੇ ਢੁਕਵੇਂ ਹੋਣ।
3. **ਵਆਿਪਕ ਭਾਸ਼ਾ ਸਹਾਇਤਾ:** Instant18n ਜੈਮ GPT ਦੇ ਵਸ਼ਾਲ ਗਆਿਨ ਅਤੇ ਭਾਸ਼ਾਈ ਸਮਰੱਥਾਵਾਂ ਦਾ ਲਾਭ ਲੈਂਦਾ ਹੈ, ਜੋ ਭਾਸ਼ਾਵਾਂ ਦੀ ਵਆਿਪਕ ਸ਼੍ਰੇਣੀ ਵੱਚਿ ਅਨੁਵਾਦ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ। ਸਪੈਨਸ਼ਿ ਅਤੇ ਫਰੈਂਚ ਵਰਗੀਆਂ ਆਮ ਭਾਸ਼ਾਵਾਂ ਤੋਂ ਲੈ ਕੇ ਕਲਿੰਗਨ ਅਤੇ ਐਲਵਸ਼ਿ ਵਰਗੀਆਂ ਵਧੇਰੇ ਅਸਪਸ਼ਟ ਜਾਂ ਕਾਲਪਨਕਿ ਭਾਸ਼ਾਵਾਂ ਤੱਕ, ਜੈਮ ਅਨੁਵਾਦ ਦੀਆਂ ਵਭਨਿੰਨ ਲੋੜਾਂ ਨੂੰ ਸੰਭਾਲ ਸਕਦਾ ਹੈ।
4. **ਲਚਕਤਾ ਅਤੇ ਰਚਨਾਤਮਕਤਾ:** ਜੈਮ ਪਰੰਪਰਾਗਤ ਭਾਸ਼ਾ ਅਨੁਵਾਦਾਂ ਤੋਂ ਅੱਗੇ ਜਾਂਦਾ ਹੈ ਅਤੇ ਰਚਨਾਤਮਕ ਅਤੇ ਗੈਰ-ਪਰੰਪਰਾਗਤ ਸਥਾਨੀਕਰਨ ਵਕਿਲਪਾਂ ਦੀ ਆਗਆਿ ਦੱਦਿਾ ਹੈ। ਡਵਿੈਲਪਰ ਵੱਖ-ਵੱਖ ਸ਼ੈਲੀਆਂ, ਬੋਲੀਆਂ, ਜਾਂ ਕਾਲਪਨਕਿ ਭਾਸ਼ਾਵਾਂ ਵੱਚਿ ਵੀ ਟੈਕਸਟ ਦਾ ਅਨੁਵਾਦ ਕਰ ਸਕਦੇ ਹਨ, ਜੋ ਵਲਿੱਖਣ ਯੂਜ਼ਰ ਅਨੁਭਵਾਂ ਅਤੇ ਦਲਿਚਸਪ ਸਮੱਗਰੀ ਲਈ ਨਵੇਂ ਮੌਕੇ ਖੋਲ੍ਹਦਾ ਹੈ।
5. **ਪ੍ਰਦਰਸ਼ਨ ਅਨੁਕੂਲਨ:** Instant18n ਜੈਮ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਬਹਿਤਰ ਬਣਾਉਣ ਅਤੇ ਦੁਹਰਾਏ ਅਨੁਵਾਦਾਂ ਦੇ ਓਵਰਹੈੱਡ ਨੂੰ ਘਟਾਉਣ ਲਈ ਕੈਸ਼ਿੰਗ ਮੈਕੇਨਜ਼ਿਮ ਨੂੰ ਸ਼ਾਮਲ ਕਰਦਾ ਹੈ। ਅਨੁਵਾਦ ਕੀਤਾ ਟੈਕਸਟ ਕੈਸ਼ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਜੋ ਉਸੇ ਅਨੁਵਾਦ ਲਈ ਬਾਅਦ ਦੀਆਂ ਬੇਨਤੀਆਂ ਨੂੰ ਵਾਧੂ API ਕਾਲਾਂ ਦੀ ਲੋੜ ਤੋਂ ਬਨਿਾਂ ਤੇਜ਼ੀ ਨਾਲ ਸੇਵਾ ਕਰਨ ਦੀ ਆਗਆਿ ਦੱਦਿਾ ਹੈ।

Instant18n ਜੈਮ AI ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਸਥਾਨੀਕ੍ਤਿ ਸਮੱਗਰੀ ਤਆਿਰ ਕਰਨ ਲਈ "ਪ੍ਰਸੰਗਿਕ ਸਮੱਗਰੀ ਨਰਿਮਾਣ" ਪੈਟਰਨ ਦੀ ਸ਼ਕਤੀ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ। ਇਹ ਦਖਿਾਉਂਦਾ ਹੈ ਕ ਿਕਵਿੇਂ AI ਨੂੰ ਇੱਕ ਰੇਲਜ਼ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਮੁੱਖ ਕਾਰਜਸ਼ੀਲਤਾ ਵੱਚਿ ਏਕੀਕ੍ਰਤਿ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਜੋ ਡਵਿੈਲਪਰਾਂ ਦੁਆਰਾ ਅੰਤਰਰਾਸ਼ਟਰੀਕਰਨ ਅਤੇ ਸਥਾਨੀਕਰਨ ਨੂੰ ਅਪਣਾਉਣ ਦੇ ਤਰੀਕੇ ਨੂੰ ਬਦਲ ਰਹਿਾ ਹੈ।

ਦਸਤੀ ਅਨੁਵਾਦ ਪ੍ਰਬੰਧਨ ਦੀ ਲੋੜ ਨੂੰ ਖਤਮ ਕਰਕੇ ਅਤੇ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਤੁਰੰਤ ਅਨੁਵਾਦ ਨੂੰ ਸਮਰੱਥ ਬਣਾ ਕੇ, Instant18n ਜੈਮ ਡਵਿੈਲਪਰਾਂ ਦਾ ਮਹੱਤਵਪੂਰਨ ਸਮਾਂ ਅਤੇ ਯਤਨ ਬਚਾਉਂਦਾ ਹੈ। ਇਹ ਉਹਨਾਂ ਨੂੰ ਆਪਣੀ

ਐਪਲੀਕੇਸ਼ਨ ਦੀਆਂ ਮੁੱਖ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦੇ ਨਿਰਮਾਣ 'ਤੇ ਧਿਆਨ ਕੇਂਦਰਿਤ ਕਰਨ ਦੀ ਆਗਿਆ ਦੇਂਦਾ ਹੈ, ਜਦੋਂ ਕਿ ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕਿ ਸਥਾਨੀਕਰਨ ਪਹਿਲੂ ਨਿਰਵਘਿਨ ਅਤੇ ਸਹੀ ਢੰਗ ਨਾਲ ਸੰਭਾਲਿਆ ਜਾਵੇ।

ਯੂਜ਼ਰ ਟੈਸਟਿੰਗ ਅਤੇ ਫੀਡਬੈਕ ਦੀ ਮਹੱਤਤਾ

ਅੰਤ ਵਿੱਚ, ਯੂਜ਼ਰ ਟੈਸਟਿੰਗ ਅਤੇ ਫੀਡਬੈਕ ਦੀ ਮਹੱਤਤਾ ਨੂੰ ਹਮੇਸ਼ਾ ਧਿਆਨ ਵਿੱਚ ਰੱਖੋ। ਇਹ ਪੁਸ਼ਟੀ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਪ੍ਰਸੰਗਿਕ ਸਮੱਗਰੀ ਨਿਰਮਾਣ ਯੂਜ਼ਰ ਦੀਆਂ ਉਮੀਦਾਂ ਨੂੰ ਪੂਰਾ ਕਰਦਾ ਹੈ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਟੀਚਿਆਂ ਦੇ ਅਨੁਕੂਲ ਹੈ। ਯੂਜ਼ਰ ਦੀ ਅੰਤਰਦ੍ਰਸ਼ਿਟੀ ਅਤੇ ਵਿਸ਼ਲੇਸ਼ਣ ਦੇ ਆਧਾਰ 'ਤੇ ਤਿਆਰ ਕੀਤੀ ਸਮੱਗਰੀ ਨੂੰ ਲਗਾਤਾਰ ਦੁਹਰਾਓ ਅਤੇ ਸੁਧਾਰੋ। ਜੇਕਰ ਤੁਸੀਂ ਵੱਡੇ ਪੱਧਰ 'ਤੇ ਗਤੀਸ਼ੀਲ ਸਮੱਗਰੀ ਤਿਆਰ ਕਰ ਰਹੇ ਹੋ ਜਿਸ ਨੂੰ ਤੁਹਾਡੇ ਅਤੇ ਤੁਹਾਡੀ ਟੀਮ ਦੁਆਰਾ ਦਸਤੀ ਤੌਰ 'ਤੇ ਪ੍ਰਮਾਣਿਤ ਕਰਨਾ ਅਸੰਭਵ ਹੋਵੇਗਾ, ਤਾਂ ਅਜਿਹੇ ਫੀਡਬੈਕ ਮੈਕੇਨਿਜ਼ਮ ਸ਼ਾਮਲ ਕਰਨ 'ਤੇ ਵਿਚਾਰ ਕਰੋ ਜੋ ਯੂਜ਼ਰਾਂ ਨੂੰ ਅਜੀਬ ਜਾਂ ਗਲਤ ਸਮੱਗਰੀ ਦੀ ਰਿਪੋਰਟ ਕਰਨ ਦੀ ਆਗਿਆ ਦੇਂਦੇ ਹਨ, ਇਸ ਦੇ ਨਾਲ ਇਹ ਵੀ ਦੱਸਣ ਕਿ ਕਿਉਂ। ਉਹ ਕੀਮਤੀ ਫੀਡਬੈਕ ਇੱਕ AI ਵਰਕਰ ਨੂੰ ਵੀ ਦਿੱਤਾ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਸਮੱਗਰੀ ਤਿਆਰ ਕਰਨ ਵਾਲੇ ਕੰਪੋਨੈਂਟ ਵਿੱਚ ਸਮਾਯੋਜਨ ਕਰਨ ਦਾ ਕੰਮ ਕਰਦਾ ਹੈ!

ਜਨਰੇਟਵਿ ਯੂਆਈ

ਅੱਜਕੱਲ੍ਹ ਧਿਆਨ ਇੰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਯੂਜ਼ਰ ਸਮੂਲੀਅਤ ਲਈ ਹੁਣ ਅਜਿਹੇ ਸਾਫਟਵੇਅਰ ਅਨੁਭਵਾਂ ਦੀ ਲੋੜ ਹੈ ਜੋ ਨਾ ਸਿਰਫ ਸਰਲ ਅਤੇ ਸਹਜਿ ਹੋਣ, ਬਲਕਿ ਵਿਅਕਤੀਗਤ ਲੋੜਾਂ, ਤਰਜੀਹਾਂ ਅਤੇ ਸੰਦਰਭਾਂ ਦੇ ਅਨੁਸਾਰ ਵੀ ਵਸ਼ਿਸ਼ ਤੌਰ ’ਤੇ ਨਿੱਜੀ ਹੋਣ। ਨਤੀਜੇ ਵਜੋਂ, ਡਜ਼ਾਈਨਰ ਅਤੇ ਡਵੈਲਪਰ ਅਜਿਹੇ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ! ਬਣਾਉਣ ਦੀ ਚੁਣੌਤੀ ਦਾ ਸਾਹਮਣਾ ਕਰ ਰਹੇ ਹਨ ਜੋ ਵੱਡੇ ਪੱਧਰ ’ਤੇ ਹਰ ਯੂਜ਼ਰ ਦੀਆਂ ਵਲਿੱਖਣ ਲੋੜਾਂ ਦੇ ਅਨੁਕੂਲ ਹੋ ਸਕਣ।

ਜਨਰੇਟਵਿ ਯੂਆਈ (ਜੈਨਯੂਆਈ) ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ ਡਜ਼ਾਈਨ! ਦਾ ਇੱਕ ਸੱਚਮੁੱਚ ਕ੍ਰਾਂਤੀਕਾਰੀ ਪਹੁੰਚ ਹੈ ਜੋ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ (ਐੱਲਐੱਲਐੱਮ) ਦੀ ਸ਼ਕਤੀ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਤੁਰੰਤ ਨਿੱਜੀ ਅਤੇ ਗਤੀਸ਼ੀਲ ਯੂਜ਼ਰ ਅਨੁਭਵ ਬਣਾਉਂਦਾ ਹੈ। ਮੈਂ ਇਸ ਕਿਤਾਬ ਵਿੱਚ ਜੈਨਯੂਆਈ ਬਾਰੇ ਘੱਟੋ-ਘੱਟ ਇੱਕ ਮੁੱਢਲੀ ਜਾਣਕਾਰੀ ਦੇਣਾ ਯਕੀਨੀ ਬਣਾਉਣਾ ਚਾਹੁੰਦਾ ਸੀ, ਕਿਉਂਕਿ ਮੇਰਾ ਮੰਨਣਾ ਹੈ ਕਿ ਇਹ ਐਪਲੀਕੇਸ਼ਨ ਡਜ਼ਾਈਨ ਅਤੇ ਫਰੇਮਵਰਕਸ ਦੇ ਖੇਤਰ ਵਿੱਚ ਮੌਜੂਦ ਸਭ ਤੋਂ ਨਵੇਂ ਮੌਕਿਆਂ ਵਿੱਚੋਂ ਇੱਕ ਹੈ। ਮੈਨੂੰ ਯਕੀਨ ਹੈ ਕਿ ਇਸ ਵਸ਼ਿਸ਼ ਖੇਤਰ ਵਿੱਚ ਦਰਜਨਾਂ ਜਾਂ ਇਸ ਤੋਂ ਵੱਧ ਨਵੇਂ ਸਫਲ ਵਪਾਰਕ ਅਤੇ ਓਪਨ-ਸੋਰਸ ਪ੍ਰੋਜੈਕਟ ਸਾਹਮਣੇ ਆਉਣਗੇ।

ਆਪਣੇ ਮੂਲ ਰੂਪ ਵਿੱਚ, ਜੈਨਯੂਆਈ ਪ੍ਰਸੰਗਿਕ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਦੇ ਸਧਿਾਂਤਾਂ ਨੂੰ ਉੱਨਤ ਏਆਈ ਤਕਨੀਕਾਂ ਨਾਲ

ਜੋੜਦਾ ਹੈ ਤਾਂ ਜੋ ਯੂਜ਼ਰ ਦੇ ਸੰਦਰਭ, ਤਰਜੀਹਾਂ ਅਤੇ ਟੀਚਿਆਂ ਦੀ ਡੂੰਘੀ ਸਮਝ ਦੇ ਆਧਾਰ 'ਤੇ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ ਤੱਤਾਂ, ਜਵਿੇਂ ਕਿ ਟੈਕਸਟ, ਚੱਤਿਰ ਅਤੇ ਲੇਆਉਟ, ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਤਆਿਰ ਕੀਤਾ ਜਾ ਸਕੇ। ਜੈਨਯੂਆਈ ਡਜ਼ਾਈਨਰਾਂ ਅਤੇ ਡਵਿੈਲਪਰਾਂ ਨੂੰ ਅਜਹਿ ਇੰਟਰਫੇਸ ਬਣਾਉਣ ਦੇ ਯੋਗ ਬਣਾਉਦਾ ਹੈ ਜੋ ਯੂਜ਼ਰ ਦੀਆਂ ਕਰਿਆਵਾਂ ਦੇ ਜਵਾਬ ਵੱਚਿ ਅਨੁਕੂਲ ਹੁੰਦੇ ਹਨ ਅਤੇ ਵਕਿਸਤ ਹੁੰਦੇ ਹਨ, ਜੋ ਨੱਜਿੀਕਰਨ ਦਾ ਇੱਕ ਅਜਹਿਾ ਪੱਧਰ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ ਜੋ ਪਹਲਿਾਂ ਪ੍ਰਾਪਤ ਨਹੀਂ ਸੀ।

ਜੈਨਯੂਆਈ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ ਡਜ਼ਾਈਨ ਦੇ ਸਾਡੇ ਪਹੁੰਚ ਵੱਚਿ ਇੱਕ ਮੌਲਕਿ ਬਦਲਾਅ ਦਰਸਾਉਦਾ ਹੈ। ਸਮੂਹਾਂ ਲਈ ਡਜ਼ਾਈਨ ਕਰਨ ਦੀ ਬਜਾਏ, ਜੈਨਯੂਆਈ ਸਾਨੂੰ ਵਅਿਕਤੀ ਲਈ ਡਜ਼ਾਈਨ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦੱਦਿਾ ਹੈ। ਨੱਜਿੀ ਸਮੱਗਰੀ ਅਤੇ ਇੰਟਰਫੇਸਾਂ ਵੱਚਿ ਅਜਹਿ ਯੂਜ਼ਰ ਅਨੁਭਵ ਬਣਾਉਣ ਦੀ ਸਮਰੱਥਾ ਹੈ ਜੋ ਹਰ ਯੂਜ਼ਰ ਨਾਲ ਡੂੰਘੇ ਪੱਧਰ 'ਤੇ ਜੁੜਦੇ ਹਨ, ਜੋ ਸ਼ਮੂਲੀਅਤ, ਸੰਤੁਸ਼ਟੀ ਅਤੇ ਵਫ਼ਾਦਾਰੀ ਨੂੰ ਵਧਾਉਂਦੇ ਹਨ।

ਇੱਕ ਅਤ-ਿਆਧੁਨਕਿ ਤਕਨੀਕ ਦੇ ਤੌਰ 'ਤੇ, ਜੈਨਯੂਆਈ ਵੱਲ ਤਬਦੀਲੀ ਸੰਕਲਪਕ ਅਤੇ ਵਵਿਹਾਰਕ ਚੁਣੌਤੀਆਂ ਨਾਲ ਭਰੀ ਹੋਈ ਹੈ। ਡਜ਼ਾਈਨ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਏਆਈ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨਾ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਕਿ ਤਆਿਰ ਕੀਤੇ ਇੰਟਰਫੇਸ ਨਾ ਸਰਿਫ ਨੱਜਿੀ ਹੋਣ ਬਲਕਿ ਵਰਤਣਯੋਗ, ਪਹੁੰਚਯੋਗ ਅਤੇ ਸਮੁੱਚੇ ਬ੍ਰਾਂਡ ਅਤੇ ਯੂਜ਼ਰ ਅਨੁਭਵ ਦੇ ਅਨੁਕੂਲ ਵੀ ਹੋਣ, ਇਹ ਸਾਰੀਆਂ ਅਜਹਿੀਆਂ ਚੁਣੌਤੀਆਂ ਹਨ ਜੋ ਜੈਨਯੂਆਈ ਨੂੰ ਬਹੁਤਆਿਂ ਲਈ ਨਹੀਂ, ਸਗੋਂ ਕੁਝ ਲਈ ਇੱਕ ਖੋਜ ਬਣਾਉਂਦੀਆਂ ਹਨ। ਇਸ ਤੋਂ ਇਲਾਵਾ, ਏਆਈ ਦੀ ਸ਼ਮੂਲੀਅਤ ਡੇਟਾ ਗੋਪਨੀਯਤਾ, ਪਾਰਦਰਸ਼ਤਾ, ਅਤੇ ਸ਼ਾਇਦ ਨੈਤਕਿ ਪ੍ਰਭਾਵਾਂ ਬਾਰੇ ਸਵਾਲ ਖੜ੍ਹੇ ਕਰਦੀ ਹੈ।

ਚੁਣੌਤੀਆਂ ਦੇ ਬਾਵਜੂਦ, ਵੱਡੇ ਪੱਧਰ 'ਤੇ ਨੱਜਿੀ ਅਨੁਭਵ ਡਜਿੀਟਲ ਉਤਪਾਦਾਂ ਅਤੇ ਸੇਵਾਵਾਂ ਨਾਲ ਸਾਡੀ ਅੰਤਰਕਰਿਆ ਦੇ ਤਰੀਕੇ ਨੂੰ ਪੂਰੀ ਤਰ੍ਹਾਂ ਬਦਲਣ ਦੀ ਸ਼ਕਤੀ ਰੱਖਦੇ ਹਨ। ਇਹ ਸਮਾਵੇਸ਼ੀ ਅਤੇ ਪਹੁੰਚਯੋਗ ਇੰਟਰਫੇਸ ਬਣਾਉਣ ਦੀਆਂ ਸੰਭਾਵਨਾਵਾਂ ਖੋਲ੍ਹਦਾ ਹੈ ਜੋ ਉਪਭੋਗਤਾਵਾਂ ਦੀਆਂ ਵੱਖ-ਵੱਖ ਲੋੜਾਂ ਨੂੰ ਪੂਰਾ ਕਰਦੇ ਹਨ, ਭਾਵੇਂ ਉਨ੍ਹਾਂ ਦੀਆਂ ਯੋਗਤਾਵਾਂ, ਪਛਿੱਕੜ, ਜਾਂ ਤਰਜੀਹਾਂ ਕੁਝ ਵੀ ਹੋਣ।

ਇਸ ਅਧਆਿਇ ਵੱਚਿ, ਅਸੀਂ GenUI ਦੀ ਧਾਰਨਾ ਦੀ ਪੜਚੋਲ ਕਰਾਂਗੇ, ਕੁਝ ਪ੍ਰਮੁੱਖ ਵਸ਼ਿੇਸ਼ਤਾਵਾਂ, ਮੁੱਖ ਲਾਭ, ਅਤੇ ਸੰਭਾਵੀ ਚੁਣੌਤੀਆਂ ਦੀ ਜਾਂਚ ਕਰਾਂਗੇ। ਅਸੀਂ GenUI ਦੇ ਸਭ ਤੋਂ ਬੁਨਆਿਦੀ ਅਤੇ ਪਹੁੰਚਯੋਗ ਰੂਪ 'ਤੇ ਵਚਿਾਰ ਕਰਕੇ ਸ਼ੁਰੂ ਕਰਦੇ ਹਾਂ: ਰਵਾਇਤੀ ਤੌਰ 'ਤੇ ਡਜ਼ਾਈਨ ਕੀਤੇ ਅਤੇ ਲਾਗੂ ਕੀਤੇ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸਾਂ ਲਈ ਟੈਕਸਟ ਕਾਪੀ ਤਆਿਰ ਕਰਨਾ।

ਯੂਜ਼ਰ ਇੰਟਰਫੇਸਾਂ ਲਈ ਕਾਪੀ ਤਆਿਰ ਕਰਨਾ

ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਕਰੋਮ ਵੱਚਿ ਮੌਜੂਦ ਟੈਕਸਟ ਐਲੀਮੈਂਟਸ, ਜਵਿੇਂ ਫਾਰਮ ਲੇਬਲ, ਟੂਲਟਪਿਸ, ਅਤੇ ਵਆਿਖਆਿਤਮਕ ਟੈਕਸਟ, ਆਮ ਤੌਰ 'ਤੇ ਟੈਂਪਲੇਟਸ ਜਾਂ UI ਕੰਪੋਨੈਂਟਸ ਵੱਚਿ ਹਾਰਡਕੋਡ ਕੀਤੇ ਜਾਂਦੇ ਹਨ, ਜੋ ਸਾਰੇ

ਉਪਭੋਗਤਾਵਾਂ ਲਈ ਇੱਕ ਸਥਰਿ ਪਰ ਆਮ ਅਨੁਭਵ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ। ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪੈਟਰਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਤੁਸੀਂ ਇਨ੍ਹਾਂ ਸਥਰਿ ਐਲੀਮੈਂਟਸ ਨੂੰ ਗਤੀਸ਼ੀਲ, ਸੰਦਰਭ-ਜਾਗਰੂਕ, ਅਤੇ ਨਜ਼ਿੀ ਕੰਪੋਨੈਂਟਸ ਵਚਿ ਬਦਲ ਸਕਦੇ ਹੋ।

ਨਜ਼ਿੀ ਫਾਰਮ

ਫਾਰਮ ਵੈੱਬ ਅਤੇ ਮੋਬਾਈਲ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦਾ ਇੱਕ ਸਰਵਵਆਿਪੀ ਹਸਿਾ ਹਨ, ਜੋ ਉਪਭੋਗਤਾ ਇਨਪੁਟ ਇਕੱਠਾ ਕਰਨ ਦਾ ਮੁੱਖ ਸਾਧਨ ਹਨ। ਹਾਲਾਂਕ,ਿ ਰਵਾਇਤੀ ਫਾਰਮ ਅਕਸਰ ਇੱਕ ਆਮ ਅਤੇ ਨਰਿਵਅਿਕਤੀਗਤ ਅਨੁਭਵ ਪੇਸ਼ ਕਰਦੇ ਹਨ, ਜਸਿ ਵਚਿ ਮਆਿਰੀ ਲੇਬਲ ਅਤੇ ਖੇਤਰ ਹੁੰਦੇ ਹਨ ਜੋ ਹਮੇਸ਼ਾ ਉਪਭੋਗਤਾ ਦੇ ਵਸ਼ਿੇਸ਼ ਸੰਦਰਭ ਜਾਂ ਲੋੜਾਂ ਨਾਲ ਮੇਲ ਨਹੀਂ ਖਾਂਦੇ। ਉਪਭੋਗਤਾ ਉਨ੍ਹਾਂ ਫਾਰਮਾਂ ਨੂੰ ਪੂਰਾ ਕਰਨ ਦੀ ਵਧੇਰੇ ਸੰਭਾਵਨਾ ਰੱਖਦੇ ਹਨ ਜੋ ਉਨ੍ਹਾਂ ਦੀਆਂ ਲੋੜਾਂ ਅਤੇ ਤਰਜੀਹਾਂ ਦੇ ਅਨੁਕੂਲ ਮਹਸੂਿਸ ਹੁੰਦੇ ਹਨ, ਜਸਿ ਨਾਲ ਉੱਚ ਰੂਪਾਂਤਰਣ ਦਰਾਂ ਅਤੇ ਉਪਭੋਗਤਾ ਸੰਤੁਸ਼ਟੀ ਪ੍ਰਾਪਤ ਹੁੰਦੀ ਹੈ।

ਹਾਲਾਂਕ,ਿ ਨਜ਼ਿੀਕਰਨ ਅਤੇ ਸਥਰਿਤਾ ਵਚਿਕਾਰ ਸੰਤੁਲਨ ਬਣਾਉਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਜਦੋਂ ਕ ਿਵਅਿਕਤੀਗਤ ਉਪਭੋਗਤਾਵਾਂ ਲਈ ਫਾਰਮਾਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣਾ ਲਾਭਦਾਇਕ ਹੋ ਸਕਦਾ ਹੈ, ਜਾਣੂਪਣ ਅਤੇ ਅਨੁਮਾਨਯੋਗਤਾ ਦਾ ਪੱਧਰ ਬਣਾਈ ਰੱਖਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਉਪਭੋਗਤਾਵਾਂ ਨੂੰ ਨਜ਼ਿੀ ਐਲੀਮੈਂਟਸ ਦੇ ਨਾਲ ਵੀ ਫਾਰਮਾਂ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਪਛਾਣਨ ਅਤੇ ਨੈਵੀਗੇਟ ਕਰਨ ਦੇ ਯੋਗ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ।

ਇੱਥੇ ਪ੍ਰੇਰਣਾ ਲਈ ਕੁਝ ਨਜ਼ਿੀ ਫਾਰਮ ਵਚਿਾਰ ਹਨ:

ਸੰਦਰਭਕਿ ਖੇਤਰ ਸੁਝਾਅ

GenUI ਭਵਖਿਬਾਣੀਆਂ ਵਜੋਂ ਬੁੱਧੀਮਾਨ ਖੇਤਰ ਸੁਝਾਅ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਉਪਭੋਗਤਾ ਦੀਆਂ ਪਛਿਲੀਆਂ ਅੰਤਰਕਰਿਆਵਾਂ, ਤਰਜੀਹਾਂ, ਅਤੇ ਡੇਟਾ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ, ਜੇਕਰ ਉਪਭੋਗਤਾ ਨੇ ਪਹਲਿਾਂ ਆਪਣਾ ਸ਼ਪਿਿੰਗ ਪਤਾ ਦਰਜ ਕੀਤਾ ਹੈ, ਤਾਂ ਫਾਰਮ ਉਨ੍ਹਾਂ ਦੀ ਸੁਰੱਖਅਿਤ ਕੀਤੀ ਜਾਣਕਾਰੀ ਨਾਲ ਸੰਬੰਧਤਿ ਖੇਤਰਾਂ ਨੂੰ ਆਪਣੇ ਆਪ ਭਰ ਸਕਦਾ ਹੈ। ਇਹ ਨਾ ਸਰ਼ਫ਼ ਸਮਾਂ ਬਚਾਉਂਦਾ ਹੈ, ਬਲਕ ਿਇਹ ਦਰਸਾਉਂਦਾ ਹੈ ਕ ਿਐਪਲੀਕੇਸ਼ਨ ਉਪਭੋਗਤਾ ਦੀਆਂ ਤਰਜੀਹਾਂ ਨੂੰ ਸਮਝਦੀ ਅਤੇ ਯਾਦ ਰੱਖਦੀ ਹੈ।

ਇੱਕ ਮੰਟਿ ਰੁਕੋ, ਕੀ ਇਹ ਤਕਨੀਕ AI ਦੀ ਵਰਤੋਂ ਕੀਤੇ ਬਨਿਾਂ ਨਹੀਂ ਕੀਤੀ ਜਾ ਸਕਦੀ? ਬੇਸ਼ੱਕ, ਪਰ AI ਨਾਲ ਇਸ ਤਰ੍ਹਾਂ ਦੀ ਕਾਰਜਸ਼ੀਲਤਾ ਨੂੰ ਚਲਾਉਣ ਦੀ ਖੂਬਸੂਰਤੀ ਦੋ-ਗੁਣੀ ਹੈ: 1) ਇਸਨੂੰ ਲਾਗੂ ਕਰਨਾ ਕਾਫ਼ੀ ਆਸਾਨ ਹੋ ਸਕਦਾ ਹੈ ਅਤੇ 2) ਤੁਹਾਡੇ UI ਦੇ ਬਦਲਣ ਅਤੇ ਵਕਿਸਤਿ ਹੋਣ ਦੇ ਨਾਲ ਇਹ ਕਾਫ਼ੀ ਲਚਕਦਾਰ ਹੋ ਸਕਦਾ ਹੈ।

ਆਓ ਆਪਣੇ ਸੋਚੇ ਹੋਏ ਆਰਡਰ ਹੈਂਡਲਿੰਗ ਸਿਸਟਮ ਲਈ ਇੱਕ ਸੇਵਾ ਬਣਾਈਏ, ਜੋ ਯੂਜ਼ਰ ਲਈ ਸਹੀ ਸ਼ਿਪਿੰਗ ਐਡਰੈੱਸ ਨੂੰ ਪਹਿਲਾਂ ਤੋਂ ਹੀ ਭਰਨ ਦੀ ਕੋਸ਼ਿਸ਼ ਕਰਦੀ ਹੈ।

```
class OrderShippingAddressSubscriber
  include Raix::ChatCompletion

  attr_accessor :order

  delegate :customer, to: :order

  DIRECTIVE = "You are a smart order processing assistant. Given the
  customer's order history, guess the most likely shipping address
  for the current order."

  def order_created(order)
    return unless order.pending? && order.shipping_address.blank?

    self.order = order

    transcript.clear
    transcript << { system: DIRECTIVE }
    transcript << { user: "Order History: #{order_history.to_json}" }
    transcript << { user: "Current Order: #{order.to_json}" }

    response = chat_completion
    apply_predicted_shipping_address(order, response)
  end

  private

  def apply_predicted_shipping_address(order, response)
    # extract the shipping address from the response...
    # ...and assume there's some sort of live update of the address fields
    order.update(shipping_address:)
  end

  def order_history
    customer.orders.successful.limit(100).map do |order|
      {
        date: order.date,
        description: order.description,
```

```
        shipping_address: order.shipping_address
      }
    end
  end
end
```

ਇਹ ਉਦਾਹਰਨ ਬਹੁਤ ਸਰਲ ਕੀਤੀ ਹੋਈ ਹੈ, ਪਰ ਜ਼ਿਆਦਾਤਰ ਮਾਮਲਿਆਂ ਲਈ ਕੰਮ ਕਰੇਗੀ। ਵਿਚਾਰ ਇਹ ਹੈ ਕਿ ਏ.ਆਈ. ਨੂੰ ਉਸੇ ਤਰ੍ਹਾਂ ਅੰਦਾਜ਼ਾ ਲਗਾਉਣ ਦੀਤਾ ਜਾਵੇ ਜਿਵੇਂ ਇੱਕ ਮਨੁੱਖ ਲਗਾਉਂਦਾ ਹੈ। ਇਸ ਨੂੰ ਸਪੱਸ਼ਟ ਕਰਨ ਲਈ ਕਿ ਮੈਂ ਕੀ ਕਹਿ ਰਿਹਾ ਹਾਂ, ਆਓ ਕੁਝ ਨਮੂਨਾ ਡਾਟਾ 'ਤੇ ਵਿਚਾਰ ਕਰੀਏ:

```
Order History:
[
  {"date": "2024-01-03", "description": "garden soil mix",
   "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-01-15", "description": "hardcover fiction novels",
   "shipping_address": "456 City Apt, Metroville"},
  {"date": "2024-01-22", "description": "baby diapers", "shipping_address":
   "789 Suburb St, Quietville"},
  {"date": "2024-02-01", "description": "organic vegetables",
   "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-02-17", "description": "mystery thriller book set",
   "shipping_address": "456 City Apt, Metroville"},
  {"date": "2024-02-25", "description": "baby wipes",
   "shipping_address": "789 Suburb St, Quietville"},
  {"date": "2024-03-05", "description": "flower seeds",
   "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-03-20", "description": "biographies",
   "shipping_address": "456 City Apt, Metroville"},
  {"date": "2024-03-30", "description": "baby formula",
   "shipping_address": "789 Suburb St, Quietville"},
  {"date": "2024-04-12", "description": "lawn fertilizer",
   "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-04-22", "description": "science fiction novels",
   "shipping_address": "456 City Apt, Metroville"},
  {"date": "2024-05-02", "description": "infant toys",
   "shipping_address": "789 Suburb St, Quietville"},
  {"date": "2024-05-14", "description": "outdoor grill",
   "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-05-29", "description": "literary classics",
  "shipping_address": "456 City Apt, Metroville"},
```

```
  {"date": "2024-06-11", "description": "baby clothes",
   "shipping_address": "789 Suburb St, Quietville"},
  {"date": "2024-07-01", "description": "watering can",
   "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-07-18", "description": "non-fiction essays",
  "shipping_address": "456 City Apt, Metroville"},
  {"date": "2024-07-28", "description": "baby bath items",
  "shipping_address": "789 Suburb St, Quietville"},
  {"date": "2024-08-09", "description": "herb garden kit",
  "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-08-24", "description": "children's books",
  "shipping_address": "456 City Apt, Metroville"}
]
```

ਕੀ ਤੁਸੀ ਡਾਟਾ ਵਚਿ ਪੈਟਰਨ ਵੇਖਆਿ? ਮੈ ਤੁਹਾਨੂੰ ਗਰੰਟੀ ਦਦਿਾ ਹਾਂ ਕਇਿ ਐਲਐਲਐਮ ਲਈ ਬਹੁਤ ਸੌਖਾ ਕੰਮ ਹੈ। ਇਸ ਨੂੰ ਦਖਿਾਉਣ ਲਈ, ਆਓ ਜੀਪੀਟੀ-4 ਤੋ ਪੁੱਛੀਏ ਕ "ਥਰਮਾਮੀਟਰ" ਲਈ ਸਭ ਤੋ ਸੰਭਾਵਤ ਸ਼ਪਿਿੰਗ ਐਡਰੈਸ ਕੀ ਹै।

```
From the order history you've provided, it looks like the purchases are
generally clustered into three main types based on the shipping addresses:

123 Country Lane, Rural Town - This address often orders garden and
outdoor-related items like soil mix, vegetables, flower seeds, lawn
fertilizer, an outdoor grill, a watering can, and an herb garden kit.

456 City Apt, Metroville - This address tends to order various types of
books, indicating an interest in reading across genres like fiction,
thrillers, biographies, science fiction, literary classics, non-fiction
essays, and children's books.

789 Suburb St, Quietville - The purchases for this address are consistently
baby-related products, such as diapers, wipes, baby formula, infant toys,
baby clothes, and baby bath items.

Given these patterns, a digital thermometer could potentially be a fit for
any of the addresses based on different needs: outdoor temperature tracking,
a tool for a passionate reader interested in science gadgets, or a health
tool for a baby. However, considering its common use in monitoring health,
especially for babies, it's most likely to be purchased by someone who's
already buying baby care products.
```

```

Therefore, the most likely shipping address for a digital thermometer, based
on the purchasing patterns, would be 789 Suburb St, Quietville.
```

ਜੇ ਤੁਸੀਂ ਇਹ ਸੋਚ ਰਹੇ ਹੋ ਕਿ ਇਸ ਕੰਮ ਲਈ GPT-4 ਵਰਗੇ ਮਹਿੰਗੇ ਮਾਡਲ ਦੀ ਵਰਤੋਂ ਕਰਨਾ ਲੋੜ ਤੋਂ ਵੱਧ ਹੈ, ਤਾਂ ਤੁਸੀਂ ਸਹੀ ਹੋ! ਮੈਂ ਇਹੀ ਪ੍ਰੌਮਪਟ Mistral 7B Instruct 'ਤੇ ਵਰਤਿਆ ਅਤੇ ਇਸ ਨੇ 75 ਟੋਕਨ ਪ੍ਰਤੀ ਸਕਿੰਟ ਦੀ ਰਫ਼ਤਾਰ ਨਾਲ, ਅਤੇ ਸਿਰਫ਼ $0.000218 USD ਦੀ ਬਹੁਤ ਘੱਟ ਕੀਮਤ 'ਤੇ ਹੇਠ ਲਿਖਿਆ ਜਵਾਬ ਦਿੱਤਾ।

```
Based on the given order history, it appears that the customer has received
orders for baby-related items (diapers, wipes, formula, clothes, bath items)
at the address "789 Suburb St, Quietville" and garden-related items (soil,
seeds, watering can, herb garden kit) at the address "123 Country Lane, Rural
Town". The customer has also received books (fiction, non-fiction, literary
classics, mystery-thriller, science fiction, biographies, children's books)
at the address "456 City Apt, Metroville".

Given that the new order is for a "thermometer", which doesn't seem to fit
into any of the categories that the customer has previously ordered, it's
hard to accurately predict the shipping address based solely on the order
history. However, given the 50%-50% split between baby-related and
garden-related items, it could somewhat lean towards the Baby-related items
address ("789 Suburb St, Quietville"). But remember, this is an assumption
and cannot be definitively confirmed without more context or information.
```

ਕੀ ਇਕ ਖਰੀਦਦਾਰੀ ਅਨੁਭਵ ਨੂੰ ਵਧੇਰੇ ਜਾਦੂਈ ਬਣਾਉਣ ਲਈ ਇਸ ਤਕਨੀਕ ਦੀ ਓਵਰਹੈੱਡ ਅਤੇ ਲਾਗਤ ਦਾ ਮੁੱਲ ਹੈ? ਕਈ ਆਨਲਾਈਨ ਰਟੇਲਰਾਂ ਲਈ, ਬਿਲਕੁਲ। ਅਤੇ ਇਸ ਤੋਂ ਜਾਪਦਾ ਹੈ ਕਿ AI ਕੰਪਿਊਟਿੰਗ ਦੀ ਲਾਗਤ ਘੱਟਦੀ ਹੀ ਜਾਵੇਗੀ, ਖਾਸ ਕਰਕੇ ਓਪਨ ਸੋਰਸ ਮਾਡਲ ਹੋਸਟਿੰਗ ਪ੍ਰਦਾਤਾਵਾਂ ਲਈ ਜੋ ਸਭ ਤੋਂ ਘੱਟ ਕੀਮਤ ਦੀ ਦੌੜ ਵਿੱਚ ਹਨ।

ਇਸ ਤਰ੍ਹਾਂ ਦੀ ਚੈਟ ਕੰਪਲੀਸ਼ਨ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣ ਲਈ Prompt Template ਅਤੇ StructuredIO ਦੇ ਨਾਲ Response Fencing ਦੀ ਵਰਤੋਂ ਕਰੋ।

ਅਨੁਕੂਲੀ ਫੀਲਡ ਕ੍ਰਮ

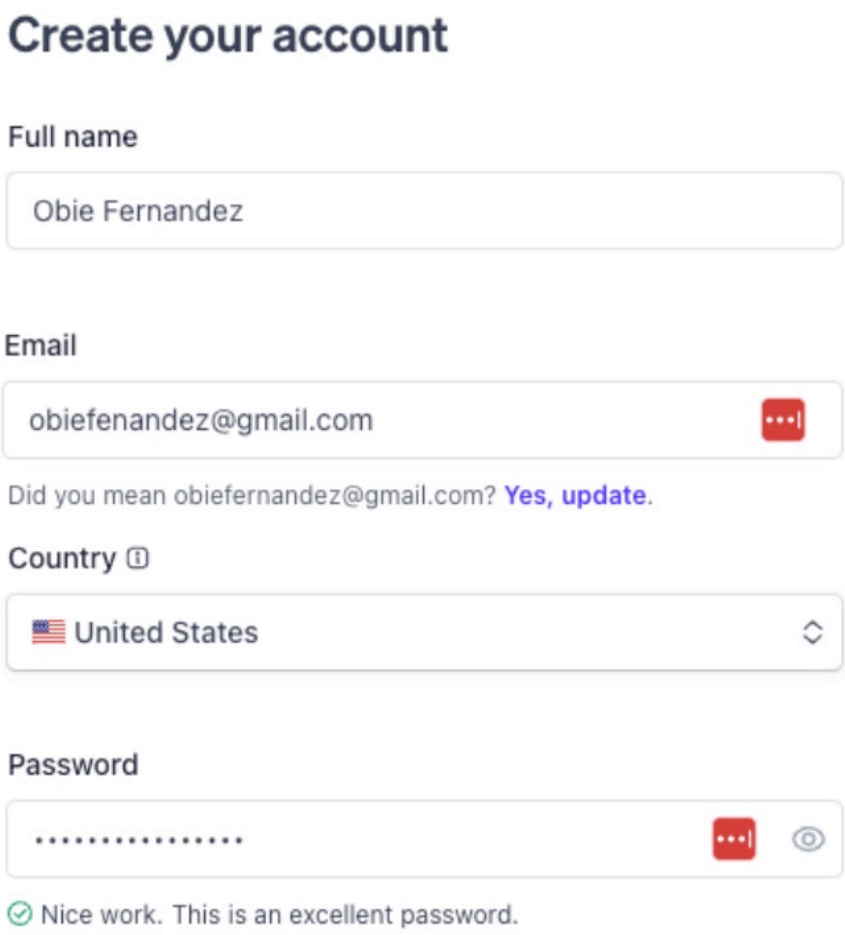
ਫਾਰਮ ਫੀਲਡਾਂ ਨੂੰ ਪੇਸ਼ ਕਰਨ ਦਾ ਕ੍ਰਮ ਯੂਜ਼ਰ ਦੇ ਅਨੁਭਵ ਅਤੇ ਪੂਰਤੀ ਦਰਾਂ 'ਤੇ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਪਾ ਸਕਦਾ ਹੈ। GenUI ਨਾਲ, ਤੁਸੀਂ ਯੂਜ਼ਰ ਦੇ ਸੰਦਰਭ ਅਤੇ ਹਰ ਫੀਲਡ ਦੀ ਮਹੱਤਤਾ ਦੇ ਆਧਾਰ 'ਤੇ ਫੀਲਡ ਕ੍ਰਮ ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦੇ ਹੋ। ਉਦਾਹਰਣ ਲਈ, ਜੇਕਰ ਯੂਜ਼ਰ ਇੱਕ ਫਟਿਨੈੱਸ ਐਪ ਲਈ ਰਜਿਸਟ੍ਰੇਸ਼ਨ ਫਾਰਮ ਭਰ ਰਹਿਾ ਹੈ, ਤਾਂ ਫਾਰਮ ਉਨ੍ਹਾਂ ਦੇ ਫਟਿਨੈੱਸ ਟੀਚਿਆਂ ਅਤੇ ਤਰਜੀਹਾਂ ਨਾਲ ਸੰਬੰਧਤਿ ਫੀਲਡਾਂ ਨੂੰ ਤਰਜੀਹ ਦੇ ਸਕਦਾ ਹੈ, ਜੋ ਪ੍ਰਕਰਿਆ ਨੂੰ ਵਧੇਰੇ ਪ੍ਰਸੰਗਕਿ ਅਤੇ ਦਲਿਚਸਪ ਬਣਾਉਂਦਾ ਹੈ।

ਨੰਿਜੀ ਮਾਈਕਰੋਕਾਪੀ

GenUI ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਫਾਰਮਾਂ ਨਾਲ ਜੁੜੇ ਨਰਿਦੇਸ਼ਾਤਮਕ ਟੈਕਸਟ, ਗਲਤੀ ਸੁਨੇਹੇ, ਅਤੇ ਹੋਰ ਮਾਈਕਰੋਕਾਪੀ ਨੂੰ ਵੀ ਨੰਿਜੀ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। "ਗਲਤ ਈਮੇਲ ਪਤਾ" ਵਰਗੇ ਆਮ ਗਲਤੀ ਸੁਨੇਹਿਆਂ ਦੀ ਬਜਾਏ, ਤੁਸੀਂ ਵਧੇਰੇ ਮਦਦਗਾਰ ਅਤੇ ਸੰਦਰਭਤਿ ਸੁਨੇਹੇ ਜਵਿੇਂ "ਕਰਿਪਾ ਕਰਕੇ ਆਪਣੀ ਆਰਡਰ ਪੁਸ਼ਟੀ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਇੱਕ ਵੈਧ ਈਮੇਲ ਪਤਾ ਦਰਜ ਕਰੋ" ਜਹਿੇ ਸੁਨੇਹੇ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹੋ। ਇਹ ਨੰਿਜੀ ਛੋਹਾਂ ਫਾਰਮ ਅਨੁਭਵ ਨੂੰ ਵਧੇਰੇ ਯੂਜ਼ਰ-ਅਨੁਕੂਲ ਅਤੇ ਘੱਟ ਨਰਿਾਸ਼ਾਜਨਕ ਬਣਾ ਸਕਦੀਆਂ ਹਨ।

ਨੰਿਜੀ ਪੁਸ਼ਟੀਕਰਨ

ਨੰਿਜੀ ਮਾਈਕਰੋਕਾਪੀ ਦੀਆਂ ਲਾਈਨਾਂ ਦੇ ਨਾਲ, ਤੁਸੀਂ ਫਾਰਮ ਨੂੰ ਅਜਹਿੇ ਤਰੀਕਿਆਂ ਨਾਲ ਪ੍ਰਮਾਣਤਿ ਕਰਨ ਲਈ AI ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹੋ ਜੋ ਜਾਦੂਈ ਜਾਪਦੇ ਹਨ। ਕਲਪਨਾ ਕਰੋ ਕ ਿਇੱਕ AI ਨੂੰ ਯੂਜ਼ਰ ਪ੍ਰੋਫਾਈਲ ਫਾਰਮ ਦੀ ਪੁਸ਼ਟੀ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦੰਿਤੀ ਜਾਵੇ, ਜੋ ਅਰਥ-ਪੂਰਨ ਪੱਧਰ 'ਤੇ ਸੰਭਾਵੀ ਗਲਤੀਆਂ ਦੀ ਜਾਂਚ ਕਰ ਰਹੀ ਹੋਵੇ।



ਚਤਿਰ 8. ਕੀ ਤੁਸੀਂ ਹੋ ਰਹੇ ਅਰਥ-ਪੂਰਨ ਪੁਸਟੀਕਰਨ ਨੂੰ ਦੇਖ ਸਕਦੇ ਹੋ?

ਕ੍ਰਮਕਿ ਪ੍ਰਗਟਾਵਾ

GenUI ਯੂਜ਼ਰ ਦੇ ਸੰਦਰਭ ਦੇ ਆਧਾਰ ’ਤੇ ਬੁੱਧੀਮਾਨੀ ਨਾਲ ਇਹ ਨਰਿਧਾਰਤ ਕਰ ਸਕਦਾ ਹੈ ਕਿ ਕਹਿੜੇ ਫਾਰਮ ਫੀਲਡ ਜ਼ਰੂਰੀ ਹਨ ਅਤੇ ਲੋੜ ਅਨੁਸਾਰ ਹੌਲੀ-ਹੌਲੀ ਵਾਧੂ ਫੀਲਡਾਂ ਨੂੰ ਪ੍ਰਗਟ ਕਰ ਸਕਦਾ ਹੈ। ਇਹ ਕ੍ਰਮਕਿ ਪ੍ਰਗਟਾਵਾ ਤਕਨੀਕ ਬੋਧਾਤਮਕ ਭਾਰ ਨੂੰ ਘਟਾਉਣ ਵੱਚਿ ਮਦਦ ਕਰਦੀ ਹੈ ਅਤੇ ਫਾਰਮ ਭਰਨ ਦੀ ਪ੍ਰਕਰਿਆ ਨੂੰ ਵਧੇਰੇ ਪ੍ਰਬੰਧਨਯੋਗ ਬਣਾਉਂਦੀ ਹੈ। ਉਦਾਹਰਣ ਲਈ, ਜੇ ਕੋਈ ਯੂਜ਼ਰ ਬੇਸਕਿ ਸਬਸਕ੍ਰਪਿਸ਼ਨ ਲਈ ਸਾਈਨ ਅੱਪ

ਕਰ ਰਹਿਾ ਹੈ, ਤਾਂ ਫਾਰਮ ਸ਼ੁਰੂ ਵਚਿ ਸਰਿਫ਼ ਜ਼ਰੂਰੀ ਫੀਲਡਾਂ ਨੂੰ ਪੇਸ਼ ਕਰ ਸਕਦਾ ਹੈ, ਅਤੇ ਜਵਿੇਂ-ਜਵਿੇਂ ਯੂਜ਼ਰ ਅੱਗੇ ਵਧਦਾ ਹੈ ਜਾਂ ਵਸਿ਼ਸ਼ ਵਕਿਲਪਾਂ ਦੀ ਚੋਣ ਕਰਦਾ ਹੈ, ਵਾਧੂ ਸੰਬੰਧਤਿ ਫੀਲਡਾਂ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਪੇਸ਼ ਕੀਤੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ।

ਸੰਦਰਭ-ਜਾਗਰੂਕ ਵਆਿਖਆਿਤਮਕ ਟੈਕਸਟ

ਟੂਲਟਪਿਸ ਦਾ ਅਕਸਰ ਇਸਤੇਮਾਲ ਯੂਜ਼ਰਾਂ ਨੂੰ ਵਾਧੂ ਜਾਣਕਾਰੀ ਜਾਂ ਮਾਰਗਦਰਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਉਹ ਵਸਿ਼ਸ਼ ਤੱਤਾਂ 'ਤੇ ਹੋਵਰ ਕਰਦੇ ਹਨ ਜਾਂ ਉਨ੍ਹਾਂ ਨਾਲ ਅੰਤਰਕਰਿਆ ਕਰਦੇ ਹਨ। "ਸੰਦਰਭੀ ਸਮੱਗਰੀ ਨਰਿਮਾਣ" ਪਹੁੰਚ ਨਾਲ, ਤੁਸੀਂ ਅਜਹਿ ਟੂਲਟਪਿਸ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹੋ ਜੋ ਯੂਜ਼ਰ ਦੇ ਸੰਦਰਭ ਅਨੁਸਾਰ ਢਲਦੇ ਹਨ ਅਤੇ ਢੁਕਵੀਂ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ। ਉਦਾਹਰਣ ਲਈ, ਜੇ ਕੋਈ ਯੂਜ਼ਰ ਕਸਿੇ ਗੁੰਝਲਦਾਰ ਵਸਿ਼ਸ਼ਤਾ ਦੀ ਪੜਚੋਲ ਕਰ ਰਹਿਾ ਹੈ, ਤਾਂ ਟੂਲਟਪਿ ਉਨ੍ਹਾਂ ਦੀਆਂ ਪਛਲਿੀਆਂ ਅੰਤਰਕਰਿਆਵਾਂ ਜਾਂ ਹੁਨਰ ਪੱਧਰ ਦੇ ਆਧਾਰ 'ਤੇ ਨਜਿੀ ਸੁਝਾਅ ਜਾਂ ਉਦਾਹਰਣਾਂ ਪੇਸ਼ ਕਰ ਸਕਦਾ ਹੈ।

ਵਆਿਖਆਿਤਮਕ ਟੈਕਸਟ, ਜਵਿੇਂ ਕਾਿ ਹਦਾਇਤਾਂ, ਵੇਰਵੇ, ਜਾਂ ਮਦਦ ਸੰਦੇਸ਼, ਯੂਜ਼ਰ ਦੇ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਤਆਿਰ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਆਮ ਵਆਿਖਆਿਵਾਂ ਪੇਸ਼ ਕਰਨ ਦੀ ਬਜਾਏ, ਤੁਸੀਂ LLMs ਦੀ ਵਰਤੋਂ ਯੂਜ਼ਰ ਦੀਆਂ ਵਸਿ਼ਸ਼ ਲੋੜਾਂ ਜਾਂ ਸਵਾਲਾਂ ਦੇ ਅਨੁਕੂਲ ਟੈਕਸਟ ਤਆਿਰ ਕਰਨ ਲਈ ਕਰ ਸਕਦੇ ਹੋ। ਉਦਾਹਰਣ ਲਈ, ਜੇ ਕੋਈ ਯੂਜ਼ਰ ਕਸਿ ਪ੍ਰਕਰਿਆ ਦੇ ਕਸਿ ਖਾਸ ਕਦਮ ਨਾਲ ਜੂਝ ਰਹਿਾ ਹੈ, ਤਾਂ ਵਆਿਖਆਿਤਮਕ ਟੈਕਸਟ ਨਜਿੀ ਮਾਰਗਦਰਸ਼ਨ ਜਾਂ ਸਮੱਸਆਿ ਨਵਾਰਣ ਸੁਝਾਅ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦਾ ਹੈ।

ਮਾਈਕਰੋਕਾਪੀ ਛੋਟੇ ਟੈਕਸਟ ਦੇ ਟੁਕੜਆਿਂ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ ਜੋ ਯੂਜ਼ਰਾਂ ਨੂੰ ਤੁਹਾਡੀ ਐਪਲੀਕੇਸ਼ਨ ਵਚਿ ਮਾਰਗਦਰਸ਼ਨ ਕਰਦੇ ਹਨ, ਜਵਿੇਂ ਕਾਿ ਬਟਨ ਲੇਬਲ, ਗਲਤੀ ਸੰਦੇਸ਼, ਜਾਂ ਪੁਸ਼ਟੀਕਰਨ ਪ੍ਰੌਂਪਟ। ਮਾਈਕਰੋਕਾਪੀ 'ਤੇ ਸੰਦਰਭੀ ਸਮੱਗਰੀ ਨਰਿਮਾਣ ਪਹੁੰਚ ਨੂੰ ਲਾਗੂ ਕਰਕੇ, ਤੁਸੀਂ ਇੱਕ ਅਨੁਕੂਲੀ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ ਬਣਾ ਸਕਦੇ ਹੋ ਜੋ ਯੂਜ਼ਰ ਦੀਆਂ ਕਾਰਵਾਈਆਂ ਦਾ ਜਵਾਬ ਦਂਦਿਾ ਹੈ ਅਤੇ ਢੁਕਵਾਂ ਅਤੇ ਮਦਦਗਾਰ ਟੈਕਸਟ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ, ਜੇ ਕੋਈ ਯੂਜ਼ਰ ਕੋਈ ਮਹੱਤਵਪੂਰਨ ਕਾਰਵਾਈ ਕਰਨ ਵਾਲਾ ਹੈ, ਤਾਂ ਪੁਸ਼ਟੀਕਰਨ ਪ੍ਰੌਂਪਟ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਤਆਿਰ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਤਾਂ ਜੋ ਇੱਕ ਸਪੱਸ਼ਟ ਅਤੇ ਨਜਿੀ ਸੰਦੇਸ਼ ਪ੍ਰਦਾਨ ਕੀਤਾ ਜਾ ਸਕੇ।

ਨਜਿੀ ਵਆਿਖਆਿਤਮਕ ਟੈਕਸਟ ਅਤੇ ਟੂਲਟਪਿਸ ਨਵੇਂ ਯੂਜ਼ਰਾਂ ਲਈ ਔਨਬੋਰਡਂਿਗ ਪ੍ਰਕਰਿਆ ਨੂੰ ਬਹੁਤ ਵਧਾ ਸਕਦੇ ਹਨ। ਸੰਦਰਭ-ਵਸਿ਼ਸ਼ ਮਾਰਗਦਰਸ਼ਨ ਅਤੇ ਉਦਾਹਰਣਾਂ ਪ੍ਰਦਾਨ ਕਰਕੇ, ਤੁਸੀਂ ਯੂਜ਼ਰਾਂ ਨੂੰ ਐਪਲੀਕੇਸ਼ਨ ਨੂੰ ਜਲਦੀ ਸਮਝਣ ਅਤੇ ਨੈਵੀਗੇਟ ਕਰਨ ਵਚਿ ਮਦਦ ਕਰ ਸਕਦੇ ਹੋ, ਜੋ ਸਖਿਣ ਦੀ ਪ੍ਰਕਰਿਆ ਨੂੰ ਘਟਾਉਂਦਾ ਹੈ ਅਤੇ ਅਪਣਾਉਣ ਨੂੰ ਵਧਾਉਂਦਾ ਹੈ।

ਗਤੀਸ਼ੀਲ ਅਤੇ ਸੰਦਰਭ-ਜਾਗਰੂਕ ਕਰੋਮ ਐਲੀਮੈਂਟਸ ਐਪਲੀਕੇਸ਼ਨ ਨੂੰ ਵਧੇਰੇ ਸਹਜਿ ਅਤੇ ਦਲਿਚਸਪ ਬਣਾ ਸਕਦੇ

ਹਨ। ਯੂਜ਼ਰ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਨਾਲ ਅੰਤਰਕਰਿਆ ਕਰਨ ਅਤੇ ਉਨ੍ਹਾਂ ਦੀ ਪੜਚੋਲ ਕਰਨ ਦੀ ਵਧੇਰੇ ਸੰਭਾਵਨਾ ਰੱਖਦੇ ਹਨ ਜਦੋਂ ਨਾਲ ਦਾ ਟੈਕਸਟ ਉਨ੍ਹਾਂ ਦੀਆਂ ਵਸ਼ਿਸ਼ ਲੋੜਾਂ ਅਤੇ ਰੁਚੀਆਂ ਦੇ ਅਨੁਕੂਲ ਹੁੰਦਾ ਹੈ।

ਹੁਣ ਤੱਕ ਅਸੀਂ AI ਨਾਲ ਮੌਜੂਦਾ UI ਪੈਰਾਡਾਈਮਜ਼ ਨੂੰ ਵਧਾਉਣ ਦੇ ਵਚਿਾਰਾਂ ਨੂੰ ਕਵਰ ਕੀਤਾ ਹੈ, ਪਰ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸਾਂ ਨੂੰ ਇੱਕ ਵਧੇਰੇ ਮੌਲਕਿ ਢੰਗ ਨਾਲ ਡਜ਼ਿਾਈਨ ਅਤੇ ਲਾਗੂ ਕਰਨ ਦੇ ਤਰੀਕੇ ਬਾਰੇ ਮੁੜ ਵਚਿਾਰ ਕਰਨ ਬਾਰੇ ਕੀ?

ਜਨਰੇਟਵਿ ਯੂਆਈ ਦੀ ਪਰਭਾਸ਼ਾ

ਪਰੰਪਰਾਗਤ ਯੂਆਈ ਡਜ਼ਿਾਈਨ ਦੇ ਉਲਟ, ਜੱਥੇ ਡਜ਼ਿਾਈਨਰ ਸਥਰਿ, ਸਥਾਈ ਇੰਟਰਫੇਸ ਬਣਾਉਂਦੇ ਹਨ, GenUI ਅਜਹਿ ਭਵੱਖ ਵੱਲ ਇਸ਼ਾਰਾ ਕਰਦਾ ਹੈ ਜੱਥੇ ਸਾਡੇ ਸਾਫਟਵੇਅਰ ਵੱਚਿ ਲਚਕਦਾਰ, ਨੱਜੀ ਤਜਰਬੇ ਹੋਣਗੇ ਜੋ ਅਸਲ ਸਮੇਂ ਵੱਚਿ ਵਕਿਸਤਿ ਅਤੇ ਅਨੁਕੂਲ ਹੋ ਸਕਦੇ ਹਨ। ਹਰ ਵਾਰ ਜਦੋਂ ਅਸੀਂ AI-ਸੰਚਾਲਤਿ ਗੱਲਬਾਤ ਇੰਟਰਫੇਸ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ, ਅਸੀਂ AI ਨੂੰ ਯੂਜ਼ਰ ਦੀਆਂ ਵਸ਼ਿਸ਼ ਲੋੜਾਂ ਦੇ ਅਨੁਕੂਲ ਹੋਣ ਦੱਿਦੇ ਹਾਂ। GenUI ਇਸ ਅਨੁਕੂਲਤਾ ਦੇ ਪੱਧਰ ਨੂੰ ਸਾਫਟਵੇਅਰ ਦੇ ਦ੍ਰਸ਼ਿਟੀ ਇੰਟਰਫੇਸ 'ਤੇ ਲਾਗੂ ਕਰਕੇ ਇੱਕ ਕਦਮ ਅੱਗੇ ਲੈ ਜਾਂਦਾ ਹੈ।

ਅੱਜ GenUI ਵਚਿਾਰਾਂ ਨਾਲ ਪ੍ਰਯੋਗ ਕਰਨਾ ਸੰਭਵ ਹੈ ਕਉਂਕਿ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ ਪਹਲਿਾਂ ਹੀ ਪ੍ਰੋਗਰਾਮੰਗਿ ਨੂੰ ਸਮਝਦੇ ਹਨ ਅਤੇ ਉਨ੍ਹਾਂ ਦੇ ਮੂਲ ਗਆਿਨ ਵੱਚਿ UI ਤਕਨਾਲੋਜੀਆਂ ਅਤੇ ਫਰੇਮਵਰਕ ਸ਼ਾਮਲ ਹਨ। ਸਵਾਲ ਇਹ ਹੈ ਕ ਿਕੀ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੀ ਵਰਤੋਂ UI ਐਲੀਮੈਂਟਸ, ਜਵਿੇਂ ਕ ਿਟੈਕਸਟ, ਚੱਤਿਰ, ਲੇਆਉਟ, ਅਤੇ ਇੱਥੋਂ ਤੱਕ ਕ ਿਪੂਰੇ ਇੰਟਰਫੇਸ ਬਣਾਉਣ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ, ਜੋ ਹਰ ਵਅਿਕਤੀਗਤ ਯੂਜ਼ਰ ਲਈ ਅਨੁਕੂਲ ਹੋਵੇ। ਮਾਡਲ ਨੂੰ ਵੱਖ-ਵੱਖ ਕਾਰਕਾਂ ਨੂੰ ਧਆਿਨ ਵੱਚਿ ਰੱਖਣ ਲਈ ਨਰਦੇਸ਼ ਦੱਤਿਾ ਜਾ ਸਕਦਾ ਹੈ, ਜਵਿੇਂ ਕ ਿਯੂਜ਼ਰ ਦੀਆਂ ਪਛਲਿੀਆਂ ਕਰਿਆਵਾਂ, ਦੱਸੀਆਂ ਗਈਆਂ ਤਰਜੀਹਾਂ, ਜਨਸੰਖਅਿਕ ਜਾਣਕਾਰੀ, ਅਤੇ ਵਰਤੋਂ ਦਾ ਮੌਜੂਦਾ ਸੰਦਰਭ, ਤਾਂ ਜੋ ਬੇਹੱਦ ਨੱਜੀ ਅਤੇ ਢੁਕਵੇਂ ਇੰਟਰਫੇਸ ਬਣਾਏ ਜਾ ਸਕਣ।

GenUI ਪਰੰਪਰਾਗਤ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ ਡਜ਼ਿਾਈਨ ਤੋਂ ਕਈ ਮੁੱਖ ਤਰੀਕਆਿਂ ਨਾਲ ਵੱਖਰਾ ਹੈ:

1. **ਗਤੀਸ਼ੀਲ ਅਤੇ ਅਨੁਕੂਲ:** ਪਰੰਪਰਾਗਤ UI ਡਜ਼ਿਾਈਨ ਵੱਚਿ ਸਥਰਿ, ਸਥਾਈ ਇੰਟਰਫੇਸ ਬਣਾਉਣਾ ਸ਼ਾਮਲ ਹੈ ਜੋ ਸਾਰੇ ਯੂਜ਼ਰਾਂ ਲਈ ਇੱਕੋ ਜਹਿ ਰਹੱਿਦੇ ਹਨ। ਇਸਦੇ ਉਲਟ, GenUI ਅਜਹਿ ਇੰਟਰਫੇਸ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ ਜੋ ਯੂਜ਼ਰ ਦੀਆਂ ਲੋੜਾਂ ਅਤੇ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਅਨੁਕੂਲ ਅਤੇ ਬਦਲ ਸਕਦੇ ਹਨ। ਇਸਦਾ ਮਤਲਬ ਹੈ ਕ ਿਇੱਕੋ ਐਪਲੀਕੇਸ਼ਨ ਵੱਖ-ਵੱਖ ਯੂਜ਼ਰਾਂ ਨੂੰ ਜਾਂ ਇੱਥੋਂ ਤੱਕ ਕ ਿਇੱਕੋ ਯੂਜ਼ਰ ਨੂੰ ਵੱਖ-ਵੱਖ ਸਥਤਿੀਆਂ ਵੱਚਿ ਵੱਖ-ਵੱਖ ਇੰਟਰਫੇਸ ਪੇਸ਼ ਕਰ ਸਕਦੀ ਹੈ।

2. **ਵੱਡੇ ਪੱਧਰ 'ਤੇ ਨਿੱਜੀਕਰਨ:** ਪਰੰਪਰਾਗਤ ਡਿਜ਼ਾਈਨ ਨਾਲ, ਹਰ ਯੂਜ਼ਰ ਲਈ ਨਿੱਜੀ ਤਜਰਬੇ ਬਣਾਉਣਾ ਅਕਸਰ ਅਵਿਹਾਰਕ ਹੁੰਦਾ ਹੈ ਕਿਉਂਕਿ ਇਸ ਲਈ ਲੋੜੀਂਦੇ ਸਮੇਂ ਅਤੇ ਸਰੋਤਾਂ ਕਰਕੇ। ਦੂਜੇ ਪਾਸੇ, GenUI ਵੱਡੇ ਪੱਧਰ 'ਤੇ ਨਿੱਜੀਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ। AI ਦਾ ਲਾਭ ਲੈਂਦੇ ਹੋਏ, ਡਿਜ਼ਾਈਨਰ ਅਜਿਹੇ ਇੰਟਰਫੇਸ ਬਣਾ ਸਕਦੇ ਹਨ ਜੋ ਹਰ ਯੂਜ਼ਰ ਦੀਆਂ ਵਿਲੱਖਣ ਲੋੜਾਂ ਅਤੇ ਤਰਜੀਹਾਂ ਦੇ ਅਨੁਕੂਲ ਆਪਣੇ ਆਪ ਢਲ ਜਾਂਦੇ ਹਨ, ਹਰ ਯੂਜ਼ਰ ਸੈਗਮੈਂਟ ਲਈ ਵੱਖਰੇ ਇੰਟਰਫੇਸ ਨੂੰ ਮੈਨੂਅਲ ਤੌਰ 'ਤੇ ਡਿਜ਼ਾਈਨ ਅਤੇ ਵਿਕਸਿਤ ਕੀਤੇ ਬਿਨਾਂ।
3. **ਨਤੀਜਿਆਂ 'ਤੇ ਫੋਕਸ:** ਪਰੰਪਰਾਗਤ UI ਡਿਜ਼ਾਈਨ ਅਕਸਰ ਦਿੱਖ ਪੱਖੋਂ ਆਕਰਸ਼ਕ ਅਤੇ ਕਾਰਜਸ਼ੀਲ ਇੰਟਰਫੇਸ ਬਣਾਉਣ 'ਤੇ ਕੇਂਦਰਿਤ ਹੁੰਦਾ ਹੈ। ਹਾਲਾਂਕਿ ਇਹ ਪਹਲੂ GenUI ਵਿੱਚ ਵੀ ਮਹੱਤਵਪੂਰਨ ਹਨ, ਮੁੱਖ ਫੋਕਸ ਲੋੜੀਂਦੇ ਯੂਜ਼ਰ ਨਤੀਜਿਆਂ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਵੱਲ ਬਦਲ ਜਾਂਦਾ ਹੈ। GenUI ਦਾ ਉਦੇਸ਼ ਅਜਿਹੇ ਇੰਟਰਫੇਸ ਬਣਾਉਣਾ ਹੈ ਜੋ ਹਰ ਯੂਜ਼ਰ ਦੇ ਵਿਸ਼ੇਸ਼ ਟੀਚਿਆਂ ਅਤੇ ਕਾਰਜਾਂ ਲਈ ਅਨੁਕੂਲ ਹੋਣ, ਸਿਰਫ਼ ਸੁਹਜਾਤਮਕ ਵਿਚਾਰਾਂ ਦੀ ਬਜਾਏ ਵਰਤੋਂ-ਯੋਗਤਾ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ੀਲਤਾ ਨੂੰ ਤਰਜੀਹ ਦਿੰਦੇ ਹੋਏ।
4. **ਲਗਾਤਾਰ ਸਿੱਖਣਾ ਅਤੇ ਸੁਧਾਰ:** GenUI ਸਿਸਟਮ ਯੂਜ਼ਰ ਕਿਰਿਆਵਾਂ ਅਤੇ ਫੀਡਬੈਕ ਦੇ ਆਧਾਰ 'ਤੇ ਲਗਾਤਾਰ ਸਿੱਖ ਅਤੇ ਸੁਧਰ ਸਕਦੇ ਹਨ। ਜਿਵੇਂ-ਜਿਵੇਂ ਯੂਜ਼ਰ ਜਨਰੇਟ ਕੀਤੇ ਇੰਟਰਫੇਸ ਨਾਲ ਜੁੜਦੇ ਹਨ, AI ਮਾਡਲ ਯੂਜ਼ਰ ਵਿਵਹਾਰ, ਤਰਜੀਹਾਂ, ਅਤੇ ਨਤੀਜਿਆਂ ਬਾਰੇ ਡੇਟਾ ਇਕੱਠਾ ਕਰ ਸਕਦੇ ਹਨ, ਇਸ ਜਾਣਕਾਰੀ ਦੀ ਵਰਤੋਂ ਭਵਿੱਖ ਦੇ ਇੰਟਰਫੇਸ ਜਨਰੇਸ਼ਨ ਨੂੰ ਸੁਧਾਰਨ ਅਤੇ ਅਨੁਕੂਲ ਬਣਾਉਣ ਲਈ ਕਰਦੇ ਹਨ। ਇਹ ਦੁਹਰਾਈ ਵਾਲੀ ਸਿੱਖਣ ਪ੍ਰਕਿਰਿਆ GenUI ਸਿਸਟਮਾਂ ਨੂੰ ਸਮੇਂ ਦੇ ਨਾਲ ਯੂਜ਼ਰ ਦੀਆਂ ਲੋੜਾਂ ਨੂੰ ਪੂਰਾ ਕਰਨ ਵਿੱਚ ਵਧੇਰੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਬਣਨ ਦੀ ਆਗਿਆ ਦਿੰਦੀ ਹੈ।

ਇਹ ਨੋਟ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਜੈਨਯੂਆਈ, ਏਆਈ-ਸਹਾਇਤਾ ਪ੍ਰਾਪਤ ਡਿਜ਼ਾਈਨ ਟੂਲਜ਼ ਤੋਂ ਵੱਖਰਾ ਹੈ, ਜਿਵੇਂ ਕਿ ਉਹ ਜੋ ਸੁਝਾਅ ਦਿੰਦੇ ਹਨ ਜਾਂ ਕੁਝ ਡਿਜ਼ਾਈਨ ਕਾਰਜਾਂ ਨੂੰ ਸਵੈਚਲਿਤ ਕਰਦੇ ਹਨ। ਭਾਵੇਂ ਇਹ ਟੂਲ ਡਿਜ਼ਾਈਨ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਸੁਚਾਰੂ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦਗਾਰ ਹੋ ਸਕਦੇ ਹਨ, ਪਰ ਉਹ ਫਿਰ ਵੀ ਅੰਤਿਮ ਫੈਸਲੇ ਲੈਣ ਅਤੇ ਸਥਿਰ ਇੰਟਰਫੇਸ ਬਣਾਉਣ ਲਈ ਡਿਜ਼ਾਈਨਰਾਂ 'ਤੇ ਨਿਰਭਰ ਕਰਦੇ ਹਨ। ਦੂਜੇ ਪਾਸੇ, ਜੈਨਯੂਆਈ ਵਿੱਚ ਏਆਈ ਸਿਸਟਮ ਯੂਜ਼ਰ ਡੇਟਾ ਅਤੇ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਇੰਟਰਫੇਸ ਦੀ ਅਸਲ ਉਤਪਤੀ ਅਤੇ ਅਨੁਕੂਲਤਾ ਵਿੱਚ ਵਧੇਰੇ ਸਰਗਰਮ ਭੂਮਿਕਾ ਨਿਭਾਉਂਦਾ ਹੈ।

ਜੈਨਯੂਆਈ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ ਡਿਜ਼ਾਈਨ ਦੇ ਪਹੁੰਚ ਵਿੱਚ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਬਦਲਾਅ ਦਰਸਾਉਂਦਾ ਹੈ, ਜੋ ਇੱਕ-ਆਕਾਰ-ਸਭ-ਲਈ-ਫਿੱਟ ਹੱਲਾਂ ਤੋਂ ਹਟ ਕੇ ਬਹੁਤ ਜ਼ਿਆਦਾ ਨਿੱਜੀਕਰਨ ਕੀਤੇ, ਅਨੁਕੂਲ ਤਜਰਬਿਆਂ ਵੱਲ ਜਾ ਰਹਿਆ ਹੈ। ਏਆਈ ਦੀ ਸ਼ਕਤੀ ਦਾ ਲਾਭ ਲੈਂਦੇ ਹੋਏ, ਜੈਨਯੂਆਈ ਕੋਲ ਡਿਜੀਟਲ ਉਤਪਾਦਾਂ ਅਤੇ ਸੇਵਾਵਾਂ ਨਾਲ ਸਾਡੀ ਅੰਤਰਕਰਿਆ ਦੇ ਤਰੀਕੇ ਨੂੰ ਕ੍ਰਾਂਤੀਕਾਰੀ ਬਣਾਉਣ ਦੀ ਸੰਭਾਵਨਾ ਹੈ, ਜੋ ਹਰ ਵਿਅਕਤੀਗਤ ਯੂਜ਼ਰ ਲਈ ਵਧੇਰੇ ਸਹਜਿ, ਦਿਲਚਸਪ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਇੰਟਰਫੇਸ ਬਣਾਉਂਦਾ ਹੈ।

ਉਦਾਹਰਨ

ਜੈਨਯੂਆਈ ਦੀ ਧਾਰਨਾ ਨੂੰ ਦਰਸਾਉਣ ਲਈ, ਆਓ "FitAI" ਨਾਂ ਦੀ ਇੱਕ ਕਲਪਤਿ ਫਟਿਨੈੱਸ ਐਪਲੀਕੇਸ਼ਨ 'ਤੇ ਵਚਿਾਰ ਕਰੀਏ। ਇਹ ਐਪ ਯੂਜ਼ਰਾਂ ਦੇ ਵਅਿਕਤੀਗਤ ਟੀਚਆਿਂ, ਫਟਿਨੈੱਸ ਪੱਧਰਾਂ, ਅਤੇ ਤਰਜੀਹਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਨਜਿੀ ਵਰਕਆਉਟ ਯੋਜਨਾਵਾਂ ਅਤੇ ਪੋਸ਼ਣ ਸਲਾਹ ਪ੍ਰਦਾਨ ਕਰਨ ਦਾ ਟੀਚਾ ਰੱਖਦੀ ਹੈ।

ਇੱਕ ਪਰੰਪਰਾਗਤ ਯੂਆਈ ਡਜਿ਼ਾਈਨ ਪਹੁੰਚ ਵੱਚਿ, FitAI ਕੋਲ ਸਕਰੀਨਾਂ ਅਤੇ ਤੱਤਾਂ ਦਾ ਇੱਕ ਨਸ਼ਿਚਤ ਸੈੱਟ ਹੋ ਸਕਦਾ ਹੈ ਜੋ ਸਾਰੇ ਯੂਜ਼ਰਾਂ ਲਈ ਇੱਕੋ ਜਹਿਾ ਹੈ। ਹਾਲਾਂਕ,ਿ ਜੈਨਯੂਆਈ ਨਾਲ, ਐਪ ਦਾ ਇੰਟਰਫੇਸ ਹਰ ਯੂਜ਼ਰ ਦੀਆਂ ਵਲਿੱਖਣ ਜਰੂਰਤਾਂ ਅਤੇ ਸੰਦਰਭ ਦੇ ਅਨੁਸਾਰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਅਨੁਕੂਲ ਹੋ ਸਕਦਾ ਹੈ।

ਇਹ ਪਹੁੰਚ 2024 ਵੱਚਿ ਲਾਗੂ ਕਰਨ ਲਈ ਥੋੜ੍ਹੀ ਜਹਿੀ ਕਲਪਨਾ ਦੀ ਲੋੜ ਹੈ ਅਤੇ ਸ਼ਾਇਦ ਇਸਦਾ ਢੁਕਵਾਂ ਨਵਿੇਸ਼ 'ਤੇ ਵਾਪਸੀ ਵੀ ਨਾ ਹੋਵੇ, ਪਰ ਇਹ ਸੰਭਵ ਹੈ।

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰ ਸਕਦਾ ਹੈ:

1. **ਸ਼ੁਰੂਆਤੀ ਪ੍ਰਕਰਿਆ:**

    - ਇੱਕ ਮਆਿਰੀ ਪ੍ਰਸ਼ਨਾਵਲੀ ਦੀ ਬਜਾਏ, FitAI ਯੂਜ਼ਰ ਦੇ ਟੀਚਆਿਂ, ਮੌਜੂਦਾ ਫਟਿਨੈੱਸ ਪੱਧਰ, ਅਤੇ ਤਰਜੀਹਾਂ ਬਾਰੇ ਜਾਣਕਾਰੀ ਇਕੱਠੀ ਕਰਨ ਲਈ ਗੱਲਬਾਤ ਵਾਲੀ ਏਆਈ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ।
    - ਇਸ ਮੁੱਢਲੀ ਅੰਤਰਕਰਿਆ ਦੇ ਆਧਾਰ 'ਤੇ, ਏਆਈ ਇੱਕ ਨਜਿੀ ਡੈਸ਼ਬੋਰਡ ਲੇਆਉਟ ਤਆਿਰ ਕਰਦੀ ਹੈ, ਜੋ ਯੂਜ਼ਰ ਦੇ ਟੀਚਆਿਂ ਨਾਲ ਸਭ ਤੋਂ ਵੱਧ ਸੰਬੰਧਤਿ ਵਸ਼ਿੇਸ਼ਤਾਵਾਂ ਅਤੇ ਜਾਣਕਾਰੀ ਨੂੰ ਉਜਾਗਰ ਕਰਦੀ ਹੈ।
    - ਮੌਜੂਦਾ ਏਆਈ ਤਕਨਾਲੋਜੀ ਕੋਲ ਨਜਿੀ ਡੈਸ਼ਬੋਰਡ ਬਣਾਉਣ ਲਈ ਵਰਤਣ ਲਈ ਸਕਰੀਨ ਕੰਪੋਨੈਂਟਸ ਦੀ ਇੱਕ ਚੋਣ ਹੋ ਸਕਦੀ ਹੈ।
    - ਭਵੱਖਿ ਦੀ ਏਆਈ ਤਕਨਾਲੋਜੀ ਇੱਕ ਤਜਰਬੇਕਾਰ ਯੂਆਈ ਡਜਿ਼ਾਈਨਰ ਦੀ ਭੂਮਕਿਾ ਨਭਿਾ ਸਕਦੀ ਹੈ ਅਤੇ ਅਸਲ ਵੱਚਿ ਡੈਸ਼ਬੋਰਡ ਨੂੰ ਸ਼ੁਰੂ ਤੋਂ ਬਣਾ ਸਕਦੀ ਹੈ।

2. **ਵਰਕਆਉਟ ਪਲੈਨਰ:**

    - ਏਆਈ ਦੁਆਰਾ ਵਰਕਆਉਟ ਪਲੈਨਰ ਇੰਟਰਫੇਸ ਨੂੰ ਯੂਜ਼ਰ ਦੇ ਤਜਰਬੇ ਦੇ ਪੱਧਰ ਅਤੇ ਉਪਲਬਧ ਉਪਕਰਣਾਂ ਦੇ ਅਨੁਸਾਰ ਅਨੁਕੂਲ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ।

- ਬਨਿਾਂ ਕਸਿੇ ਉਪਕਰਣ ਵਾਲੇ ਸ਼ੁਰੂਆਤੀ ਯੂਜ਼ਰ ਲਈ, ਇਹ ਵਸਿਤ੍ਰਤਿ ਹਦਿਾਇਤਾਂ ਅਤੇ ਵੀਡੀਓ ਨਾਲ ਸਧਾਰਨ ਬਾਡੀਵੇਟ ਕਸਰਤਾਂ ਦਖਿਾ ਸਕਦਾ ਹੈ।
- ਜਮ੍ਹਿ ਤੱਕ ਪਹੁੰਚ ਵਾਲੇ ਉੱਨਤ ਯੂਜ਼ਰ ਲਈ, ਇਹ ਘੱਟ ਵਆਿਖਆਿਤਮਕ ਸਮੱਗਰੀ ਨਾਲ ਵਧੇਰੇ ਗੁੰਝਲਦਾਰ ਰੁਟੀਨ ਦਖਿਾ ਸਕਦਾ ਹੈ।
- ਵਰਕਆਉਟ ਪਲੈਨਰ ਦੀ ਸਮੱਗਰੀ ਸਰਫ਼ ਇੱਕ ਵੱਡੇ ਸੁਪਰਸੈੱਟ ਤੋਂ ਫਲਿਟਰ ਨਹੀਂ ਕੀਤੀ ਜਾਂਦੀ। ਇਹ ਤੁਰੰਤ ਇੱਕ ਗਆਿਨ ਅਧਾਰ ਤੋਂ ਤਆਿਰ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ ਜਸ੍ਿ ਨੂੰ ਯੂਜ਼ਰ ਬਾਰੇ ਜਾਣੀ ਜਾਂਦੀ ਹਰ ਜਾਣਕਾਰੀ ਸਮੇਤ ਸੰਦਰਭ ਨਾਲ ਕੁਏਰੀ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

3. **ਤਰੱਕੀ ਦੀ ਨਗਿਰਾਨੀ:**

- ਤਰੱਕੀ ਦੀ ਨਗਿਰਾਨੀ ਇੰਟਰਫੇਸ ਯੂਜ਼ਰ ਦੇ ਟੀਚਆਿਂ ਅਤੇ ਰੁਝਾਨ ਪੈਟਰਨਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਵਕਿਸਤਿ ਹੁੰਦਾ ਹੈ।
- ਜੇ ਕੋਈ ਯੂਜ਼ਰ ਮੁੱਖ ਤੌਰ 'ਤੇ ਭਾਰ ਘਟਾਉਣ 'ਤੇ ਕੇਂਦਰਤਿ ਹੈ, ਤਾਂ ਇੰਟਰਫੇਸ ਭਾਰ ਦੇ ਰੁਝਾਨ ਗ੍ਰਾਫ਼ ਅਤੇ ਕੈਲੋਰੀ ਬਰਨ ਅੰਕੜਆਿਂ ਨੂੰ ਪ੍ਰਮੁੱਖਤਾ ਨਾਲ ਦਖਿਾ ਸਕਦਾ ਹੈ।
- ਮਾਸਪੇਸ਼ੀਆਂ ਬਣਾ ਰਹੇ ਯੂਜ਼ਰ ਲਈ, ਇਹ ਤਾਕਤ ਵੱਚਿ ਵਾਧੇ ਅਤੇ ਸਰੀਰ ਦੀ ਬਣਤਰ ਵੱਚਿ ਤਬਦੀਲੀਆਂ ਨੂੰ ਉਜਾਗਰ ਕਰ ਸਕਦਾ ਹੈ।
- ਏਆਈ ਐਪਲੀਕੇਸ਼ਨ ਦੇ ਇਸ ਹੱਸ੍ਿ ਨੂੰ ਯੂਜ਼ਰ ਦੀ ਅਸਲ ਤਰੱਕੀ ਦੇ ਅਨੁਸਾਰ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦੀ ਹੈ। ਜੇ ਤਰੱਕੀ ਕੁਝ ਸਮੇਂ ਲਈ ਰੁਕ ਜਾਂਦੀ ਹੈ, ਤਾਂ ਐਪ ਅਜਹਿੀ ਮੋਡ ਵੱਚਿ ਬਦਲ ਸਕਦੀ ਹੈ ਜੱਥ੍ਿ ਇਹ ਯੂਜ਼ਰ ਨੂੰ ਰੁਕਾਵਟ ਦੇ ਕਾਰਨਾਂ ਨੂੰ ਦੱਸਣ ਲਈ ਪ੍ਰੇਰਤਿ ਕਰਦੀ ਹੈ, ਤਾਂ ਜੋ ਉਨ੍ਹਾਂ ਨੂੰ ਘੱਟ ਕੀਤਾ ਜਾ ਸਕੇ।

4. **ਪੋਸ਼ਣ ਸਲਾਹ:**

- ਪੋਸ਼ਣ ਭਾਗ ਯੂਜ਼ਰ ਦੀਆਂ ਖੁਰਾਕੀ ਤਰਜੀਹਾਂ ਅਤੇ ਪਾਬੰਦੀਆਂ ਦੇ ਅਨੁਸਾਰ ਢਲਦਾ ਹੈ।
- ਇੱਕ ਸ਼ਾਕਾਹਾਰੀ ਯੂਜ਼ਰ ਲਈ, ਇਹ ਪਲਾਂਟ-ਬੇਸਡ ਖਾਣੇ ਦੇ ਸੁਝਾਅ ਅਤੇ ਪ੍ਰੋਟੀਨ ਸਰੋਤ ਦਖਿਾ ਸਕਦਾ ਹੈ।
- ਗਲੂਟੇਨ ਅਸਹਣਿਸ਼ੀਲਤਾ ਵਾਲੇ ਯੂਜ਼ਰ ਲਈ, ਇਹ ਸਫ਼ਾਰਸ਼ਾਂ ਵੱਚਿੋਂ ਗਲੂਟੇਨ-ਯੁਕਤ ਭੋਜਨ ਨੂੰ ਆਪਣੇ ਆਪ ਫਲਿਟਰ ਕਰ ਦੇਵੇਗਾ।
- ਫਰਿ ਵੀ, ਸਮੱਗਰੀ ਸਾਰੇ ਯੂਜ਼ਰਾਂ 'ਤੇ ਲਾਗੂ ਹੋਣ ਵਾਲੇ ਭੋਜਨ ਡੇਟਾ ਦੇ ਵਸ਼ਿਾਲ ਸੁਪਰਸੈੱਟ ਤੋਂ ਨਹੀਂ ਲਈ ਜਾਂਦੀ, ਸਗੋਂ ਇੱਕ ਗਆਿਨ ਅਧਾਰ ਤੋਂ ਸੰਸਲੇਸ਼ਤਿ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜੋ ਯੂਜ਼ਰ ਦੀ ਵਸ਼ਿੇਸ਼ ਸਥਤਿੀ ਅਤੇ ਪਾਬੰਦੀਆਂ ਦੇ ਆਧਾਰ 'ਤੇ ਅਨੁਕੂਲ ਜਾਣਕਾਰੀ ਰੱਖਦੀ ਹੈ।

    - ਉਦਾਹਰਨ ਲਈ, ਪਕਵਾਨਾਂ ਨੂੰ ਸਮੱਗਰੀ ਦੀਆਂ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਨਾਲ ਤਿਆਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜੋ ਯੂਜ਼ਰ ਦੀ ਫਿਟਨੈੱਸ ਪੱਧਰ ਅਤੇ ਸਰੀਰਕ ਅੰਕੜਿਆਂ ਦੇ ਵਿਕਾਸ ਦੇ ਨਾਲ ਲਗਾਤਾਰ ਬਦਲਦੀਆਂ ਕੈਲੋਰੀ ਦੀਆਂ ਜ਼ਰੂਰਤਾਂ ਨਾਲ ਮੇਲ ਖਾਂਦੀਆਂ ਹਨ।

5. **ਪ੍ਰੇਰਕ ਤੱਤ:**

    - ਐਪ ਦੀ ਪ੍ਰੇਰਕ ਸਮੱਗਰੀ ਅਤੇ ਸੂਚਨਾਵਾਂ ਯੂਜ਼ਰ ਦੀ ਸ਼ਖਸੀਅਤ ਦੀ ਕਿਸਮ ਅਤੇ ਵੱਖ-ਵੱਖ ਪ੍ਰੇਰਕ ਰਣਨੀਤੀਆਂ ਪ੍ਰਤੀ ਪ੍ਰਤੀਕਰਿਆ ਦੇ ਆਧਾਰ 'ਤੇ ਨਿੱਜੀ ਬਣਾਈਆਂ ਜਾਂਦੀਆਂ ਹਨ।
    - ਕੁਝ ਯੂਜ਼ਰਾਂ ਨੂੰ ਉਤਸ਼ਾਹਜਨਕ ਸੁਨੇਹੇ ਮਿਲ ਸਕਦੇ ਹਨ, ਜਦੋਂ ਕਿ ਦੂਜਿਆਂ ਨੂੰ ਵਧੇਰੇ ਡੇਟਾ-ਆਧਾਰਿਤ ਫੀਡਬੈਕ ਮਿਲਦੀ ਹੈ।

ਇਸ ਉਦਾਹਰਨ ਵਿੱਚ, ਜੈਨਯੂਆਈ ਫਿੱਟਏਆਈ ਨੂੰ ਹਰ ਯੂਜ਼ਰ ਲਈ ਇੱਕ ਬੇਹੱਦ ਨਿੱਜੀ ਤਜਰਬਾ ਬਣਾਉਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ, ਜੋ ਸੰਭਾਵਤ ਤੌਰ 'ਤੇ ਰੁਝੇਵੇਂ, ਸੰਤੁਸ਼ਟੀ, ਅਤੇ ਫਿਟਨੈੱਸ ਟੀਚਿਆਂ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਦੀ ਸੰਭਾਵਨਾ ਨੂੰ ਵਧਾਉਂਦਾ ਹੈ। ਇੰਟਰਫੇਸ ਤੱਤ, ਸਮੱਗਰੀ, ਅਤੇ ਇੱਥੋਂ ਤੱਕ ਕਿ ਐਪ ਦੀ "ਸ਼ਖਸੀਅਤ" ਹਰੇਕ ਵਿਅਕਤੀਗਤ ਯੂਜ਼ਰ ਦੀਆਂ ਜ਼ਰੂਰਤਾਂ ਅਤੇ ਤਰਜੀਹਾਂ ਦੀ ਸਭ ਤੋਂ ਵਧੀਆ ਸੇਵਾ ਕਰਨ ਲਈ ਅਨੁਕੂਲ ਬਣਦੇ ਹਨ।

ਨਤੀਜਾ-ਕੇਂਦਰਿਤ ਡਿਜ਼ਾਈਨ ਵੱਲ ਤਬਦੀਲੀ

ਜੈਨਯੂਆਈ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ ਡਿਜ਼ਾਈਨ! ਦੇ ਪਹੁੰਚ ਵਿੱਚ ਇੱਕ ਬੁਨਿਆਦੀ ਤਬਦੀਲੀ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ, ਜੋ ਵਸ਼ਿਸ਼ ਇੰਟਰਫੇਸ ਤੱਤਾਂ ਨੂੰ ਬਣਾਉਣ ਤੋਂ ਇੱਕ ਵਧੇਰੇ ਸਮੁੱਚੇ, ਨਤੀਜਾ-ਕੇਂਦਰਿਤ ਪਹੁੰਚ ਵੱਲ ਜਾ ਰਹਿਆ ਹੈ। ਇਸ ਤਬਦੀਲੀ ਦੇ ਕਈ ਮਹੱਤਵਪੂਰਨ ਪ੍ਰਭਾਵ ਹਨ:

1. **ਯੂਜ਼ਰ ਟੀਚਿਆਂ 'ਤੇ ਧਿਆਨ:**

    - ਡਿਜ਼ਾਈਨਰਾਂ ਨੂੰ ਵਸ਼ਿਸ਼ ਇੰਟਰਫੇਸ ਭਾਗਾਂ ਦੀ ਬਜਾਏ ਯੂਜ਼ਰ ਦੇ ਟੀਚਿਆਂ ਅਤੇ ਇੱਛਿਤ ਨਤੀਜਿਆਂ ਬਾਰੇ ਵਧੇਰੇ ਡੂੰਘਾਈ ਨਾਲ ਸੋਚਣ ਦੀ ਲੋੜ ਹੋਵੇਗੀ।
    - ਜ਼ੋਰ ਅਜਿਹੀਆਂ ਪ੍ਰਣਾਲੀਆਂ ਬਣਾਉਣ 'ਤੇ ਹੋਵੇਗਾ ਜੋ ਯੂਜ਼ਰਾਂ ਨੂੰ ਆਪਣੇ ਉਦੇਸ਼ਾਂ ਨੂੰ ਕੁਸ਼ਲਤਾ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਪ੍ਰਾਪਤ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰਨ ਵਾਲੇ ਇੰਟਰਫੇਸ ਤਿਆਰ ਕਰ ਸਕਣ।
    - ਨਵੇਂ ਯੂਆਈ ਫਰੇਮਵਰਕ ਉਭਰਨਗੇ ਜੋ ਏਆਈ-ਆਧਾਰਿਤ ਡਿਜ਼ਾਈਨਰਾਂ ਨੂੰ ਪਹਿਲਾਂ ਤੋਂ ਨਿਰਧਾਰਿਤ ਸਕ੍ਰੀਨ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਦੀ ਬਜਾਏ ਢੌਂਗੀ ਤੌਰ 'ਤੇ ਅਤੇ ਸ਼ੁਰੂ ਤੋਂ ਯੂਜ਼ਰ ਤਜਰਬਿਆਂ ਨੂੰ ਤਿਆਰ ਕਰਨ ਲਈ ਲੋੜੀਂਦੇ ਟੂਲ ਪ੍ਰਦਾਨ ਕਰਨਗੇ।

2. **ਡਿਜ਼ਾਈਨਰਾਂ ਦੀ ਬਦਲਦੀ ਭੂਮਿਕਾ:**
    - ਡਿਜ਼ਾਈਨਰ ਸਥਿਰ ਲੇਆਉਟ ਬਣਾਉਣ ਤੋਂ ਨਿਯਮ, ਸੀਮਾਵਾਂ, ਅਤੇ ਦਿਸ਼ਾ-ਨਿਰਦੇਸ਼ ਨਿਰਧਾਰਤ ਕਰਨ ਵੱਲ ਤਬਦੀਲ ਹੋਣਗੇ ਜਿਨ੍ਹਾਂ ਦੀ ਏਆਈ ਸਿਸਟਮ ਇੰਟਰਫੇਸ ਬਣਾਉਣ ਵੇਲੇ ਪਾਲਣਾ ਕਰੇਗਾ।
    - ਉਨ੍ਹਾਂ ਨੂੰ ਡਾਟਾ ਵਿਸ਼ਲੇਸ਼ਣ, ਏਆਈ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ, ਅਤੇ ਸਿਸਟਮ ਸੋਚ ਵਰਗੇ ਖੇਤਰਾਂ ਵਿੱਚ ਹੁਨਰ ਵਿਕਸਿਤ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇਗੀ ਤਾਂ ਜੋ GenUI ਸਿਸਟਮਾਂ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਮਾਰਗਦਰਸ਼ਨ ਕੀਤਾ ਜਾ ਸਕੇ।
3. **ਯੂਜ਼ਰ ਖੋਜ ਦੀ ਮਹੱਤਤਾ:**
    - GenUI ਸੰਦਰਭ ਵਿੱਚ ਯੂਜ਼ਰ ਖੋਜ ਹੋਰ ਵੀ ਮਹੱਤਵਪੂਰਨ ਹੋ ਜਾਂਦੀ ਹੈ, ਕਿਉਂਕਿ ਡਿਜ਼ਾਈਨਰਾਂ ਨੂੰ ਨਾ ਸਿਰਫ਼ ਯੂਜ਼ਰ ਪਸੰਦਾਂ ਨੂੰ, ਬਲਕਿ ਇਹ ਵੀ ਸਮਝਣ ਦੀ ਲੋੜ ਹੈ ਕਿ ਵੱਖ-ਵੱਖ ਸੰਦਰਭਾਂ ਵਿੱਚ ਇਹ ਪਸੰਦਾਂ ਅਤੇ ਲੋੜਾਂ ਕਿਵੇਂ ਬਦਲਦੀਆਂ ਹਨ।
    - ਨਿਰੰਤਰ ਯੂਜ਼ਰ ਟੈਸਟਿੰਗ ਅਤੇ ਫੀਡਬੈਕ ਲੂਪ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਇੰਟਰਫੇਸ ਤਿਆਰ ਕਰਨ ਲਈ ਏਆਈ ਦੀ ਯੋਗਤਾ ਨੂੰ ਸੁਧਾਰਨ ਅਤੇ ਬਹਿਤਰ ਬਣਾਉਣ ਲਈ ਜ਼ਰੂਰੀ ਹੋਣਗੇ।
4. **ਪਰਿਵਰਤਨਸ਼ੀਲਤਾ ਲਈ ਡਿਜ਼ਾਈਨ:**
    - ਇੱਕ "ਸੰਪੂਰਨ" ਇੰਟਰਫੇਸ ਬਣਾਉਣ ਦੀ ਬਜਾਏ, ਡਿਜ਼ਾਈਨਰਾਂ ਨੂੰ ਕਈ ਸੰਭਾਵੀ ਵਿਭਿੰਨਤਾਵਾਂ 'ਤੇ ਵਿਚਾਰ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇਗੀ ਅਤੇ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਹੋਵੇਗਾ ਕਿ ਸਿਸਟਮ ਵੱਖ-ਵੱਖ ਯੂਜ਼ਰ ਲੋੜਾਂ ਲਈ ਢੁਕਵੇਂ ਇੰਟਰਫੇਸ ਤਿਆਰ ਕਰ ਸਕੇ।
    - ਇਸ ਵਿੱਚ ਸੀਮਾ ਮਾਮਲਿਆਂ ਲਈ ਡਿਜ਼ਾਈਨ ਕਰਨਾ ਅਤੇ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਸ਼ਾਮਲ ਹੈ ਕਿ ਤਿਆਰ ਕੀਤੇ ਇੰਟਰਫੇਸ ਵੱਖ-ਵੱਖ ਕੌਨਫਿਗਿਰੇਸ਼ਨਾਂ ਵਿੱਚ ਵਰਤੋਂਯੋਗਤਾ ਅਤੇ ਪਹੁੰਚਯੋਗਤਾ ਬਣਾਈ ਰੱਖਣ।
    - ਉਤਪਾਦ ਵਿਭਿੰਦੀਕਰਨ ਯੂਜ਼ਰ ਮਨੋਵਿਗਿਆਨ ਬਾਰੇ ਵੱਖ-ਵੱਖ ਦ੍ਰਿਸ਼ਟੀਕੋਣਾਂ ਅਤੇ ਵਿਲੱਖਣ ਡਾਟਾ ਸੈੱਟਾਂ ਅਤੇ ਗਿਆਨ ਅਧਾਰਾਂ ਦੀ ਵਰਤੋਂ ਨਾਲ ਨਵੇਂ ਆਯਾਮ ਲੈਂਦਾ ਹੈ ਜੋ ਪ੍ਰਤੀਯੋਗੀਆਂ ਲਈ ਉਪਲਬਧ ਨਹੀਂ ਹਨ।

ਚੁਣੌਤੀਆਂ ਅਤੇ ਵਿਚਾਰ

ਜਦੋਂ ਕਿ GenUI ਰੋਮਾਂਚਕ ਸੰਭਾਵਨਾਵਾਂ ਪੇਸ਼ ਕਰਦਾ ਹੈ, ਇਹ ਕਈ ਚੁਣੌਤੀਆਂ ਅਤੇ ਵਿਚਾਰਾਂ ਨੂੰ ਵੀ ਪੇਸ਼ ਕਰਦਾ ਹੈ:

1. **ਤਕਨੀਕੀ ਸੀਮਾਵਾਂ:**

    - ਮੌਜੂਦਾ ਏਆਈ ਤਕਨਾਲੋਜੀ, ਭਾਵੇਂ ਉੱਨਤ ਹੈ, ਫਰਿ ਵੀ ਗੁੰਝਲਦਾਰ ਯੂਜ਼ਰ ਇਰਾਦਿਆਂ ਨੂੰ ਸਮਝਣ ਅਤੇ ਅਸਲ ਸੰਦਰਭ-ਜਾਗਰੂਕ ਇੰਟਰਫੇਸ ਤਿਆਰ ਕਰਨ ਵਿੱਚ ਸੀਮਾਵਾਂ ਹਨ।
    - ਇੰਟਰਫੇਸ ਐਲੀਮੈਂਟਸ ਦੀ ਰੀਅਲ-ਟਾਈਮ ਜਨਰੇਸ਼ਨ ਨਾਲ ਜੁੜੀਆਂ ਪਰਫਾਰਮੈਂਸ ਸਮੱਸਿਆਵਾਂ, ਖ਼ਾਸ ਕਰਕੇ ਘੱਟ ਸ਼ਕਤੀਸ਼ਾਲੀ ਡਿਵਾਈਸਾਂ ’ਤੇ।

2. **ਡਾਟਾ ਲੋੜਾਂ:**

    - ਵਰਤੋਂ ਦੇ ਮਾਮਲੇ ’ਤੇ ਨਿਰਭਰ ਕਰਦੇ ਹੋਏ, ਪ੍ਰਭਾਵਸ਼ਾਲੀ GenUI ਸਿਸਟਮਾਂ ਨੂੰ ਨਿੱਜੀ ਇੰਟਰਫੇਸ ਤਿਆਰ ਕਰਨ ਲਈ ਯੂਜ਼ਰ ਡਾਟਾ ਦੀ ਕਾਫ਼ੀ ਮਾਤਰਾ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ।
    - ਪ੍ਰਮਾਣਿਕ ਯੂਜ਼ਰ ਡਾਟਾ ਨੂੰ ਨੈਤਿਕ ਤੌਰ ’ਤੇ ਪ੍ਰਾਪਤ ਕਰਨ ਦੀਆਂ ਚੁਣੌਤੀਆਂ ਡਾਟਾ ਪਰਦੇਦਾਰੀ ਅਤੇ ਸੁਰੱਖਿਆ ਬਾਰੇ ਚਿੰਤਾਵਾਂ ਪੈਦਾ ਕਰਦੀਆਂ ਹਨ, ਨਾਲ ਹੀ GenUI ਮਾਡਲਾਂ ਨੂੰ ਸਿਖਲਾਈ ਦੇਣ ਲਈ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਡਾਟਾ ਵਿੱਚ ਸੰਭਾਵੀ ਪੱਖਪਾਤ।

3. **ਵਰਤੋਂਯੋਗਤਾ ਅਤੇ ਇਕਸਾਰਤਾ:**

    - ਘੱਟੋ-ਘੱਟ ਜਦੋਂ ਤੱਕ ਇਹ ਅਭਿਆਸ ਵਿਆਪਕ ਨਹੀਂ ਹੋ ਜਾਂਦਾ, ਲਗਾਤਾਰ ਬਦਲਦੇ ਇੰਟਰਫੇਸਾਂ ਵਾਲੀ ਐਪਲੀਕੇਸ਼ਨ ਵਰਤੋਂਯੋਗਤਾ ਸਮੱਸਿਆਵਾਂ ਦਾ ਕਾਰਨ ਬਣ ਸਕਦੀ ਹੈ, ਕਿਉਂਕਿ ਯੂਜ਼ਰਾਂ ਨੂੰ ਜਾਣੇ-ਪਛਾਣੇ ਐਲੀਮੈਂਟਸ ਲੱਭਣ ਜਾਂ ਕੁਸ਼ਲਤਾ ਨਾਲ ਨੈਵੀਗੇਟ ਕਰਨ ਵਿੱਚ ਮੁਸ਼ਕਲ ਹੋ ਸਕਦੀ ਹੈ।
    - ਨਿੱਜੀਕਰਨ ਅਤੇ ਇੱਕ ਸਥਿਰ, ਸਿੱਖਣਯੋਗ ਇੰਟਰਫੇਸ ਬਣਾਈ ਰੱਖਣ ਵਿਚਕਾਰ ਸੰਤੁਲਨ ਬਣਾਉਣਾ ਮਹੱਤਵਪੂਰਨ ਹੋਵੇਗਾ।

4. **ਏਆਈ ’ਤੇ ਜ਼ਿਆਦਾ ਨਿਰਭਰਤਾ:**

    - ਏਆਈ ਸਿਸਟਮਾਂ ਨੂੰ ਡਿਜ਼ਾਈਨ ਫੈਸਲੇ ਜ਼ਿਆਦਾ ਸੌਂਪਣ ਦਾ ਜੋਖਮ ਹੈ, ਜੋ ਸੰਭਾਵੀ ਤੌਰ ’ਤੇ ਬੇਪ੍ਰੇਰਣਾ, ਸਮੱਸਿਆਤਮਕ, ਜਾਂ ਬਸ ਟੁੱਟੇ ਹੋਏ ਇੰਟਰਫੇਸ ਵਿਕਲਪਾਂ ਵੱਲ ਲੈ ਜਾ ਸਕਦਾ ਹੈ।
    - ਮਨੁੱਖੀ ਨਿਗਰਾਨੀ ਅਤੇ ਏਆਈ-ਜਨਰੇਟਡ ਡਿਜ਼ਾਈਨਾਂ ਨੂੰ ਓਵਰਰਾਈਡ ਕਰਨ ਦੀ ਯੋਗਤਾ ਨਜ਼ਦੀਕੀ ਭਵਿੱਖ ਵਿੱਚ ਮਹੱਤਵਪੂਰਨ ਰਹੇਗੀ।

5. **ਪਹੁੰਚਯੋਗਤਾ ਸੰਬੰਧੀ ਚਿੰਤਾਵਾਂ:**

- ਗਤੀਸ਼ੀਲ ਤੌਰ 'ਤੇ ਤਆਿਰ ਕੀਤੇ ਇੰਟਰਫੇਸਾਂ ਨੂੰ ਅਪਾਹਜ ਯੂਜ਼ਰਾਂ ਲਈ ਪਹੁੰਚਯੋਗ ਬਣਾਈ ਰੱਖਣਾ ਬਲਿਕੁਲ ਨਵੀਆਂ ਚੁਣੌਤੀਆਂ ਪੇਸ਼ ਕਰਦਾ ਹੈ, ਜੋ ਕਿਚੰਤਾਜਨਕ ਹੈ ਕਉਿਕਿਆਮ ਸਸਿਟਮਾਂ ਵੱਚਿ ਪਹੁੰਚਯੋਗਤਾ ਦੀ ਪਾਲਣਾ ਦਾ ਪੱਧਰ ਘੱਟ ਹੈ।
- ਦੂਜੇ ਪਾਸੇ, AI ਡਜ਼ਾਇਨਰਾਂ ਨੂੰ ਅੰਦਰੂਨੀ ਪਹੁੰਚਯੋਗਤਾ ਦੀ ਚੰਤਿਾ ਨਾਲ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਅਤੇ ਗੈਰ-ਅਪਾਹਜ ਯੂਜ਼ਰਾਂ ਲਈ UI ਬਣਾਉਣ ਵਾਂਗ ਹੀ ਤੁਰੰਤ ਪਹੁੰਚਯੋਗ ਇੰਟਰਫੇਸ ਬਣਾਉਣ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ।
- ਕਸਿੇ ਵੀ ਤਰ੍ਹਾਂ, GenUI ਸਸਿਟਮਾਂ ਨੂੰ ਮਜ਼ਬੂਤ ਪਹੁੰਚਯੋਗਤਾ ਦਸ਼ਾ-ਨਰਦੇਸ਼ਾਂ ਅਤੇ ਟੈਸਟਿੰਗ ਪ੍ਰਕਰਿਆਵਾਂ ਨਾਲ ਡਜ਼ਾਇਨ ਕੀਤਾ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ।

6. **ਯੂਜ਼ਰ ਭਰੋਸਾ ਅਤੇ ਪਾਰਦਰਸ਼ਤਾ:**

- ਯੂਜ਼ਰ ਅਜਹਿੇ ਇੰਟਰਫੇਸਾਂ ਨਾਲ ਬੇਆਰਾਮ ਮਹਸੂਸ ਕਰ ਸਕਦੇ ਹਨ ਜੋ ਉਨ੍ਹਾਂ ਬਾਰੇ "ਬਹੁਤ ਜ਼ਆਿਦਾ ਜਾਣਦੇ" ਜਾਪਦੇ ਹਨ ਜਾਂ ਅਜਹ�ੇ ਤਰੀਕਆਿਂ ਨਾਲ ਬਦਲਦੇ ਹਨ ਜੋ ਉਹ ਨਹੀਂ ਸਮਝਦੇ।
- ਯੂਜ਼ਰ ਦਾ ਭਰੋਸਾ ਬਣਾਉਣ ਲਈ ਇਹ ਦੱਸਣਾ ਮਹੱਤਵਪੂਰਨ ਹੋਵੇਗਾ ਕਿ ਇੰਟਰਫੇਸ ਕਵਿੇਂ ਅਤੇ ਕਉਿ ਨੱਜੀ ਬਣਾਏ ਜਾਂਦੇ ਹਨ।

ਭਵੱਖਿ ਦਾ ਨਜ਼ਰੀਆ ਅਤੇ ਮੌਕੇ

ਜਨਰੇਟਵਿ UI (GenUI) ਦਾ ਭਵੱਖਿ ਡਜ਼ਿੀਟਲ ਉਤਪਾਦਾਂ ਅਤੇ ਸੇਵਾਵਾਂ ਨਾਲ ਸਾਡੀ ਅੰਤਰਕਰਿਆ ਦੇ ਤਰੀਕੇ ਨੂੰ ਕ੍ਰਾਂਤੀਕਾਰੀ ਬਣਾਉਣ ਲਈ ਵਸ਼ਾਲ ਵਾਅਦਾ ਰੱਖਦਾ ਹੈ। ਜਵਿੇਂ-ਜਵਿੇਂ ਇਹ ਤਕਨਾਲੋਜੀ ਵਕਿਸਤਿ ਹੁੰਦੀ ਜਾਂਦੀ ਹੈ, ਅਸੀਂ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸਾਂ ਦੇ ਡਜ਼ਾਇਨ, ਲਾਗੂਕਰਨ, ਅਤੇ ਅਨੁਭਵ ਵੱਚਿ ਇੱਕ ਭੂਚਾਲੀ ਬਦਲਾਅ ਦੀ ਉਮੀਦ ਕਰ ਸਕਦੇ ਹਾਂ। ਮੈਂ ਸੋਚਦਾ ਹਾਂ ਕਿ GenUI ਉਹ ਵਰਤਾਰਾ ਹੈ ਜੋ ਆਖਰਕਾਰ ਸਾਡੇ ਸਾਫਟਵੇਅਰ ਨੂੰ ਉਸ ਖੇਤਰ ਵੱਚਿ ਧੱਕ ਦੇਵੇਗਾ ਜੋ ਹੁਣ ਵਗਿਆਨ-ਕਲਪਨਾ ਮੰਨਆਿ ਜਾਂਦਾ ਹੈ।

GenUI ਦੀਆਂ ਸਭ ਤੋਂ ਰੋਮਾਂਚਕ ਸੰਭਾਵਨਾਵਾਂ ਵੱਚੋਂ ਇੱਕ ਇਸਦੀ ਪਹੁੰਚਯੋਗਤਾ ਨੂੰ ਇੱਕ ਵਸ਼ਾਲ ਪੱਧਰ 'ਤੇ ਵਧਾਉਣ ਦੀ ਸਮਰੱਥਾ ਹੈ ਜੋ ਸਰਫ਼ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣ ਤੋਂ ਅੱਗੇ ਜਾਂਦੀ ਹੈ ਕਿ ਗੰਭੀਰ ਅਪਾਹਜਤਾ ਵਾਲੇ ਲੋਕਾਂ ਨੂੰ ਤੁਹਾਡੇ ਸਾਫਟਵੇਅਰ ਦੀ ਵਰਤੋਂ ਤੋਂ ਪੂਰੀ ਤਰ੍ਹਾਂ ਬਾਹਰ ਨਾ ਰੱਖਆਿ ਜਾਵੇ। ਇੰਟਰਫੇਸਾਂ ਨੂੰ ਵਅਿਕਤੀਗਤ ਯੂਜ਼ਰ ਦੀਆਂ ਲੋੜਾਂ ਅਨੁਸਾਰ ਆਪਣੇ ਆਪ ਢਾਲ ਕੇ, GenUI ਡਜ਼ਿੀਟਲ ਅਨੁਭਵਾਂ ਨੂੰ ਪਹਲਿਾਂ ਨਾਲੋਂ ਵੀ ਵੱਧ ਸਮਾਵੇਸ਼ੀ ਬਣਾ ਸਕਦਾ ਹੈ। ਅਜਹिे ਇੰਟਰਫੇਸਾਂ ਦੀ ਕਲਪਨਾ ਕਰੋ ਜੋ ਛੋਟੇ ਜਾਂ ਨਜ਼ਰ ਦੀ ਕਮਜ਼ੋਰੀ ਵਾਲੇ ਯੂਜ਼ਰਾਂ ਲਈ ਵੱਡਾ ਟੈਕਸਟ ਜਾਂ ਬੌਧਕਿ ਅਪਾਹਜਤਾ ਵਾਲੇ ਲੋਕਾਂ ਲਈ ਸਰਲ ਲੇਆਉਟ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਨਰਵਘਿਨ ਢੰਗ ਨਾਲ

ਅਨੁਕੂਲ ਹੋ ਜਾਂਦੇ ਹਨ, ਇਹ ਸਭ ਮੈਨੂਅਲ ਕੌਂਫਿਗਰੇਸ਼ਨ ਜਾਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦੇ ਵੱਖਰੇ "ਪਹੁੰਚਯੋਗ" ਵਰਜਨਾਂ ਦੀ ਲੋੜ ਤੋਂ ਬਨਿਾਂ।

ਨੰਜੀਕਰਨ ਦੀਆਂ ਸਮਰੱਥਾਵਾਂ ਡਜਿੀਟਲ ਉਤਪਾਦਾਂ ਦੀ ਵਸਿ਼ਾਲ ਸ਼੍ਰੇਣੀ ਵੱਚਿ ਵਧੀ ਹੋਈ ਯੂਜ਼ਰ ਸ਼ਮੂਲੀਅਤ, ਸੰਤੁਸ਼ਟੀ, ਅਤੇ ਵਫ਼ਾਦਾਰੀ ਨੂੰ ਚਲਾਉਣ ਦੀ ਸੰਭਾਵਨਾ ਹੈ। ਜਵਿੇਂ-ਜਵਿੇਂ ਇੰਟਰਫੇਸ ਵਅਿਕਤੀਗਤ ਤਰਜੀਹਾਂ ਅਤੇ ਵਵਿਹਾਰਾਂ ਦੇ ਅਨੁਕੂਲ ਹੁੰਦੇ ਜਾਂਦੇ ਹਨ, ਯੂਜ਼ਰ ਡਜਿੀਟਲ ਅਨੁਭਵਾਂ ਨੂੰ ਵਧੇਰੇ ਸਹਜਿ ਅਤੇ ਮਨੋਰੰਜਕ ਪਾਉਣਗੇ, ਜੋ ਸੰਭਾਵਤ ਤੌਰ 'ਤੇ ਤਕਨਾਲੋਜੀ ਨਾਲ ਡੂੰਘੇ ਅਤੇ ਵਧੇਰੇ ਅਰਥਪੂਰਨ ਅੰਤਰਕਰਿਆਵਾਂ ਵੱਲ ਲੈ ਜਾਵੇਗਾ।

GenUI ਵੱਚਿ ਨਵੇਂ ਯੂਜ਼ਰਾਂ ਲਈ ਆਨਬੋਰਡਿੰਗ ਪ੍ਰਕਰਿਆ ਨੂੰ ਬਦਲਣ ਦੀ ਸਮਰੱਥਾ ਵੀ ਹੈ। ਸਹਜਿ, ਨੰਜੀ ਪਹਲਿੇ-ਵਾਰ ਯੂਜ਼ਰ ਅਨੁਭਵ ਬਣਾ ਕੇ ਜੋ ਹਰੇਕ ਯੂਜ਼ਰ ਦੀ ਮੁਹਾਰਤ ਦੇ ਪੱਧਰ ਦੇ ਅਨੁਸਾਰ ਤੇਜ਼ੀ ਨਾਲ ਢਲ ਜਾਂਦੇ ਹਨ, GenUI ਨਵੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨਾਲ ਜੁੜੀ ਸੰਖਿਣ ਦੀ ਪ੍ਰਕਰਿਆ ਨੂੰ ਕਾਫ਼ੀ ਘਟਾ ਸਕਦਾ ਹੈ। ਇਹ ਤੇਜ਼ ਅਪਣਾਉਣ ਦੀ ਦਰ ਅਤੇ ਨਵੀਆਂ ਵਸਿ਼ਸ਼ਤਾਵਾਂ ਅਤੇ ਕਾਰਜਸ਼ੀਲਤਾਵਾਂ ਦੀ ਖੋਜ ਕਰਨ ਵੱਚਿ ਵਧੇ ਹੋਏ ਯੂਜ਼ਰ ਆਤਮਵਸ਼ਿਵਾਸ ਵੱਲ ਲੈ ਜਾ ਸਕਦਾ ਹੈ।

ਇੱਕ ਹੋਰ ਰੋਮਾਂਚਕ ਸੰਭਾਵਨਾ GenUI ਦੀ ਵੱਖ-ਵੱਖ ਡਵਿਾਈਸਾਂ ਅਤੇ ਪਲੇਟਫਾਰਮਾਂ 'ਤੇ ਇੱਕ ਸਥਰਿ ਯੂਜ਼ਰ ਅਨੁਭਵ ਨੂੰ ਬਣਾਈ ਰੱਖਣ ਦੀ ਯੋਗਤਾ ਹੈ, ਜਦੋਂ ਕਹਿਰ ਖਾਸ ਵਰਤੋਂ ਦੇ ਸੰਦਰਭ ਲਈ ਇਸਨੂੰ ਅਨੁਕੂਲ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਸਮਾਰਟਫੋਨਾਂ ਅਤੇ ਟੈਬਲੇਟਾਂ ਤੋਂ ਲੈ ਕੇ ਡੈਸਕਟੌਪ ਕੰਪਊਿਟਰਾਂ ਅਤੇ ਵਧਾਈ ਹੋਈ ਅਸਲੀਅਤ ਐਨਕਾਂ ਵਰਗੀਆਂ ਉੱਭਰ ਰਹੀਆਂ ਤਕਨਾਲੋਜੀਆਂ ਤੱਕ, ਵੱਧ ਰਹੇ ਵੰਡੇ ਹੋਏ ਡਵਿਾਈਸ ਲੈਂਡਸਕੇਪ ਵੱਚਿ ਸੁਸੰਗਤ ਅਨੁਭਵ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਲੰਬੇ ਸਮੇਂ ਤੋਂ ਚੱਲੀ ਆ ਰਹੀ ਚੁਣੌਤੀ ਨੂੰ ਹੱਲ ਕਰ ਸਕਦਾ ਹੈ।

GenUI ਦੀ ਡਾਟਾ-ਸੰਚਾਲਤਿ ਪ੍ਰਕਰਿਤੀ UI ਡਜਿ਼ਾਈਨ ਵੱਚਿ ਤੇਜ਼ ਦੁਹਰਾਓ ਅਤੇ ਸੁਧਾਰ ਲਈ ਮੌਕੇ ਖੋਲ੍ਹਦੀ ਹੈ। ਇਸ ਬਾਰੇ ਅਸਲ-ਸਮੇਂ ਦਾ ਡਾਟਾ ਇਕੱਠਾ ਕਰਕੇ ਕਿ ਯੂਜ਼ਰ ਜਨਰੇਟ ਕੀਤੇ ਇੰਟਰਫੇਸਾਂ ਨਾਲ ਕਵਿੇਂ ਇੰਟਰੈਕਟ ਕਰਦੇ ਹਨ, ਡਜਿ਼ਾਈਨਰ ਅਤੇ ਡਵਿੈਲਪਰ ਯੂਜ਼ਰ ਵਵਿਹਾਰ ਅਤੇ ਤਰਜੀਹਾਂ ਬਾਰੇ ਬੇਮਸਿਾਲ ਅੰਤਰਦ੍ਰਸ਼ਟਿੀ ਪ੍ਰਾਪਤ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਫੀਡਬੈਕ ਲੂਪ UI ਡਜਿ਼ਾਈਨ ਵੱਚਿ ਲਗਾਤਾਰ ਸੁਧਾਰ ਲਆਿ ਸਕਦਾ ਹੈ, ਜੋ ਧਾਰਨਾਵਾਂ ਜਾਂ ਸੀਮਤਿ ਯੂਜ਼ਰ ਟੈਸਟਿੰਗ ਦੀ ਬਜਾਏ ਅਸਲ ਵਰਤੋਂ ਦੇ ਪੈਟਰਨਾਂ ਦੁਆਰਾ ਸੰਚਾਲਤਿ ਹੈ।

ਇਸ ਬਦਲਾਅ ਲਈ ਤਆਿਰੀ ਕਰਨ ਵਾਸਤੇ, ਡਜਿ਼ਾਈਨਰਾਂ ਨੂੰ ਆਪਣੇ ਹੁਨਰ ਸੈੱਟ ਅਤੇ ਮਾਨਸਕਿਤਾ ਨੂੰ ਵਕਿਸਤਿ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇਗੀ। ਧਆਿਨ ਸਥਰਿ ਲੇਆਉਟ ਬਣਾਉਣ ਤੋਂ ਵਆਿਪਕ ਡਜਿ਼ਾਈਨ ਸਸਿਟਮ ਅਤੇ ਦਸਿ਼ਾ-ਨਰਿਦੇਸ਼ ਵਕਿਸਤਿ ਕਰਨ ਵੱਲ ਬਦਲ ਜਾਵੇਗਾ ਜੋ AI-ਸੰਚਾਲਤਿ ਇੰਟਰਫੇਸ ਜਨਰੇਸ਼ਨ ਨੂੰ ਸੂਚਤਿ ਕਰ ਸਕਦੇ ਹਨ। ਡਜਿ਼ਾਈਨਰਾਂ ਨੂੰ GenUI ਸਸਿਟਮਾਂ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਮਾਰਗਦਰਸ਼ਨ ਕਰਨ ਲਈ ਡਾਟਾ ਵਸਿ਼ਲੇਸ਼ਣ, AI ਤਕਨਾਲੋਜੀਆਂ, ਅਤੇ ਸਸਿਟਮ ਸੋਚ ਦੀ ਡੂੰਘੀ ਸਮਝ ਵਕਿਸਤਿ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇਗੀ।

ਇਸ ਤੋਂ ਇਲਾਵਾ, ਜਵਿੇਂ-ਜਵਿੇਂ GenUI ਡਜਿ਼ਾਈਨ ਅਤੇ ਤਕਨਾਲੋਜੀ ਵਚਿਕਾਰ ਦੀਆਂ ਲਾਈਨਾਂ ਨੂੰ ਧੁੰਦਲਾ

ਕਰਦਾ ਹੈ, ਡਿਜ਼ਾਈਨਰਾਂ ਨੂੰ ਡਵੈਲਪਰਾਂ ਅਤੇ ਡਾਟਾ ਵਿਗਿਆਨੀਆਂ ਨਾਲ ਵਧੇਰੇ ਨੇੜਿਓਂ ਸਹਿਯੋਗ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇਗੀ। ਇਹ ਅੰਤਰ-ਅਨੁਸ਼ਾਸਨੀ ਪਹੁੰਚ GenUI ਸਿਸਟਮ ਬਣਾਉਣ ਵੱਚਿ ਮਹੱਤਵਪੂਰਨ ਹੋਵੇਗੀ ਜੋ ਨਾ ਸਿਰਫ਼ ਦ੍ਰਸ਼ਟੀਗਤ ਤੌਰ 'ਤੇ ਆਕਰਸ਼ਕ ਅਤੇ ਯੂਜ਼ਰ-ਅਨੁਕੂਲ ਹਨ, ਬਲਕਿ ਤਕਨੀਕੀ ਤੌਰ 'ਤੇ ਮਜ਼ਬੂਤ ਅਤੇ ਨੈਤਿਕ ਤੌਰ 'ਤੇ ਸਹੀ ਵੀ ਹਨ।

ਜਵਿੇਂ-ਜਵਿੇਂ ਤਕਨਾਲੋਜੀ ਪਰਪਿੱਕ ਹੁੰਦੀ ਹੈ, GenUI ਦੇ ਨੈਤਿਕ ਪ੍ਰਭਾਵ ਵੀ ਸਾਹਮਣੇ ਆਉਣਗੇ। ਡਿਜ਼ਾਈਨਰ ਇੰਟਰਫੇਸ ਡਿਜ਼ਾਈਨ ਵੱਚਿ ਜ਼ਿੰਮੇਵਾਰ AI ਵਰਤੋਂ ਲਈ ਫਰੇਮਵਰਕ ਵਿਕਸਿਤ ਕਰਨ ਵੱਚਿ ਮਹੱਤਵਪੂਰਨ ਭੂਮਿਕਾ ਨਭਿਾਉਣਗੇ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦੇ ਹੋਏ ਕਿ ਨਿੱਜੀਕਰਨ ਯੂਜ਼ਰ ਅਨੁਭਵਾਂ ਨੂੰ ਵਧਾਉਂਦਾ ਹੈ, ਪਰ ਗੋਪਨੀਯਤਾ ਨੂੰ ਖ਼ਤਰੇ ਵੱਚਿ ਪਾਏ ਬਨਿਾਂ ਜਾਂ ਅਨੈਤਿਕ ਤਰੀਕਿਆਂ ਨਾਲ ਯੂਜ਼ਰ ਵਵਿਹਾਰ ਨੂੰ ਪ੍ਰਭਾਵਤਿ ਕੀਤੇ ਬਨਿਾਂ।

ਜਵਿੇਂ ਅਸੀਂ ਭਵਿੱਖ ਵੱਲ ਦੇਖਦੇ ਹਾਂ, GenUI ਰੋਮਾਂਚਕ ਮੌਕੇ ਅਤੇ ਮਹੱਤਵਪੂਰਨ ਚੁਣੌਤੀਆਂ ਦੋਵੇਂ ਪੇਸ਼ ਕਰਦਾ ਹੈ। ਇਸ ਵੱਚਿ ਦੁਨੀਆ ਭਰ ਦੇ ਯੂਜ਼ਰਾਂ ਲਈ ਵਧੇਰੇ ਸਹਜਿ, ਕੁਸ਼ਲ, ਅਤੇ ਸੰਤੁਸ਼ਟੀਜਨਕ ਡਜਿੀਟਲ ਅਨੁਭਵ ਬਣਾਉਣ ਦੀ ਸਮਰੱਥਾ ਹੈ। ਭਾਵੇਂ ਇਸ ਲਈ ਡਿਜ਼ਾਈਨਰਾਂ ਨੂੰ ਅਨੁਕੂਲ ਹੋਣ ਅਤੇ ਨਵੇਂ ਹੁਨਰ ਪ੍ਰਾਪਤ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇਗੀ, ਇਹ ਡੂੰਘੇ ਅਤੇ ਅਰਥਪੂਰਨ ਤਰੀਕਿਆਂ ਨਾਲ ਮਨੁੱਖ-ਕੰਪਿਊਟਰ ਇੰਟਰੈਕਸ਼ਨ ਦੇ ਭਵਿੱਖ ਨੂੰ ਆਕਾਰ ਦੇਣ ਦਾ ਇੱਕ ਬੇਮਸਿਾਲ ਮੌਕਾ ਵੀ ਪੇਸ਼ ਕਰਦਾ ਹੈ। ਪੂਰੀ ਤਰ੍ਹਾਂ ਵਿਕਸਿਤ GenUI ਸਿਸਟਮਾਂ ਵੱਲ ਦੀ ਯਾਤਰਾ ਨਸ਼ਿ੍ਚਤਿ ਤੌਰ 'ਤੇ ਗੁੰਝਲਦਾਰ ਹੋਵੇਗੀ, ਪਰ ਬਹਿਤਰ ਯੂਜ਼ਰ ਅਨੁਭਵਾਂ ਅਤੇ ਡਜਿੀਟਲ ਪਹੁੰਚਯੋਗਤਾ ਦੇ ਰੂਪ ਵੱਚਿ ਸੰਭਾਵੀ ਇਨਾਮ ਇਸਨੂੰ ਇੱਕ ਅਜਹਿਾ ਭਵਿੱਖ ਬਣਾਉਂਦੇ ਹਨ ਜਸਿ ਲਈ ਯਤਨ ਕਰਨ ਦੀ ਕੀਮਤ ਹੈ।

ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ

“ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ” ਪਹੁੰਚ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦੇ ਅੰਦਰ ਜਟਲਿ ਵਰਕਫਲੋ ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਔਰਕੈਸਟ੍ਰੇਟ ਅਤੇ ਅਨੁਕੂਲ ਕਰਨ ਲਈ ਏ.ਆਈ. ਕੰਪੋਨੈਂਟਸ ਦੀ ਵਰਤੋਂ ’ਤੇ ਕੇਂਦਰਤਿ ਹੈ। ਇਸਦਾ ਉਦੇਸ਼ ਅਜਹਿੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣਾ ਹੈ ਜੋ ਵਧੇਰੇ ਕੁਸ਼ਲ, ਪ੍ਰਤੀਕਰਿਆਸ਼ੀਲ, ਅਤੇ ਰੀਅਲ-ਟਾਈਮ ਡੇਟਾ ਅਤੇ ਸੰਦਰਭ ਦੇ ਅਨੁਕੂਲ ਹੋਣ।

ਇਸ ਅਧਿਆਇ ਵਿੱਚ, ਅਸੀਂ ਉਨ੍ਹਾਂ ਮੁੱਖ ਸਧਿਾਂਤਾਂ ਅਤੇ ਪੈਟਰਨਾਂ ਦੀ ਪੜਚੋਲ ਕਰਾਂਗੇ ਜੋ ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ ਪਹੁੰਚ ਦਾ ਆਧਾਰ ਹਨ। ਅਸੀਂ ਵਚਿਾਰ ਕਰਾਂਗੇ ਕ ਿਏ.ਆਈ. ਨੂੰ ਕਾਰਜਾਂ ਨੂੰ ਬੁੱਧੀਮਾਨੀ ਨਾਲ ਰੂਟ ਕਰਨ, ਫੈਸਲਾ ਲੈਣ ਨੂੰ ਸਵੈਚਾਲਤਿ ਕਰਨ, ਅਤੇ ਯੂਜ਼ਰ ਵਵਿਹਾਰ, ਸਸਿਟਮ ਪ੍ਰਦਰਸ਼ਨ, ਅਤੇ ਵਪਾਰਕ ਨਯਿਮਾਂ ਵਰਗੇ ਵੱਖ-ਵੱਖ ਕਾਰਕਾਂ ਦੇ ਆਧਾਰ ’ਤੇ ਵਰਕਫਲੋ ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਅਨੁਕੂਲ ਕਰਨ ਲਈ ਕਵਿੇਂ ਵਰਤਆਿ ਜਾ ਸਕਦਾ ਹੈ। ਵਵਿਹਾਰਕ ਉਦਾਹਰਣਾਂ ਅਤੇ ਅਸਲ-ਸੰਸਾਰ ਦੇ ਸਥਤਿੀਆਂ ਰਾਹੀਂ, ਅਸੀਂ ਐਪਲੀਕੇਸ਼ਨ ਵਰਕਫਲੋ ਨੂੰ ਸਟ੍ਰੀਮਲਾਈਨ ਅਤੇ ਅਨੁਕੂਲ ਕਰਨ ਵਿੱਚ ਏ.ਆਈ. ਦੀ ਪਰਵਿਰਤਨਕਾਰੀ ਸੰਭਾਵਨਾ ਨੂੰ ਪ੍ਰਦਰਸ਼ਤਿ ਕਰਾਂਗੇ।

ਭਾਵੇਂ ਤੁਸੀਂ ਜਟਲਿ ਵਪਾਰਕ ਪ੍ਰਕਰਿਆਵਾਂ ਵਾਲੀਆਂ ਐਂਟਰਪ੍ਰਾਈਜ਼ ਐਪਲੀਕੇਸ਼ਨਾਂ ਜਾਂ ਗਤੀਸ਼ੀਲ ਯੂਜ਼ਰ ਯਾਤਰਾਵਾਂ ਵਾਲੀਆਂ ਉਪਭੋਗਤਾ-ਕੇਂਦਰਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾ ਰਹੇ ਹੋ, ਇਸ ਅਧਿਆਇ ਵਿੱਚ ਚਰਚਾ ਕੀਤੇ ਗਏ

ਪੈਟਰਨ ਅਤੇ ਤਕਨੀਕਾਂ ਤੁਹਾਨੂੰ ਬੁੱਧੀਮਾਨ ਅਤੇ ਕੁਸ਼ਲ ਵਰਕਫਲੋ ਬਣਾਉਣ ਲਈ ਗਿਆਨ ਅਤੇ ਟੂਲਸ ਨਾਲ ਲੈਸ ਕਰਨਗੀਆਂ ਜੋ ਸਮੁੱਚੇ ਯੂਜ਼ਰ ਅਨੁਭਵ ਨੂੰ ਵਧਾਉਂਦੀਆਂ ਹਨ ਅਤੇ ਵਪਾਰਕ ਮੁੱਲ ਨੂੰ ਵਧਾਉਂਦੀਆਂ ਹਨ।

ਵਪਾਰਕ ਲੋੜ

ਵਰਕਫਲੋ ਪ੍ਰਬੰਧਨ ਦੇ ਪਰੰਪਰਾਗਤ ਤਰੀਕੇ ਅਕਸਰ ਪਹਲਿਾਂ ਤੋਂ ਨਰਿਧਾਰਤਿ ਨਯਿਮਾਂ ਅਤੇ ਸਥਰਿ ਫੈਸਲਾ ਰੁੱਖਾਂ ’ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹਨ, ਜੋ ਕਠੋਰ, ਅਲਚਕਦਾਰ, ਅਤੇ ਆਧੁਨਕਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦੀ ਗਤੀਸ਼ੀਲ ਪ੍ਰਕਰਿਤੀ ਨਾਲ ਨਜਿੰਠਣ ਵਿੱਚ ਅਸਮਰੱਥ ਹੋ ਸਕਦੇ ਹਨ।

ਇੱਕ ਅਜਹਿੀ ਸਥਤਿੀ ’ਤੇ ਵਚਿਾਰ ਕਰੋ ਜਿੱਥੇ ਇੱਕ ਈ-ਕਾਮਰਸ ਐਪਲੀਕੇਸ਼ਨ ਨੂੰ ਇੱਕ ਜਟਲਿ ਆਰਡਰ ਪੂਰਤੀ ਪ੍ਰਕਰਿਆ ਨੂੰ ਸੰਭਾਲਣ ਦੀ ਲੋੜ ਹੈ। ਵਰਕਫਲੋ ਵਿੱਚ ਆਰਡਰ ਪ੍ਰਮਾਣੀਕਰਨ, ਇਨਵੈਂਟਰੀ ਚੈੱਕ, ਭੁਗਤਾਨ ਪ੍ਰੋਸੈਸਿੰਗ, ਸ਼ਪਿਿੰਗ, ਅਤੇ ਗਾਹਕ ਸੂਚਨਾਵਾਂ ਵਰਗੇ ਕਈ ਕਦਮ ਸ਼ਾਮਲ ਹੋ ਸਕਦੇ ਹਨ। ਹਰੇਕ ਕਦਮ ਦੇ ਆਪਣੇ ਨਯਿਮ, ਨਰਿਭਰਤਾਵਾਂ, ਬਾਹਰੀ ਏਕੀਕਰਣ, ਅਤੇ ਅਪਵਾਦ ਸੰਭਾਲ ਵਧਿੀਆਂ ਹੋ ਸਕਦੀਆਂ ਹਨ। ਅਜਹਿੇ ਵਰਕਫਲੋ ਨੂੰ ਮੈਨੂਅਲ ਤੌਰ ’ਤੇ ਜਾਂ ਹਾਰਡਕੋਡਡ ਲੌਜਕਿ ਰਾਹੀਂ ਪ੍ਰਬੰਧਤਿ ਕਰਨਾ ਜਲਦੀ ਹੀ ਬੋਝਲ, ਗਲਤੀਆਂ-ਭਰਪੂਰ, ਅਤੇ ਰੱਖ-ਰਖਾਅ ਵਿੱਚ ਮੁਸ਼ਕਲ ਹੋ ਸਕਦਾ ਹੈ।

ਇਸ ਤੋਂ ਇਲਾਵਾ, ਜਵਿੇਂ-ਜਵਿੇਂ ਐਪਲੀਕੇਸ਼ਨ ਦਾ ਪੱਧਰ ਵੱਧਦਾ ਹੈ ਅਤੇ ਇੱਕੋ ਸਮੇਂ ’ਤੇ ਵਰਤੋਂਕਾਰਾਂ ਦੀ ਗਣਿਤੀ ਵੱਧਦੀ ਹੈ, ਵਰਕਫਲੋ ਨੂੰ ਰੀਅਲ-ਟਾਈਮ ਡੇਟਾ ਅਤੇ ਸਸਿਟਮ ਪ੍ਰਦਰਸ਼ਨ ਦੇ ਆਧਾਰ ’ਤੇ ਆਪਣੇ ਆਪ ਨੂੰ ਅਨੁਕੂਲ ਅਤੇ ਆਪਟੀਮਾਈਜ਼ ਕਰਨ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ। ਉਦਾਹਰਨ ਲਈ, ਵੱਧ ਟ੍ਰੈਫਕਿ ਦੇ ਸਮੇਂ ਦੌਰਾਨ, ਐਪਲੀਕੇਸ਼ਨ ਨੂੰ ਕੁਝ ਕਾਰਜਾਂ ਨੂੰ ਤਰਜੀਹ ਦੇਣ, ਸਰੋਤਾਂ ਨੂੰ ਕੁਸ਼ਲਤਾ ਨਾਲ ਵੰਡਣ, ਅਤੇ ਵਰਤੋਂਕਾਰ ਦੇ ਤਜਰਬੇ ਨੂੰ ਸੁਚਾਰੂ ਬਣਾਉਣ ਲਈ ਵਰਕਫਲੋ ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਅਨੁਕੂਲ ਕਰਨ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ।

ਇੱਥੇ “ਇੰਟੈਲੀਜੈਂਟ ਵਰਕਫਲੋ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ” ਪਹੁੰਚ ਕੰਮ ਵਿੱਚ ਆਉਂਦੀ ਹੈ। AI ਕੰਪੋਨੈਂਟਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਡਵੈਲਪਰ ਅਜਹਿੇ ਵਰਕਫਲੋ ਬਣਾ ਸਕਦੇ ਹਨ ਜੋ ਬੁੱਧੀਮਾਨ, ਅਨੁਕੂਲ, ਅਤੇ ਸਵੈ-ਆਪਟੀਮਾਈਜ਼ਿੰਗ ਹਨ। AI ਵੱਡੀ ਮਾਤਰਾ ਵਿੱਚ ਡੇਟਾ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰ ਸਕਦਾ ਹੈ, ਪਛਿਲੇ ਤਜਰਬਿਆਂ ਤੋਂ ਸਖਿ ਸਕਦਾ ਹੈ, ਅਤੇ ਵਰਕਫਲੋ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਚਲਾਉਣ ਲਈ ਰੀਅਲ-ਟਾਈਮ ਵਿੱਚ ਸੂਝਵਾਨ ਫੈਸਲੇ ਲੈ ਸਕਦਾ ਹੈ।

ਮੁੱਖ ਲਾਭ

1. **ਵਧੀ ਹੋਈ ਕੁਸ਼ਲਤਾ:** AI ਕਾਰਜ ਵੰਡ, ਸਰੋਤ ਵਰਤੋਂ, ਅਤੇ ਵਰਕਫਲੋ ਕਾਰਜਾਂਵਧਿੀ ਨੂੰ ਆਪਟੀਮਾਈਜ਼ ਕਰ ਸਕਦਾ ਹੈ, ਜਸਿ ਨਾਲ ਤੇਜ਼ ਪ੍ਰੋਸੈਸਿੰਗ ਸਮਾਂ ਅਤੇ ਸਮੁੱਚੀ ਕੁਸ਼ਲਤਾ ਵਿੱਚ ਸੁਧਾਰ ਹੁੰਦਾ ਹੈ।

2. **ਅਨੁਕੂਲਤਾ:** AI-ਸੰਚਾਲਤਿ ਵਰਕਫਲੋ ਬਦਲਦੀਆਂ ਸਥਤਿੀਆਂ ਦੇ ਅਨੁਸਾਰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਢਲ ਸਕਦੇ ਹਨ, ਜਵਿੇਂ ਕ ਿਵਰਤੋਂਕਾਰ ਮੰਗ ਵ ਿਚ ਉਤਾਰ-ਚੜ੍ਹਾਅ, ਸਸਿਟਮ ਪ੍ਰਦਰਸ਼ਨ, ਜਾਂ ਵਪਾਰਕ ਲੋੜਾਂ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਦੇ ਹੋਏ ਕ ਿਐਪਲੀਕੇਸ਼ਨ ਪ੍ਰਤੀਕਰਿਆਸ਼ੀਲ ਅਤੇ ਲਚਕਦਾਰ ਰਹੇ।
3. **ਸਵੈਚਾਲਤਿ ਫੈਸਲਾ-ਲੈਣਾ:** AI ਵਰਕਫਲੋ ਵ ਿਚ ਗੁੰਝਲਦਾਰ ਫੈਸਲਾ-ਲੈਣ ਦੀਆਂ ਪ੍ਰਕਰਿਆਵਾਂ ਨੂੰ ਸਵੈਚਾਲਤਿ ਕਰ ਸਕਦਾ ਹੈ, ਜੋ ਮੈਨੂਅਲ ਦਖਲਅੰਦਾਜ਼ੀ ਨੂੰ ਘਟਾਉਦਾ ਹੈ ਅਤੇ ਮਨੁੱਖੀ ਗਲਤੀਆਂ ਦੇ ਜੋਖਮ ਨੂੰ ਘੱਟ ਕਰਦਾ ਹੈ।
4. **ਨਜ਼ੀਕਰਨ:** AI ਵਰਤੋਂਕਾਰ ਵਵ ਿਹਾਰ, ਤਰਜੀਹਾਂ, ਅਤੇ ਸੰਦਰਭ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਕੇ ਵਰਕਫਲੋ ਨੂੰ ਨਜ਼ੀ ਬਣਾ ਸਕਦਾ ਹੈ ਅਤੇ ਵਅ ਿਕਤੀਗਤ ਵਰਤੋਂਕਾਰਾਂ ਨੂੰ ਉਨ੍ਹਾਂ ਦੇ ਅਨੁਕੂਲ ਤਜਰਬੇ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦਾ ਹੈ।
5. **ਸਕੇਲੇਬਲਿਟੀ:** AI-ਸੰਚਾਲਤਿ ਵਰਕਫਲੋ ਪ੍ਰਦਰਸ਼ਨ ਜਾਂ ਵਸ਼ਿਵਾਸਯੋਗਤਾ ਨੂੰ ਪ੍ਰਭਾਵਤਿ ਕੀਤੇ ਬਨਿਾਂ, ਵੱਧਦੇ ਡੇਟਾ ਅਤੇ ਵਰਤੋਂਕਾਰ ਇੰਟਰੈਕਸ਼ਨਾਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਨਰਿਵਘਿਨ ਢੰਗ ਨਾਲ ਸਕੇਲ ਕਰ ਸਕਦੇ ਹਨ।

ਅਗਲੇ ਭਾਗਾਂ ਵ ਿਚ, ਅਸੀਂ ਉਨ੍ਹਾਂ ਮੁੱਖ ਪੈਟਰਨਾਂ ਅਤੇ ਤਕਨੀਕਾਂ ਦੀ ਪੜਚੋਲ ਕਰਾਂਗੇ ਜੋ ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਦੀ ਲਾਗੂਕਰਨ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਦੇ ਹਨ ਅਤੇ ਅਸਲ-ਸੰਸਾਰ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਦਖਿਾਵਾਂਗੇ ਕ ਿਕਵਿੇਂ AI ਆਧੁਨਕਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵ ਿਚ ਵਰਕਫਲੋ ਪ੍ਰਬੰਧਨ ਨੂੰ ਬਦਲ ਰਹਿਾ ਹੈ।

ਮੁੱਖ ਪੈਟਰਨ

ਐਪਲੀਕੇਸ਼ਨਾਂ ਵ ਿਚ ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਨੂੰ ਲਾਗੂ ਕਰਨ ਲਈ, ਡਵ ਿਲਪਰ ਕਈ ਮੁੱਖ ਪੈਟਰਨਾਂ ਦਾ ਲਾਭ ਉਠਾ ਸਕਦੇ ਹਨ ਜੋ AI ਦੀ ਸ਼ਕਤੀ ਦਾ ਉਪਯੋਗ ਕਰਦੇ ਹਨ। ਇਹ ਪੈਟਰਨ ਵਰਕਫਲੋ ਨੂੰ ਡਜ਼ਾਇਨ ਅਤੇ ਪ੍ਰਬੰਧਤਿ ਕਰਨ ਲਈ ਇੱਕ ਸੰਰਚਤਿ ਪਹੁੰਚ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ, ਜੋ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਰੀਅਲ-ਟਾਈਮ ਡੇਟਾ ਅਤੇ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਅਨੁਕੂਲ, ਆਪਟੀਮਾਈਜ਼, ਅਤੇ ਸਵੈਚਾਲਤਿ ਪ੍ਰਕਰਿਆਵਾਂ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਦੇ ਹਨ। ਆਓ ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਵ ਿਚ ਕੁਝ ਬੁਨਆਿਦੀ ਪੈਟਰਨਾਂ ਦੀ ਪੜਚੋਲ ਕਰੀਏ।

ਡਾਇਨਾਮਕਿ ਟਾਸਕ ਰੂਟਿੰਗ

ਇਹ ਪੈਟਰਨ AI ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਵੱਖ-ਵੱਖ ਕਾਰਕਾਂ ਜਵਿੇਂ ਕਾਰਜ ਦੀ ਤਰਜੀਹ, ਸਰੋਤਾਂ ਦੀ ਉਪਲਬਧਤਾ, ਅਤੇ ਸਸਿਟਮ ਪ੍ਰਦਰਸ਼ਨ ਦੇ ਆਧਾਰ 'ਤੇ ਵਰਕਫਲੋ ਵ ਿਚ ਕਾਰਜਾਂ ਨੂੰ ਬੁੱਧੀਮਾਨੀ ਨਾਲ ਰੂਟ ਕਰਨ ਨਾਲ ਸਬੰਧਤਿ ਹੈ। AI ਐਲਗੋਰਦਿਮ ਹਰੇਕ ਕਾਰਜ ਦੀਆਂ ਵਸ਼ਿੇਸ਼ਤਾਵਾਂ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰ ਸਕਦੇ ਹਨ, ਸਸਿਟਮ ਦੀ ਮੌਜੂਦਾ ਸਥਤਿੀ

'ਤੇ ਵਿਚਾਰ ਕਰ ਸਕਦੇ ਹਨ, ਅਤੇ ਕਾਰਜਾਂ ਨੂੰ ਸਭ ਤੋਂ ਢੁਕਵੇਂ ਸਰੋਤਾਂ ਜਾਂ ਪ੍ਰੋਸੈਸਿੰਗ ਰਸਤਿਆਂ 'ਤੇ ਨਿਰਧਾਰਤ ਕਰਨ ਲਈ ਸੂਝਵਾਨ ਫੈਸਲੇ ਲੈ ਸਕਦੇ ਹਨ। ਡਾਇਨਾਮਿਕ ਟਾਸਕ ਰੂਟਿੰਗ ਯਕੀਨੀ ਬਣਾਉਂਦੀ ਹੈ ਕਿ ਕਾਰਜ ਕੁਸ਼ਲਤਾ ਨਾਲ ਵੰਡੇ ਅਤੇ ਕਾਰਜਾਂਵਿਤ ਕੀਤੇ ਜਾਂਦੇ ਹਨ, ਜੋ ਸਮੁੱਚੇ ਵਰਕਫਲੋ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਆਪਟੀਮਾਈਜ਼ ਕਰਦੀ ਹੈ।

```
class TaskRouter
  include Raix::ChatCompletion
  include Raix::FunctionDispatch

  attr_accessor :task

  # list of functions that can be called by the AI entirely at its
  # discretion depending on the task received

  function :analyze_task_priority do
    TaskPriorityAnalyzer.perform(task)
  end

  function :check_resource_availability, # ...
  function :assess_system_performance, # ...
  function :assign_task_to_resource, # ...

  DIRECTIVE = "You are a task router, responsible for intelligently
   assigning tasks to available resources based on priority, resource
   availability, and system performance..."

  def initialize(task)
    self.task = task
    transcript << { system: DIRECTIVE }
    transcript << { user: task.to_json }
  end

  def perform
    while task.unassigned?
      chat_completion

      # todo: add max loop counter and break
    end

    # capture the transcript for later analysis
    task.update(routing_transcript: transcript)
  end
```

```
38 end
```

ਲਾਈਨ 29 'ਤੇ `while` ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੁਆਰਾ ਬਣਾਏ ਗਏ ਲੂਪ ਨੂੰ ਧਿਆਨ ਵਿੱਚ ਰੱਖੋ, ਜੋ ਕਿ ਕੰਮ ਨਿਰਧਾਰਤਿ ਹੋਣ ਤੱਕ AI ਨੂੰ ਪ੍ਰੌਂਪਟ ਕਰਦਾ ਰਹਿੰਦਾ ਹੈ। ਲਾਈਨ 35 'ਤੇ, ਅਸੀਂ ਬਾਅਦ ਵਿੱਚ ਵਿਸ਼ਲੇਸ਼ਣ ਅਤੇ ਡੀਬੱਗਿੰਗ ਲਈ ਕੰਮ ਦੀ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਨੂੰ ਸੰਭਾਲਦੇ ਹਾਂ, ਜੇਕਰ ਇਹ ਜ਼ਰੂਰੀ ਹੋਵੇ।

ਸੰਦਰਭਕਿ ਫੈਸਲਾ ਲੈਣਾ

ਤੁਸੀਂ ਵਰਕਫਲੋ ਵਿੱਚ ਸੰਦਰਭ-ਜਾਗਰੂਕ ਫੈਸਲੇ ਲੈਣ ਲਈ ਬਹੁਤ ਮਿਲਦਾ-ਜੁਲਦਾ ਕੋਡ ਵਰਤ ਸਕਦੇ ਹੋ। ਵਰਤੋਂਕਾਰ ਤਰਜੀਹਾਂ, ਇਤਿਹਾਸਕ ਪੈਟਰਨ, ਅਤੇ ਰੀਅਲ-ਟਾਈਮ ਇਨਪੁੱਟ ਵਰਗੇ ਢੁਕਵੇਂ ਡਾਟਾ ਪੁਆਇੰਟਾਂ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਕੇ, AI ਕੰਪੋਨੈਂਟ ਵਰਕਫਲੋ ਵਿੱਚ ਹਰੇਕ ਫੈਸਲੇ ਦੇ ਬਿੰਦੂ 'ਤੇ ਸਭ ਤੋਂ ਢੁਕਵੀਂ ਕਾਰਵਾਈ ਨਿਰਧਾਰਤਿ ਕਰ ਸਕਦੇ ਹਨ। ਹਰ ਵਰਤੋਂਕਾਰ ਜਾਂ ਸਥਿਤੀ ਦੇ ਵਿਸ਼ੇਸ਼ ਸੰਦਰਭ ਦੇ ਆਧਾਰ 'ਤੇ ਆਪਣੇ ਵਰਕਫਲੋ ਦੇ ਵਵਿਹਾਰ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਓ, ਨਿੱਜੀ ਅਤੇ ਅਨੁਕੂਲਤਿ ਤਜਰਬੇ ਪ੍ਰਦਾਨ ਕਰੋ।

ਅਨੁਕੂਲੀ ਵਰਕਫਲੋ ਰਚਨਾ

ਇਹ ਪੈਟਰਨ ਬਦਲਦੀਆਂ ਲੋੜਾਂ ਜਾਂ ਹਾਲਾਤਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਵਰਕਫਲੋ ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਰਚਣ ਅਤੇ ਵਵਿਸਥਤਿ ਕਰਨ 'ਤੇ ਕੇਂਦਰਤਿ ਹੈ। AI ਵਰਕਫਲੋ ਦੀ ਮੌਜੂਦਾ ਸਥਿਤੀ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰ ਸਕਦਾ ਹੈ, ਰੁਕਾਵਟਾਂ ਜਾਂ ਅਕੁਸ਼ਲਤਾਵਾਂ ਦੀ ਪਛਾਣ ਕਰ ਸਕਦਾ ਹੈ, ਅਤੇ ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣ ਲਈ ਵਰਕਫਲੋ ਢਾਂਚੇ ਨੂੰ ਆਪਣੇ ਆਪ ਸੰਸ਼ੋਧਤਿ ਕਰ ਸਕਦਾ ਹੈ। ਅਨੁਕੂਲੀ ਵਰਕਫਲੋ ਰਚਨਾ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਮੈਨੂਅਲ ਦਖਲ ਦੀ ਲੋੜ ਤੋਂ ਬਨਿਾਂ ਲਗਾਤਾਰ ਵਕਿਸਤਿ ਹੋਣ ਅਤੇ ਆਪਣੀਆਂ ਪ੍ਰਕਰਿਆਵਾਂ ਨੂੰ ਬਹਿਤਰ ਬਣਾਉਣ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦੀ ਹੈ।

ਅਪਵਾਦ ਸੰਭਾਲ ਅਤੇ ਰਕਿਵਰੀ

ਅਪਵਾਦ ਸੰਭਾਲ ਅਤੇ ਰਕਿਵਰੀ ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਦੇ ਮਹੱਤਵਪੂਰਨ ਪਹਲਿੂ ਹਨ। AI ਕੰਪੋਨੈਂਟਾਂ ਅਤੇ ਗੁੰਝਲਦਾਰ ਵਰਕਫਲੋ ਨਾਲ ਕੰਮ ਕਰਦੇ ਸਮੇਂ, ਸਸਿਟਮ ਦੀ ਸਥਰਿਤਾ ਅਤੇ ਵਿਸ਼ਵਾਸਯੋਗਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਅਪਵਾਦਾਂ ਦੀ ਅਗਾਊਂ ਕਲਪਨਾ ਕਰਨਾ ਅਤੇ ਉਨ੍ਹਾਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲਣਾ ਜ਼ਰੂਰੀ ਹੈ।

ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਵਿੱਚ ਅਪਵਾਦ ਸੰਭਾਲ ਅਤੇ ਰਕਿਵਰੀ ਲਈ ਕੁਝ ਮੁੱਖ ਵਚਿਾਰ ਅਤੇ ਤਕਨੀਕਾਂ ਹਨ:

1. **ਅਪਵਾਦ ਪ੍ਰਸਾਰ:** ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟਾਂ ਵਿੱਚ ਅਪਵਾਦਾਂ ਦੇ ਪ੍ਰਸਾਰ ਲਈ ਇੱਕ ਸਥਰਿ ਪਹੁੰਚ ਲਾਗੂ ਕਰੋ। ਜਦੋਂ ਕਸਿੇ ਕੰਪੋਨੈਂਟ ਵਿੱਚ ਕੋਈ ਅਪਵਾਦ ਵਾਪਰਦਾ ਹੈ, ਇਸਨੂੰ ਫੜਿਆ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ, ਲੌਗ ਕੀਤਾ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ, ਅਤੇ ਆਰਕੈਸਟ੍ਰੇਟਰ ਜਾਂ ਅਪਵਾਦਾਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਜ਼ਿੰਮੇਵਾਰ ਵੱਖਰੇ ਕੰਪੋਨੈਂਟ ਤੱਕ ਪ੍ਰਸਾਰਤਿ ਕੀਤਾ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ। ਵਚਿਾਰ ਇਹ ਹੈ ਕ ਿਅਪਵਾਦ ਸੰਭਾਲ ਨੂੰ ਕੇਂਦਰੀਕ੍ਰਤਿ ਕੀਤਾ ਜਾਵੇ ਅਤੇ ਅਪਵਾਦਾਂ ਨੂੰ ਚੁੱਪ-ਚਾਪ ਨਗਿਲਣ ਤੋਂ ਰੋਕਿਆ ਜਾਵੇ, ਨਾਲ ਹੀ ਬੁੱਧੀਮਾਨ ਗਲਤੀ ਸੰਭਾਲ ਲਈ ਸੰਭਾਵਨਾਵਾਂ ਖੋਲ੍ਹੀਆਂ ਜਾਣ।
2. **ਮੁੜ-ਕੋਸ਼ਸ਼ਿ ਵਧਿੀਆਂ:** ਮੁੜ-ਕੋਸ਼ਸ਼ਿ ਵਧਿੀਆਂ ਵਰਕਫਲੋ ਦੀ ਲਚਕਤਾ ਨੂੰ ਬਹਿਤਰ ਬਣਾਉਣ ਅਤੇ ਅਸਥਾਈ ਅਸਫਲਤਾਵਾਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲਣ ਵਿੱਚ ਮਦਦ ਕਰਦੀਆਂ ਹਨ। ਅਸਥਾਈ ਜਾਂ ਰਕਿਵਰ ਕਰਨ ਯੋਗ ਅਪਵਾਦਾਂ ਲਈ ਮੁੜ-ਕੋਸ਼ਸ਼ਿ ਵਧਿੀਆਂ ਨੂੰ ਜ਼ਰੂਰ ਲਾਗੂ ਕਰੋ, ਜਵਿੇਂ ਕ ਿਨੈਟਵਰਕ ਕਨੈਕਟੀਵਟਿੀ ਜਾਂ ਸਰੋਤ ਦੀ ਅਣਉਪਲਬਧਤਾ ਜਸਿ ਨੂੰ ਨਰਿਧਾਰਤਿ ਦੇਰੀ ਤੋਂ ਬਾਅਦ ਆਪਣੇ ਆਪ ਮੁੜ ਕੋਸ਼ਸ਼ਿ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। AI-ਸੰਚਾਲਤਿ ਆਰਕੈਸਟ੍ਰੇਟਰ ਜਾਂ ਅਪਵਾਦ ਹੈਂਡਲਰ ਹੋਣ ਦਾ ਮਤਲਬ ਹੈ ਕ ਿਤੁਹਾਡੀਆਂ ਮੁੜ-ਕੋਸ਼ਸ਼ਿ ਰਣਨੀਤੀਆਂ ਨੂੰ ਮਸ਼ੀਨੀ ਰੂਪ ਵਿੱਚ ਹੋਣ ਦੀ ਲੋੜ ਨਹੀਂ ਹੈ, ਜੋ ਐਕਸਪੋਨੈਂਸ਼ਲ ਫਾਲਬੈਕ ਵਰਗੇ ਨਸ਼ਿਚਤ ਐਲਗੋਰਦਿਮ 'ਤੇ ਨਰਿਭਰ ਕਰਦੀਆਂ ਹਨ। ਤੁਸੀਂ ਅਪਵਾਦ ਨੂੰ ਕਵਿੇਂ ਸੰਭਾਲਣਾ ਹੈ, ਇਸ ਬਾਰੇ ਫੈਸਲਾ ਲੈਣ ਲਈ ਜ਼ਿੰਮੇਵਾਰ AI ਕੰਪੋਨੈਂਟ ਦੇ "ਵਵਿੇਕ" 'ਤੇ ਮੁੜ-ਕੋਸ਼ਸ਼ਿ ਦੀ ਸੰਭਾਲ ਛੱਡ ਸਕਦੇ ਹੋ।
3. **ਫਾਲਬੈਕ ਰਣਨੀਤੀਆਂ:** ਜੇਕਰ ਕੋਈ AI ਕੰਪੋਨੈਂਟ ਵੈਧ ਜਵਾਬ ਦੇਣ ਵਿੱਚ ਅਸਫਲ ਰਹਿੰਦਾ ਹੈ ਜਾਂ ਕਸਿ ਗਲਤੀ ਦਾ ਸਾਹਮਣਾ ਕਰਦਾ ਹੈ—ਜੋ ਕ ਿਇਸਦੀ ਨਵੀਨਤਮ ਪ੍ਰਕਰਿਤੀ ਨੂੰ ਦੇਖਦੇ ਹੋਏ ਆਮ ਘਟਨਾ ਹੈ—ਤਾਂ ਵਰਕਫਲੋ ਨੂੰ ਜਾਰੀ ਰੱਖਣ ਲਈ ਇੱਕ ਫਾਲਬੈਕ ਵਧਿੀ ਹੋਣੀ ਚਾਹੀਦੀ ਹੈ। ਇਸ ਵਿੱਚ ਡਫਿਾਲਟ ਮੁੱਲਾਂ ਦੀ ਵਰਤੋਂ, ਵਕਿਲਪਕ ਐਲਗੋਰਦਿਮ, ਜਾਂ ਫੈਸਲੇ ਲੈਣ ਅਤੇ ਵਰਕਫਲੋ ਨੂੰ ਅੱਗੇ ਵਧਾਉਣ ਲਈ ਮਨੁੱਖੀ-ਇਨ-ਦ-ਲੂਪ ਦੀ ਵਰਤੋਂ ਸ਼ਾਮਲ ਹੋ ਸਕਦੀ ਹੈ।
4. **ਮੁਆਵਜ਼ਾ ਕਾਰਵਾਈਆਂ:** ਆਰਕੈਸਟ੍ਰੇਟਰ ਦੇ ਨਰਿਦੇਸ਼ਾਂ ਵਿੱਚ ਉਹਨਾਂ ਅਪਵਾਦਾਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਮੁਆਵਜ਼ਾ ਕਾਰਵਾਈਆਂ ਬਾਰੇ ਹਦਾਇਤਾਂ ਸ਼ਾਮਲ ਹੋਣੀਆਂ ਚਾਹੀਦੀਆਂ ਹਨ ਜੋ ਆਟੋਮੈਟਕਿ ਤੌਰ 'ਤੇ ਹੱਲ ਨਹੀਂ ਕੀਤੇ ਜਾ ਸਕਦੇ। ਮੁਆਵਜ਼ਾ ਕਾਰਵਾਈਆਂ ਉਹ ਕਦਮ ਹਨ ਜੋ ਅਸਫਲ ਓਪਰੇਸ਼ਨ ਦੇ ਪ੍ਰਭਾਵਾਂ ਨੂੰ ਰੱਦ ਕਰਨ ਜਾਂ ਘੱਟ ਕਰਨ ਲਈ ਚੁੱਕੇ ਜਾਂਦੇ ਹਨ। ਉਦਾਹਰਨ ਲਈ, ਜੇਕਰ ਭੁਗਤਾਨ ਪ੍ਰੋਸੈਸਿੰਗ ਸਟੈਪ ਅਸਫਲ ਹੋ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਮੁਆਵਜ਼ਾ ਕਾਰਵਾਈ ਲੈਣ-ਦੇਣ ਨੂੰ ਰੋਲਬੈਕ ਕਰਨਾ ਅਤੇ ਯੂਜ਼ਰ ਨੂੰ ਸੂਚਤਿ ਕਰਨਾ ਹੋ ਸਕਦੀ ਹੈ। ਮੁਆਵਜ਼ਾ ਕਾਰਵਾਈਆਂ ਅਪਵਾਦਾਂ ਦੇ ਮੌਜੂਦਗੀ ਵਿੱਚ ਡੇਟਾ ਸਥਰਿਤਾ ਅਤੇ ਅਖੰਡਤਾ ਨੂੰ ਬਣਾਈ ਰੱਖਣ ਵਿੱਚ ਮਦਦ ਕਰਦੀਆਂ ਹਨ।
5. **ਅਪਵਾਦ ਨਗਿਰਾਨੀ ਅਤੇ ਚੇਤਾਵਨੀ:** ਮਹੱਤਵਪੂਰਨ ਅਪਵਾਦਾਂ ਬਾਰੇ ਪਤਾ ਲਗਾਉਣ ਅਤੇ ਸੰਬੰਧਤਿ ਹੱਸਿੇਦਾਰਾਂ ਨੂੰ ਸੂਚਤਿ ਕਰਨ ਲਈ ਨਗਿਰਾਨੀ ਅਤੇ ਚੇਤਾਵਨੀ ਵਧਿੀਆਂ ਸਥਾਪਤ ਕਰੋ। ਆਰਕੈਸਟ੍ਰੇਟਰ ਨੂੰ

ਸੀਮਾਵਾਂ ਅਤੇ ਨਯਿਮਾਂ ਤੋਂ ਜਾਣੂ ਕਰਵਾਇਆ ਜਾ ਸਕਦਾ ਹੈ ਤਾਂ ਜੋ ਜਦੋਂ ਅਪਵਾਦ ਕੁਝ ਸੀਮਾਵਾਂ ਤੋਂ ਵੱਧ ਜਾਂਦੇ ਹਨ ਜਾਂ ਜਦੋਂ ਖਾਸ ਕਸਿਮ ਦੇ ਅਪਵਾਦ ਵਾਪਰਦੇ ਹਨ ਤਾਂ ਚੇਤਾਵਨੀਆਂ ਟਰਗਿਰ ਹੋ ਜਾਣ। ਇਹ ਸਮੱਸਿਆਵਾਂ ਦੀ ਪੂਰੀ ਪ੍ਰਣਾਲੀ 'ਤੇ ਪ੍ਰਭਾਵ ਪਾਉਣ ਤੋਂ ਪਹਲਿਾਂ ਉਨ੍ਹਾਂ ਦੀ ਸਕਰਿਆ ਪਛਾਣ ਅਤੇ ਹੱਲ ਕਰਨ ਦੀ ਆਗਿਆ ਦੇਂਦਾ ਹੈ।

ਇੱਥੇ Ruby ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟ ਵੱਚਿ ਅਪਵਾਦ ਪ੍ਰਬੰਧਨ ਅਤੇ ਰਕਿਵਰੀ ਦੀ ਇੱਕ ਉਦਾਹਰਨ ਹੈ:

```ruby
class InventoryManager
  def check_availability(order)
    begin
      # Perform inventory check logic
      inventory = Inventory.find_by(product_id: order.product_id)
      if inventory.available_quantity >= order.quantity
        return true
      else
        raise InsufficientInventoryError,
              "Insufficient inventory for product #{order.product_id}"
      end
    rescue InsufficientInventoryError => e
      # Log the exception
      logger.error("Inventory check failed: #{e.message}")

      # Retry the operation after a delay
      retry_count ||= 0
      if retry_count < MAX_RETRIES
        retry_count += 1
        sleep(RETRY_DELAY)
        retry
      else
        # Fallback to manual intervention
        NotificationService.admin("Inventory check failed: Order #{order.id}")
        return false
      end
    end
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵੱਚਿ, `InventoryManager` ਕੰਪੋਨੈਂਟ ਕਸਿੇ ਦੱਤਿੇ ਆਰਡਰ ਲਈ ਉਤਪਾਦ ਦੀ ਉਪਲਬਧਤਾ ਦੀ ਜਾਂਚ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਉਪਲਬਧ ਮਾਤਰਾ ਨਾਕਾਫ਼ੀ ਹੈ, ਤਾਂ ਇਹ `InsufficientInventoryError`

ਉਠਾਉਂਦਾ ਹੈ। ਅਪਵਾਦ ਨੂੰ ਫੜਿਆ ਜਾਂਦਾ ਹੈ, ਲੌਗ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਅਤੇ ਇੱਕ ਮੁੜ-ਕੋਸ਼ਿਸ਼ ਪ੍ਰਣਾਲੀ ਲਾਗੂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜੇਕਰ ਮੁੜ-ਕੋਸ਼ਿਸ਼ ਦੀ ਸੀਮਾ ਪਾਰ ਹੋ ਜਾਂਦੀ ਹੈ, ਤਾਂ ਕੰਪੋਨੈਂਟ ਐਡਮਿਨ ਨੂੰ ਸੂਚਿਤ ਕਰਕੇ ਮੈਨੂਅਲ ਦਖਲਅੰਦਾਜ਼ੀ 'ਤੇ ਵਾਪਸ ਆ ਜਾਂਦਾ ਹੈ।

ਮਜ਼ਬੂਤ ਅਪਵਾਦ ਪ੍ਰਬੰਧਨ ਅਤੇ ਰਿਕਵਰੀ ਮਕੈਨਿਜ਼ਮ ਨੂੰ ਲਾਗੂ ਕਰਕੇ, ਤੁਸੀਂ ਯਕੀਨੀ ਬਣਾ ਸਕਦੇ ਹੋ ਕਿ ਤੁਹਾਡੇ ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਲਚਕਦਾਰ, ਰੱਖ-ਰਖਾਵ ਯੋਗ, ਅਤੇ ਅਣਚਾਹੀਆਂ ਸਥਿਤੀਆਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲਣ ਦੇ ਯੋਗ ਹਨ।

ਇਹ ਪੈਟਰਨ ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਯੋਜਨ ਦੀ ਨੀਂਹ ਬਣਾਉਂਦੇ ਹਨ ਅਤੇ ਵੱਖ-ਵੱਖ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ ਲੋੜਾਂ ਦੇ ਅਨੁਕੂਲ ਜੋੜੇ ਅਤੇ ਅਨੁਕੂਲ ਬਣਾਏ ਜਾ ਸਕਦੇ ਹਨ। ਇਨ੍ਹਾਂ ਪੈਟਰਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਡਿਵੈਲਪਰ ਅਜਿਹੇ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਬਣਾ ਸਕਦੇ ਹਨ ਜੋ ਲਚਕਦਾਰ, ਮਜ਼ਬੂਤ ਅਤੇ ਪ੍ਰਦਰਸ਼ਨ ਅਤੇ ਯੂਜ਼ਰ ਅਨੁਭਵ ਲਈ ਅਨੁਕੂਲ ਹਨ।

ਅਗਲੇ ਭਾਗ ਵਿੱਚ, ਅਸੀਂ ਦੇਖਾਂਗੇ ਕਿ ਇਹ ਪੈਟਰਨ ਅਮਲੀ ਤੌਰ 'ਤੇ ਕਿਵੇਂ ਲਾਗੂ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ, ਅਸਲ-ਸੰਸਾਰ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਅਤੇ ਕੋਡ ਸਨਿੱਪਟਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਪ੍ਰਬੰਧਨ ਵਿੱਚ AI ਕੰਪੋਨੈਂਟਾਂ ਦੇ ਏਕੀਕਰਣ ਨੂੰ ਦਰਸਾਉਣ ਲਈ।

ਅਮਲੀ ਤੌਰ 'ਤੇ ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਯੋਜਨ ਨੂੰ ਲਾਗੂ ਕਰਨਾ

ਹੁਣ ਜਦੋਂ ਅਸੀਂ ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਯੋਜਨ ਵਿੱਚ ਮੁੱਖ ਪੈਟਰਨਾਂ ਦੀ ਪੜਚੋਲ ਕਰ ਲਈ ਹੈ, ਆਓ ਦੇਖੀਏ ਕਿ ਇਹ ਪੈਟਰਨ ਅਸਲ-ਸੰਸਾਰ ਦੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਕਿਵੇਂ ਲਾਗੂ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਅਸੀਂ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਪ੍ਰਬੰਧਨ ਵਿੱਚ AI ਕੰਪੋਨੈਂਟਾਂ ਦੇ ਏਕੀਕਰਣ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਅਮਲੀ ਉਦਾਹਰਣਾਂ ਅਤੇ ਕੋਡ ਸਨਿੱਪਟ ਪ੍ਰਦਾਨ ਕਰਾਂਗੇ।

ਬੁੱਧੀਮਾਨ ਆਰਡਰ ਪ੍ਰੋਸੈਸਰ

ਆਓ Ruby on Rails ਈ-ਕਾਮਰਸ ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ AI-ਸੰਚਾਲਿਤ `OrderProcessor` ਕੰਪੋਨੈਂਟ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਯੋਜਨ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦੀ ਇੱਕ ਅਮਲੀ ਉਦਾਹਰਣ ਵਿੱਚ ਡੂੰਘਾਈ ਨਾਲ

ਜਾਈਏ। `OrderProcessor` ਪ੍ਰੋਸੈਸ ਮੈਨੇਜਰ ਐਟਰਪ੍ਰਾਈਜ਼ ਏਕੀਕਰਣ ਦੀ ਧਾਰਨਾ ਨੂੰ ਸਾਕਾਰ ਕਰਦਾ ਹੈ ਜਿਸ ਨਾਲ ਅਸੀਂ ਪਹਿਲੀ ਵਾਰ ਅਧਿਆਇ 3 ਵਿੱਚ ਵਰਕਰਾਂ ਦੀ ਬਹੁਤਾਤ ਬਾਰੇ ਚਰਚਾ ਕਰਦੇ ਸਮੇਂ ਮਿਲੇ ਸੀ। ਇਹ ਕੰਪੋਨੈਂਟ ਆਰਡਰ ਪੂਰਤੀ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਦੇ ਪ੍ਰਬੰਧਨ, ਵਿਚਿਕਾਰਲੇ ਨਤੀਜਿਆਂ ਦੇ ਆਧਾਰ 'ਤੇ ਰੂਟਿੰਗ ਫੈਸਲੇ ਲੈਣ, ਅਤੇ ਵੱਖ-ਵੱਖ ਪ੍ਰੋਸੈਸਿੰਗ ਕਦਮਾਂ ਦੇ ਨਿਸ਼ਪਾਦਨ ਦੇ ਆਯੋਜਨ ਲਈ ਜ਼ਿੰਮੇਵਾਰ ਹੋਵੇਗਾ।

ਆਰਡਰ ਪੂਰਤੀ ਪ੍ਰਕਰਿਆ ਵਿੱਚ ਕਈ ਕਦਮ ਸ਼ਾਮਲ ਹਨ ਜਿਵੇਂ ਕਿ ਆਰਡਰ ਪ੍ਰਮਾਣੀਕਰਨ, ਇਨਵੈਂਟਰੀ ਜਾਂਚ, ਭੁਗਤਾਨ ਪ੍ਰੋਸੈਸਿੰਗ, ਅਤੇ ਸ਼ਿਪਿੰਗ। ਹਰ ਕਦਮ ਇੱਕ ਵੱਖਰੀ ਵਰਕਰ ਪ੍ਰਕਰਿਆ ਵਜੋਂ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜੋ ਇੱਕ ਵਿਸ਼ੇਸ਼ ਕਾਰਜ ਕਰਦੀ ਹੈ ਅਤੇ ਨਤੀਜਾ `OrderProcessor` ਨੂੰ ਵਾਪਸ ਕਰਦੀ ਹੈ। ਕਦਮ ਲਾਜ਼ਮੀ ਨਹੀਂ ਹਨ, ਅਤੇ ਜ਼ਰੂਰੀ ਨਹੀਂ ਕਿ ਇੱਕ ਸਟੀਕ ਕ੍ਰਮ ਵਿੱਚ ਕੀਤੇ ਜਾਣ।

ਇੱਥੇ `OrderProcessor` ਦੀ ਇੱਕ ਉਦਾਹਰਣ ਲਾਗੂਕਰਨ ਹੈ। ਇਹ Raix ਤੋਂ ਦੋ ਮਕਿਸਨਿ ਨੂੰ ਵਿਸ਼ੇਸ਼ਤਾ ਦਿੰਦਾ ਹੈ। ਪਹਿਲਾ (`ChatCompletion`) ਇਸਨੂੰ ਚੈਟ ਪੂਰਨਤਾ ਕਰਨ ਦੀ ਯੋਗਤਾ ਦਿੰਦਾ ਹੈ, ਜੋ ਇਸਨੂੰ ਇੱਕ AI ਕੰਪੋਨੈਂਟ ਬਣਾਉਂਦਾ ਹੈ। ਦੂਜਾ (`FunctionDispatch`) AI ਦੁਆਰਾ ਫੰਕਸ਼ਨ ਕਾਲਿੰਗ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ, ਜੋ ਇਸਨੂੰ ਟੈਕਸਟ ਸੰਦੇਸ਼ ਦੀ ਬਜਾਏ ਫੰਕਸ਼ਨ ਇਨਵੋਕੇਸ਼ਨ ਨਾਲ ਪ੍ਰੌਂਪਟ ਦਾ ਜਵਾਬ ਦੇਣ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦਾ ਹੈ।

ਵਰਕਰ ਫੰਕਸ਼ਨ (`validate_order`, `check_inventory`, ਆਦਿ) ਆਪਣੇ ਸਬੰਧਤ ਵਰਕਰ ਕਲਾਸਾਂ ਨੂੰ ਕੰਮ ਸੌਂਪਦੇ ਹਨ, ਜੋ ਏ.ਆਈ. ਜਾਂ ਗੈਰ-ਏ.ਆਈ. ਕੰਪੋਨੈਂਟਸ ਹੋ ਸਕਦੇ ਹਨ, ਜਿਨ੍ਹਾਂ ਲਈ ਸਿਰਫ਼ ਇਹ ਲੋੜ ਹੈ ਕਿ ਉਹ ਆਪਣੇ ਕੰਮ ਦੇ ਨਤੀਜੇ ਅਜਿਹੇ ਫਾਰਮੈਟ ਵਿੱਚ ਵਾਪਸ ਕਰਨ ਜੋ ਸਟ੍ਰਿੰਗ ਦੇ ਰੂਪ ਵਿੱਚ ਦਰਸਾਏ ਜਾ ਸਕਣ।

> ਕਿਤਾਬ ਦੇ ਇਸ ਭਾਗ ਵਿੱਚ ਦਿੱਤੇ ਗਏ ਸਾਰੇ ਹੋਰ ਉਦਾਹਰਣਾਂ ਵਾਂਗ, ਇਹ ਕੋਡ ਅਸਲ ਵਿੱਚ ਸੂਡੋ-ਕੋਡ ਹੈ ਅਤੇ ਸਿਰਫ਼ ਪੈਟਰਨ ਦਾ ਮਤਲਬ ਦੱਸਣ ਅਤੇ ਤੁਹਾਡੀਆਂ ਆਪਣੀਆਂ ਰਚਨਾਵਾਂ ਨੂੰ ਪ੍ਰੇਰਿਤ ਕਰਨ ਲਈ ਹੈ। ਪੈਟਰਨਾਂ ਦੇ ਪੂਰੇ ਵੇਰਵੇ ਅਤੇ ਸੰਪੂਰਨ ਕੋਡ ਉਦਾਹਰਣਾਂ ਭਾਗ 2 ਵਿੱਚ ਸ਼ਾਮਲ ਹਨ।

```ruby
class OrderProcessor
  include Raix::ChatCompletion
  include Raix::FunctionDispatch

  SYSTEM_DIRECTIVE = "You are an order processor, tasked with..."

  def initialize(order)
    self.order = order
    transcript << { system: SYSTEM_DIRECTIVE }
    transcript << { user: order.to_json }
  end

  def perform
    # will continue looping until `stop_looping!` is called
    chat_completion(loop: true)
  end

  # list of functions available to be called by the AI
  # truncated for brevity

  def functions
    [
      {
        name: "validate_order",
        description: "Invoke to check validity of order",
        parameters: {
          ...
      },
      ...
    ]
  end

  # implementation of functions that can be called by the AI
  # entirely at its discretion, depending on the needs of the order

  def validate_order
    OrderValidationWorker.perform(@order)
  end

  def check_inventory
    InventoryCheckWorker.perform(@order)
  end
```

```
43
44    def process_payment
45      PaymentProcessingWorker.perform(@order)
46    end
47
48    def schedule_shipping
49      ShippingSchedulerWorker.perform(@order)
50    end
51
52    def send_confirmation
53      OrderConfirmationWorker.perform(@order)
54    end
55
56    def finished_processing
57      @order.update!(transcript:, processed_at: Time.current)
58      stop_looping!
59    end
60  end
```

ਉਦਾਹਰਣ ਵਾਂਚਿ, ਆਰਡਰ ਪ੍ਰੋਸੈਸਰ ਨੂੰ ਇੱਕ ਆਰਡਰ ਔਬਜੈਕਟ ਨਾਲ ਸ਼ੁਰੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਦਾ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਬਣਾਈ ਰੱਖਦਾ ਹੈ, ਜੋ ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲਾਂ ਦੇ ਮੂਲ ਗੱਲਬਾਤ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਫਾਰਮੈਟ ਵਾਂਚਿ ਹੁੰਦਾ ਹੈ। ਏਆਈ ਨੂੰ ਵੱਖ-ਵੱਖ ਪ੍ਰੋਸੈਸਿੰਗ ਕਦਮਾਂ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਨਿਯੰਤਰਿਤ ਕਰਨ ਲਈ ਪੂਰਾ ਨਿਯੰਤਰਣ ਦੱਤਿਆ ਜਾਂਦਾ ਹੈ, ਜਵਿੇਂ ਕਿ ਆਰਡਰ ਪ੍ਰਮਾਣੀਕਰਨ, ਇਨਵੈਂਟਰੀ ਜਾਂਚ, ਭੁਗਤਾਨ ਪ੍ਰੋਸੈਸਿੰਗ, ਅਤੇ ਸ਼ਿਪਿੰਗ।

ਹਰ ਵਾਰ ਜਦੋਂ `chat_completion` ਮੈਥਡ ਨੂੰ ਕਾਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਨੂੰ ਏਆਈ ਨੂੰ ਇੱਕ ਫੰਕਸ਼ਨ ਕਾਲ ਵਜੋਂ ਪੂਰਤੀ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਭੇਜਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਪੂਰੀ ਤਰ੍ਹਾਂ ਏਆਈ 'ਤੇ ਨਿਰਭਰ ਕਰਦਾ ਹੈ ਕਿ ਉਹ ਪਛਿਲੇ ਕਦਮ ਦੇ ਨਤੀਜੇ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰੇ ਅਤੇ ਉਚਤਿ ਕਾਰਵਾਈ ਕਰਨ ਦਾ ਫੈਸਲਾ ਕਰੇ। ਉਦਾਹਰਣ ਲਈ, ਜੇਕਰ ਇਨਵੈਂਟਰੀ ਜਾਂਚ ਘੱਟ ਸਟਾਕ ਪੱਧਰਾਂ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ, ਤਾਂ `OrderProcessor` ਇੱਕ ਰੀਪਲੇਨਸ਼ਿਮੈਂਟ ਟਾਸਕ ਨੂੰ ਸ਼ੈਡਿਊਲ ਕਰ ਸਕਦਾ ਹੈ। ਜੇਕਰ ਭੁਗਤਾਨ ਪ੍ਰੋਸੈਸਿੰਗ ਅਸਫਲ ਹੋ ਜਾਂਦੀ ਹੈ, ਤਾਂ ਇਹ ਮੁੜ ਕੋਸ਼ਿਸ਼ ਕਰ ਸਕਦਾ ਹੈ ਜਾਂ ਗਾਹਕ ਸਹਾਇਤਾ ਨੂੰ ਸੂਚਤਿ ਕਰ ਸਕਦਾ ਹੈ।

> ਉੱਪਰ ਦੱਤੀ ਉਦਾਹਰਣ ਵਾਂਚਿ ਰੀਪਲੇਨਸ਼ਿਮੈਂਟ ਜਾਂ ਗਾਹਕ ਸਹਾਇਤਾ ਨੂੰ ਸੂਚਤਿ ਕਰਨ ਲਈ ਫੰਕਸ਼ਨ ਪਰਭਾਸ਼ਤਿ

> ਨਹੀਂ ਹਨ, ਪਰ ਬਲਿਕੁਲ ਹੋ ਸਕਦੇ ਹਨ।

ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਹਰ ਵਾਰ ਵੱਧਦੀ ਹੈ ਜਦੋਂ ਇੱਕ ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਦੇ ਰਕਿਾਰਡ ਵਜੋਂ ਕੰਮ ਕਰਦੀ ਹੈ, ਜਸਿ ਵੱਚਿ ਹਰ ਕਦਮ ਦੇ ਨਤੀਜੇ ਅਤੇ ਅਗਲੇ ਕਦਮਾਂ ਲਈ ਏਆਈ-ਜਨਰੇਟਡ ਨਰਿਦੇਸ਼ ਸ਼ਾਮਲ ਹਨ। ਇਸ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਦੀ ਵਰਤੋਂ ਡੀਬੱਗਿੰਗ, ਆਡਟਿੰਗ, ਅਤੇ ਆਰਡਰ ਪੂਰਤੀ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਦਖਿਾਈ ਦੇਣ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।

`OrderProcessor` ਵੱਚਿ ਏਆਈ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਈ-ਕਾਮਰਸ ਐਪਲੀਕੇਸ਼ਨ ਰੀਅਲ-ਟਾਈਮ ਡੇਟਾ ਦੇ ਆਧਾਰ 'ਤੇ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਅਨੁਕੂਲ ਕਰ ਸਕਦੀ ਹੈ ਅਤੇ ਬੁੱਧੀਮਾਨੀ ਨਾਲ ਅਪਵਾਦਾਂ ਨੂੰ ਸੰਭਾਲ ਸਕਦੀ ਹੈ। ਏਆਈ ਕੰਪੋਨੈਂਟ ਸੂਚਤਿ ਫੈਸਲੇ ਲੈ ਸਕਦਾ ਹੈ, ਕਾਰਜ-ਪ੍ਰਵਾਹ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦਾ ਹੈ, ਅਤੇ ਗੁੰਝਲਦਾਰ ਸਥਤਿੀਆਂ ਵੱਚਿ ਵੀ ਨਰਿਵਘਿਨ ਆਰਡਰ ਪ੍ਰੋਸੈਸਿੰਗ ਯਕੀਨੀ ਬਣਾ ਸਕਦਾ ਹੈ।

ਇਹ ਤੱਥ ਕਿ ਵਰਕਰ ਪ੍ਰੋਸੈਸਾਂ 'ਤੇ ਸਰਿਫ਼ ਇੱਕੋ ਲੋੜ ਹੈ ਕਿ ਉਹ ਏਆਈ ਲਈ ਕੁਝ ਸਮਝਣਯੋਗ ਆਉਟਪੁੱਟ ਵਾਪਸ ਕਰਨ ਜਦੋਂ ਅਗਲਾ ਕੀ ਕਰਨਾ ਹੈ ਇਸਦਾ ਫੈਸਲਾ ਕਰ ਰਹੇ ਹੋਣ, ਤੁਹਾਨੂੰ ਇਹ ਸਮਝ ਆਉਣ ਲੱਗ ਸਕਦੀ ਹੈ ਕਿ ਇਹ ਪਹੁੰਚ ਕਵਿੇਂ ਵੱਖ-ਵੱਖ ਸਸਿਟਮਾਂ ਨੂੰ ਇੱਕ ਦੂਜੇ ਨਾਲ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਵੇਲੇ ਆਮ ਤੌਰ 'ਤੇ ਸ਼ਾਮਲ ਇਨਪੁੱਟ/ਆਉਟਪੁੱਟ ਮੈਪਿੰਗ ਕੰਮ ਨੂੰ ਘਟਾ ਸਕਦੀ ਹੈ।

ਬੁੱਧੀਮਾਨ ਕੰਟੈਂਟ ਮੌਡਰੇਟਰ

ਸੋਸ਼ਲ ਮੀਡੀਆ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਆਮ ਤੌਰ 'ਤੇ ਇੱਕ ਸੁਰੱਖਿਅਤ ਅਤੇ ਸਹਿਤਮੰਦ ਕਮਊਿਨਟੀ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਘੱਟੋ-ਘੱਟ ਕੰਟੈਂਟ ਮੌਡਰੇਸ਼ਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਇਹ ਉਦਾਹਰਣ `ContentModerator` ਕੰਪੋਨੈਂਟ ਏਆਈ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਬੁੱਧੀਮਾਨੀ ਨਾਲ ਮੌਡਰੇਸ਼ਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਨੂੰ ਨਯਿੰਤਰਤਿ ਕਰਦਾ ਹੈ, ਕੰਟੈਂਟ ਦੀਆਂ ਵਸ਼ਿੇਸ਼ਤਾਵਾਂ ਅਤੇ ਵੱਖ-ਵੱਖ ਮੌਡਰੇਸ਼ਨ ਕਦਮਾਂ ਦੇ ਨਤੀਜਆਿਂ ਦੇ ਆਧਾਰ 'ਤੇ ਫੈਸਲੇ ਲੈਂਦਾ ਹੈ।

ਮੌਡਰੇਸ਼ਨ ਪ੍ਰਕਰਿਆ ਵੱਚਿ ਕਈ ਕਦਮ ਸ਼ਾਮਲ ਹਨ ਜਵਿੇਂ ਕਿ ਟੈਕਸਟ ਵਸ਼ਿਲੇਸ਼ਣ, ਚੱਤਿਰ ਪਛਾਣ, ਯੂਜ਼ਰ ਪ੍ਰਤਸ਼ਿਠਾ ਮੁਲਾਂਕਣ, ਅਤੇ ਮੈਨੂਅਲ ਸਮੀਖਆਿ। ਹਰੇਕ ਕਦਮ ਨੂੰ ਇੱਕ ਵੱਖਰੇ ਵਰਕਰ ਪ੍ਰੋਸੈੱਸ ਵਜੋਂ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜੋ ਇੱਕ ਵਸ਼ਿੇਸ਼ ਕੰਮ ਕਰਦਾ ਹੈ ਅਤੇ ਨਤੀਜਾ `ContentModerator` ਨੂੰ ਵਾਪਸ ਕਰਦਾ ਹੈ।

ਇੱਥੇ `ContentModerator` ਦੀ ਇੱਕ ਉਦਾਹਰਣ ਲਾਗੂਕਰਨ ਹੈ:

```ruby
class ContentModerator
  include Raix::ChatCompletion
  include Raix::FunctionDispatch

  SYSTEM_DIRECTIVE = "You are a content moderator process manager,
    tasked with the workflow involved in moderating user-generated content..."

  def initialize(content)
    @content = content
    @transcript = [
      { system: SYSTEM_DIRECTIVE },
      { user: content.to_json }
    ]
  end

  def perform
    complete(@transcript)
  end

  def model
    "openai/gpt-4"
  end

  # list of functions available to be called by the AI
  # truncated for brevity

  def functions
    [
      {
        name: "analyze_text",
        # ...
      },
      {
        name: "recognize_image",
        description: "Invoke to describe images...",
        # ...
      },
      {
        name: "assess_user_reputation",
        # ...
      },
      {
```

```
        name: "escalate_to_manual_review",
        # ...
      },
      {
        name: "approve_content",
        # ...
      },
      {
        name: "reject_content",
        # ...
      }
    ]
  end

  # implementation of functions that can be called by the AI
  # entirely at its discretion, depending on the needs of the order

  def analyze_text
    result = TextAnalysisWorker.perform(@content)
    continue_with(result)
  end

  def recognize_image
    result = ImageRecognitionWorker.perform(@content)
    continue_with(result)
  end

  def assess_user_reputation
    result = UserReputationWorker.perform(@content.user)
    continue_with(result)
  end

  def escalate_to_manual_review
    ManualReviewWorker.perform(@content)
    @content.update!(status: 'pending', transcript: @transcript)
  end

  def approve_content
    @content.update!(status: 'approved', transcript: @transcript)
  end

  def reject_content
```

```
    @content.update!(status: 'rejected', transcript: @transcript)
  end

  private

  def continue_with(result)
    @transcript << { function: result }
    complete(@transcript)
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ, `ContentModerator` ਨੂੰ ਇੱਕ ਸਮੱਗਰੀ ਔਬਜੈਕਟ ਨਾਲ ਸ਼ੁਰੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਗੱਲਬਾਤ ਫਾਰਮੈਟ ਵਿੱਚ ਇੱਕ ਨਗਿਰਾਨੀ ਲਖਿਤੀ ਰਕਿਾਰਡ ਬਣਾਈ ਰੱਖਦਾ ਹੈ। AI ਕੰਪੋਨੈਂਟ ਕੋਲ ਨਗਿਰਾਨੀ ਕਾਰਜ-ਪ੍ਰਵਾਹ 'ਤੇ ਪੂਰਾ ਨਯਿੰਤਰਣ ਹੁੰਦਾ ਹੈ, ਜੋ ਸਮੱਗਰੀ ਦੀਆਂ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਅਤੇ ਹਰ ਕਦਮ ਦੇ ਨਤੀਜਿਆਂ ਦੇ ਆਧਾਰ 'ਤੇ ਕਹਿੜੇ ਕਦਮ ਚੁੱਕਣੇ ਹਨ, ਇਹ ਫੈਸਲਾ ਕਰਦਾ ਹੈ।

AI ਦੁਆਰਾ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਉਪਲਬਧ ਕਾਰਜਕਰਤਾ ਫੰਕਸ਼ਨਾਂ ਵਿੱਚ `analyze_text`, `recognize_image`, `assess_user_reputation`, ਅਤੇ `escalate_to_manual_review` ਸ਼ਾਮਲ ਹਨ। ਹਰੇਕ ਫੰਕਸ਼ਨ ਕਾਰਜ ਨੂੰ ਸੰਬੰਧਤਿ ਕਾਰਜਕਰਤਾ ਪ੍ਰਕਰਿਆ (`TextAnalysisWorker`, `ImageRecognitionWorker`, ਆਦਿ) ਨੂੰ ਸੌਂਪਦਾ ਹੈ ਅਤੇ ਨਤੀਜੇ ਨੂੰ ਨਗਿਰਾਨੀ ਲਖਿਤੀ ਰਕਿਾਰਡ ਵਿੱਚ ਜੋੜਦਾ ਹੈ, ਐਸਕਲੇਸ਼ਨ ਫੰਕਸ਼ਨ ਨੂੰ ਛੱਡ ਕੇ, ਜੋ ਇੱਕ ਅੰਤਮਿ ਸਥਤਿੀ ਵਜੋਂ ਕੰਮ ਕਰਦਾ ਹੈ। ਅੰਤ ਵਿੱਚ, `approve_content` ਅਤੇ `reject_content` ਫੰਕਸ਼ਨ ਵੀ ਅੰਤਮਿ ਸਥਤਿੀਆਂ ਵਜੋਂ ਕੰਮ ਕਰਦੇ ਹਨ।

AI ਕੰਪੋਨੈਂਟ ਸਮੱਗਰੀ ਦਾ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਦਾ ਹੈ ਅਤੇ ਢੁਕਵੀਂ ਕਾਰਵਾਈ ਕਰਨ ਦਾ ਫੈਸਲਾ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਸਮੱਗਰੀ ਵਿੱਚ ਚੱਤਿਰ ਹਵਾਲੇ ਸ਼ਾਮਲ ਹਨ, ਤਾਂ ਇਹ ਵਜ਼ੁਅਲ ਸਮੀਖਿਆ ਲਈ `recognize_image` ਕਾਰਜਕਰਤਾ ਨੂੰ ਕਾਲ ਕਰ ਸਕਦਾ ਹੈ। ਜੇਕਰ ਕੋਈ ਕਾਰਜਕਰਤਾ ਸੰਭਾਵੀ ਨੁਕਸਾਨਦੇਹ ਸਮੱਗਰੀ ਬਾਰੇ ਚੇਤਾਵਨੀ ਦੱਿਦਾ ਹੈ, ਤਾਂ AI ਸਮੱਗਰੀ ਨੂੰ ਮੈਨੂਅਲ ਸਮੀਖਿਆ ਲਈ ਭੇਜਣ ਜਾਂ ਸੱਿਧਾ ਰੱਦ ਕਰਨ ਦਾ ਫੈਸਲਾ ਕਰ ਸਕਦਾ ਹੈ। ਪਰ ਚੇਤਾਵਨੀ ਦੀ ਗੰਭੀਰਤਾ 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹੋਏ, AI ਉਸ ਸਮੱਗਰੀ ਨਾਲ ਕਵਿੇਂ ਨਜੰਠਿਣਾ ਹੈ ਜਸਿ ਬਾਰੇ ਇਹ ਹੋਰ ਯਕੀਨੀ ਨਹੀ ਹੈ, ਇਸ ਬਾਰੇ ਫੈਸਲਾ ਲੈਣ ਲਈ ਯੂਜ਼ਰ ਪ੍ਰਤਸਿ਼ਠਾ ਮੁਲਾਂਕਣ ਦੇ ਨਤੀਜਿਆਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦੀ ਚੋਣ ਕਰ ਸਕਦਾ ਹੈ। ਵਰਤੋਂ ਦੇ ਕੇਸ 'ਤੇ ਨਰਿਭਰ ਕਰਦੇ ਹੋਏ, ਸ਼ਾਇਦ ਭਰੋਸੇਯੋਗ ਯੂਜ਼ਰਾਂ ਨੂੰ ਉਹਨਾਂ ਦੀ ਪੋਸਟ ਕਰਨ ਦੀ ਸਮਰੱਥਾ ਵਿੱਚ ਵਧੇਰੇ ਛੋਟ ਮਲਿਦੀ ਹੈ। ਅਤੇ ਇਸੇ ਤਰ੍ਹਾਂ ਹੋਰ...

ਪਛਿਲੀ ਪ੍ਰਕਰਿਆ ਪ੍ਰਬੰਧਕ ਉਦਾਹਰਣ ਵਾਂਗ, ਨਗਿਰਾਨੀ ਲਖਿਤੀ ਰਕਿਾਰਡ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਦਾ ਰਕਿਾਰਡ ਹੈ, ਜਸਿ ਵਿੱਚ ਹਰ ਕਦਮ ਦੇ ਨਤੀਜੇ ਅਤੇ AI-ਜਨਰੇਟ ਕੀਤੇ ਫੈਸਲੇ ਸ਼ਾਮਲ ਹਨ। ਇਹ ਲਖਿਤੀ ਰਕਿਾਰਡ

ਆਡਿਟਿੰਗ, ਪਾਰਦਰਸ਼ਤਾ, ਅਤੇ ਸਮੇਂ ਦੇ ਨਾਲ ਨਿਗਰਾਨੀ ਪ੍ਰਕਰਿਆ ਨੂੰ ਬਹਿਤਰ ਬਣਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।

`ContentModerator` ਵੱਚਿ AI ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਸੋਸ਼ਲ ਮੀਡੀਆ ਐਪਲੀਕੇਸ਼ਨ ਸਮੱਗਰੀ ਦੀਆਂ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਨਿਗਰਾਨੀ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਨੂੰ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਅਨੁਕੂਲ ਕਰ ਸਕਦੀ ਹੈ ਅਤੇ ਜਟਲਿ ਨਿਗਰਾਨੀ ਸਥਤਿੀਆਂ ਨਾਲ ਬੁੱਧੀਮਾਨੀ ਨਾਲ ਨਜਾਂਠਿ ਸਕਦੀ ਹੈ। AI ਕੰਪੋਨੈਂਟ ਸੂਚਤਿ ਫੈਸਲੇ ਲੈ ਸਕਦਾ ਹੈ, ਕਾਰਜ-ਪ੍ਰਵਾਹ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦਾ ਹੈ, ਅਤੇ ਇੱਕ ਸੁਰੱਖਅਿਤ ਅਤੇ ਸਹਿਤਮੰਦ ਕਮਊਿਨਟਿੀ ਅਨੁਭਵ ਨੂੰ ਯਕੀਨੀ ਬਣਾ ਸਕਦਾ ਹੈ।

ਆਓ ਦੋ ਹੋਰ ਉਦਾਹਰਣਾਂ ਦੀ ਪੜਚੋਲ ਕਰੀਏ ਜੋ ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਦੇ ਸੰਦਰਭ ਵੱਚਿ ਭਵਖਿ-ਸੂਚਕ ਕਾਰਜ ਸ਼ੈਡਊਿਲੰਗ ਅਤੇ ਅਪਵਾਦ ਹੈਂਡਲੰਗ ਅਤੇ ਰਕਿਵਰੀ ਨੂੰ ਦਰਸਾਉਂਦੀਆਂ ਹਨ।

ਗਾਹਕ ਸਹਾਇਤਾ ਸਸਿਟਮ ਵੱਚਿ ਭਵਖਿ-ਸੂਚਕ ਕਾਰਜ ਸ਼ੈਡਊਿਲੰਗ

Ruby on Rails ਨਾਲ ਬਣਾਈ ਗਈ ਗਾਹਕ ਸਹਾਇਤਾ ਐਪਲੀਕੇਸ਼ਨ ਵੱਚਿ, ਗਾਹਕਾਂ ਨੂੰ ਸਮੇਂ ਸਰਿ ਸਹਾਇਤਾ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਸਹਾਇਤਾ ਟਕਿਟਾਂ ਦਾ ਕੁਸ਼ਲਤਾ ਨਾਲ ਪ੍ਰਬੰਧਨ ਅਤੇ ਤਰਜੀਹੀਕਰਨ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। `SupportTicketScheduler` ਕੰਪੋਨੈਂਟ ਟਕਿਟ ਦੀ ਤਤਕਾਲਤਾ, ਏਜੰਟ ਦੀ ਮੁਹਾਰਤ, ਅਤੇ ਕੰਮ ਦੇ ਬੋਝ ਵਰਗੇ ਵੱਖ-ਵੱਖ ਕਾਰਕਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਸਹਾਇਤਾ ਟਕਿਟਾਂ ਨੂੰ ਭਵਖਿ-ਸੂਚਕ ਢੰਗ ਨਾਲ ਸ਼ੈਡਊਿਲ ਅਤੇ ਉਪਲਬਧ ਏਜੰਟਾਂ ਨੂੰ ਅਸਾਈਨ ਕਰਨ ਲਈ AI ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ।

```ruby
class SupportTicketScheduler
  include Raix::ChatCompletion
  include Raix::FunctionDispatch

  SYSTEM_DIRECTIVE = "You are a support ticket scheduler,
    tasked with intelligently assigning tickets to available agents..."

  def initialize(ticket)
    @ticket = ticket
    @transcript = [
      { system: SYSTEM_DIRECTIVE },
      { user: ticket.to_json }
    ]
  end

  def perform
```

```
    complete(@transcript)
  end

  def model
    "openai/gpt-4"
  end

  def functions
    [
      {
        name: "analyze_ticket_urgency",
        # ...
      },
      {
        name: "list_available_agents",
        description: "Includes expertise of available agents",
        # ...
      },
      {
        name: "predict_agent_workload",
        description: "Uses historical data to predict upcoming workloads",
        # ...
      },
      {
        name: "assign_ticket_to_agent",
        # ...
      },
      {
        name: "reschedule_ticket",
        # ...
      }
    ]
  end

  # implementation of functions that can be called by the AI
  # entirely at its discretion, depending on the needs of the order

  def analyze_ticket_urgency
    result = TicketUrgencyAnalyzer.perform(@ticket)
    continue_with(result)
  end

```

```
  def list_available_agents
    result = ListAvailableAgents.perform
    continue_with(result)
  end

  def predict_agent_workload
    result = AgentWorkloadPredictor.perform
    continue_with(result)
  end

  def assign_ticket_to_agent
    TicketAssigner.perform(@ticket, @transcript)
  end

  def delay_assignment(until)
    until = DateTimeStandardizer.process(until)
    SupportTicketScheduler.delay(@ticket, @transcript, until)
  end

  private

  def continue_with(result)
    @transcript << { function: result }
    complete(@transcript)
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ, `SupportTicketScheduler` ਨੂੰ ਇੱਕ ਸਹਾਇਤਾ ਟਿਕਟ ਆਬਜੈਕਟ ਨਾਲ ਸ਼ੁਰੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਇੱਕ ਸ਼ੈਡਊਲਿੰਗ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਬਣਾਈ ਰੱਖਦਾ ਹੈ। AI ਭਾਗ ਟਿਕਟ ਦੇ ਵੇਰਵਿਆਂ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਦਾ ਹੈ ਅਤੇ ਟਿਕਟ ਦੀ ਤਾਕੀਦ, ਏਜੰਟ ਮੁਹਾਰਤ, ਅਤੇ ਅਨੁਮਾਨਤਿ ਏਜੰਟ ਕੰਮ ਦੇ ਭਾਰ ਵਰਗੇ ਕਾਰਕਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਟਿਕਟ ਨਿਰਧਾਰਨ ਦੀ ਭਵਿੱਖਬਾਣੀ ਕਰਦਾ ਹੈ।

AI ਦੁਆਰਾ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਉਪਲਬਧ ਫੰਕਸ਼ਨਾਂ ਵਿੱਚ `analyze_ticket_urgency`, `list_available_agents`, `predict_agent_workload`, ਅਤੇ `assign_ticket_to_agent` ਸ਼ਾਮਲ ਹਨ। ਹਰੇਕ ਫੰਕਸ਼ਨ ਕੰਮ ਨੂੰ ਸੰਬੰਧਤਿ ਵਿਸ਼ਲੇਸ਼ਕ ਜਾਂ ਭਵਿੱਖਬਾਣੀ ਕਰਨ ਵਾਲੇ ਭਾਗ ਨੂੰ ਸੌਂਪਦਾ ਹੈ ਅਤੇ ਨਤੀਜੇ ਨੂੰ ਸ਼ੈਡਊਲਿੰਗ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਵਿੱਚ ਜੋੜਦਾ ਹੈ। AI ਕੋਲ `delay_assignment` ਫੰਕਸ਼ਨ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਨਿਰਧਾਰਨ ਨੂੰ ਦੇਰੀ ਕਰਨ ਦਾ ਵਿਕਲਪ ਵੀ ਹੈ।

AI ਭਾਗ ਸ਼ੈਡਊਲਿੰਗ ਟ੍ਰਾਂਸਕ੍ਰਿਪਟ ਦੀ ਜਾਂਚ ਕਰਦਾ ਹੈ ਅਤੇ ਟਿਕਟ ਨਿਰਧਾਰਨ 'ਤੇ ਜਾਣਕਾਰੀ ਭਰਪੂਰ ਫੈਸਲੇ

ਲੈਂਦਾ ਹੈ। ਇਹ ਟਿਕਟ ਦੀ ਤਾਕੀਦ, ਉਪਲਬਧ ਏਜੰਟਾਂ ਦੀ ਮੁਹਾਰਤ, ਅਤੇ ਹਰੇਕ ਏਜੰਟ ਦੇ ਅਨੁਮਾਨਤਿ ਕੰਮ ਦੇ ਭਾਰ ਨੂੰ ਧਿਆਨ ਵਿੱਚ ਰੱਖਦਾ ਹੈ ਤਾਂ ਜੋ ਟਿਕਟ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਸਭ ਤੋਂ ਢੁਕਵੇਂ ਏਜੰਟ ਦਾ ਨਿਰਧਾਰਨ ਕੀਤਾ ਜਾ ਸਕੇ।

ਭਵਿੱਖਬਾਣੀ ਕਰਨ ਵਾਲੀ ਕਾਰਜ ਸ਼ੈਡਊਿਲੰਗ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਗਾਹਕ ਸਹਾਇਤਾ ਐਪਲੀਕੇਸ਼ਨ ਟਿਕਟ ਨਿਰਧਾਰਨ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦੀ ਹੈ, ਜਵਾਬ ਦੇਣ ਦੇ ਸਮੇਂ ਨੂੰ ਘਟਾ ਸਕਦੀ ਹੈ, ਅਤੇ ਸਮੁੱਚੀ ਗਾਹਕ ਸੰਤੁਸ਼ਟੀ ਨੂੰ ਵਧਾ ਸਕਦੀ ਹੈ। ਸਹਾਇਤਾ ਟਿਕਟਾਂ ਦਾ ਸਰਗਰਮ ਅਤੇ ਕੁਸ਼ਲ ਪ੍ਰਬੰਧਨ ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕਿ ਸਹੀ ਟਿਕਟਾਂ ਸਹੀ ਏਜੰਟਾਂ ਨੂੰ ਸਹੀ ਸਮੇਂ 'ਤੇ ਸੌਂਪੀਆਂ ਜਾਂਦੀਆਂ ਹਨ।

ਡਾਟਾ ਪ੍ਰੋਸੈਸਿੰਗ ਪਾਈਪਲਾਈਨ ਵਿੱਚ ਐਕਸੈਪਸ਼ਨ ਹੈਂਡਲਿੰਗ ਅਤੇ ਰਿਕਵਰੀ

ਐਕਸੈਪਸ਼ਨਾਂ ਨੂੰ ਸੰਭਾਲਣਾ ਅਤੇ ਅਸਫਲਤਾਵਾਂ ਤੋਂ ਰਿਕਵਰ ਕਰਨਾ ਡਾਟਾ ਅਖੰਡਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਅਤੇ ਡਾਟਾ ਦੇ ਨੁਕਸਾਨ ਨੂੰ ਰੋਕਣ ਲਈ ਜ਼ਰੂਰੀ ਹੈ।। `DataProcessingOrchestrator` ਭਾਗ AI ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ ਤਾਂ ਜੋ ਡਾਟਾ ਪ੍ਰੋਸੈਸਿੰਗ ਪਾਈਪਲਾਈਨ ਵਿੱਚ ਬੁੱਧੀਮਾਨੀ ਨਾਲ ਐਕਸੈਪਸ਼ਨਾਂ ਨੂੰ ਸੰਭਾਲਿਆ ਜਾ ਸਕੇ ਅਤੇ ਰਿਕਵਰੀ ਪ੍ਰਕਿਰਿਆ ਦਾ ਪ੍ਰਬੰਧਨ ਕੀਤਾ ਜਾ ਸਕੇ

```
class DataProcessingOrchestrator
  include Raix::ChatCompletion
  include Raix::FunctionDispatch

  SYSTEM_DIRECTIVE = "You are a data processing orchestrator..."

  def initialize(data_batch)
    @data_batch = data_batch
    @transcript = [
      { system: SYSTEM_DIRECTIVE },
      { user: data_batch.to_json }
    ]
  end

  def perform
    complete(@transcript)
  end

  def model
    "openai/gpt-4"
  end

```

```
  def functions
    [
      {
        name: "validate_data",
        # ...
      },
      {
        name: "process_data",
        # ...
      },
      {
        name: "request_fix",
        # ...
      },
      {
        name: "retry_processing",
        # ...
      },
      {
        name: "mark_data_as_failed",
        # ...
      },
      {
        name: "finished",
        # ...
      }
    ]
  end

  # implementation of functions that can be called by the AI
  # entirely at its discretion, depending on the needs of the order

  def validate_data
    result = DataValidator.perform(@data_batch)
    continue_with(result)
  rescue ValidationException => e
    handle_validation_exception(e)
  end

  def process_data
    result = DataProcessor.perform(@data_batch)
    continue_with(result)
```

```
  rescue ProcessingException => e
    handle_processing_exception(e)
  end

  def request_fix(description_of_fix)
    result = SmartDataFixer.new(description_of_fix, @data_batch)
    continue_with(result)
  end

  def retry_processing(timeout_in_seconds)
    wait(timeout_in_seconds)
    process_data
  end

  def mark_data_as_failed
    @data_batch.update!(status: 'failed', transcript: @transcript)
  end

  def finished
    @data_batch.update!(status: 'finished', transcript: @transcript)
  end

  private

  def continue_with(result)
    @transcript << { function: result }
    complete(@transcript)
  end

  def handle_validation_exception(exception)
    @transcript << { exception: exception.message }
    complete(@transcript)
  end

  def handle_processing_exception(exception)
    @transcript << { exception: exception.message }
    complete(@transcript)
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵੱਚਿ, `DataProcessingOrchestrator` ਨੂੰ ਡਾਟਾ ਬੈਚ ਆਬਜੈਕਟ ਨਾਲ ਸੁਰੂ ਕੀਤਾ

ਜਾਂਦਾ ਹੈ ਅਤੇ ਇਹ ਪ੍ਰੋਸੈਸਿੰਗ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਨੂੰ ਬਣਾਈ ਰੱਖਦਾ ਹੈ। AI ਕੰਪੋਨੈਂਟ ਡਾਟਾ ਪ੍ਰੋਸੈਸਿੰਗ ਪਾਈਪਲਾਈਨ ਦਾ ਪ੍ਰਬੰਧਨ ਕਰਦਾ ਹੈ, ਐਕਸੈਪਸ਼ਨਾਂ ਨੂੰ ਸੰਭਾਲਦਾ ਹੈ ਅਤੇ ਲੋੜ ਅਨੁਸਾਰ ਅਸਫਲਤਾਵਾਂ ਤੋਂ ਰਕਿਵਰ ਕਰਦਾ ਹੈ।

AI ਦੁਆਰਾ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਉਪਲਬਧ ਫੰਕਸ਼ਨਾਂ ਵਿੱਚ `validate_data`, `process_data`, `request_fix`, `retry_processing`, ਅਤੇ `mark_data_as_failed` ਸ਼ਾਮਲ ਹਨ। ਹਰ ਫੰਕਸ਼ਨ ਕਾਰਜ ਨੂੰ ਸੰਬੰਧਤਿ ਡਾਟਾ ਪ੍ਰੋਸੈਸਿੰਗ ਕੰਪੋਨੈਂਟ ਨੂੰ ਸੌਂਪਦਾ ਹੈ ਅਤੇ ਨਤੀਜੇ ਜਾਂ ਐਕਸੈਪਸ਼ਨ ਵੇਰਵਿਆਂ ਨੂੰ ਪ੍ਰੋਸੈਸਿੰਗ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਵਿੱਚ ਜੋੜਦਾ ਹੈ।

ਜੇਕਰ `validate_data` ਸਟੈੱਪ ਦੌਰਾਨ ਵੈਲੀਡੇਸ਼ਨ ਐਕਸੈਪਸ਼ਨ ਆਉਂਦੀ ਹੈ, ਤਾਂ `handle_validation_exception` ਫੰਕਸ਼ਨ ਐਕਸੈਪਸ਼ਨ ਡਾਟਾ ਨੂੰ ਟ੍ਰਾਂਸਕ੍ਰਪਿਟ ਵਿੱਚ ਜੋੜਦਾ ਹੈ ਅਤੇ ਕੰਟਰੋਲ AI ਨੂੰ ਵਾਪਸ ਕਰ ਦਿੰਦਾ ਹੈ। ਇਸੇ ਤਰ੍ਹਾਂ, ਜੇਕਰ `process_data` ਸਟੈੱਪ ਦੌਰਾਨ ਪ੍ਰੋਸੈਸਿੰਗ ਐਕਸੈਪਸ਼ਨ ਆਉਂਦੀ ਹੈ, ਤਾਂ AI ਰਕਿਵਰੀ ਰਣਨੀਤੀ ਦਾ ਫੈਸਲਾ ਕਰ ਸਕਦੀ ਹੈ।

ਐਕਸੈਪਸ਼ਨ ਦੀ ਪ੍ਰਕਰਿਤੀ ਦੇ ਆਧਾਰ 'ਤੇ, AI ਆਪਣੇ ਵਵਿੇਕ ਨਾਲ `request_fix` ਨੂੰ ਕਾਲ ਕਰਨ ਦਾ ਫੈਸਲਾ ਕਰ ਸਕਦੀ ਹੈ, ਜੋ AI-ਸੰਚਾਲਤਿ `SmartDataFixer` ਕੰਪੋਨੈਂਟ ਨੂੰ ਡੈਲੀਗੇਟ ਕਰਦੀ ਹੈ (ਸੈਲਫ ਹੀਲਿੰਗ ਡਾਟਾ ਚੈਪਟਰ ਦੇਖੋ)। ਡਾਟਾ ਫਕਿਸਰ ਨੂੰ ਸਧਾਰਨ ਅੰਗਰੇਜ਼ੀ ਵਿੱਚ ਵੇਰਵਾ ਮਲਿਦਾ ਹੈ ਕਿ ਇਹ `@data_batch` ਨੂੰ ਕਵਿੇਂ ਸੋਧ ਸਕਦਾ ਹੈ ਤਾਂ ਜੋ ਪ੍ਰੋਸੈਸਿੰਗ ਨੂੰ ਦੁਬਾਰਾ ਕੋਸ਼ਸ਼ਿ ਕੀਤੀ ਜਾ ਸਕੇ। ਸ਼ਾਇਦ ਇੱਕ ਸਫਲ ਮੁੜ-ਕੋਸ਼ਸ਼ਿ ਵਿੱਚ ਡਾਟਾ ਬੈਚ ਤੋਂ ਉਹ ਰਕਿਾਰਡ ਹਟਾਉਣਾ ਸ਼ਾਮਲ ਹੋਵੇਗਾ ਜੋ ਵੈਲੀਡੇਸ਼ਨ ਵਿੱਚ ਅਸਫਲ ਹੋ ਗਏ ਹਨ ਅਤੇ/ਜਾਂ ਉਨ੍ਹਾਂ ਨੂੰ ਮਨੁੱਖੀ ਸਮੀਖਿਆ ਲਈ ਵੱਖਰੀ ਪ੍ਰੋਸੈਸਿੰਗ ਪਾਈਪਲਾਈਨ 'ਤੇ ਕਾਪੀ ਕਰਨਾ? ਸੰਭਾਵਨਾਵਾਂ ਲਗਭਗ ਅਨੰਤ ਹਨ।

AI-ਸੰਚਾਲਤਿ ਐਕਸੈਪਸ਼ਨ ਹੈਂਡਲਿੰਗ ਅਤੇ ਰਕਿਵਰੀ ਨੂੰ ਸ਼ਾਮਲ ਕਰਕੇ, ਡਾਟਾ ਪ੍ਰੋਸੈਸਿੰਗ ਐਪਲੀਕੇਸ਼ਨ ਵਧੇਰੇ ਲਚਕਦਾਰ ਅਤੇ ਗਲਤੀ-ਸਹਣਿਸ਼ੀਲ ਬਣ ਜਾਂਦੀ ਹੈ। `DataProcessingOrchestrator` ਬੁੱਧੀਮਾਨੀ ਨਾਲ ਐਕਸੈਪਸ਼ਨਾਂ ਦਾ ਪ੍ਰਬੰਧਨ ਕਰਦਾ ਹੈ, ਡਾਟਾ ਦੇ ਨੁਕਸਾਨ ਨੂੰ ਘੱਟ ਕਰਦਾ ਹੈ, ਅਤੇ ਡਾਟਾ ਪ੍ਰੋਸੈਸਿੰਗ ਵਰਕਫਲੋ ਦੇ ਸੁਚਾਰੂ ਕਾਰਜਾਂਵੀ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ।

ਨਗਿਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ

ਨਗਿਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ AI-ਸੰਚਾਲਤਿ ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟਾਂ ਦੀ ਪ੍ਰਗਤੀ, ਕਾਰਗੁਜ਼ਾਰੀ, ਅਤੇ ਸਹਿਤ ਦੀ ਦ੍ਰਸ਼ਿਟੀ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ, ਜੋ ਡਵਿੈਲਪਰਾਂ ਨੂੰ ਸਸਿਟਮ ਦੇ ਵਵਿਹਾਰ ਨੂੰ ਟਰੈਕ ਅਤੇ ਵਸ਼ਿਲੇਸ਼ਣ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦੇ ਹਨ। ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਨਗਿਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ ਵਧਿੀਆਂ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋਜ਼ ਦੀ ਡੀਬੱਗਿੰਗ, ਆਡਟਿੰਗ, ਅਤੇ ਨਰਿੰਤਰ ਸੁਧਾਰ ਲਈ ਜ਼ਰੂਰੀ ਹੈ।

ਵਰਕਫਲੋ ਪ੍ਰਗਤੀ ਅਤੇ ਕਾਰਗੁਜ਼ਾਰੀ ਦੀ ਨਗਿਰਾਨੀ

ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋਜ਼ ਦੇ ਸੁਚਾਰੂ ਕਾਰਜਾਂਵੀ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ, ਹਰ ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟ ਦੀ ਪ੍ਰਗਤੀ ਅਤੇ ਕਾਰਗੁਜ਼ਾਰੀ ਦੀ ਨਗਿਰਾਨੀ ਕਰਨਾ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਇਸ ਵਿੱਚ ਵਰਕਫਲੋ ਜੀਵਨ-ਚੱਕਰ ਦੌਰਾਨ ਮੁੱਖ ਮੈਟ੍ਰਕਿਸ ਅਤੇ ਘਟਨਾਵਾਂ ਦੀ ਟਰੈਕਿੰਗ ਸ਼ਾਮਲ ਹੈ।

ਨਗਿਰਾਨੀ ਕਰਨ ਲਈ ਕੁਝ ਮਹੱਤਵਪੂਰਨ ਪਹਲੂ ਹਨ:

1. ਵਰਕਫਲੋ ਕਾਰਜਾਂਵੀ ਸਮਾਂ: ਹਰ ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟ ਦੁਆਰਾ ਆਪਣਾ ਕਾਰਜ ਪੂਰਾ ਕਰਨ ਵਿੱਚ ਲੱਗੇ ਸਮੇਂ ਦੀ ਮਾਪ ਕਰੋ। ਇਹ ਕਾਰਗੁਜ਼ਾਰੀ ਦੀਆਂ ਰੁਕਾਵਟਾਂ ਦੀ ਪਛਾਣ ਕਰਨ ਅਤੇ ਸਮੁੱਚੀ ਵਰਕਫਲੋ ਕੁਸ਼ਲਤਾ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ।

2. ਸਰੋਤ ਵਰਤੋਂ: ਹਰ ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟ ਦੁਆਰਾ ਸਿਸਟਮ ਸਰੋਤਾਂ, ਜਿਵੇਂ ਕਿ CPU, ਮੈਮੋਰੀ, ਅਤੇ ਸਟੋਰੇਜ ਦੀ ਵਰਤੋਂ ਦੀ ਨਗਿਰਾਨੀ ਕਰੋ। ਇਹ ਯਕੀਨੀ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ ਕਿ ਸਿਸਟਮ ਆਪਣੀ ਸਮਰੱਥਾ ਦੇ ਅੰਦਰ ਕੰਮ ਕਰ ਰਿਹਾ ਹੈ ਅਤੇ ਵਰਕਲੋਡ ਨੂੰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਸੰਭਾਲ ਸਕਦਾ ਹੈ।

3. ਗਲਤੀ ਦਰਾਂ ਅਤੇ ਅਪਵਾਦ: ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟਸ ਵਿੱਚ ਗਲਤੀਆਂ ਅਤੇ ਅਪਵਾਦਾਂ ਦੀ ਵਾਪਰਨ ਦੀ ਨਗਿਰਾਨੀ ਕਰੋ। ਇਹ ਸੰਭਾਵੀ ਸਮੱਸਿਆਵਾਂ ਦੀ ਪਛਾਣ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ ਅਤੇ ਸਰਗਰਮ ਗਲਤੀ ਨਪਿਟਾਰੇ ਅਤੇ ਰਿਕਵਰੀ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ।

4. ਫੈਸਲੇ ਦੇ ਬਿੰਦੂ ਅਤੇ ਨਤੀਜੇ: ਵਰਕਫਲੋ ਵਿੱਚ ਫੈਸਲੇ ਦੇ ਬਿੰਦੂਆਂ ਅਤੇ AI-ਸੰਚਾਲਿਤ ਫੈਸਲਿਆਂ ਦੇ ਨਤੀਜਿਆਂ ਦੀ ਨਗਿਰਾਨੀ ਕਰੋ। ਇਹ AI ਕੰਪੋਨੈਂਟਸ ਦੇ ਵਵਿਹਾਰ ਅਤੇ ਪ੍ਰਭਾਵਸ਼ੀਲਤਾ ਬਾਰੇ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।

ਨਗਿਰਾਨੀ ਪ੍ਰਕਰਿਆਵਾਂ ਦੁਆਰਾ ਇਕੱਤਰ ਕੀਤਾ ਡਾਟਾ ਡੈਸ਼ਬੋਰਡਾਂ ਵਿੱਚ ਦਖਿਾਇਆ ਜਾ ਸਕਦਾ ਹੈ ਜਾਂ ਨਯਿਤ ਰਪਿਰਟਾਂ ਲਈ ਇਨਪੁੱਟ ਵਜੋਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਸਿਸਟਮ ਐਡਮਨਿਸਿਟ੍ਰੇਟਰਾਂ ਨੂੰ ਸਿਸਟਮ ਦੀ ਸਹਿਤ ਬਾਰੇ ਜਾਣੂ ਕਰਵਾਉਂਦੀਆਂ ਹਨ।

ਨਗਿਰਾਨੀ ਡਾਟਾ ਨੂੰ ਸਮੀਖਿਆ ਅਤੇ ਸੰਭਾਵੀ ਕਾਰਵਾਈ ਲਈ AI-ਸੰਚਾਲਿਤ ਸਿਸਟਮ ਐਡਮਨਿਸਿਟ੍ਰੇਟਰ ਪ੍ਰਕਰਿਆ ਵਿੱਚ ਭੇਜਿਆ ਜਾ ਸਕਦਾ ਹੈ!

ਮਹੱਤਵਪੂਰਨ ਘਟਨਾਵਾਂ ਅਤੇ ਫੈਸਲਿਆਂ ਦੀ ਲੌਗਿੰਗ

ਲੌਗਿੰਗ ਇੱਕ ਜ਼ਰੂਰੀ ਅਭਿਆਸ ਹੈ ਜਿਸ ਵਿੱਚ ਵਰਕਫਲੋ ਦੀ ਕਾਰਵਾਈ ਦੌਰਾਨ ਵਾਪਰਨ ਵਾਲੀਆਂ ਮਹੱਤਵਪੂਰਨ ਘਟਨਾਵਾਂ, ਫੈਸਲਿਆਂ, ਅਤੇ ਅਪਵਾਦਾਂ ਬਾਰੇ ਢੁਕਵੀਂ ਜਾਣਕਾਰੀ ਨੂੰ ਕੈਪਚਰ ਕਰਨਾ ਅਤੇ ਸਟੋਰ ਕਰਨਾ ਸ਼ਾਮਲ ਹੈ।

ਲੌਗ ਕਰਨ ਲਈ ਕੁਝ ਮਹੱਤਵਪੂਰਨ ਪਹਲੂ ਹਨ:

1. ਵਰਕਫਲੋ ਦੀ ਸ਼ੁਰੂਆਤ ਅਤੇ ਸਮਾਪਤੀ: ਹਰੇਕ ਵਰਕਫਲੋ ਇੰਸਟੈਂਸ ਦੇ ਸ਼ੁਰੂ ਅਤੇ ਅੰਤ ਦੇ ਸਮੇਂ ਨੂੰ ਲੌਗ ਕਰੋ, ਨਾਲ ਹੀ ਕੋਈ ਵੀ ਢੁਕਵਾਂ ਮੈਟਾਡਾਟਾ ਜਿਵੇਂ ਕਿ ਇਨਪੁੱਟ ਡਾਟਾ ਅਤੇ ਯੂਜ਼ਰ ਕੰਟੈਕਸਟ।

2. ਕੰਪੋਨੈਂਟ ਐਗਜ਼ੀਕਿਊਸ਼ਨ: ਹਰੇਕ ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟ ਦੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਵੇਰਵਿਆਂ ਨੂੰ ਲੌਗ ਕਰੋ, ਜਿਸ ਵਿੱਚ ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰ, ਆਉਟਪੁੱਟ ਨਤੀਜੇ, ਅਤੇ ਕੋਈ ਵੀ ਵਿਚਕਾਰਲਾ ਡਾਟਾ ਜੋ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ, ਸ਼ਾਮਲ ਹੈ।

3. AI ਫੈਸਲੇ ਅਤੇ ਤਰਕ: AI ਕੰਪੋਨੈਂਟਸ ਦੁਆਰਾ ਲਏ ਗਏ ਫੈਸਲਿਆਂ ਨੂੰ ਲੌਗ ਕਰੋ, ਨਾਲ ਹੀ ਅੰਦਰੂਨੀ ਤਰਕ ਜਾਂ ਭਰੋਸੇਯੋਗਤਾ ਸਕੋਰ। ਇਹ ਪਾਰਦਰਸ਼ਤਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਅਤੇ AI-ਸੰਚਾਲਿਤ ਫੈਸਲਿਆਂ ਦੀ ਆਡਿਟਿੰਗ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ।

4. ਅਪਵਾਦ ਅਤੇ ਗਲਤੀ ਸੁਨੇਹੇ: ਵਰਕਫਲੋ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੌਰਾਨ ਆਉਣ ਵਾਲੇ ਕਿਸੇ ਵੀ ਅਪਵਾਦ ਜਾਂ ਗਲਤੀ ਸੁਨੇਹਿਆਂ ਨੂੰ ਲੌਗ ਕਰੋ, ਜਿਸ ਵਿੱਚ ਸਟੈਕ ਟਰੇਸ ਅਤੇ ਢੁਕਵੀਂ ਕੰਟੈਕਸਟ ਜਾਣਕਾਰੀ ਸ਼ਾਮਲ ਹੈ।

ਲੌਗਿੰਗ ਨੂੰ ਵੱਖ-ਵੱਖ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਨਾਲ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਜਿਵੇਂ ਕਿ ਲੌਗ ਫਾਈਲਾਂ ਵਿੱਚ ਲਿਖਣਾ, ਲੌਗਾਂ ਨੂੰ ਡਾਟਾਬੇਸ ਵਿੱਚ ਸਟੋਰ ਕਰਨਾ, ਜਾਂ ਲੌਗਾਂ ਨੂੰ ਕੇਂਦਰੀਕ੍ਰਿਤ ਲੌਗਿੰਗ ਸੇਵਾ ਵਿੱਚ ਭੇਜਣਾ। ਇਹ ਮਹੱਤਵਪੂਰਨ ਹੈ ਕਿ ਅਜਿਹਾ ਲੌਗਿੰਗ ਫਰੇਮਵਰਕ ਚੁਣਿਆ ਜਾਵੇ ਜੋ ਲਚਕਤਾ, ਸਕੇਲੇਬਲਿਟੀ, ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਆਰਕੀਟੈਕਚਰ ਨਾਲ ਆਸਾਨ ਏਕੀਕਰਣ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੋਵੇ।

ਇੱਥੇ ਇੱਕ ਉਦਾਹਰਣ ਹੈ ਕਿ Ruby on Rails ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ `ActiveSupport::Logger` ਕਲਾਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਲੌਗਿੰਗ ਕਿਵੇਂ ਲਾਗੂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ:

```ruby
class WorkflowLogger
  def self.log(message, severity = :info)
    @logger ||= ActiveSupport::Logger.new('workflow.log')
    @logger.formatter ||= proc do |severity, datetime, progname, msg|
      "#{datetime} [#{severity}] #{msg}\n"
    end
    @logger.send(severity, message)
  end
end

# Usage example
WorkflowLogger.log("Workflow initiated for order ##{@order.id}")
WorkflowLogger.log("Payment processing completed successfully")
WorkflowLogger.log("Inventory check failed for item ##{item.id}", :error)
```

ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਅਤੇ AI ਫ਼ੈਸਲੇ ਦੇ ਬਿੰਦੂਆਂ ਵਿੱਚ ਰਣਨੀਤਕ ਤੌਰ 'ਤੇ ਲੌਗਿੰਗ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਰੱਖ ਕੇ, ਡਵੈਲਪਰ ਡੀਬੱਗਿੰਗ, ਲੇਖਾ-ਪੜਤਾਲ, ਅਤੇ ਵਿਸ਼ਲੇਸ਼ਣ ਲਈ ਮਹੱਤਵਪੂਰਨ ਜਾਣਕਾਰੀ ਪ੍ਰਾਪਤ ਕਰ ਸਕਦੇ ਹਨ।

ਨਿਗਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ ਦੇ ਲਾਭ

ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਵਿੱਚ ਨਿਗਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦੇ ਕਈ ਲਾਭ ਹਨ:

1. ਡੀਬੱਗਿੰਗ ਅਤੇ ਸਮੱਸਿਆ ਨਿਵਾਰਣ: ਵਿਸਤ੍ਰਤਿ ਲੌਗ ਅਤੇ ਨਿਗਰਾਨੀ ਡਾਟਾ ਡਵੈਲਪਰਾਂ ਨੂੰ ਜਲਦੀ ਸਮੱਸਿਆਵਾਂ ਦੀ ਪਛਾਣ ਅਤੇ ਨਦਿਾਨ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰਦੇ ਹਨ। ਇਹ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਦੇ ਕਾਰਜਾਂ, ਭਾਗਾਂ ਦੀ ਅੰਤਰਕਰਿਆ, ਅਤੇ ਕਸਿੇ ਵੀ ਗਲਤੀਆਂ ਜਾਂ ਅਪਵਾਦਾਂ ਬਾਰੇ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

2. ਕਾਰਗੁਜ਼ਾਰੀ ਅਨੁਕੂਲਨ: ਕਾਰਗੁਜ਼ਾਰੀ ਮੈਟ੍ਰਕਿਸ ਦੀ ਨਿਗਰਾਨੀ ਡਵੈਲਪਰਾਂ ਨੂੰ ਰੁਕਾਵਟਾਂ ਦੀ ਪਛਾਣ ਕਰਨ ਅਤੇ ਬਹਿਤਰ ਕੁਸ਼ਲਤਾ ਲਈ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣ ਦੀ ਇਜਾਜ਼ਤ ਦੇਂਦੀ ਹੈ। ਕਾਰਜ ਸਮੇਂ, ਸਰੋਤ ਵਰਤੋਂ, ਅਤੇ ਹੋਰ ਮੈਟ੍ਰਕਿਸ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਕੇ, ਡਵੈਲਪਰ ਸਸਿਟਮ ਦੀ ਸਮੁੱਚੀ ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਸੁਧਾਰਨ ਲਈ ਸੂਚਤਿ ਫੈਸਲੇ ਲੈ ਸਕਦੇ ਹਨ।

3. ਲੇਖਾ-ਪੜਤਾਲ ਅਤੇ ਪਾਲਣਾ: ਮੁੱਖ ਘਟਨਾਵਾਂ ਅਤੇ ਫੈਸਲਆਿਂ ਦੀ ਲੌਗਿੰਗ ਨਯਿਾਮਕ ਪਾਲਣਾ ਅਤੇ ਜਵਾਬਦੇਹੀ ਲਈ ਲੇਖਾ-ਪੜਤਾਲ ਰਕਿਾਰਡ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ। ਇਹ ਸੰਗਠਨਾਂ ਨੂੰ AI ਭਾਗਾਂ ਦੁਆਰਾ ਕੀਤੀਆਂ ਗਈਆਂ ਕਾਰਵਾਈਆਂ ਦੀ ਟ੍ਰੈਕਿੰਗ ਅਤੇ ਪੜਤਾਲ ਕਰਨ ਅਤੇ ਵਪਾਰਕ ਨਯਿਮਾਂ ਅਤੇ ਕਾਨੂੰਨੀ ਜ਼ਰੂਰਤਾਂ ਦੀ ਪਾਲਣਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ।

4. ਨਰਿੰਤਰ ਸੁਧਾਰ: ਨਗਿਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ ਡਾਟਾ ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹਾਂ ਦੇ ਨਰਿੰਤਰ ਸੁਧਾਰ ਲਈ ਮਹੱਤਵਪੂਰਨ ਇਨਪੁੱਟ ਵਜੋਂ ਕੰਮ ਕਰਦੇ ਹਨ। ਇਤਿਹਾਸਕ ਡਾਟਾ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਕੇ, ਪੈਟਰਨਾਂ ਦੀ ਪਛਾਣ ਕਰਕੇ, ਅਤੇ AI ਫੈਸਲਿਆਂ ਦੀ ਪ੍ਰਭਾਵਸ਼ੀਲਤਾ ਨੂੰ ਮਾਪ ਕੇ, ਡਵੈਲਪਰ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਤਰਕ ਨੂੰ ਲਗਾਤਾਰ ਸੁਧਾਰ ਅਤੇ ਵਧਾ ਸਕਦੇ ਹਨ।

ਵਿਚਾਰ ਅਤੇ ਸਰਵੋਤਮ ਅਭਿਆਸ

ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਵਿੱਚ ਨਗਿਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ ਨੂੰ ਲਾਗੂ ਕਰਦੇ ਸਮੇਂ, ਹੇਠ ਲਿਖੇ ਸਰਵੋਤਮ ਅਭਿਆਸਾਂ 'ਤੇ ਵਿਚਾਰ ਕਰੋ:

1. ਸਪਸ਼ਟ ਨਗਿਰਾਨੀ ਮੈਟ੍ਰਿਕਸ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ: ਕਾਰਜ-ਪ੍ਰਵਾਹ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ ਜ਼ਰੂਰਤਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਨਗਿਰਾਨੀ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਮੁੱਖ ਮੈਟ੍ਰਿਕਸ ਅਤੇ ਘਟਨਾਵਾਂ ਦੀ ਪਛਾਣ ਕਰੋ। ਉਹਨਾਂ ਮੈਟ੍ਰਿਕਸ 'ਤੇ ਧਿਆਨ ਕੇਂਦਰਿਤ ਕਰੋ ਜੋ ਸਿਸਟਮ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ, ਸਿਹਤ, ਅਤੇ ਵਵਿਹਾਰ ਬਾਰੇ ਅਰਥਪੂਰਨ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।

2. ਵਿਸਤ੍ਰਿਤ ਲੌਗਿੰਗ ਲਾਗੂ ਕਰੋ: ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਲੌਗਿੰਗ ਸਟੇਟਮੈਂਟਾਂ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਅਤੇ AI ਫੈਸਲੇ ਦੇ ਬਿੰਦੂਆਂ ਵਿੱਚ ਢੁਕਵੇਂ ਸਥਾਨਾਂ 'ਤੇ ਰੱਖੀਆਂ ਗਈਆਂ ਹਨ। ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰ, ਆਉਟਪੁੱਟ ਨਤੀਜੇ, ਅਤੇ ਕੋਈ ਵੀ ਵਿਚਕਾਰਲਾ ਡਾਟਾ ਜੋ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ, ਵਰਗੀ ਪ੍ਰਸੰਗਿਕ ਜਾਣਕਾਰੀ ਨੂੰ ਕੈਪਚਰ ਕਰੋ।

3. ਸੰਰਚਿਤ ਲੌਗਿੰਗ ਦੀ ਵਰਤੋਂ ਕਰੋ: ਲੌਗ ਡਾਟਾ ਦੀ ਆਸਾਨ ਪਾਰਸਿੰਗ ਅਤੇ ਵਿਸ਼ਲੇਸ਼ਣ ਦੀ ਸਹੂਲਤ ਲਈ ਇੱਕ ਸੰਰਚਿਤ ਲੌਗਿੰਗ ਫਾਰਮੈਟ ਨੂੰ ਅਪਣਾਓ। ਸੰਰਚਿਤ ਲੌਗਿੰਗ ਲੌਗ ਐਂਟਰੀਆਂ ਦੀ ਬਿਹਤਰ ਖੋਜ, ਫਿਲਟਰਿੰਗ, ਅਤੇ ਏਕੀਕਰਣ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦੀ ਹੈ।

4. ਲੌਗ ਰੀਟੈਂਸ਼ਨ ਅਤੇ ਰੋਟੇਸ਼ਨ ਦਾ ਪ੍ਰਬੰਧਨ ਕਰੋ: ਲੌਗ ਫਾਈਲਾਂ ਦੇ ਸਟੋਰੇਜ ਅਤੇ ਜੀਵਨ-ਚੱਕਰ ਦਾ ਪ੍ਰਬੰਧਨ ਕਰਨ ਲਈ ਲੌਗ ਰੀਟੈਂਸ਼ਨ ਅਤੇ ਰੋਟੇਸ਼ਨ ਨੀਤੀਆਂ ਲਾਗੂ ਕਰੋ। ਕਾਨੂੰਨੀ ਜ਼ਰੂਰਤਾਂ, ਸਟੋਰੇਜ ਸੀਮਾਵਾਂ, ਅਤੇ ਵਿਸ਼ਲੇਸ਼ਣ ਦੀਆਂ ਜ਼ਰੂਰਤਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਢੁਕਵੀਂ ਰੀਟੈਂਸ਼ਨ ਮਿਆਦ ਨਿਰਧਾਰਤ ਕਰੋ। ਜੇ ਸੰਭਵ ਹੋਵੇ, ਲੌਗਿੰਗ ਨੂੰ Papertrail ਵਰਗੀ ਤੀਜੀ-ਧਿਰ ਸੇਵਾ 'ਤੇ ਭੇਜੋ।

5. ਸੰਵੇਦਨਸ਼ੀਲ ਜਾਣਕਾਰੀ ਨੂੰ ਸੁਰੱਖਿਅਤ ਕਰੋ: ਨਿੱਜੀ ਪਛਾਣਯੋਗ ਜਾਣਕਾਰੀ (ਪੀਆਈਆਈ) ਜਾਂ ਗੁਪਤ ਵਪਾਰਕ ਡਾਟਾ ਵਰਗੀ ਸੰਵੇਦਨਸ਼ੀਲ ਜਾਣਕਾਰੀ ਨੂੰ ਲੌਗ ਕਰਦੇ ਸਮੇਂ ਸਾਵਧਾਨ ਰਹੋ। ਲੌਗ ਫਾਈਲਾਂ ਵਿੱਚ ਸੰਵੇਦਨਸ਼ੀਲ ਜਾਣਕਾਰੀ ਦੀ ਸੁਰੱਖਿਆ ਲਈ ਡਾਟਾ ਮਾਸਕਿੰਗ ਜਾਂ ਐਨਕ੍ਰਿਪਸ਼ਨ ਵਰਗੇ ਢੁਕਵੇਂ ਸੁਰੱਖਿਆ ਉਪਾਅ ਲਾਗੂ ਕਰੋ।

6. ਨਗਿਰਾਨੀ ਅਤੇ ਚੇਤਾਵਨੀ ਟੂਲਜ਼ ਨਾਲ ਏਕੀਕਰਨ ਕਰੋ: ਨਗਿਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ ਡਾਟਾ ਦੇ ਇਕੱਤਰੀਕਰਨ, ਵਿਸ਼ਲੇਸ਼ਣ, ਅਤੇ ਵਿਜ਼ੁਅਲਾਈਜ਼ੇਸ਼ਨ ਨੂੰ ਕੇਂਦਰੀਕ੍ਰਤਿ ਕਰਨ ਲਈ ਨਗਿਰਾਨੀ ਅਤੇ ਚੇਤਾਵਨੀ ਟੂਲਜ਼ ਦਾ ਲਾਭ ਉਠਾਓ। ਇਹ ਟੂਲਜ਼ ਰੀਅਲ-ਟਾਈਮ ਅੰਤਰਦ੍ਰਸ਼ਿਟੀ ਪ੍ਰਦਾਨ ਕਰ ਸਕਦੇ ਹਨ, ਪਹਲਿਾਂ-ਨਰਿਧਾਰਤ ਸੀਮਾਵਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਚੇਤਾਵਨੀਆਂ ਤਆਿਰ ਕਰ ਸਕਦੇ ਹਨ, ਅਤੇ ਸਰਗਰਮ ਸਮੱਸਆਿ ਦੀ ਪਛਾਣ ਅਤੇ ਹੱਲ ਨੂੰ ਸੌਖਾ ਬਣਾ ਸਕਦੇ ਹਨ। ਇਹਨਾਂ ਟੂਲਜ਼ ਵੱਚਿੋਂ ਮੇਰਾ ਪਸੰਦੀਦਾ Datadog ਹੈ।

ਵਆਿਪਕ ਨਗਿਰਾਨੀ ਅਤੇ ਲੌਗਿੰਗ ਵਧਿੀਆਂ ਨੂੰ ਲਾਗੂ ਕਰਕੇ, ਡਵੈਿਲਪਰ ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋਜ਼ ਦੇ ਵਵਿਹਾਰ ਅਤੇ ਪ੍ਰਦਰਸ਼ਨ ਬਾਰੇ ਮੁੱਲਵਾਨ ਜਾਣਕਾਰੀ ਪ੍ਰਾਪਤ ਕਰ ਸਕਦੇ ਹਨ। ਇਹ ਅੰਤਰਦ੍ਰਸ਼ਿਟੀਆਂ AI-ਸੰਚਾਲਤਿ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ ਸਸਿਟਮਾਂ ਦੀ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਡੀਬੱਗਿੰਗ, ਆਪਟੀਮਾਈਜ਼ੇਸ਼ਨ, ਅਤੇ ਲਗਾਤਾਰ ਸੁਧਾਰ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਦੀਆਂ ਹਨ।

ਸਕੇਲੇਬਲਿਟੀ ਅਤੇ ਪ੍ਰਦਰਸ਼ਨ ਵਚਿਾਰ

ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ ਸਸਿਟਮਾਂ ਦੀ ਡਜ਼ਾਈਨਿੰਗ ਅਤੇ ਲਾਗੂਕਰਨ ਵੇਲੇ ਸਕੇਲੇਬਲਿਟੀ ਅਤੇ ਪ੍ਰਦਰਸ਼ਨ ਮਹੱਤਵਪੂਰਨ ਪਹਲੂ ਹਨ। ਜਵਿੇਂ-ਜਵਿੇਂ ਸਮਕਾਲੀ ਵਰਕਫਲੋਜ਼ ਦੀ ਮਾਤਰਾ ਅਤੇ AI-ਸੰਚਾਲਤਿ ਕੰਪੋਨੈਂਟਸ ਦੀ ਜਟਲਿਤਾ ਵਧਦੀ ਹੈ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਜ਼ਰੂਰੀ ਹੋ ਜਾਂਦਾ ਹੈ ਕ ਿਸਸਿਟਮ ਕੁਸ਼ਲਤਾ ਨਾਲ ਵਰਕਲੋਡ ਨੂੰ ਸੰਭਾਲ ਸਕੇ ਅਤੇ ਵਧਦੀਆਂ ਮੰਗਾਂ ਨੂੰ ਪੂਰਾ ਕਰਨ ਲਈ ਨਰਿਵਘਿਨ ਢੰਗ ਨਾਲ ਸਕੇਲ ਕਰ ਸਕੇ।

ਸਮਕਾਲੀ ਵਰਕਫਲੋਜ਼ ਦੀ ਉੱਚ ਮਾਤਰਾ ਨੂੰ ਸੰਭਾਲਣਾ

ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ ਸਸਿਟਮਾਂ ਨੂੰ ਅਕਸਰ ਸਮਕਾਲੀ ਵਰਕਫਲੋਜ਼ ਦੀ ਵੱਡੀ ਸੰਖਆਿ ਨੂੰ ਸੰਭਾਲਣ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਸਕੇਲੇਬਲਿਟੀ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ, ਹੇਠ ਲਖਿੀਆਂ ਰਣਨੀਤੀਆਂ 'ਤੇ ਵਚਿਾਰ ਕਰੋ:

1. ਅਸਂਕ੍ਰੋਨਸ ਪ੍ਰੋਸੈਸਿੰਗ: ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟਸ ਦੀ ਐਗਜ਼ੀਕਊਿਸ਼ਨ ਨੂੰ ਵੱਖ ਕਰਨ ਲਈ ਅਸਂਕ੍ਰੋਨਸ ਪ੍ਰੋਸੈਸਿੰਗ ਵਧਿੀਆਂ ਨੂੰ ਲਾਗੂ ਕਰੋ। ਇਹ ਸਸਿਟਮ ਨੂੰ ਹਰ ਕੰਪੋਨੈਂਟ ਦੇ ਪੂਰਾ ਹੋਣ ਦੀ ਉਡੀਕ ਕੀਤੇ ਬਨਿਾਂ ਜਾਂ ਬਲਾਕ ਕੀਤੇ ਬਨਿਾਂ ਕਈ ਵਰਕਫਲੋਜ਼ ਨੂੰ ਸਮਕਾਲੀ ਤੌਰ 'ਤੇ ਸੰਭਾਲਣ ਦੀ ਇਜਾਜ਼ਤ ਦਂਦਿਾ ਹੈ। ਅਸਂਕ੍ਰੋਨਸ ਪ੍ਰੋਸੈਸਿੰਗ ਮੈਸੇਜ ਕਊਿਜ਼, ਈਵੈਂਟ-ਡਰਾਈਵਨ ਆਰਕੀਟੈਕਚਰ, ਜਾਂ Sidekiq ਵਰਗੇ ਬੈਕਗਰਾਊਂਡ ਜੌਬ ਪ੍ਰੋਸੈਸਿੰਗ ਫਰੇਮਵਰਕਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪ੍ਰਾਪਤ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।

2. ਵੰਡੀ ਹੋਈ ਆਰਕੀਟੈਕਚਰ: ਸਸਿਟਮ ਆਰਕੀਟੈਕਚਰ ਨੂੰ ਸਰਵਰਲੈੱਸ ਕੰਪੋਨੈਂਟਸ (ਜਵਿੇਂ AWS Lambda) ਦੀ ਵਰਤੋਂ ਕਰਨ ਲਈ ਜਾਂ ਬਸ ਤੁਹਾਡੇ ਮੁੱਖ ਐਪਲੀਕੇਸ਼ਨ ਸਰਵਰ ਦੇ ਨਾਲ ਕਈ ਨੋਡਸ ਜਾਂ ਸਰਵਰਾਂ ਵੱਚਿ ਵਰਕਲੋਡ

ਨੂੰ ਵੰਡਣ ਲਈ ਡਿਜ਼ਾਈਨ ਕਰੋ। ਇਹ ਹੋਰੀਜ਼ੌਂਟਲ ਸਕੇਲੇਬਲਿਟੀ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ, ਜੱਥੇ ਵਧੇ ਹੋਏ ਵਰਕਫਲੋ ਵਾਲੀਊਮ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਵਾਧੂ ਨੋਡਸ ਜੋੜੇ ਜਾ ਸਕਦੇ ਹਨ।

3. ਸਮਾਨੰਤਰ ਐਗਜ਼ੀਕਊਸ਼ਨ: ਵਰਕਫਲੋਜ਼ ਦੇ ਅੰਦਰ ਸਮਾਨੰਤਰ ਐਗਜ਼ੀਕਊਸ਼ਨ ਦੇ ਮੌਕਿਆਂ ਦੀ ਪਛਾਣ ਕਰੋ। ਕੁਝ ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟਸ ਇੱਕ ਦੂਜੇ ਤੋਂ ਸੁਤੰਤਰ ਹੋ ਸਕਦੇ ਹਨ ਅਤੇ ਸਮਕਾਲੀ ਤੌਰ 'ਤੇ ਚਲਾਏ ਜਾ ਸਕਦੇ ਹਨ। ਮਲਟੀ-ਥਰੈਡਿੰਗ ਜਾਂ ਵੰਡੀਆਂ ਹੋਈਆਂ ਟਾਸਕ ਕਊਜ਼ ਵਰਗੀਆਂ ਸਮਾਨੰਤਰ ਪ੍ਰੋਸੈਸਿੰਗ ਤਕਨੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਸਿਸਟਮ ਸਰੋਤਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦਾ ਹੈ ਅਤੇ ਸਮੁੱਚੇ ਵਰਕਫਲੋ ਐਗਜ਼ੀਕਊਸ਼ਨ ਸਮੇਂ ਨੂੰ ਘਟਾ ਸਕਦਾ ਹੈ।

AI-ਸੰਚਾਲਿਤ ਕੰਪੋਨੈਂਟਸ ਦੇ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣਾ

ਏਆਈ-ਸੰਚਾਲਿਤ ਕੰਪੋਨੈਂਟਸ, ਜਿਵੇਂ ਕਿ ਮਸ਼ੀਨ ਲਰਨਿੰਗ ਮਾਡਲ ਜਾਂ ਨੈਚੁਰਲ ਲੈਂਗੁਏਜ ਪ੍ਰੋਸੈਸਿੰਗ ਇੰਜਣ, ਕੰਪਊਟੇਸ਼ਨਲ ਤੌਰ 'ਤੇ ਗੰਭੀਰ ਹੋ ਸਕਦੇ ਹਨ ਅਤੇ ਵਰਕਫਲੋ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਸਿਸਟਮ ਦੇ ਸਮੁੱਚੇ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਪ੍ਰਭਾਵਿਤ ਕਰ ਸਕਦੇ ਹਨ। ਏਆਈ ਕੰਪੋਨੈਂਟਸ ਦੇ ਪ੍ਰਦਰਸ਼ਨ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣ ਲਈ, ਹੇਠ ਲਿਖੀਆਂ ਤਕਨੀਕਾਂ 'ਤੇ ਵਿਚਾਰ ਕਰੋ:

1. ਕੈਸ਼ਿੰਗ: ਜੇਕਰ ਤੁਹਾਡੀ ਏਆਈ ਪ੍ਰੋਸੈਸਿੰਗ ਪੂਰੀ ਤਰ੍ਹਾਂ ਜਨਰੇਟਿਵ ਹੈ ਅਤੇ ਇਸ ਵਿੱਚ ਰੀਅਲਟਾਈਮ ਜਾਣਕਾਰੀ ਲੱਭਣ ਜਾਂ ਚੈਟ ਕੰਪਲੀਸ਼ਨਜ਼ ਨੂੰ ਜਨਰੇਟ ਕਰਨ ਲਈ ਬਾਹਰੀ ਏਕੀਕਰਣ ਸ਼ਾਮਲ ਨਹੀਂ ਹਨ, ਤਾਂ ਤੁਸੀਂ ਅਕਸਰ ਵਰਤੀਆਂ ਜਾਂਦੀਆਂ ਜਾਂ ਕੰਪਊਟੇਸ਼ਨਲ ਤੌਰ 'ਤੇ ਮਹਿੰਗੀਆਂ ਕਾਰਵਾਈਆਂ ਦੇ ਨਤੀਜਿਆਂ ਨੂੰ ਸਟੋਰ ਕਰਨ ਅਤੇ ਮੁੜ ਵਰਤਣ ਲਈ ਕੈਸ਼ਿੰਗ ਮੈਕੇਨਜ਼ਿਮ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹੋ।

2. ਮਾਡਲ ਅਨੁਕੂਲਣ: ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟਸ ਵਿੱਚ ਏਆਈ ਮਾਡਲਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਲਗਾਤਾਰ ਅਨੁਕੂਲ ਬਣਾਓ। ਇਸ ਵਿੱਚ ਪ੍ਰੌਮਪਟ ਡਿਸਟੀਲੇਸ਼ਨ ਵਰਗੀਆਂ ਤਕਨੀਕਾਂ ਸ਼ਾਮਲ ਹੋ ਸਕਦੀਆਂ ਹਨ ਜਾਂ ਇਹ ਸਿਰਫ਼ ਨਵੇਂ ਮਾਡਲਾਂ ਦੀ ਜਾਂਚ ਕਰਨ ਦਾ ਮਾਮਲਾ ਹੋ ਸਕਦਾ ਹੈ ਜਦੋਂ ਉਹ ਉਪਲਬਧ ਹੁੰਦੇ ਹਨ।

3. ਬੈਚ ਪ੍ਰੋਸੈਸਿੰਗ: ਜੇਕਰ ਤੁਸੀਂ GPT-4 ਕਲਾਸ ਮਾਡਲਾਂ ਨਾਲ ਕੰਮ ਕਰ ਰਹੇ ਹੋ, ਤਾਂ ਤੁਸੀਂ ਕਈ ਡਾਟਾ ਪੁਆਇੰਟਸ ਜਾਂ ਬੇਨਤੀਆਂ ਨੂੰ ਵਅਿਕਤੀਗਤ ਤੌਰ 'ਤੇ ਪ੍ਰੋਸੈਸ ਕਰਨ ਦੀ ਬਜਾਏ ਇੱਕ ਬੈਚ ਵਿੱਚ ਪ੍ਰੋਸੈਸ ਕਰਨ ਲਈ ਬੈਚ ਪ੍ਰੋਸੈਸਿੰਗ ਤਕਨੀਕਾਂ ਦਾ ਲਾਭ ਲੈ ਸਕਦੇ ਹੋ। ਡਾਟਾ ਨੂੰ ਬੈਚਾਂ ਵਿੱਚ ਪ੍ਰੋਸੈਸ ਕਰਕੇ, ਸਿਸਟਮ ਸਰੋਤਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦਾ ਹੈ ਅਤੇ ਦੁਹਰਾਏ ਮਾਡਲ ਬੇਨਤੀਆਂ ਦੇ ਓਵਰਹੈੱਡ ਨੂੰ ਘਟਾ ਸਕਦਾ ਹੈ।

ਨਗਿਰਾਨੀ ਅਤੇ ਪ੍ਰੋਫਾਈਲਿੰਗ ਪ੍ਰਦਰਸ਼ਨ

ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਸਿਸਟਮ ਦੇ ਪ੍ਰਦਰਸ਼ਨ ਦੇ ਬੋਤਲਨੈੱਕਸ ਦੀ ਪਛਾਣ ਕਰਨ ਅਤੇ ਸਕੇਲੇਬਲਿਟੀ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਉਣ ਲਈ, ਨਗਿਰਾਨੀ ਅਤੇ ਪ੍ਰੋਫਾਈਲਿੰਗ ਮੈਕੇਨਜ਼ਿਮ ਲਾਗੂ ਕਰਨਾ ਜ਼ਰੂਰੀ ਹੈ। ਹੇਠ ਲਿਖੇ ਪਹੁੰਚਾਂ 'ਤੇ ਵਿਚਾਰ ਕਰੋ:

1. ਪ੍ਰਦਰਸ਼ਨ ਮੈਟ੍ਰਿਕਸ: ਮੁੱਖ ਪ੍ਰਦਰਸ਼ਨ ਮੈਟ੍ਰਿਕਸ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਅਤੇ ਟਰੈਕ ਕਰੋ, ਜਿਵੇਂ ਕਿ ਜਵਾਬ ਸਮਾਂ, ਥਰੂਪੁੱਟ, ਸਰੋਤ ਵਰਤੋਂ, ਅਤੇ ਲੇਟੈਂਸੀ। ਇਹ ਮੈਟ੍ਰਿਕਸ ਸਿਸਟਮ ਦੇ ਪ੍ਰਦਰਸ਼ਨ ਬਾਰੇ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ ਅਤੇ ਅਨੁਕੂਲਣ ਲਈ ਖੇਤਰਾਂ ਦੀ ਪਛਾਣ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰਦੇ ਹਨ। ਪ੍ਰਸਿੱਧ ਏਆਈ ਮਾਡਲ ਐਗਰੀਗੇਟਰ OpenRouter ਹਰੇਕ API ਜਵਾਬ ਵਿੱਚ ਹੋਸਟ[1] ਅਤੇ ਸਪੀਡ[2] ਮੈਟ੍ਰਿਕਸ ਸ਼ਾਮਲ ਕਰਦਾ ਹੈ, ਜੋ ਇਨ੍ਹਾਂ ਮੁੱਖ ਮੈਟ੍ਰਿਕਸ ਨੂੰ ਟਰੈਕ ਕਰਨਾ ਬਹੁਤ ਸੌਖਾ ਬਣਾਉਂਦਾ ਹੈ।

2. ਪ੍ਰੋਫਾਈਲਿੰਗ ਟੂਲਜ਼: ਵਿਅਕਤੀਗਤ ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟਸ ਅਤੇ ਏਆਈ ਕਾਰਵਾਈਆਂ ਦੇ ਪ੍ਰਦਰਸ਼ਨ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਨ ਲਈ ਪ੍ਰੋਫਾਈਲਿੰਗ ਟੂਲਜ਼ ਦੀ ਵਰਤੋਂ ਕਰੋ। ਪ੍ਰੋਫਾਈਲਿੰਗ ਟੂਲਜ਼ ਪ੍ਰਦਰਸ਼ਨ ਹੌਟਸਪੌਟਸ, ਅਕੁਸ਼ਲ ਕੋਡ ਪਾਥਸ, ਜਾਂ ਸਰੋਤ-ਗਹਿਨ ਕਾਰਵਾਈਆਂ ਦੀ ਪਛਾਣ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰ ਸਕਦੇ ਹਨ। ਪ੍ਰਸਿੱਧ ਪ੍ਰੋਫਾਈਲਿੰਗ ਟੂਲਜ਼ ਵਿੱਚ New Relic, Scout, ਜਾਂ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਜਾਂ ਫਰੇਮਵਰਕ ਦੁਆਰਾ ਪ੍ਰਦਾਨ ਕੀਤੇ ਬਿਲਟ-ਇਨ ਪ੍ਰੋਫਾਈਲਰ ਸ਼ਾਮਲ ਹਨ।

3. ਲੋਡ ਟੈਸਟਿੰਗ: ਵੱਖ-ਵੱਖ ਪੱਧਰਾਂ ਦੇ ਸਮਕਾਲੀ ਵਰਕਲੋਡ ਅਧੀਨ ਸਿਸਟਮ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਦਾ ਮੁਲਾਂਕਣ ਕਰਨ ਲਈ ਲੋਡ ਟੈਸਟਿੰਗ ਕਰੋ। ਲੋਡ ਟੈਸਟਿੰਗ ਸਿਸਟਮ ਦੀਆਂ ਸਕੇਲੇਬਲਿਟੀ ਸੀਮਾਵਾਂ ਦੀ ਪਛਾਣ ਕਰਨ, ਕਾਰਗੁਜ਼ਾਰੀ ਵਿੱਚ ਕਮੀ ਦਾ ਪਤਾ ਲਗਾਉਣ, ਅਤੇ ਇਹ ਯਕੀਨੀ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਦੀ ਹੈ ਕਿ ਸਿਸਟਮ ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਪ੍ਰਭਾਵਿਤ ਕੀਤੇ ਬਿਨਾਂ ਉਮੀਦ ਕੀਤੀ ਟ੍ਰੈਫਿਕ ਨੂੰ ਸੰਭਾਲ ਸਕਦਾ ਹੈ।

4. ਲਗਾਤਾਰ ਨਗਿਰਾਨੀ: ਕਾਰਗੁਜ਼ਾਰੀ ਦੀਆਂ ਸਮੱਸਿਆਵਾਂ ਅਤੇ ਬੋਤਲਨੈੱਕਾਂ ਦਾ ਸਰਗਰਮੀ ਨਾਲ ਪਤਾ ਲਗਾਉਣ ਲਈ ਲਗਾਤਾਰ ਨਗਿਰਾਨੀ ਅਤੇ ਚੇਤਾਵਨੀ ਵਿਧੀਆਂ ਨੂੰ ਲਾਗੂ ਕਰੋ। ਮੁੱਖ ਕਾਰਗੁਜ਼ਾਰੀ ਸੂਚਕਾਂ (KPIs) ਦੀ ਨਗਿਰਾਨੀ ਕਰਨ ਅਤੇ ਪਹਿਲਾਂ-ਨਿਰਧਾਰਤ ਸੀਮਾਵਾਂ ਦੀ ਉਲੰਘਣਾ ਹੋਣ 'ਤੇ ਸੂਚਨਾਵਾਂ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਨਗਿਰਾਨੀ ਡੈਸ਼ਬੋਰਡ ਅਤੇ ਚੇਤਾਵਨੀਆਂ ਸਥਾਪਿਤ ਕਰੋ। ਇਹ ਕਾਰਗੁਜ਼ਾਰੀ ਸਮੱਸਿਆਵਾਂ ਦੀ ਤੁਰੰਤ ਪਛਾਣ ਅਤੇ ਹੱਲ ਨੂੰ ਸਮਰੱਥ ਬਣਾਉਂਦਾ ਹੈ।

---

[1] ਹੋਸਟ ਉਹ ਸਮਾਂ ਹੈ ਜੋ ਮਾਡਲ ਹੋਸਟ ਤੋਂ ਸਟ੍ਰੀਮ ਕੀਤੀ ਜਨਰੇਸ਼ਨ ਦਾ ਪਹਿਲਾ ਬਾਈਟ ਪ੍ਰਾਪਤ ਕਰਨ ਵਿੱਚ ਲੱਗਿਆ, ਯਾਨੀ "ਪਹਿਲੇ ਬਾਈਟ ਤੱਕ ਦਾ ਸਮਾਂ।"

[2] ਸਪੀਡ ਦੀ ਗਣਨਾ ਕੰਪਲੀਸ਼ਨ ਟੋਕਨਾਂ ਦੀ ਸੰਖਿਆ ਨੂੰ ਕੁੱਲ ਜਨਰੇਸ਼ਨ ਸਮੇਂ ਨਾਲ ਭਾਗ ਕਰਕੇ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਨਾਨ-ਸਟ੍ਰੀਮਡ ਬੇਨਤੀਆਂ ਲਈ ਲੇਟੈਂਸੀ ਨੂੰ ਜਨਰੇਸ਼ਨ ਸਮੇਂ ਦਾ ਹਿੱਸਾ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ।

ਸਕੇਲਿੰਗ ਰਣਨੀਤੀਆਂ

ਵਧਦੇ ਵਰਕਲੋਡ ਨੂੰ ਸੰਭਾਲਣ ਅਤੇ ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ ਸਿਸਟਮ ਦੀ ਸਕੇਲੇਬਲਿਟੀ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ, ਹੇਠ ਲਿਖੀਆਂ ਸਕੇਲਿੰਗ ਰਣਨੀਤੀਆਂ 'ਤੇ ਵਿਚਾਰ ਕਰੋ:

1. ਵਰਟੀਕਲ ਸਕੇਲਿੰਗ: ਵਰਟੀਕਲ ਸਕੇਲਿੰਗ ਵਿੱਚ ਵੱਧ ਵਰਕਲੋਡ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਵਿਅਕਤੀਗਤ ਨੋਡਾਂ ਜਾਂ ਸਰਵਰਾਂ ਦੇ ਸਰੋਤਾਂ (ਜਿਵੇਂ CPU, ਮੈਮੋਰੀ) ਨੂੰ ਵਧਾਉਣਾ ਸ਼ਾਮਲ ਹੈ। ਇਹ ਪਹੁੰਚ ਉਦੋਂ ਢੁਕਵੀਂ ਹੁੰਦੀ ਹੈ ਜਦੋਂ ਸਿਸਟਮ ਨੂੰ ਗੁੰਝਲਦਾਰ ਵਰਕਫਲੋ ਜਾਂ AI ਓਪਰੇਸ਼ਨਾਂ ਨੂੰ ਸੰਭਾਲਣ ਲਈ ਵਧੇਰੇ ਪ੍ਰੋਸੈਸਿੰਗ ਪਾਵਰ ਜਾਂ ਮੈਮੋਰੀ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

2. ਹੋਰੀਜ਼ੌਨਟਲ ਸਕੇਲਿੰਗ: ਹੋਰੀਜ਼ੌਨਟਲ ਸਕੇਲਿੰਗ ਵਿੱਚ ਵਰਕਲੋਡ ਨੂੰ ਵੰਡਣ ਲਈ ਸਿਸਟਮ ਵਿੱਚ ਹੋਰ ਨੋਡਾਂ ਜਾਂ ਸਰਵਰਾਂ ਨੂੰ ਜੋੜਨਾ ਸ਼ਾਮਲ ਹੈ। ਇਹ ਪਹੁੰਚ ਉਦੋਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਹੁੰਦੀ ਹੈ ਜਦੋਂ ਸਿਸਟਮ ਨੂੰ ਬਹੁਤ ਸਾਰੇ ਸਮਕਾਲੀ ਵਰਕਫਲੋ ਨੂੰ ਸੰਭਾਲਣ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ ਜਾਂ ਜਦੋਂ ਵਰਕਲੋਡ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਕਈ ਨੋਡਾਂ ਵਿੱਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਹੋਰੀਜ਼ੌਨਟਲ ਸਕੇਲਿੰਗ ਲਈ ਟ੍ਰੈਫਿਕ ਦੀ ਸਮਾਨ ਵੰਡ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਇੱਕ ਵੰਡੀ ਹੋਈ ਆਰਕੀਟੈਕਚਰ ਅਤੇ ਲੋਡ ਬੈਲੇਂਸਿੰਗ ਵਿਧੀਆਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ।

3. ਆਟੋ-ਸਕੇਲਿੰਗ: ਵਰਕਲੋਡ ਦੀ ਮੰਗ ਦੇ ਆਧਾਰ 'ਤੇ ਨੋਡਾਂ ਜਾਂ ਸਰੋਤਾਂ ਦੀ ਸੰਖਿਆ ਨੂੰ ਆਪਣੇ ਆਪ ਵਿਵਿਸਥਤਿ ਕਰਨ ਲਈ ਆਟੋ-ਸਕੇਲਿੰਗ ਵਿਧੀਆਂ ਨੂੰ ਲਾਗੂ ਕਰੋ। ਆਟੋ-ਸਕੇਲਿੰਗ ਸਿਸਟਮ ਨੂੰ ਆਉਣ ਵਾਲੀ ਟ੍ਰੈਫਿਕ ਦੇ ਆਧਾਰ 'ਤੇ ਗਤੀਸ਼ੀਲ ਢੰਗ ਨਾਲ ਵੱਧਣ ਜਾਂ ਘਟਣ ਦੀ ਆਗਿਆ ਦੇਂਦੀ ਹੈ, ਜੋ ਸਰੋਤਾਂ ਦੀ ਇਸ਼ਟਤਮ ਵਰਤੋਂ ਅਤੇ ਲਾਗਤ-ਕੁਸ਼ਲਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਂਦੀ ਹੈ। Amazon Web Services (AWS) ਜਾਂ Google Cloud Platform (GCP) ਵਰਗੇ ਕਲਾਉਡ ਪਲੇਟਫਾਰਮ ਆਟੋ-ਸਕੇਲਿੰਗ ਸਮਰੱਥਾਵਾਂ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ ਜਿਨ੍ਹਾਂ ਨੂੰ ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ ਸਿਸਟਮਾਂ ਲਈ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।

ਕਾਰਗੁਜ਼ਾਰੀ ਅਨੁਕੂਲਨ ਤਕਨੀਕਾਂ

ਸਕੇਲਿੰਗ ਰਣਨੀਤੀਆਂ ਦੇ ਨਾਲ-ਨਾਲ, ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ ਸਿਸਟਮ ਦੀ ਕੁਸ਼ਲਤਾ ਨੂੰ ਵਧਾਉਣ ਲਈ ਹੇਠ ਲਿਖੀਆਂ ਕਾਰਗੁਜ਼ਾਰੀ ਅਨੁਕੂਲਨ ਤਕਨੀਕਾਂ 'ਤੇ ਵਿਚਾਰ ਕਰੋ:

1. ਕੁਸ਼ਲ ਡਾਟਾ ਸਟੋਰੇਜ ਅਤੇ ਪੁਨਰਪ੍ਰਾਪਤੀ: ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟਾਂ ਦੁਆਰਾ ਵਰਤੀਆਂ ਜਾਂਦੀਆਂ ਡਾਟਾ ਸਟੋਰੇਜ ਅਤੇ ਪੁਨਰਪ੍ਰਾਪਤੀ ਵਿਧੀਆਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਓ। ਡਾਟਾ-ਗਹਨਿ ਓਪਰੇਸ਼ਨਾਂ ਦੀ ਦੇਰੀ ਨੂੰ ਘੱਟ ਕਰਨ ਅਤੇ ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਬਹਿਤਰ ਬਣਾਉਣ ਲਈ ਕੁਸ਼ਲ ਡਾਟਾਬੇਸ ਇੰਡੈਕਸਿੰਗ, ਕਵੇਰੀ ਅਨੁਕੂਲਨ ਤਕਨੀਕਾਂ, ਅਤੇ ਡਾਟਾ ਕੈਸ਼ਿੰਗ ਦੀ ਵਰਤੋਂ ਕਰੋ।

2. ਅਸਿੰਕ੍ਰੋਨਸ I/O: ਬਲਾਕਿੰਗ ਨੂੰ ਰੋਕਣ ਅਤੇ ਸਿਸਟਮ ਦੀ ਪ੍ਰਤਿਕਿਰਿਆਸ਼ੀਲਤਾ ਨੂੰ ਬਹਿਤਰ ਬਣਾਉਣ ਲਈ ਅਸਿੰਕ੍ਰੋਨਸ I/O ਓਪਰੇਸ਼ਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰੋ। ਅਸਿੰਕ੍ਰੋਨਸ I/O ਸਿਸਟਮ ਨੂੰ I/O ਓਪਰੇਸ਼ਨਾਂ ਦੇ ਪੂਰਾ ਹੋਣ ਦੀ ਉਡੀਕ ਕੀਤੇ ਬਿਨਾਂ ਕਈ ਬੇਨਤੀਆਂ ਨੂੰ ਸਮਕਾਲੀ ਤੌਰ 'ਤੇ ਸੰਭਾਲਣ ਦੀ ਆਗਿਆ ਦੇਂਦਾ ਹੈ, ਇਸ ਤਰ੍ਹਾਂ ਸਰੋਤਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਵੱਧ ਤੋਂ ਵੱਧ ਕਰਦਾ ਹੈ।

3. ਕੁਸ਼ਲ ਲੜੀਬੱਧਤਾ ਅਤੇ ਵਿ-ਲੜੀਬੱਧਤਾ: ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਵਚਿਕਾਰ ਡਾਟਾ ਵਟਾਂਦਰੇ ਲਈ ਵਰਤੀਆਂ ਜਾਂਦੀਆਂ ਲੜੀਬੱਧਤਾ ਅਤੇ ਵਿ-ਲੜੀਬੱਧਤਾ ਪ੍ਰਕਰਿਆਵਾਂ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾਓ। ਕੁਸ਼ਲ ਲੜੀਬੱਧਤਾ ਫਾਰਮੈਟਾਂ ਦੀ ਵਰਤੋਂ ਕਰੋ, ਜਵਿੇਂ ਕਿ Protocol Buffers ਜਾਂ MessagePack, ਡਾਟਾ ਲੜੀਬੱਧਤਾ ਦੇ ਓਵਰਹੈੱਡ ਨੂੰ ਘਟਾਉਣ ਅਤੇ ਅੰਤਰ-ਭਾਗ ਸੰਚਾਰ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਬਹਿਤਰ ਬਣਾਉਣ ਲਈ।

> Ruby-ਆਧਾਰਤਿ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ, Universal ID ਦੀ ਵਰਤੋਂ 'ਤੇ ਵਚਿਾਰ ਕਰੋ। Universal ID MessagePack ਅਤੇ Brotli ਦੋਵਾਂ ਦਾ ਲਾਭ ਲੈਂਦਾ ਹੈ (ਇੱਕ ਕੰਬੋ ਜੋ ਗਤੀ ਅਤੇ ਸਰਵੋਤਮ-ਸ਼੍ਰੇਣੀ ਡਾਟਾ ਸੰਕੁਚਨ ਲਈ ਬਣਾਇਆ ਗਿਆ ਹੈ)। ਜਦੋਂ ਇਕੱਠੇ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਇਹ ਲਾਇਬ੍ਰੇਰੀਆਂ Protocol Buffers ਦੇ ਮੁਕਾਬਲੇ 30% ਤੱਕ ਤੇਜ਼ ਹਨ ਅਤੇ 2-5% ਸੰਕੁਚਨ ਦਰਾਂ ਦੇ ਅੰਦਰ ਹਨ।

4. ਸੰਕੁਚਨ ਅਤੇ ਇਨਕੋਡਿੰਗ: ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਵਚਿਕਾਰ ਟ੍ਰਾਂਸਫਰ ਕੀਤੇ ਡਾਟਾ ਦੇ ਆਕਾਰ ਨੂੰ ਘਟਾਉਣ ਲਈ ਸੰਕੁਚਨ ਅਤੇ ਇਨਕੋਡਿੰਗ ਤਕਨੀਕਾਂ ਲਾਗੂ ਕਰੋ। ਸੰਕੁਚਨ ਐਲਗੋਰਦਿਮ, ਜਵਿੇਂ ਕਿ gzip ਜਾਂ Brotli, ਨੈੱਟਵਰਕ ਬੈਂਡਵਡਿਥ ਦੀ ਵਰਤੋਂ ਨੂੰ ਕਾਫ਼ੀ ਘਟਾ ਸਕਦੇ ਹਨ ਅਤੇ ਸਿਸਟਮ ਦੀ ਸਮੁੱਚੀ ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਬਹਿਤਰ ਬਣਾ ਸਕਦੇ ਹਨ।

ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਸਿਸਟਮਾਂ ਦੇ ਡਜ਼ਿਾਈਨ ਅਤੇ ਕਾਰਜਾਂਵਤਿ ਦੌਰਾਨ ਸਕੇਲੇਬਲਿਟੀ ਅਤੇ ਕਾਰਗੁਜ਼ਾਰੀ ਪਹਲ਼ੂਆਂ 'ਤੇ ਵਚਿਾਰ ਕਰਕੇ, ਤੁਸੀਂ ਯਕੀਨੀ ਬਣਾ ਸਕਦੇ ਹੋ ਕਿ ਤੁਹਾਡਾ ਸਿਸਟਮ ਸਮਕਾਲੀ ਕਾਰਜ-ਪ੍ਰਵਾਹਾਂ ਦੀ ਉੱਚ ਮਾਤਰਾ ਨੂੰ ਸੰਭਾਲ ਸਕਦਾ ਹੈ, AI-ਸੰਚਾਲਤਿ ਭਾਗਾਂ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਨੂੰ ਅਨੁਕੂਲ ਬਣਾ ਸਕਦਾ ਹੈ, ਅਤੇ ਵਧਦੀਆਂ ਮੰਗਾਂ ਨੂੰ ਪੂਰਾ ਕਰਨ ਲਈ ਨਰਿਵਘਿਨ ਢੰਗ ਨਾਲ ਸਕੇਲ ਕਰ ਸਕਦਾ ਹੈ। ਲਗਾਤਾਰ ਨਗਿਰਾਨੀ, ਪ੍ਰੋਫਾਈਲਿੰਗ, ਅਤੇ ਅਨੁਕੂਲਨ ਯਤਨ ਸਿਸਟਮ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਅਤੇ ਪ੍ਰਤਿਕਿਰਿਆਸ਼ੀਲਤਾ ਨੂੰ ਬਣਾਈ ਰੱਖਣ ਲਈ ਜ਼ਰੂਰੀ ਹਨ ਜਵਿੇਂ-ਜਵਿੇਂ ਕੰਮ ਦਾ ਭਾਰ ਅਤੇ ਗੁੰਝਲਤਾ ਸਮੇਂ ਦੇ ਨਾਲ ਵਧਦੀ ਹੈ।

ਕਾਰਜ-ਪ੍ਰਵਾਹਾਂ ਦੀ ਟੈਸਟਿੰਗ ਅਤੇ ਪ੍ਰਮਾਣੀਕਰਨ

ਟੈਸਟਿੰਗ ਅਤੇ ਪ੍ਰਮਾਣੀਕਰਨ ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ ਸਿਸਟਮਾਂ ਦੇ ਵਿਕਾਸ ਅਤੇ ਰੱਖ-ਰਖਾਅ ਦੇ ਮਹੱਤਵਪੂਰਨ ਪਹਲੂ ਹਨ। AI-ਸੰਚਾਲਤਿ ਕਾਰਜ-ਪ੍ਰਵਾਹਾਂ ਦੀ ਗੁੰਝਲ ਪ੍ਰਕਰਿਤੀ ਨੂੰ ਦੇਖਦੇ ਹੋਏ, ਇਹ ਯਕੀਨੀ ਬਣਾਉਣਾ ਜ਼ਰੂਰੀ ਹੈ ਕਿ ਹਰ ਭਾਗ ਉਮੀਦ ਮੁਤਾਬਕ ਕੰਮ ਕਰਦਾ ਹੈ, ਸਮੁੱਚਾ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਸਹੀ ਢੰਗ ਨਾਲ ਵਵਿਹਾਰ ਕਰਦਾ ਹੈ, ਅਤੇ AI ਫੈਸਲੇ ਸਹੀ ਅਤੇ ਭਰੋਸੇਯੋਗ ਹਨ। ਇਸ ਭਾਗ ਵੱਚਿ, ਅਸੀ ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹਾਂ ਦੀ ਟੈਸਟਿੰਗ ਅਤੇ ਪ੍ਰਮਾਣੀਕਰਨ ਲਈ ਵੱਖ-ਵੱਖ ਤਕਨੀਕਾਂ ਅਤੇ ਵਚਿਾਰਾਂ ਦੀ ਪੜਚੋਲ ਕਰਾਂਗੇ।

ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਦੀ ਇਕਾਈ ਟੈਸਟਿੰਗ

ਇਕਾਈ ਟੈਸਟਿੰਗ ਵੱਚਿ ਉਨ੍ਹਾਂ ਦੀ ਸ਼ੁੱਧਤਾ ਅਤੇ ਮਜ਼ਬੂਤੀ ਦੀ ਜਾਂਚ ਕਰਨ ਲਈ ਵੱਖਰੇ ਤੌਰ 'ਤੇ ਵਅਿਕਤੀਗਤ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਦੀ ਟੈਸਟਿੰਗ ਸ਼ਾਮਲ ਹੈ। AI-ਸੰਚਾਲਤਿ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਦੀ ਇਕਾਈ ਟੈਸਟਿੰਗ ਕਰਦੇ ਸਮੇਂ, ਹੇਠ ਲਖਿਆਂ 'ਤੇ ਵਚਿਾਰ ਕਰੋ:

1. ਇਨਪੁੱਟ ਪ੍ਰਮਾਣੀਕਰਨ: ਵੱਖ-ਵੱਖ ਕਸਿਮਾਂ ਦੇ ਇਨਪੁੱਟ, ਸਮੇਤ ਵੈਧ ਅਤੇ ਅਵੈਧ ਡਾਟਾ ਨੂੰ ਸੰਭਾਲਣ ਦੀ ਭਾਗ ਦੀ ਯੋਗਤਾ ਦੀ ਜਾਂਚ ਕਰੋ। ਪੁਸ਼ਟੀ ਕਰੋ ਕਿ ਭਾਗ ਸੀਮਾ ਮਾਮਲਆਿਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲਦਾ ਹੈ ਅਤੇ ਢੁਕਵੇਂ ਗਲਤੀ ਸੰਦੇਸ਼ ਜਾਂ ਅਪਵਾਦ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।

2. ਆਊਟਪੁੱਟ ਤਸਦੀਕ: ਪੁਸ਼ਟੀ ਕਰੋ ਕਿ ਭਾਗ ਦੱਤਿ ਇਨਪੁੱਟ ਸੈੱਟ ਲਈ ਉਮੀਦ ਕੀਤੇ ਆਊਟਪੁੱਟ ਪੈਦਾ ਕਰਦਾ ਹੈ। ਸ਼ੁੱਧਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਅਸਲ ਆਊਟਪੁੱਟ ਦੀ ਉਮੀਦ ਕੀਤੇ ਨਤੀਜਆਿਂ ਨਾਲ ਤੁਲਨਾ ਕਰੋ।

3. ਗਲਤੀ ਸੰਭਾਲ: ਵੱਖ-ਵੱਖ ਗਲਤੀ ਦੀਆਂ ਸਥਤਿੀਆਂ ਦੀ ਨਕਲ ਕਰਕੇ ਕੰਪੋਨੈਂਟ ਦੇ ਗਲਤੀ ਸੰਭਾਲ ਤੰਤਰਾਂ ਦੀ ਜਾਂਚ ਕਰੋ, ਜਵਿੇਂ ਕਿ ਅਵੈਧ ਇਨਪੁੱਟ, ਸਰੋਤ ਦੀ ਅਣਉਪਲਬਧਤਾ, ਜਾਂ ਅਣਚਾਹੇ ਅਪਵਾਦ। ਪੁਸ਼ਟੀ ਕਰੋ ਕਿ ਕੰਪੋਨੈਂਟ ਗਲਤੀਆਂ ਨੂੰ ਢੁਕਵੇਂ ਢੰਗ ਨਾਲ ਫੜਦਾ ਅਤੇ ਸੰਭਾਲਦਾ ਹੈ।

4. ਸੀਮਾ ਸਥਤিੀਆਂ: ਸੀਮਾ ਸਥਤিੀਆਂ ਦੇ ਅਧੀਨ ਕੰਪੋਨੈਂਟ ਦੇ ਵਵਿਹਾਰ ਦੀ ਜਾਂਚ ਕਰੋ, ਜਵਿੇਂ ਕਿ ਖਾਲੀ ਇਨਪੁੱਟ, ਵੱਧ ਤੋਂ ਵੱਧ ਇਨਪੁੱਟ ਆਕਾਰ, ਜਾਂ ਅਤਅਿੰਤ ਮੁੱਲ। ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਕੰਪੋਨੈਂਤ ਇਨ੍ਹਾਂ ਸਥਤিੀਆਂ ਨੂੰ ਕ੍ਰੈਸ਼ ਹੋਏ ਬਨਿਾਂ ਜਾਂ ਗਲਤ ਨਤੀਜੇ ਪੈਦਾ ਕੀਤੇ ਬਨਿਾਂ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲਦਾ ਹੈ।

ਇੱਥੇ Ruby ਵੱਚਿ RSpec ਟੈਸਟਿੰਗ ਫਰੇਮਵਰਕ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇੱਕ ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟ ਲਈ ਯੂਨਟਿ ਟੈਸਟ ਦੀ ਉਦਾਹਰਣ ਹੈ:

```
RSpec.describe OrderValidator do
  describe '#validate' do
    context 'when order is valid' do
      let(:order) { build(:order) }

      it 'returns true' do
        expect(subject.validate(order)).to be true
      end
    end

    context 'when order is invalid' do
      let(:order) { build(:order, total_amount: -100) }

      it 'returns false' do
        expect(subject.validate(order)).to be false
      end
    end
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ, `OrderValidator` ਭਾਗ ਦੀ ਜਾਂਚ ਦੋ ਟੈਸਟ ਕੇਸਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਕੀਤੀ ਜਾਂਦੀ ਹੈ: ਇੱਕ ਵੈਧ ਆਰਡਰ ਲਈ ਅਤੇ ਦੂਜਾ ਅਵੈਧ ਆਰਡਰ ਲਈ। ਟੈਸਟ ਕੇਸ ਇਹ ਪੁਸ਼ਟੀ ਕਰਦੇ ਹਨ ਕਿ `validate` ਮੈਥਡ ਆਰਡਰ ਦੀ ਵੈਧਤਾ ਦੇ ਆਧਾਰ 'ਤੇ ਉਮੀਦ ਕੀਤਾ ਬੂਲੀਅਨ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ।

ਏਕੀਕਰਣ ਟੈਸਟਿੰਗ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਅੰਤਰਕਰਿਆਵਾਂ

ਏਕੀਕਰਣ ਟੈਸਟਿੰਗ ਵੱਖ-ਵੱਖ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਵਿਚਕਾਰ ਅੰਤਰਕਰਿਆਵਾਂ ਅਤੇ ਡਾਟਾ ਪ੍ਰਵਾਹ ਦੀ ਜਾਂਚ 'ਤੇ ਕੇਂਦਰਿਤ ਹੈ। ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦੀ ਹੈ ਕਿ ਭਾਗ ਇਕੱਠੇ ਨਿਰਵਘਿਨ ਕੰਮ ਕਰਦੇ ਹਨ ਅਤੇ ਉਮੀਦ ਕੀਤੇ ਨਤੀਜੇ ਪੈਦਾ ਕਰਦੇ ਹਨ। ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹਾਂ ਦੀ ਏਕੀਕਰਣ ਟੈਸਟਿੰਗ ਕਰਦੇ ਸਮੇਂ, ਹੇਠ ਲਿਖੀਆਂ ਗੱਲਾਂ 'ਤੇ ਵਿਚਾਰ ਕਰੋ:

1. **ਭਾਗ ਅੰਤਰਕਰਿਆ:** ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਵਿਚਕਾਰ ਸੰਚਾਰ ਅਤੇ ਡਾਟਾ ਵਟਾਂਦਰੇ ਦੀ ਜਾਂਚ ਕਰੋ। ਪੁਸ਼ਟੀ ਕਰੋ ਕਿ ਇੱਕ ਭਾਗ ਦਾ ਆਉਟਪੁੱਟ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਵਿੱਚ ਅਗਲੇ ਭਾਗ ਲਈ ਇਨਪੁੱਟ ਵਜੋਂ ਸਹੀ ਢੰਗ ਨਾਲ ਪਾਸ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

2. **ਡਾਟਾ ਸਥਿਰਤਾ:** ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਵਿੱਚੋਂ ਲੰਘਦੇ ਹੋਏ ਡਾਟਾ ਸਥਿਰ ਅਤੇ ਸਹੀ ਰਹਿੰਦਾ ਹੈ। ਪੁਸ਼ਟੀ ਕਰੋ ਕਿ ਡਾਟਾ ਰੂਪਾਂਤਰਣ, ਗਣਨਾਵਾਂ, ਅਤੇ ਸੰਗ੍ਰਹਿ ਸਹੀ ਢੰਗ ਨਾਲ ਕੀਤੇ ਜਾਂਦੇ ਹਨ।

3. ਅਪਵਾਦ ਪ੍ਰਸਾਰ: ਜਾਂਚ ਕਰੋ ਕਿ ਅਪਵਾਦ ਅਤੇ ਗਲਤੀਆਂ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਭਾਗਾਂ ਵਿੱਚ ਕਿਵੇਂ ਫੈਲਦੀਆਂ ਅਤੇ ਸੰਭਾਲੀਆਂ ਜਾਂਦੀਆਂ ਹਨ। ਪੁਸ਼ਟੀ ਕਰੋ ਕਿ ਅਪਵਾਦਾਂ ਨੂੰ ਫੜਿਆ ਜਾਂਦਾ ਹੈ, ਲੌਗ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਅਤੇ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਵਿੱਚ ਵਘਿਨ ਨੂੰ ਰੋਕਣ ਲਈ ਢੁਕਵੇਂ ਢੰਗ ਨਾਲ ਨਜਿੱਠਿਆ ਜਾਂਦਾ ਹੈ।

4. ਅਸਿੰਕਰੋਨਸ ਵਵਿਹਾਰ: ਜੇਕਰ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਵਿੱਚ ਅਸਿੰਕਰੋਨਸ ਭਾਗ ਜਾਂ ਸਮਾਨਾਂਤਰ ਕਾਰਜ ਸ਼ਾਮਲ ਹਨ, ਤਾਂ ਤਾਲਮੇਲ ਅਤੇ ਸਿੰਕਰੋਨਾਈਜ਼ੇਸ਼ਨ ਵਧਿੀਆਂ ਦੀ ਜਾਂਚ ਕਰੋ। ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਸਮਕਾਲੀ ਅਤੇ ਅਸਿੰਕਰੋਨਸ ਸਥਤਿੀਆਂ ਵਿੱਚ ਸਹੀ ਢੰਗ ਨਾਲ ਕੰਮ ਕਰਦਾ ਹੈ।

ਇੱਥੇ Ruby ਵਿੱਚ RSpec ਟੈਸਟਿੰਗ ਫਰੇਮਵਰਕ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇੱਕ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਲਈ ਏਕੀਕਰਣ ਟੈਸਟ ਦੀ ਉਦਾਹਰਣ ਹੈ:

```ruby
RSpec.describe OrderProcessingWorkflow do

  let(:order) { build(:order) }

  it 'processes the order successfully' do
    expect(OrderValidator).to receive(:validate).and_return(true)
    expect(InventoryManager).to receive(:check_availability).and_return(true)
    expect(PaymentProcessor).to receive(:process_payment).and_return(true)
    expect(ShippingService).to receive(:schedule_shipping).and_return(true)

    workflow = OrderProcessingWorkflow.new(order)
    result = workflow.process

    expect(result).to be true
    expect(order.status).to eq('processed')
  end

end
```

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ, `OrderProcessingWorkflow` ਦੀ ਜਾਂਚ ਵੱਖ-ਵੱਖ ਵਰਕਫਲੋ ਭਾਗਾਂ ਵਚਿਕਾਰ ਅੰਤਰਕਰਿਆਵਾਂ ਦੀ ਪੁਸ਼ਟੀ ਕਰਕੇ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਟੈਸਟ ਕੇਸ ਹਰੇਕ ਭਾਗ ਦੇ ਵਵਿਹਾਰ ਲਈ ਉਮੀਦਾਂ ਨਰਿਧਾਰਤ ਕਰਦਾ ਹੈ ਅਤੇ ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕਿ ਵਰਕਫਲੋ ਆਰਡਰ ਨੂੰ ਸਫਲਤਾਪੂਰਵਕ ਪ੍ਰੋਸੈਸ ਕਰਦਾ ਹੈ, ਅਤੇ ਆਰਡਰ ਦੀ ਸਥਤਿੀ ਨੂੰ ਉਸ ਅਨੁਸਾਰ ਅੱਪਡੇਟ ਕਰਦਾ ਹੈ।

ਏ.ਆਈ. ਫੈਸਲਾ ਬਿੰਦੂਆਂ ਦੀ ਜਾਂਚ

ਏ.ਆਈ. ਫੈਸਲਾ ਬਿੰਦੂਆਂ ਦੀ ਜਾਂਚ ਕਰਨਾ ਏ.ਆਈ.-ਸੰਚਾਲਤਿ ਵਰਕਫਲੋਜ਼ ਦੀ ਸ਼ੁੱਧਤਾ ਅਤੇ ਭਰੋਸੇਯੋਗਤਾ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਮਹੱਤਵਪੂਰਨ ਹੈ। ਏ.ਆਈ. ਫੈਸਲਾ ਬਿੰਦੂਆਂ ਦੀ ਜਾਂਚ ਕਰਦੇ ਸਮੇਂ, ਹੇਠ ਲਿਖੀਆਂ ਗੱਲਾਂ ’ਤੇ ਵਿਚਾਰ ਕਰੋ:

1. ਫੈਸਲੇ ਦੀ ਸ਼ੁੱਧਤਾ: ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਏ.ਆਈ. ਭਾਗ ਇਨਪੁੱਟ ਡੇਟਾ ਅਤੇ ਸਿਖਲਾਈ ਪ੍ਰਾਪਤ ਮਾਡਲ ਦੇ ਆਧਾਰ ’ਤੇ ਸਹੀ ਫੈਸਲੇ ਲੈਂਦਾ ਹੈ। ਏ.ਆਈ. ਦੇ ਫੈਸਲਿਆਂ ਦੀ ਉਮੀਦ ਕੀਤੇ ਨਤੀਜਿਆਂ ਜਾਂ ਗਰਾਊਂਡ ਟਰੁੱਥ ਡੇਟਾ ਨਾਲ ਤੁਲਨਾ ਕਰੋ।

2. ਸੀਮਾ ਕੇਸ: ਸੀਮਾ ਕੇਸਾਂ ਅਤੇ ਅਸਧਾਰਨ ਸਥਿਤੀਆਂ ਵਿੱਚ ਏ.ਆਈ. ਭਾਗ ਦੇ ਵਿਵਹਾਰ ਦੀ ਜਾਂਚ ਕਰੋ। ਪੁਸ਼ਟੀ ਕਰੋ ਕਿ ਏ.ਆਈ. ਭਾਗ ਇਹਨਾਂ ਕੇਸਾਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲਦਾ ਹੈ ਅਤੇ ਉਚਿਤ ਫੈਸਲੇ ਲੈਂਦਾ ਹੈ।

3. ਪੱਖਪਾਤ ਅਤੇ ਨਿਰਪੱਖਤਾ: ਏ.ਆਈ. ਭਾਗ ਦੀ ਸੰਭਾਵੀ ਪੱਖਪਾਤਾਂ ਲਈ ਜਾਂਚ ਕਰੋ ਅਤੇ ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਇਹ ਨਿਰਪੱਖ ਅਤੇ ਬਿਨਾਂ ਪੱਖਪਾਤ ਵਾਲੇ ਫੈਸਲੇ ਲੈਂਦਾ ਹੈ। ਵੱਖ-ਵੱਖ ਇਨਪੁੱਟ ਡੇਟਾ ਨਾਲ ਭਾਗ ਦੀ ਜਾਂਚ ਕਰੋ ਅਤੇ ਕਿਸੇ ਵੀ ਭੇਦਭਾਵ ਵਾਲੇ ਪੈਟਰਨ ਲਈ ਨਤੀਜਿਆਂ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰੋ।

4. ਵਿਆਖਿਆਯੋਗਤਾ: ਜੇਕਰ ਏ.ਆਈ. ਭਾਗ ਆਪਣੇ ਫੈਸਲਿਆਂ ਲਈ ਵਿਆਖਿਆਵਾਂ ਜਾਂ ਤਰਕ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ, ਤਾਂ ਵਿਆਖਿਆਵਾਂ ਦੀ ਸ਼ੁੱਧਤਾ ਅਤੇ ਸਪੱਸ਼ਟਤਾ ਦੀ ਪੁਸ਼ਟੀ ਕਰੋ। ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਵਿਆਖਿਆਵਾਂ ਅੰਦਰੂਨੀ ਫੈਸਲਾ-ਲੈਣ ਦੀ ਪ੍ਰਕਰਿਆ ਨਾਲ ਮੇਲ ਖਾਂਦੀਆਂ ਹਨ।

ਇੱਥੇ Ruby ਵਿੱਚ RSpec ਟੈਸਟਿੰਗ ਫਰੇਮਵਰਕ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਏ.ਆਈ. ਫੈਸਲਾ ਬਿੰਦੂ ਦੀ ਜਾਂਚ ਦਾ ਇੱਕ ਉਦਾਹਰਣ ਹੈ:

```
RSpec.describe FraudDetector do
  describe '#detect_fraud' do
    context 'when transaction is fraudulent' do
      let(:tx) do
        build(:transaction, amount: 10_000, location: 'High-Risk Country')
      end

      it 'returns true' do
        expect(subject.detect_fraud(tx)).to be true
      end
    end

    context 'when transaction is legitimate' do
      let(:tx) do
        build(:transaction, amount: 100, location: 'Low-Risk Country')
      end

      it 'returns false' do
        expect(subject.detect_fraud(tx)).to be false
      end
    end
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵੱਚਿ, `FraudDetector` AI ਕੰਪੋਨੈਂਟ ਨੂੰ ਦੋ ਟੈਸਟ ਕੇਸਾਂ ਨਾਲ ਟੈਸਟ ਕੀਤਾ ਗਆਿ ਹੈ: ਇੱਕ ਧੋਖਾਧੜੀ ਵਾਲੇ ਲੈਣ-ਦੇਣ ਲਈ ਅਤੇ ਦੂਜਾ ਵੈਧ ਲੈਣ-ਦੇਣ ਲਈ। ਟੈਸਟ ਕੇਸ ਇਹ ਪੁਸ਼ਟੀ ਕਰਦੇ ਹਨ ਕ ਿ `detect_fraud` ਮੈਥਡ ਲੈਣ-ਦੇਣ ਦੀਆਂ ਵਸ਼ਿਸ਼ਤਾਵਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਉਮੀਦ ਕੀਤੇ ਬੂਲੀਅਨ ਮੁੱਲ ਨੂੰ ਵਾਪਸ ਕਰਦਾ ਹੈ।

ਐਂਡ-ਟੂ-ਐਂਡ ਟੈਸਟਿੰਗ

ਐਂਡ-ਟੂ-ਐਂਡ ਟੈਸਟਿੰਗ ਵੱਚਿ ਸ਼ੁਰੂ ਤੋਂ ਅੰਤ ਤੱਕ ਪੂਰੇ ਵਰਕਫਲੋ ਦੀ ਜਾਂਚ ਕਰਨਾ ਸ਼ਾਮਲ ਹੈ, ਜੋ ਅਸਲ-ਦੁਨੀਆ ਦੇ ਸਨੇਰੀਓ ਅਤੇ ਯੂਜ਼ਰ ਇੰਟਰੈਕਸ਼ਨਾਂ ਦੀ ਨਕਲ ਕਰਦਾ ਹੈ। ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕ ਿਵਰਕਫਲੋ ਸਹੀ ਢੰਗ ਨਾਲ ਕੰਮ ਕਰਦਾ ਹੈ ਅਤੇ ਲੋੜੀਂਦੇ ਨਤੀਜੇ ਪੈਦਾ ਕਰਦਾ ਹੈ। ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਲਈ ਐਂਡ-ਟੂ-ਐਂਡ ਟੈਸਟਿੰਗ ਕਰਦੇ ਸਮੇਂ, ਹੇਠ ਲਖਿੀਆਂ ਗੱਲਾਂ 'ਤੇ ਵਚਿਾਰ ਕਰੋ:

1. **ਯੂਜ਼ਰ ਸਨੇਰੀਓ:** ਆਮ ਯੂਜ਼ਰ ਸਨੇਰੀਓ ਦੀ ਪਛਾਣ ਕਰੋ ਅਤੇ ਇਹਨਾਂ ਸਨੇਰੀਓ ਤਹਤਿ ਵਰਕਫਲੋ ਦੇ ਵਵਿਹਾਰ

ਦੀ ਜਾਂਚ ਕਰੋ। ਪੁਸ਼ਟੀ ਕਰੋ ਕਿ ਵਰਕਫਲੋ ਯੂਜ਼ਰ ਇਨਪੁੱਟ ਨੂੰ ਸਹੀ ਢੰਗ ਨਾਲ ਸੰਭਾਲਦਾ ਹੈ, ਢੁਕਵੇਂ ਫੈਸਲੇ ਲੈਂਦਾ ਹੈ, ਅਤੇ ਉਮੀਦ ਕੀਤੇ ਆਉਟਪੁੱਟ ਪੈਦਾ ਕਰਦਾ ਹੈ।

2. ਡਾਟਾ ਵੈਲੀਡੇਸ਼ਨ: ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਵਰਕਫਲੋ ਡਾਟਾ ਅਸੰਗਤੀਆਂ ਜਾਂ ਸੁਰੱਖਿਆ ਕਮਜ਼ੋਰੀਆਂ ਨੂੰ ਰੋਕਣ ਲਈ ਯੂਜ਼ਰ ਇਨਪੁੱਟ ਦੀ ਪੁਸ਼ਟੀ ਅਤੇ ਸਫਾਈ ਕਰਦਾ ਹੈ। ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਇਨਪੁੱਟ ਡਾਟਾ ਨਾਲ ਵਰਕਫਲੋ ਦੀ ਜਾਂਚ ਕਰੋ, ਜਿਸ ਵਿੱਚ ਵੈਧ ਅਤੇ ਅਵੈਧ ਡਾਟਾ ਸ਼ਾਮਲ ਹੈ।

3. ਗਲਤੀ ਰਿਕਵਰੀ: ਵਰਕਫਲੋ ਦੀ ਗਲਤੀਆਂ ਅਤੇ ਅਪਵਾਦਾਂ ਤੋਂ ਰਿਕਵਰ ਹੋਣ ਦੀ ਯੋਗਤਾ ਦੀ ਜਾਂਚ ਕਰੋ। ਗਲਤੀ ਦੇ ਸਨੇਰੀਓ ਦੀ ਨਕਲ ਕਰੋ ਅਤੇ ਪੁਸ਼ਟੀ ਕਰੋ ਕਿ ਵਰਕਫਲੋ ਉਹਨਾਂ ਨੂੰ ਸੁਚੱਜੇ ਢੰਗ ਨਾਲ ਸੰਭਾਲਦਾ ਹੈ, ਗਲਤੀਆਂ ਨੂੰ ਲੌਗ ਕਰਦਾ ਹੈ, ਅਤੇ ਢੁਕਵੀਆਂ ਰਿਕਵਰੀ ਕਾਰਵਾਈਆਂ ਕਰਦਾ ਹੈ।

4. ਕਾਰਗੁਜ਼ਾਰੀ ਅਤੇ ਸਕੇਲੇਬਲਿਟੀ: ਵੱਖ-ਵੱਖ ਲੋਡ ਸਥਿਤੀਆਂ ਤਹਿਤ ਵਰਕਫਲੋ ਦੀ ਕਾਰਗੁਜ਼ਾਰੀ ਅਤੇ ਸਕੇਲੇਬਲਿਟੀ ਦਾ ਮੁਲਾਂਕਣ ਕਰੋ। ਵੱਡੀ ਮਾਤਰਾ ਵਿੱਚ ਇਕੋ ਸਮੇਂ 'ਤੇ ਬੇਨਤੀਆਂ ਨਾਲ ਵਰਕਫਲੋ ਦੀ ਜਾਂਚ ਕਰੋ ਅਤੇ ਜਵਾਬ ਸਮੇਂ, ਸਰੋਤ ਵਰਤੋਂ, ਅਤੇ ਸਮੁੱਚੀ ਸਿਸਟਮ ਸਥਿਰਤਾ ਨੂੰ ਮਾਪੋ।

ਇੱਥੇ Ruby ਵਿੱਚ RSpec ਟੈਸਟਿੰਗ ਫਰੇਮਵਰਕ ਅਤੇ ਯੂਜ਼ਰ ਇੰਟਰੈਕਸ਼ਨਾਂ ਦੀ ਨਕਲ ਲਈ Capybara ਲਾਇਬ੍ਰੇਰੀ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇੱਕ ਵਰਕਫਲੋ ਲਈ ਐਂਡ-ਟੂ-ਐਂਡ ਟੈਸਟ ਦੀ ਉਦਾਹਰਣ ਹੈ:

```
RSpec.describe 'Order Processing Workflow' do
  scenario 'User places an order successfully' do
    visit '/orders/new'
    fill_in 'Product', with: 'Sample Product'
    fill_in 'Quantity', with: '2'
    fill_in 'Shipping Address', with: '123 Main St'
    click_button 'Place Order'

    expect(page).to have_content('Order Placed Successfully')
    expect(Order.count).to eq(1)
    expect(Order.last.status).to eq('processed')
  end
end
```

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ, ਐਂਡ-ਟੂ-ਐਂਡ ਟੈਸਟ ਵੈੱਬ ਇੰਟਰਫੇਸ ਰਾਹੀਂ ਇੱਕ ਯੂਜ਼ਰ ਦੁਆਰਾ ਆਰਡਰ ਪਲੇਸ ਕਰਨ ਦੀ ਨਕਲ ਕਰਦਾ ਹੈ। ਇਹ ਲੋੜੀਂਦੇ ਫਾਰਮ ਫੀਲਡਾਂ ਨੂੰ ਭਰਦਾ ਹੈ, ਆਰਡਰ ਸਬਮਿਟ ਕਰਦਾ ਹੈ, ਅਤੇ ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕਿ ਆਰਡਰ ਸਫਲਤਾਪੂਰਵਕ ਪ੍ਰੋਸੈਸ ਹੋ ਗਿਆ ਹੈ, ਢੁਕਵਾਂ ਪੁਸ਼ਟੀਕਰਨ ਸੁਨੇਹਾ ਦਖਿਾ ਰਹਿਾ ਹੈ ਅਤੇ ਡੇਟਾਬੇਸ ਵਿੱਚ ਆਰਡਰ ਦੀ ਸਥਿਤੀ ਅੱਪਡੇਟ ਕਰ ਰਹਿਾ ਹੈ।

ਨਰਿੰਤਰ ਏਕੀਕਰਨ ਅਤੇ ਤੈਨਾਤੀ

ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋਜ਼ ਦੀ ਭਰੋਸੇਯੋਗਤਾ ਅਤੇ ਰੱਖ-ਰਖਾਅ ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ, ਨਰਿੰਤਰ ਏਕੀਕਰਨ ਅਤੇ ਤੈਨਾਤੀ (CI/CD) ਪਾਈਪਲਾਈਨ ਵੱਚਿ ਟੈਸਟਿੰਗ ਅਤੇ ਵੈਲੀਡੇਸ਼ਨ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰਨ ਦੀ ਸਫਿਾਰਸ਼ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਪ੍ਰੋਡਕਸ਼ਨ ’ਤੇ ਤੈਨਾਤ ਕਰਨ ਤੋਂ ਪਹਲਿਾਂ ਵਰਕਫਲੋ ਤਬਦੀਲੀਆਂ ਦੀ ਸਵੈਚਲਤਿ ਟੈਸਟਿੰਗ ਅਤੇ ਵੈਲੀਡੇਸ਼ਨ ਦੀ ਆਗਆਿ ਦੰਦਿਾ ਹੈ। ਹੇਠ ਲਖਿੀਆਂ ਪ੍ਰਥਾਵਾਂ ’ਤੇ ਵਚਿਾਰ ਕਰੋ:

1. ਸਵੈਚਲਤਿ ਟੈਸਟ ਐਗਜ਼ੀਕਊਿਸ਼ਨ: CI/CD ਪਾਈਪਲਾਈਨ ਨੂੰ ਕੌਨਫਗਿਰ ਕਰੋ ਤਾਂ ਜੋ ਜਦੋਂ ਵੀ ਵਰਕਫਲੋ ਕੋਡਬੇਸ ਵੱਚਿ ਤਬਦੀਲੀਆਂ ਕੀਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ ਤਾਂ ਟੈਸਟ ਸੂਟ ਆਪਣੇ ਆਪ ਚੱਲ ਜਾਵੇ। ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ ਕ ਿਕੋਈ ਵੀ ਰਗਿਰੈਸ਼ਨ ਜਾਂ ਅਸਫਲਤਾ ਵਕਿਾਸ ਪ੍ਰਕਰਿਆ ਦੇ ਸ਼ੁਰੂਆਤੀ ਪੜਾਅ ’ਤੇ ਹੀ ਪਤਾ ਲੱਗ ਜਾਵੇ।

2. ਟੈਸਟ ਕਵਰੇਜ ਮਾਨੀਟਰਿੰਗ: ਵਰਕਫਲੋ ਕੰਪੋਨੈਂਟਸ ਅਤੇ AI ਡਸਿੀਜ਼ਨ ਪੁਆਇੰਟਸ ਦੀ ਟੈਸਟ ਕਵਰੇਜ ਨੂੰ ਮਾਪੋ ਅਤੇ ਮਾਨੀਟਰ ਕਰੋ। ਇਹ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਉੱਚ ਟੈਸਟ ਕਵਰੇਜ ਦਾ ਟੀਚਾ ਰੱਖੋ ਕ ਿਮਹੱਤਵਪੂਰਨ ਮਾਰਗਾਂ ਅਤੇ ਸਥਤਿੀਆਂ ਦੀ ਚੰਗੀ ਤਰ੍ਹਾਂ ਜਾਂਚ ਕੀਤੀ ਜਾਵੇ।

3. ਨਰਿੰਤਰ ਫੀਡਬੈਕ: ਵਕਿਾਸ ਵਰਕਫਲੋ ਵੱਚਿ ਟੈਸਟ ਨਤੀਜਆਿਂ ਅਤੇ ਕੋਡ ਕੁਆਲਟਿੀ ਮੈਟ੍ਰਕਿਸ ਨੂੰ ਏਕੀਕ੍ਰਤਿ ਕਰੋ। ਡਵਿੈਲਪਰਾਂ ਨੂੰ ਟੈਸਟਾਂ ਦੀ ਸਥਤਿੀ, ਕੋਡ ਕੁਆਲਟਿੀ, ਅਤੇ CI/CD ਪ੍ਰਕਰਿਆ ਦੌਰਾਨ ਪਤਾ ਲੱਗੀਆਂ ਕਸਿੇ ਵੀ ਸਮੱਸਆਿਵਾਂ ਬਾਰੇ ਨਰਿੰਤਰ ਫੀਡਬੈਕ ਪ੍ਰਦਾਨ ਕਰੋ।

4. ਸਟੇਜਿੰਗ ਵਾਤਾਵਰਣ: ਵਰਕਫਲੋ ਨੂੰ ਸਟੇਜਿੰਗ ਵਾਤਾਵਰਣਾਂ ਵੱਚਿ ਤੈਨਾਤ ਕਰੋ ਜੋ ਪ੍ਰੋਡਕਸ਼ਨ ਵਾਤਾਵਰਣ ਦੇ ਬਹੁਤ ਨੇੜੇ ਹੋਣ। ਇੰਫਰਾਸਟ੍ਰਕਚਰ, ਕੌਨਫਗਿਰੇਸ਼ਨ, ਜਾਂ ਡੇਟਾ ਏਕੀਕਰਨ ਨਾਲ ਸਬੰਧਤ ਕਸਿੇ ਵੀ ਸਮੱਸਆਿ ਨੂੰ ਫੜਨ ਲਈ ਸਟੇਜਿੰਗ ਵਾਤਾਵਰਣ ਵੱਚਿ ਵਾਧੂ ਟੈਸਟਿੰਗ ਅਤੇ ਵੈਲੀਡੇਸ਼ਨ ਕਰੋ।

5. ਰੋਲਬੈਕ ਮਕੈਨਜ਼ਿਮ: ਤੈਨਾਤੀ ਅਸਫਲਤਾਵਾਂ ਜਾਂ ਪ੍ਰੋਡਕਸ਼ਨ ਵੱਚਿ ਪਤਾ ਲੱਗੀਆਂ ਗੰਭੀਰ ਸਮੱਸਆਿਵਾਂ ਦੀ ਸਥਤਿੀ ਵੱਚਿ ਰੋਲਬੈਕ ਮਕੈਨਜ਼ਿਮ ਲਾਗੂ ਕਰੋ। ਇਹ ਯਕੀਨੀ ਬਣਾਓ ਕ ਿਡਾਊਨਟਾਈਮ ਅਤੇ ਯੂਜ਼ਰਾਂ ’ਤੇ ਪ੍ਰਭਾਵ ਨੂੰ ਘੱਟ ਕਰਨ ਲਈ ਵਰਕਫਲੋ ਨੂੰ ਤੇਜ਼ੀ ਨਾਲ ਪਛਿਲੇ ਸਥਰਿ ਵਰਜਨ ’ਤੇ ਵਾਪਸ ਕੀਤਾ ਜਾ ਸਕੇ।

ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋਜ਼ ਦੇ ਵਕਿਾਸ ਜੀਵਨ-ਚੱਕਰ ਵੱਚਿ ਟੈਸਟਿੰਗ ਅਤੇ ਵੈਲੀਡੇਸ਼ਨ ਨੂੰ ਸ਼ਾਮਲ ਕਰਕੇ, ਸੰਸਥਾਵਾਂ ਆਪਣੇ AI-ਪਾਵਰਡ ਸਸਿਟਮਾਂ ਦੀ ਭਰੋਸੇਯੋਗਤਾ, ਸ਼ੁੱਧਤਾ, ਅਤੇ ਰੱਖ-ਰਖਾਅ ਨੂੰ ਯਕੀਨੀ ਬਣਾ ਸਕਦੀਆਂ ਹਨ। ਨਯਿਮਤਿ ਟੈਸਟਿੰਗ ਅਤੇ ਵੈਲੀਡੇਸ਼ਨ ਬੱਗਾਂ ਨੂੰ ਫੜਨ, ਰਗਿਰੈਸ਼ਨਾਂ ਨੂੰ ਰੋਕਣ, ਅਤੇ ਵਰਕਫਲੋ ਦੇ ਵਵਿਹਾਰ ਅਤੇ ਨਤੀਜਆਿਂ ਵੱਚਿ ਭਰੋਸਾ ਬਣਾਉਣ ਵੱਚਿ ਮਦਦ ਕਰਦੀ ਹੈ।

ਭਾਗ 2: ਪੈਟਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰਿੰਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਵਿਚਾਰਾਂ ਦੀ ਲੜੀ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਿਵੇਂ ਕੰਮ ਕਰਦੀ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਣਾਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸਮੱਗਰੀ ਨਿਰਮਾਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਢਾਂਚਾਗਤ ਇਕਾਈ ਨਿਰਮਾਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

LLM ਏਜੰਟ ਮਾਰਗਦਰਸ਼ਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਲਾਭ ਅਤੇ ਵਿਚਾਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮੋਡ ਸਵਿੱਚ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਿਵੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਸਦੀ ਵਰਤੋਂ ਕਦੋਂ ਕਰੀਏ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਭੂਮਿਕਾ ਨਿਰਧਾਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਿਵੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਸਨੂੰ ਕਦੋਂ ਵਰਤਣਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਨਾਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪ੍ਰੌਮਪਟ ਔਬਜੈਕਟ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

Prompt Template

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਲਾਭ ਅਤੇ ਵਚਿਾਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਸਨੂੰ ਕਦੋਂ ਵਰਤਣਾ ਹੈ:

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੰਰਚਤਿ ਇਨਪੁੱਟ-ਆਊਟਪੁੱਟ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੰਰਚਤਿ IO ਨੂੰ ਵਧਾਉਣਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਲਾਭ ਅਤੇ ਵਚਿਾਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪ੍ਰੌਮਪਟ ਚੇਨਿੰਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਸਨੂੰ ਕਦੋਂ ਵਰਤਣਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਣ: Olympia ਦੀ ਔਨਬੋਰਡਿੰਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪ੍ਰੌਮਪਟ ਰੀਰਾਈਟਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਰਸਿਪਾਂਸ ਫੈਸਿੰਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਲਾਭ ਅਤੇ ਵਚਿਾਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਗਲਤੀ ਸੰਭਾਲ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਕਵੇਰੀ ਐਨਾਲਾਈਜ਼ਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਲਾਗੂਕਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਭਾਸ਼ਣ-ਦੇ-ਭਾਗ (POS) ਟੈਗਿੰਗ ਅਤੇ ਨਾਮਤਿ ਇਕਾਈ ਪਛਾਣ (NER)

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਰਾਦਾ ਵਰਗੀਕਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮੁੱਖ-ਸ਼ਬਦ ਕੱਢਣਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਲਾਭ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਕੁਐਰੀ ਰੀਰਾਈਟਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਲਾਭ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

Ventriloquist

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਕਦੋਂ ਵਰਤਣਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਵੱਖਰੇ ਭਾਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪ੍ਰੈਡੀਕੇਟ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਸ ਨੂੰ ਕਦੋਂ ਵਰਤਣਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਏ.ਪੀ.ਆਈ. ਫਸਾਡ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮੁੱਖ ਲਾਭ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਕਦੋਂ ਵਰਤੀਏ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪ੍ਰਮਾਣੀਕਰਨ ਅਤੇ ਅਧਕਿਾਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਬੇਨਤੀ ਪ੍ਰਬੰਧਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਜਵਾਬ ਫਾਰਮੈਟਿੰਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਗਲਤੀ ਪ੍ਰਬੰਧਨ ਅਤੇ ਸੀਮਾ ਕੇਸ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸਕੇਲੇਬਲਿਟੀ ਅਤੇ ਪ੍ਰਦਰਸ਼ਨ ਵਚਿਾਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਹੋਰ ਡਜ਼ਿਾਈਨ ਪੈਟਰਨਾਂ ਨਾਲ ਤੁਲਨਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਨਤੀਜਾ ਵਆਿਖਆਿਕਾਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਸਦੀ ਵਰਤੋਂ ਕਦੋਂ ਕਰਨੀ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਵਰਚੁਅਲ ਮਸ਼ੀਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਸਨੂੰ ਕਦੋਂ ਵਰਤਣਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਜਾਦੂ ਦੇ ਪੱਛੇ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸਪੈਸੀਫਕਿਸ਼ਨ ਅਤੇ ਟੈਸਟਿੰਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਵਵਿਹਾਰ ਦੀ ਸਪੈਸੀਫਕਿੇਸ਼ਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਟੈਸਟ ਕੇਸ ਲਖਿਣਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਨ: ਅਨੁਵਾਦਕ ਕੰਪੋਨੈਂਟ ਦੀ ਟੈਸਟੰਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

HTTP ਇੰਟਰੈਕਸ਼ਨਾਂ ਦਾ ਰੀਪਲੇਅ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮਨੁੱਖੀ-ਸ਼ਮੂਲੀਅਤ ਪ੍ਰਣਾਲੀ (HITL)

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉੱਚ-ਪੱਧਰੀ ਪੈਟਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਹਾਈਬ੍ਰਡਿ ਬੁੱਧੀ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਅਨੁਕੂਲੀ ਪ੍ਰਤੀਕਰਿਆ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮਨੁੱਖ-ਏਆਈ ਭੂਮਕਿਾ ਬਦਲੀ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਵਧਾਅ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮੁੱਖ ਲਾਭ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਅਸਲ-ਦੁਨੀਆ ਦੀ ਵਰਤੋਂ: ਸਹਿਤ ਸੰਭਾਲ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਫੀਡਬੈਕ ਲੂਪ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਐਪਲੀਕੇਸ਼ਨਾਂ ਅਤੇ ਉਦਾਹਰਣਾਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮਨੁੱਖੀ ਫੀਡਬੈਕ ਏਕੀਕਰਣ ਵੱਚਿ ਉੱਨਤ ਤਕਨੀਕਾਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਨਸ਼ਿਕਰਿਆ ਜਾਣਕਾਰੀ ਵਿਕੀਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੰਦਰਭਕਿ ਜਾਣਕਾਰੀ ਪ੍ਰਦਰਸ਼ਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਪ੍ਰੋਐਕਟਵਿ ਸੂਚਨਾਵਾਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਵਆਿਖਆਿਤਮਕ ਅੰਤਰਦ੍ਰਸ਼ਿਟੀਆਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇੰਟਰੈਕਟਵਿ ਖੋਜ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਮੁੱਖ ਲਾਭ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਐਪਲੀਕੇਸ਼ਨਾਂ ਅਤੇ ਉਦਾਹਰਨਾਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸਹਯੋਗੀ ਫੈਸਲਾ ਲੈਣਾ (CDM)

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਲਗਾਤਾਰ ਸੱਖਿਣਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਐਪਲੀਕੇਸ਼ਨਾਂ ਅਤੇ ਉਦਾਹਰਣਾਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਨੈਤਕਿ ਵਚਿਾਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

AI ਜੋਖਮਾਂ ਨੂੰ ਘਟਾਉਣ ਵੱਚਿ HITL ਦੀ ਭੂਮਕਿਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਤਕਨੀਕੀ ਤਰੱਕੀ ਅਤੇ ਭਵੱਖਿ ਦਾ ਦ੍ਰਸ਼ਿਟੀਕੋਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਐੱਚਆਈਟੀਐੱਲ (HITL) ਸਸਿਟਮਾਂ ਦੀਆਂ ਚੁਣੌਤੀਆਂ ਅਤੇ ਸੀਮਾਵਾਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਬੁੱਧੀਮਾਨ ਗਲਤੀ ਸੰਭਾਲ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਰਵਾਇਤੀ ਗਲਤੀ ਸੰਭਾਲ ਪਹੁੰਚਾਂ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੰਦਰਭਕਿ ਗਲਤੀ ਨਦਿਾਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੰਦਰਭਕਿ ਤਰੁੱਟੀ ਨਦਿਾਨ ਲਈ ਪ੍ਰੌਮਪਟ ਇੰਜੀਨੀਅਰੰਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੰਦਰਭਤਿ ਗਲਤੀ ਜਾਂਚ ਲਈ ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਵਧਤਿ ਉਤਪਾਦਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਬੁੱਧੀਮਾਨ ਗਲਤੀ ਰਿਪੋਰਟਿੰਗ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਭਵਿੱਖ-ਸੂਚਕ ਗਲਤੀ ਰੋਕਥਾਮ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸਮਾਰਟ ਗਲਤੀ ਰਕਿਵਰੀ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਨੌਜੀ ਗਲਤੀ ਸੰਚਾਰ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਅਨੁਕੂਲ ਗਲਤੀ ਨਪਿਟਾਰਾ ਵਰਕਫਲੋ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਗੁਣਵੱਤਾ ਨਯਿੰਤਰਣ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਈਵੈਲ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸਮੱਸਆਿ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਹੱਲ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੇਂ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਵਚਿਾਰ-ਵਟਾਂਦਰਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੁਨਹਿਰੀ ਹਵਾਲਿਆਂ ਨੂੰ ਸਮਝਣਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਰੈਫਰੈਂਸ-ਫ੍ਰੀ ਮੁਲਾਂਕਣ ਕਿਵੇਂ ਕੰਮ ਕਰਦੇ ਹਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੁਰੱਖਿਆ ਉਪਾਅ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸਮੱਸਿਆ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਹੱਲ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਇਹ ਕਵਿੰ ਕੰਮ ਕਰਦਾ ਹੈ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਉਦਾਹਰਨ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਵਿਚਾਰ-ਵਟਾਂਦਰੇ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੁਰੱਖਿਆ ਰੇਲਾਂ ਅਤੇ ਮੁਲਾਂਕਣ: ਇੱਕੋ ਸਿੱਕੇ ਦੇ ਦੋ ਪਾਸੇ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸੁਰੱਖਿਆ ਰੇਲਾਂ ਅਤੇ ਹਵਾਲਾ-ਮੁਕਤ ਮੁਲਾਂਕਣਾਂ ਦੀ ਅੰਤਰ-ਬਦਲੀ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਦੋਹਰੇ-ਉਦੇਸ਼ ਵਾਲੇ ਸੁਰੱਖਿਆ ਰੇਲਿੰਗ ਅਤੇ ਮੁਲਾਂਕਣਾਂ ਨੂੰ ਲਾਗੂ ਕਰਨਾ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਿਤਾਬ ਵਿਚ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਿਤਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸ਼ਬਦਾਵਲੀ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

ਸ਼ਬਦਾਵਲੀ

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

A

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

B

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

C

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

D

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

E

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

F

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

G

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

H

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

I

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

J

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

K

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

L

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

M

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

N

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

O

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

P

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

Q

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

R

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

S

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

T

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

U

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

V

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

W

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

Z

ਇਸ ਸਮੱਗਰੀ ਦੀ ਨਮੂਨਾ ਕਤਿਾਬ ਵਚਿ ਉਪਲਬਧਤਾ ਨਹੀਂ ਹੈ। ਇਹ ਕਤਿਾਬ Leanpub ਤੇ ਖਰੀਦੀ ਜਾ ਸਕਦੀ ਹੈ http://leanpub.com/patterns-of-application-development-using-ai-pa.

Index

ACID ਗੁਣ, 101
AI, 68, 91, 139, 194
    applications, 128
    compound systems, 27, 28
    conversational, 28, 195
    model, 90, 91
    ਮਾਡਲ, 194
    ਮਸ਼ਿਰਤਿ ਸਸਿਟਮ, 31
    ਸੰਵਾਦੀ, 6
AI ਵਰਕਰਾਂ ਦੀ ਲੜੀਬੱਧਤਾ, 102
Alpaca, 12
Altman, Sam, 16
Amazon Web Services, 232
Anthropic, 20, 36, 67, 119, 127
APIs, 114
audit logging, 98

BERT, 12
Brotli, 233
Byte Pair Encoding (BPE), 13

C (ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ), 107
Capybara ਲਾਇਬ੍ਰੇਰੀ, 239
ChatGPT, 27, 48
classification, 111
Claude, 7, 40, 71
Claude 3, 45, 117, 120, 125, 127
Claude 3 Opus, 68
Claude v1, 15
Claude v2, 15
Cohere (LLM Provider), 20
concurrent workflows, 233
Customer Sentiment Analysis, 91

Databricks ਕਰਮਚਾਰੀ, 47
Datadog, 229
decision
    -making capabilities, 91
document clustering, 111

ELK ਸਟੈਕ, 102
errors
    handling, 98

F#, 85
finalize ਮੈਥਡ, 147
FitAI, 195

Gemma 7B, 10
GitLab, 85
Google, 20
    API, 57, 59
    Cloud Platform, 232
    Gemini, 19
    Gemini 1.5 Pro, 13, 16, 17

PaLM (Pathways Language Model), 16
T5, 12
ਕਲਾਉਡ ਏਆਈ ਪਲੇਟਫਾਰਮ, 22
ਪਾਮ (ਪਾਥਵੇਜ਼ ਲੈਂਗਵੇਜ ਮਾਡਲ), 22
GPT-3, 12, 15
GPT-4, 6, 12, 15, 16, 19, 28, 40, 45, 57, 96, 108, 110, 118, 124, 189, 230
Graham, Paul, 17
Groq, 24, 110
gzip, 233

Hohpe, Gregor, 96
Honeybadger, 86
HTTP, 139

input
prompts, 51
validation, 234
intelligent workflow orchestration, 233

JSON (JavaScript Object Notation), 117, 121, 122, 125, 154
JSON (ਜਾਵਾਸਕ੍ਰਿਪਟ ਆਬਜੈਕਟ ਨੋਟੇਸ਼ਨ), 137

K-means, 112

Large Language Model (LLM), 111
Latent Dirichlet Allocation, 112
Llama, 12
Llama 2-70B, 46
Llama 3 70B, 10
Llama 3 8B, 10
LLMs ਨੂੰ ਏਕੀਕ੍ਰਿਤ ਕਰਨਾ, 174
Louvre, 39

MessagePack, 233
Metropolitan Museum of Art, 39
Mistral
7B, 10
7B Instruct, 15, 189
Mixtral
8x22B, 10
8x7B, 51
Multi-Agent
Problem Solvers, 28

Naive Bayes, 111
natural language
Natural Language Processing (NLP), 111
New Relic, 231

Olympia, 30, 57, 119, 140, 155
Olympia ਦਾ ਗਿਆਨ ਅਧਾਰ, 84
OpenAI, 3, 20, 36, 67
OpenRouter, 25, 26, 140, 231
output verification, 234

Perplexity (ਪ੍ਰਦਾਤਾ), 11
Process Manager, 99
prompts
engineering, 51
Protocol Buffers, 233

Qwen2 70B, 10

Rails, 180
Railway Oriented Programming (ROP), 87
Raix, 211
    ਲਾਇਬ੍ਰੇਰੀ, 89
Retrieval Augmented Generation (RAG), 29
RSpec, 234, 236, 239
Ruby, 85, 86, 104, 151, 239
Ruby on Rails, 1, 102, 210, 218
Rudall, Alex, 21
Rust (Programming Language), 85
Rust (ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ), 107

Scout, 231
Stripe, 120
Support Vector Machines (SVM), 111
system directive, 90

Together.ai, 24
topic identification, 111

Unicode-encodable language, 13
Universal ID, 233

Wall, Larry, 3
Wisper, 86, 98, 140, 147
Wooley, Chad, 85

XML, 124

Yi-34B, 46

ਅਣਸੁਪਰਵਾਈਜ਼ਡ ਲਰਨਿੰਗ, 4
ਅਨੁਕੂਲਤਾ, 24
ਅਨੁਕੂਲੀ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ, 192
ਅਨੁਕੂਲੀ ਵਰਕਫਲੋ
    ਅਨੁਕੂਲੀ ਵਰਕਫਲੋ ਰਚਨਾ, 207
ਅਨੁਮਾਨ, 5
ਅਨੁਵਾਦ, 15, 181
ਅਪਵਾਦ ਪ੍ਰਬੰਧਨ, 210
ਅਪਵਾਦ ਸੰਭਾਲ, 208
ਅਸਿੰਕ੍ਰੋਨਸ ਪ੍ਰੋਸੈਸਿੰਗ, 229
ਅੰਤਰਰਾਸ਼ਟਰੀਕਰਨ, 180
ਆਟੋ ਕੰਟੀਨਊਇਟੇਸ਼ਨ, 148
ਆਟੋ-ਸਕੇਲਿੰਗ, 232
ਆਧੁਨਿਕ ਐਪਲੀਕੇਸ਼ਨਾਂ, 205
ਆਨਲਾਈਨ ਰਟਿਲਰ, 189
ਆਵਾਜ਼-ਨਿਯੰਤਰਿਤ ਇੰਟਰਫੇਸ, 30
ਆਸ਼ਾਵਾਦੀ ਲੌਕਿੰਗ, 101
ਇਕਸਾਰਤਾ
    ਅਤੇ ਦੁਹਰਾਉਣਯੋਗਤਾ, 123
ਇਤਿਹਾਸਕ ਪੈਟਰਨ, 207
ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰ, 119
ਈ-ਕਾਮਰਸ, 177, 204
ਈ-ਕਾਮਰਸ ਐਪਲੀਕੇਸ਼ਨਾਂ, 84
ਈਕੋਸਿਸਟਮ, 137
ਉਤਪਾਦ ਸਫਿਾਰਸ਼ਾਂ, 84
ਉਤਪਾਦਕਤਾ, 176
ਉਪਭੋਗਤਾ-ਨਿਰਮਿਤ ਸਮੱਗਰੀ, 102
ਉੱਚ-ਪ੍ਰਦਰਸ਼ਨ ਕੰਪਲੀਸ਼ਨ, 24
ਏ.ਆਈ., 59, 125, 133, 187
    ਐਪਲੀਕੇਸ਼ਨਾਂ, 116, 138, 151
    ਫੈਸਲਾ ਬਿੰਦੂ, 237
    ਮਾਡਲ, 81, 144, 147
ਏ.ਪੀ.ਆਈ., 65, 142
ਏਆਈ, 119

ਮਾਡਲ, 145
ਏਕੀਕਰਣ ਟੈਸਟਿੰਗ, 235
ਏਜੰਟਕਿ, 29
ਐਟਰਪ੍ਰਾਈਜ਼ ਇੰਟੀਗ੍ਰੇਸ਼ਨ ਪੈਟਰਨ, 96
ਐਟਰਪ੍ਰਾਈਜ਼ ਐਪਲੀਕੇਸ਼ਨ ਆਰਕੀਟੈਕਚਰ, 35
ਐਡ-ਟੂ-ਐਂਡ ਟੈਸਟਿੰਗ, 238, 239
ਐਨਸੈਬਲ, 108
ਐਪਲੀਕੇਸ਼ਨ ਡਿਜ਼ਾਈਨ ਅਤੇ ਫਰੇਮਵਰਕਸ, 183
ਐਮਰਜੈਂਸੀ ਪ੍ਰਤੀਕਰਿਆ ਯੋਜਨਾਬੰਦੀ, 30
ਐਰੇ, 121
ਐਂਸਕਊਐਲ ਇੰਜੈਕਸ਼ਨਾਂ, 64
ਓਪਨ ਸੋਰਸ ਮਾਡਲ ਹੋਸਟਿੰਗ ਪ੍ਰਦਾਤਾ, 189
ਓਪੀਟੀ ਮਾਡਲ, 22
ਓਲਾਮਾ, 23
ਓਲੰਪੀਆ, 133
ਕਮਾਂਡ ਲਾਈਨ
ਕਮਾਂਡ-ਲਾਈਨ ਇੰਟਰਫੇਸ (CLI), 23
ਕਰਮਚਾਰੀਆਂ ਦੀ ਬਰੁਤਾਤ, 109
ਕਰਾਸ-ਮਾਧਅਿਮ ਉਤਪਾਦਨ, 20
ਕਲੀਨਕਿਲ ਫ਼ੈਸਲਾ ਸਹਾਇਤਾ, 95
ਕਹਾਣੀ ਨਰਿਮਾਣ, 18
ਕਾਰਗੁਜ਼ਾਰੀ
ਅਨੁਕੂਲਤਾ, 123
ਅਨੁਕੂਲਨ, 227
ਸਮਝੌਤੇ, 5
ਸਮੱਸਆਿਵਾਂ, 231
ਕੁਆਂਟੀਕਰਨ, 26
ਕੁਦਰਤੀ ਭਾਸ਼ਾ
ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਪ੍ਰੋਸੈਸਿੰਗ (NLP), 93
ਕੁਸ਼ਲਤਾ, 204
ਕੈਸ਼ਿੰਗ, 230
ਕੋਹੇਅਰ (LLM ਪ੍ਰਦਾਤਾ), 23
ਕ੍ਰਮਕਿ ਪ੍ਰਗਟਾਵਾ, 191
ਕੰਪਊਿਟਰ ਵਗਿਆਨ, 64, 66
ਖਾਤਾ, 83
ਗਤੀਸ਼ੀਲ UI ਨਰਿਮਾਣ, 174
ਗਤੀਸ਼ੀਲ ਟੂਲ ਚੋਣ, 121
ਗਲਤੀ ਸੰਭਾਲ, 234
ਗਲਤੀਆਂ
ਦਰਾਂ, 102
ਬੁੱਧੀਮਾਨ ਗਲਤੀ ਪ੍ਰਬੰਧਨ, 133
ਰਕਿਵਰੀ, 239
ਸੰਭਾਲ, 101, 132
ਗਲੋਬਲ ਇੰਟਰਪ੍ਰੇਟਰ ਲੌਕ (GIL), 106
ਗਾਹਕ ਸਹਾਇਤਾ, 29
ਗਾਹਕ ਸੇਵਾ ਚੈਟਬੋਟ, 30
ਗਆਿਨ ਪ੍ਰਬੰਧਨ, 29
ਗਆਿਨ ਭੰਡਾਰ, 7
ਗੁੰਝਲਦਾਰ ਕੰਮ, 136
ਗ੍ਰਾਫਕਊਐਲ, 99
ਗ੍ਰਾਫਕਿਲ ਮਾਡਲ, 40
ਗੱਲਬਾਤ
ਲਖਿਤ, 145, 148
ਲੂਪ, 146, 148
ਘਟਨਾ-ਚਾਲਤਿ ਆਰਕੀਟੈਕਚਰ, 100
ਘੱਟੋ-ਘੱਟ ਵਸ਼ਿਸ਼-ਅਧਕਿਾਰ ਦਾ ਸਧਿਾਂਤ, 65
ਚੈਟਬੌਟ ਐਪਲੀਕੇਸ਼ਨ, 109
ਛਪਿਆ ਹੋਇਆ ਸਪੇਸ, 37
ਜਨਰੇਟਵਿ UI (GenUI), 190, 200
ਜਨਰੇਟਵਿ ਪ੍ਰੀ-ਟਰੇਨਡ ਟ੍ਰਾਂਸਫਾਰਮਰ (GPT), 7
ਜਨਰੇਟਵਿ ਪ੍ਰੀ-ਟ੍ਰੇਨਡ ਟ੍ਰਾਂਸਫਾਰਮਰ (GPT), 61
ਜਨਰੇਟਵਿ ਯੂਆਈ (GenUI), 193
ਜਨਰੇਟਵਿ ਯੂਆਈ (ਜੈਨਯੂਆਈ), 183, 197
ਜ਼ਬਰਦਸਤੀ ਟੂਲ ਚੋਣ, 122

ਜ਼ੀਰੋ-ਸ਼ਾਟ ਲਰਨਿੰਗ, 54
ਜ਼ੀਰੋ-ਸੌਟ ਲਰਨਿੰਗ, 53
ਜਾਣਕਾਰੀ
    ਕੱਢਣਾ, 47
    ਪੁਨਰ-ਪ੍ਰਾਪਤੀ, 6, 116
ਜੀਪੀਟੀ-4, 188
ਜੋਖਮ ਕਾਰਕ, 88, 89
ਜੋਖਮ ਵਰਗੀਕਰਨ, 94
ਟਕਿਟ ਨਿਰਧਾਰਨ, 221
ਟੀ5, 22
ਟੂਲ ਕਾਲ, 142
ਟੂਲ ਵਰਤੋਂ, 114
ਟੈਕਸਟ ਸਫਾਈ, 103
ਟੈਬਲੇਟ, 201
ਟੋਕਨ, 5, 11
ਟੋਕਨਾਈਜ਼ੇਸ਼ਨ, 11
ਟੌਪ-ਕੇ ਸੈਂਪਲਿੰਗ, 44
ਟੌਪ-ਪੀ (ਨਿਊਕਲੀਅਸ) ਸੈਂਪਲਿੰਗ, 44
ਟ੍ਰਾਂਸਫਾਰਮਰ ਆਰਕੀਟੈਕਚਰ, 6
ਟ੍ਰਿੰਗਿਰ ਸੁਨੇਹਾ, 96
ਟ੍ਰੈਫਿਕ ਪ੍ਰਬੰਧਨ, 30
ਡਾਇਨਾਮਿਕ ਟਾਸਕ ਰੂਟਿੰਗ, 206
ਡਾਟਾ
    ਅਖੰਡਤਾ, 221
    ਗੋਪਨੀਯਤਾ, 24
    ਡਾਟਾ ਪ੍ਰਾਪਤੀ, 101
    ਡਾਟਾ ਵੈਲੀਡੇਸ਼ਨ, 239
    ਡਾਟਾ ਸਿੰਕਰੋਨਾਈਜ਼ੇਸ਼ਨ, 101
    ਤਿਆਰੀ, 100
    ਪਰਦੇਦਾਰੀ, 199
    ਪ੍ਰੋਸੈਸਿੰਗ ਕਾਰਜ, 116
    ਪ੍ਰੋਸੈਸਿੰਗ ਪਾਈਪਲਾਈਨ, 221
    ਫਲੋ, 101
    ਵਿਸ਼ਲੇਸ਼ਣ, 31, 137
    ਸਥਰਿਤਾ, 100
ਡਾਟਾਬੇਸ, 114
    -ਅਧਾਰਤਿ ਔਬਜੈਕਟ, 97
    ਲੌਕਿੰਗ ਰਣਨੀਤੀਆਂ, 101
ਡਕਿਸ਼ਨਰੀਆਂ, 121
ਡਜਿੀਟਲ ਖੇਤਰ, 179
ਡੀਬੱਗਿੰਗ, 207
    ਅਤੇ ਟੈਸਟਿੰਗ, 122
    ਅਤੇ ਸਮੱਸਿਆ ਨਵਿਾਰਣ, 227
ਡੈਸਕਟੌਪ ਕੰਪਿਊਟਰ, 201
ਡੋਹਨ ਅਤੇ ਹੋਰ, 40
ਢਾਂਚਾਗਤ ਡਾਟਾ, 124
ਤਾਪਮਾਨ, 49
ਤੰਤੂ ਨੈੱਟਵਰਕ, 3
ਤੰਤ੍ਰਕਿਾ ਨੈੱਟਵਰਕ, 6
ਥਰੂਪੁੱਟ, 25
ਦੁਹਰਾਉਣ ਵਾਲੀ ਸੁਧਾਰ, 69
ਦੁਹਰਾਓ ਜੁਰਮਾਨੇ, 46
ਦੇਰੀ, 25
ਦ੍ਰਸ਼ਿਟੀ ਇੰਟਰਫੇਸ, 193
ਧੋਖਾਧੜੀ ਦੀ ਪਛਾਣ
    ਸਸਿਟਮ, 89
ਨਤੀਜਾ ਵਆਿਖਆਿਕਾਰ, 132
ਨਗਿਰਾਨੀ
    ਅਤੇ ਚੇਤਾਵਨੀ, 208
    ਅਤੇ ਲੌਗਿੰਗ, 102, 227
    ਮੈਟ੍ਰਕਿਸ, 228
ਨਤਿਾਰਨ ਦੀ ਪ੍ਰਕਰਿਆ, 69
ਨਰਿਦੇਸ਼ ਟਊਿਨਿੰਗ
    ਇੰਸਟ੍ਰਕਟ-ਟਊਿਨਡ ਮਾਡਲ, 47

ਨਰਿਯਾਰਤ ਵਵਿਹਾਰ, 53
ਨਰਿਾਸ਼ਾਵਾਦੀ ਲੌਕਿੰਗ, 101
ਨਰਿੰਤਰ ਏਕੀਕਰਨ ਅਤੇ ਤੈਨਾਤੀ (CI/CD), 240
    ਪਾਈਪਲਾਈਨ, 240
ਨੰਜੀਕਰਣ ਕੀਤੀਆਂ ਉਤਪਾਦ ਸਫਿਾਰਸ਼ਾਂ, 84
ਨੰਜੀਕਰਨ, 174, 201, 205
    ਨੰਜੀ ਫਾਰਮ, 185
    ਨੰਜੀ ਮਾਈਕਰੋਕਾਪੀ, 190
ਨੈਤਕਿਤਾ
    ਪ੍ਰਭਾਵ, 184
ਨੈੱਟਵਰਕ ਕਨੈਕਟੀਵਟਿੀ, 208
ਪਹਲਿੇ ਟੋਕਨ ਤੱਕ ਦਾ ਸਮਾਂ (TTFT), 25
ਪਹੁੰਚਯੋਗਤਾ, 200
ਪਾਈਟੌਰਚ, 22
ਪੁਨਰ-ਪ੍ਰਾਪਤੀ ਵਧਾਇਆ ਉਤਪਾਦਨ (RAG), 116
ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਆਧਾਰਤਿ ਮਾਡਲ, 6
ਪੁਨਰ-ਪ੍ਰਾਪਤੀ-ਵਧਾਇਆ ਉਤਪਾਦਨ (RAG), 42
ਪੈਟਰਨ ਮੈਚਿੰਗ, 141
ਪੈਰਾਫਰੇਜ਼ਿੰਗ, 48
ਪੈਰਾਮੀਟਰ
    ਪੈਰਾਮੀਟਰ ਗਣਿਤੀ, 26
    ਪ੍ਰਭਾਵ, 119
    ਰੇਂਜ, 10
ਪ੍ਰਕਾਸ਼ਤਿ-ਗਾਹਕੀ ਪ੍ਰਣਾਲੀਆਂ, 100
ਪ੍ਰਦਰਸ਼ਨ
    ਅਨੁਕੂਲਨ, 181
ਪ੍ਰਯੋਗ
    ਢਾਂਚਾ, 179
ਪ੍ਰਸੰਗ
    ਪ੍ਰਸੰਗਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ, 173, 183
ਪ੍ਰੇਰਕ ਰਣਨੀਤੀਆਂ, 197
ਪ੍ਰੋਸੈਸ ਮੈਨੇਜਰ, 96
    ਐਂਟਰਪ੍ਰਾਈਜ਼ ਏਕੀਕਰਣ, 211
ਪ੍ਰੋਸੈਸਿੰਗ ਸਮਾਂ, 102
ਪ੍ਰੌਮਪਟ
    ਚੇਨਿੰਗ, 65
ਪ੍ਰੌਮਪਟਸ
    ਇੰਜੀਨੀਅਰਿੰਗ, 37, 41, 42, 54, 59, 61, 198
    ਚੇਨਿੰਗ, 54
    ਡਜ਼ਿਾਈਨ, 53, 62
    ਪ੍ਰੌਮਪਟ ਔਬਜੈਕਟ, 68
    ਪ੍ਰੌਮਪਟ ਟੈਂਪਲੇਟ, 54, 189
    ਪ੍ਰੌਮਪਟ ਡਸਿਟੀਲੇਸ਼ਨ, 67, 230
    ਪ੍ਰੌਮਪਟ ਨਖਿਾਰ, 71
    ਪ੍ਰੌਮਪਟ ਨਤਿਾਰਨਾ, 42
    ਸੁਧਾਰ, 62
ਪੱਖਪਾਤ
    ਅਤੇ ਏ.ਆਈ. ਵਚਿ ਨਰਿਪੱਖਤਾ, 237
ਫਾਈਨ-ਟਊਿਨਿੰਗ, 73
ਫਾਈਨਲਾਈਜ਼ ਮੈਥਡ, 145
ਫਾਲਬੈਕ ਰਣਨੀਤੀਆਂ, 101
ਫਊਿ-ਸ਼ਾਟ
    ਪ੍ਰੌਮਪਟਿੰਗ, 57
    ਲਰਨਿੰਗ, 56
ਫੀਡਬੈਕ
    ਫੀਡਬੈਕ ਲੂਪ, 54
ਫੇਸਬੁੱਕ, 22
ਫੈਸਲਾ
    -ਲੈਣ ਦੇ ਮਾਮਲੇ, 123
    ਰੁੱਖ, 204
ਫੈਸਲੇ
    ਬਦ੍ਧਿ, 225
ਫੰਕਸ਼ਨ

ਕਾਲ ਇਤਿਹਾਸ, 145
ਕਾਲਿੰਗ, 114, 146
ਨਾਂ, 143
ਫੰਕਸ਼ਨ ਕਾਲ ਅਸਫਲਤਾ, 124
ਫੰਕਸ਼ਨਲ ਪ੍ਰੋਗਰਾਮਿੰਗ, 84
ਬਰਟ, 22
ਬਹੁ-ਪੜਾਵੀ ਕਾਰਜ-ਪ੍ਰਵਾਹ, 102
ਬਹੁ-ਮਾਧਅਿਮ
ਮਾਡਲ, 18
ਬਹੁ-ਮਾਧਅਿਮੀ
ਭਾਸ਼ਾ ਮਾਡਲ, 19
ਬਹੁਮਤ ਵੋਟਿੰਗ, 108
ਬਾਈਟ ਪੇਅਰ ਇਨਕੋਡਿੰਗ (BPE), 12
ਬਾਹਰੀ ਸੇਵਾਵਾਂ ਜਾਂ ਏ.ਪੀ.ਆਈ., 117
ਬੀਮਾ ਤਸਦੀਕ, 93
ਬੁੱਧੀਮਾਨ ਕਾਰਜ-ਪ੍ਰਵਾਹ ਆਯੋਜਨ, 210
ਬੁੱਧੀਮਾਨ ਕੰਟੈਂਟ ਮੌਡਰੇਟਰ, 214
ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਆਰਕੈਸਟ੍ਰੇਸ਼ਨ, 231
ਬੁੱਧੀਮਾਨ ਵਰਕਫਲੋ ਔਰਕੈਸਟ੍ਰੇਸ਼ਨ, 203
ਬੇਸ ਮਾਡਲ, 49
ਬੈਚ ਪ੍ਰੋਸੈਸਿੰਗ, 230
ਬੰਦ ਅਤੇ ਖੁੱਲ੍ਹੇ ਸਵਾਲਾਂ ਦੇ ਜਵਾਬ, 47
ਭਵਿੱਖਬਾਣੀਆਂ, 5
ਭਾਵਨਾ ਵਸ਼ਿਲੇਸ਼ਣ, 15, 92, 103–105, 108, 109, 125
ਭਾਵਨਾਤਮਕ ਵਸ਼ਿਲੇਸ਼ਣ, 135
ਭਾਵਨਾਤਮਕ ਸੁਰ, 135
ਭਾਸ਼ਾ
-ਸੰਬੰਧਤਿ ਕੰਮ, 4
ਭਾਸ਼ਾ ਪਛਾਣ, 103
ਮਾਡਲ, 39, 60
ਮੌਡਲ, 67
ਮਨ ਦਾ ਸਧਿਾਂਤ, 37
ਮਨੁੱਖੀ-ਸੰਚਾਲਤਿ ਪ੍ਰਕਰਿਆ (HITL), 166
ਮਨੁੱਖੀਕਰਨ, 63
ਮਰਕਰੀ (ਗ੍ਰਹਿ), 41
ਮਰਕਰੀ (ਤੱਤ), 41
ਮਰਕਰੀ (ਰੋਮਨ ਦੇਵਤਾ), 41
ਮਾਈਕਰੋਸਰਵਸਿਜ਼ ਆਰਕੀਟੈਕਚਰ, 82
ਮਾਰਕਅੱਪ-ਸ਼ੈਲੀ ਟੈਗਿੰਗ, 65
ਮਾਰਕਡਾਊਨ, 137
ਮਸਿਤਰਾਲ, 23
ਮੁੜ-ਕੋਸ਼ਸ਼ ਮੈਕੇਨਜ਼ਿਮ, 101
ਮੁੱਖ ਪੈਟਰਨ, 205
ਮੁੱਖ ਮੈਟ੍ਰਕਿਸ ਟਰੈਕਿੰਗ, 225
ਮੇਟਾ, 22
ਮੈਡੀਕਲ ਇਤਿਹਾਸ ਇਕੱਤਰੀਕਰਨ, 93
ਮੈਡੀਕਲ ਖੋਜਾਂ, 92
ਮੈਨੂਅਲ ਦਖਲਅੰਦਾਜ਼ੀ, 210
ਮੈਨੇਜਡ ਸਟ੍ਰੀਮਿੰਗ ਫਾਰ ਅਪਾਚੇ ਕਾਫ਼ਕਾ, 38
ਮੈਮੋਰੀਅਲ ਸਲੋਨ ਕੈਟਰਿੰਗ ਕੈਂਸਰ ਸੈਂਟਰ, 38
ਮੋਡੀਊਲੈਰਟਿੀ, 81
ਮੌਜੂਦਗੀ ਪੈਨਲਟੀ, 44
ਯੂਜ਼ਰ ਅਨੁਭਵ, 179
ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ (UI)
ਡਜ਼ਾਈਨ, 201
ਤਕਨਾਲੋਜੀਆਂ, 193
ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ (ਯੂਆਈ)
ਇੰਟਰਫੇਸ, 183, 197
ਫਰੇਮਵਰਕ, 197
ਯੂਜ਼ਰ ਟੈਸਟਿੰਗ ਅਤੇ ਫੀਡਬੈਕ, 182
ਯੂਜ਼ਰ ਭਰੋਸਾ, 200
ਯੂਜ਼ਰ ਮਨੋਵਗਿਆਨ, 198
ਰਚਨਾਤਮਕ ਲੇਖਣ, 31, 47

ਰਾਹ ਨੂੰ ਤੰਗ ਕਰਨਾ, 36
ਰਾਹ ਨੂੰ ਸੀਮਤ ਕਰਨਾ, 35
ਰਟਿਰੀਵਲ ਔਗਮੈਂਟਡ ਜਨਰੇਸ਼ਨ (ਆਰ.ਏ.ਜੀ.), 73
ਰਟ੍ਰਿੀਵਲ ਔਗਮੈਂਟਡ ਜਨਰੇਸ਼ਨ (RAG), 35
ਰਸਿਪਾਂਸ ਫੈਸਾਂਗਿ, 163, 189
ਰੁਕਾਵਟਾਂ, 207
ਰੇਖਕਿ ਪ੍ਰਤੀਗਮਨ, 40
ਰੇਖਕਿ ਬੀਜਗਣਤਿ, 39
ਰੈਂਕਰ, 32
ਰੋਲ-ਪਲੇਅ-ਸ਼ੈਲੀ ਦੀਆਂ ਗੱਲਬਾਤਾਂ, 6
ਰੋਲਬੈਕ ਮਕੈਨਜ਼ਿਮ, 240
ਲਗਾਤਾਰ ਜੋਖਮ ਨਗਿਰਾਨੀ, 95
ਲਗਾਤਾਰ ਸੁਧਾਰ, 134
ਲਚਕਤਾ ਅਤੇ ਰਚਨਾਤਮਕਤਾ, 181
ਲਾਰਜ ਲੈਂਗਵੇਜ ਮਾਡਲ (LLM), 136
ਲਾਰਜ ਲੈਂਗੁਏਜ ਮਾਡਲ (LLM), 102
ਲੇਖਾ-ਪੜਤਾਲ ਅਤੇ ਪਾਲਣਾ, 227
ਲੇਟੈਂਟ ਸਪੇਸ, 39
ਲੌਂਗ ਰੀਟੈਂਸ਼ਨ ਅਤੇ ਰੋਟੇਸ਼ਨ, 228
ਲੱਛਣ ਮੁਲਾਂਕਣ ਅਤੇ ਵਰਗੀਕਰਨ, 93
ਵਧਾਈ ਹੋਈ ਅਸਲੀਅਤ ਐਨਕਾਂ, 201
ਵਨ-ਸ਼ਾਟ ਲਰਨਿੰਗ, 55
ਵਪਾਰਕ ਨਯਿਮ, 203
ਵਰਕਰਾਂ ਦੀ ਬਹੁਤਾਤ, 154
ਵਰਗੀਕਰਨ, 47
ਵਰਗੀਕਰਨ ਅਤੇ ਟਾਰਗੇਟਿੰਗ ਰਣਨੀਤੀਆਂ, 179
ਵਰਚੁਅਲ ਸਹਾਇਕ, 30
ਵਰਤੋਂਯੋਗਤਾ ਸਮੱਸਿਆਵਾਂ, 199
ਵਿਆਕਰਣ ਦੇ ਨਯਿਮ, 4
ਵਿਆਖਿਆਯੋਗਤਾ, 237
ਵਿਕਾਸ ਫਰੇਮਵਰਕ, 138
ਵਿਚਾਰਾਂ ਦੀ ਲੜੀ (CoT), 41, 129
ਵਿਸਤ੍ਰਤਿ ਲੌਂਗਾਂਗਿ, 228
ਵੀਂਦਿਅਕ ਐਪਲੀਕੇਸ਼ਨਾਂ, 29
ਵੈਟਰੀਲੋਕੁਇਸਟ, 163
ਵੰਡੀ ਹੋਈ ਆਰਕੀਟੈਕਚਰ, 230
ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ (LLM), 3, 16, 61, 62, 65, 69, 114, 115, 124, 130, 134, 155, 193, 213
ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ (ਐਲ.ਐਲ.ਐਮ.), 134
ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ (ਐੱਲ.ਐੱਲ.ਐੱਮ.), 80, 152
ਵੱਡਾ ਭਾਸ਼ਾ ਮਾਡਲ (ਐੱਲਐੱਲਐੱਮ), 14, 27, 71, 188
    ਖੇਤਰ, 25
ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ (LLM), 1, 173
ਵੱਡੇ ਭਾਸ਼ਾ ਮਾਡਲ (ਐੱਲਐੱਲਐੱਮ), 183
ਸਕੇਲੇਬਲਿਟੀ, 205, 229
ਸਟਰਕਚਰਡ IO, 189
ਸਟੇਜਿੰਗ ਵਾਤਾਵਰਣ, 240
ਸਟੇਟਲੈੱਸ, 145
ਸਟ੍ਰੀਮ ਪ੍ਰੋਸੈਸਿੰਗ, 139, 145, 151
    ਲੌਜਕਿ, 147
ਸਟ੍ਰੀਮ ਹੈਂਡਲਰ, 139
ਸਟ੍ਰੀਮਿੰਗ ਡਾਟਾ, 141
ਸਥਾਨਕ ਵਿਕਾਸ ਵਾਤਾਵਰਣ, 143
ਸਪਲਾਈ ਚੇਨ
    ਆਪਟੀਮਾਈਜ਼ੇਸ਼ਨ, 30
ਸਮਾਨੰਤਰ ਐਗਜ਼ੀਕਿਊਸ਼ਨ, 230
ਸਮਾਰਟਫੋਨ, 201
ਸਮਾਵੇਸ਼ੀ ਇੰਟਰਫੇਸ, 184
ਸਮੂਹ, 109
    ਵਰਕਰਾਂ ਦਾ ਸਮੂਹ, 109
ਸਮੱਗਰੀ
    ਫਲਿਟਰਿੰਗ, 24

ਸਮੱਗਰੀ ਵਰਗੀਕਰਨ, 103
ਸਮੱਗਰੀ-ਆਧਾਰਤਿ ਫਲਿਟਰਿੰਗ, 84
ਸਰਕਟ ਬਰੇਕਰ ਲੌਜਿਕ, 150
ਸਰਵਰ-ਭੇਜੀਆਂ ਘਟਨਾਵਾਂ (SSE), 139
ਸਵਾਲ-ਜਵਾਬ ਪ੍ਰਣਾਲੀਆਂ, 7
ਸਵੈ-ਠੀਕ ਹੋਣ ਵਾਲਾ ਡਾਟਾ, 152
ਸਵੈ-ਪ੍ਰਤੀਗਾਮੀ ਮਾਡਲਿੰਗ, 40
ਸਹਯੋਗੀ ਫਲਿਟਰਿੰਗ, 84
ਸਾਂਝੇ ਸਰੋਤਾਂ ਦੀ ਤ੍ਰਾਸਦੀ, 177
ਸਾਫਟਵੇਅਰ ਆਰਕੀਟੈਕਚਰ, 2
ਸਖਿਲਾਈ ਡਾਟਾ, 39
ਸਸਿਟਮ ਨਰਿਦੇਸ਼, 119
ਸਟੈਟਿਕ ਗਲਤੀਆਂ, 122
ਸਥਿਟਕਿ ਡੇਟਾ ਜਨਰੇਸ਼ਨ, 48
ਸੀਮਾ ਮਾਮਲੇ, 53
ਸੀਮਾ ਸਥਤਿੀਆਂ, 234
ਸੈਲਫ-ਹੀਲਿੰਗ ਡਾਟਾ, 224
ਸੰਕਲਪਕ ਅਤੇ ਵਵਿਹਾਰਕ ਚੁਣੌਤੀਆਂ, 184
ਸੰਖੇਪੀਕਰਨ, 47
ਸੰਦ ਦੀ ਵਰਤੋਂ, 138
ਸੰਦਰਭ
ਅਸੀਮਤਿ ਲੰਬੀਆਂ ਇਨਪੁੱਟਾਂ, 14
ਵਾਧਾ, 42
ਵਿੰਡੋ, 14, 207
ਸੰਦਰਭਕਿ ਖੇਤਰ ਸੁਝਾਅ, 185
ਸੰਦਰਭਕਿ ਫੈਸਲਾ ਲੈਣਾ, 207
ਸੰਦਰਭਕਿ ਸਮੱਗਰੀ ਨਰਿਮਾਣ, 177, 178, 185
ਸੰਭਾਵੀ ਮਾਡਲ, 39
ਸੰਰਚਤਿ ਲੌਗਿੰਗ, 228
ਹਦਾਇਤ ਸਖਿਲਾਈ
ਹਦਾਇਤ-ਸਖਿਲਾਈ ਮਾਡਲ, 45
ਹਾਈਪਰਪੈਰਾਮੀਟਰ, 43
ਹਾਰਡਵੇਅਰ, 26
ਹਦਿਾਇਤ ਟਊਿਨਿੰਗ, 9
ਹੈਸ਼, 141