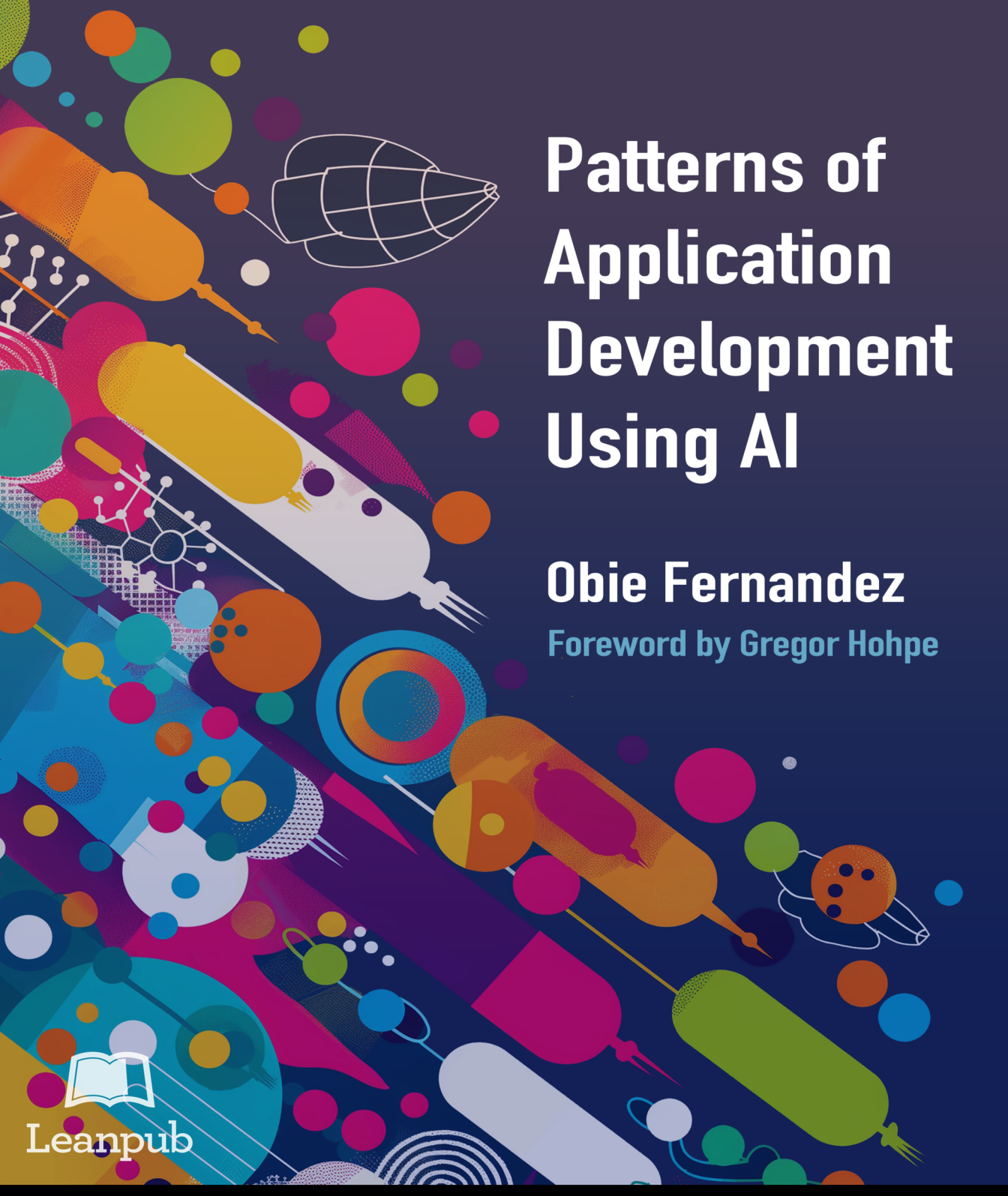

Patterns of Application Development Using AI
Obie Fernandez
Foreword by Gregor Hohpe
Leanpub

हिंदी संस्करण

एआई का उपयोग करके एप्लिकेशन विकास के पैटर्न (हिंदी संस्करण)

Obie Fernandez

यह किताब
http://leanpub.com/patterns-of-application-development-using-ai-hi पर बिक्री के लिए उपलब्ध है।

यह संस्करण 2025-01-23 को प्रकाशित हुआ था

यह एक Leanpub की पुस्तक है। Leanpub लेखकों और प्रकाशकों को लीन प्रकाशन प्रक्रिया द्वारा सक्षम बनाता है। Lean Publishing एक प्रगतिशील ईबुक प्रकाशित करने की क्रिया है जिसमें सरल उपकरणों और अनेक पुनरावृत्तियों के माध्यम से पाठकों से प्रतिक्रिया प्राप्त करने के लिए, पुस्तक को तब तक बदलते रहें जब तक कि आप सही पुस्तक नहीं बना लेते, और एक बार आप ऐसा कर लेते हैं, तो अपनी पुस्तक के प्रचार में सफलता प्राप्त कर सकें।

इस पुस्तक के बारे में ट्वीट करें!

कृपया Obie Fernandez की मदद करें और इस पुस्तक के बारे में Twitter पर शब्द फैलाएं!

इस पुस्तक के लिए सुझाया गया हैशटैग #poaduai है।

ट्विटर पर इस हैशटैग की खोज करके अन्य लोग इस पुस्तक के बारे में क्या कह रहे हैं, यह जानने के लिए इस लिंक पर क्लिक करें:

#poaduai

मेरी दुर्धर्ष रानी, मेरी प्रेरणा, मेरी ज्योति और मेरा प्रेम, विक्टोरिया को

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

विषय सूची

भाग 1: मौलिक दृष्टिकोण और तकनीकें 39

ग्रेगर होप्फ द्वारा प्राक्कथन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रस्तावना

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

पुस्तक के बारे में

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कोड उदाहरणों के बारे में

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

मैं क्या नहीं कवर करता

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह पुस्तक किसके लिए है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

एक सामान्य शब्दावली का निर्माण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-

hi पर।

शामिल होना

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

आभार

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

चित्रों के बारे में क्या है?

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

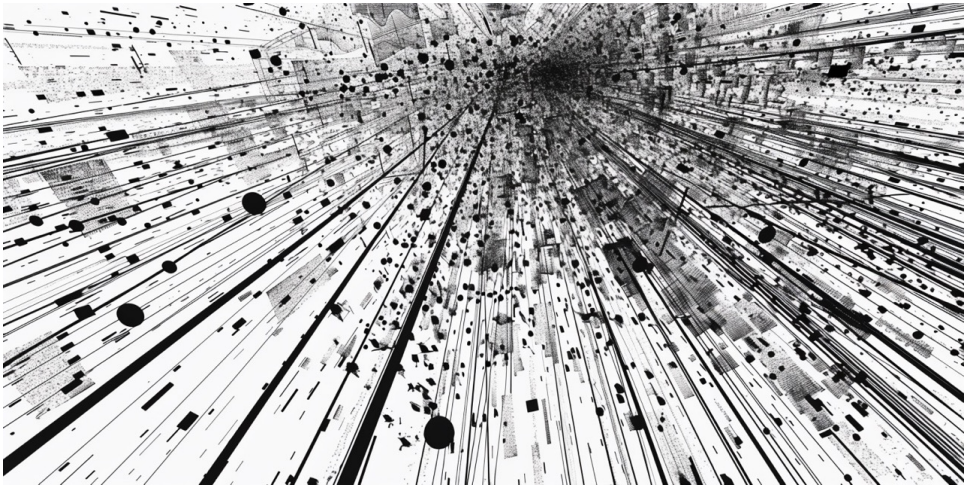
लीन पब्लिशिंग के बारे में

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

लेखक के बारे में

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

परिचय



यदि आप अपनी प्रोग्रामिंग परियोजनाओं में एआई लार्ज लैंग्वेज मॉडल्स (एलएलएम) को एकीकृत करने के लिए उत्सुक हैं, तो बाद के अध्यायों में प्रस्तुत पैटर्न और कोड उदाहरणों में सीधे डूब जाएं। हालांकि, इन पैटर्न की शक्ति और संभावना को पूरी तरह से समझने के लिए, व्यापक संदर्भ और उनके द्वारा प्रस्तुत एकजुट दृष्टिकोण को समझने में कुछ समय लगाना उपयोगी होगा।

ये पैटर्न केवल अलग-अलग तकनीकों का संग्रह नहीं हैं, बल्कि आपके एप्लिकेशन में एआई को एकीकृत करने के लिए एक एकीकृत फ्रेमवर्क हैं। मैं Ruby on Rails का उपयोग करता हूं, लेकिन ये पैटर्न लगभग किसी भी अन्य प्रोग्रामिंग वातावरण में काम करने चाहिए। ये डेटा प्रबंधन और प्रदर्शन अनुकूलन से लेकर यूजर एक्सपीरियंस और सुरक्षा तक, कई चिंताओं को संबोधित करते हैं, जो एआई की क्षमताओं के साथ पारंपरिक प्रोग्रामिंग प्रथाओं को बढ़ाने के लिए एक व्यापक टूलकिट प्रदान करते हैं।

पैटर्न की प्रत्येक श्रेणी एक विशिष्ट चुनौती या अवसर को संबोधित करती है जो आपके

एप्लिकेशन में एआई घटकों को शामिल करते समय उत्पन्न होती है। इन पैटर्न के बीच संबंधों और तालमेल को समझकर, आप एआई को सबसे प्रभावी ढंग से कहां और कैसे लागू करना है, इस बारे में सूचित निर्णय ले सकते हैं।

पैटर्न कभी भी निर्देशात्मक समाधान नहीं होते और उन्हें ऐसा नहीं माना जाना चाहिए। वे अनुकूलन योग्य बिल्डिंग ब्लॉक होने के लिए बनाए गए हैं जिन्हें आपके अपने विशिष्ट एप्लिकेशन की अनूठी आवश्यकताओं और सीमाओं के अनुरूप बनाया जाना चाहिए। इन पैटर्न का सफल अनुप्रयोग (सॉफ्टवेयर क्षेत्र में किसी अन्य की तरह) समस्या डोमेन, उपयोगकर्ता की जरूरतों और आपकी परियोजना की समग्र तकनीकी आर्किटेक्चर की गहरी समझ पर निर्भर करता है।

सॉफ्टवेयर आर्किटेक्चर पर विचार

मैंने 1980 के दशक में प्रोग्रामिंग शुरू की और हैकर समुदाय में शामिल था, और पेशेवर सॉफ्टवेयर डेवलपर बनने के बाद भी मैंने अपनी हैकर मानसिकता को कभी नहीं खोया। शुरू से ही, मुझे हमेशा इस बात पर स्वस्थ संदेह था कि अपने आइवरी टावर में बैठे सॉफ्टवेयर आर्किटेक्ट्स वास्तव में क्या मूल्य लाते हैं।

एआई तकनीक की इस शक्तिशाली नई लहर द्वारा लाए गए परिवर्तनों के बारे में मैं व्यक्तिगत रूप से इतना उत्साहित होने का एक कारण यह है कि यह सॉफ्टवेयर आर्किटेक्चर निर्णयों पर क्या प्रभाव डालता है। यह हमारी सॉफ्टवेयर परियोजनाओं को डिजाइन और कार्यान्वित करने के "सही" तरीके की पारंपरिक धारणाओं को चुनौती देता है। यह इस बात को भी चुनौती देता है कि क्या आर्किटेक्चर को अभी भी मुख्य रूप से सिस्टम के उन हिस्सों के रूप में सोचा जा सकता है जिन्हें बदलना मुश्किल है, क्योंकि एआई एन्हांसमेंट किसी भी समय आपकी परियोजना के किसी भी हिस्से को बदलना पहले से कहीं अधिक आसान बना रहा है।

शायद हम सॉफ्टवेयर इंजीनियरिंग के "उत्तर-आधुनिक" दृष्टिकोण के शीर्ष वर्षों में प्रवेश कर रहे हैं। इस संदर्भ में, उत्तर-आधुनिक का अर्थ है परंपरागत प्रतिमानों से एक मौलिक बदलाव, जहाँ डेवलपर्स हर कोड की पंक्ति को लिखने और उसके रखरखाव के लिए जिम्मेदार थे। इसके बजाय, यह डेटा में हेरफेर, जटिल एल्गोरिथ्म, और यहां तक

कि एप्लिकेशन लॉजिक के पूरे हिस्सों को थर्ड-पार्टी लाइब्रेरी और बाहरी एपीआई को सौंपने के विचार को अपनाता है। यह उत्तर-आधुनिक बदलाव एप्लिकेशन को शुरू से बनाने की पारंपरिक सोच से एक महत्वपूर्ण प्रस्थान का प्रतीक है, और यह डेवलपर्स को विकास प्रक्रिया में अपनी भूमिका पर पुनर्विचार करने की चुनौती देता है।

मैं हमेशा से मानता आया हूं कि अच्छे प्रोग्रामर केवल वही कोड लिखते हैं जो बिल्कुल आवश्यक है, जो Larry Wall और उनके जैसे अन्य हैकर दिग्गजों की शिक्षाओं पर आधारित है। लिखे गए कोड की मात्रा को कम करके, हम तेजी से आगे बढ़ सकते हैं, बग्स के लिए सतह क्षेत्र को कम कर सकते हैं, रखरखाव को सरल बना सकते हैं, और अपने एप्लिकेशन की समग्र विश्वसनीयता में सुधार कर सकते हैं। कम कोड हमें मुख्य व्यावसायिक तर्क और उपयोगकर्ता अनुभव पर ध्यान केंद्रित करने की अनुमति देता है, जबकि अन्य कार्यों को अन्य सेवाओं को सौंप दिया जाता है।

अब जबकि AI-संचालित सिस्टम उन कार्यों को संभाल सकते हैं जो पहले केवल मानव-लिखित कोड का क्षेत्र था, हमें और भी अधिक उत्पादक और चुस्त होने में सक्षम होना चाहिए, व्यावसायिक मूल्य और उपयोगकर्ता अनुभव बनाने पर पहले से कहीं अधिक ध्यान केंद्रित कर सकते हैं।

बेशक AI सिस्टम को अपनी परियोजना के बड़े हिस्सों को सौंपने के नुकसान भी हैं, जैसे नियंत्रण की संभावित हानि, और मजबूत निगरानी और प्रतिक्रिया तंत्र की आवश्यकता। इसीलिए इसके लिए कौशल और ज्ञान के एक नए सेट की आवश्यकता होती है, जिसमें AI कैसे काम करता है, इसकी कम से कम कुछ मौलिक समझ शामिल है।

बृहत भाषा मॉडल क्या है?

बृहत भाषा मॉडल (LLMs) कृत्रिम बुद्धिमत्ता के एक प्रकार के मॉडल हैं जिन्होंने हाल के वर्षों में काफी ध्यान आकर्षित किया है, खासकर 2020 में OpenAI द्वारा GPT-3 के लॉन्च के बाद से। LLMs को उल्लेखनीय सटीकता और धाराप्रवाह के साथ मानव भाषा को प्रोसेस करने, समझने और उत्पन्न करने के लिए डिज़ाइन किया गया है। इस

खंड में, हम संक्षेप में देखेंगे कि LLMs कैसे काम करते हैं और वे बुद्धिमान सिस्टम घटकों के निर्माण के लिए क्यों उपयुक्त हैं।

मूल रूप से, LLMs डीप लर्निंग एल्गोरिथ्म पर आधारित हैं, विशेष रूप से तंत्रिका नेटवर्क पर। ये नेटवर्क परस्पर जुड़े नोड्स या न्यूरॉन्स से बने होते हैं, जो सूचना को प्रोसेस और प्रेषित करते हैं। LLMs के लिए पसंद की जाने वाली आर्किटेक्चर अक्सर ट्रांसफॉर्मर मॉडल होता है, जो टेक्स्ट जैसे क्रमिक डेटा को संभालने में अत्यंत प्रभावी सिद्ध हुआ है।

Transformer मॉडल ध्यान तंत्र पर आधारित हैं और मुख्य रूप से अनुक्रमिक डेटा से जुड़े कार्यों के लिए उपयोग किए जाते हैं, जैसे प्राकृतिक भाषा प्रसंस्करण। Transformer इनपुट डेटा को क्रमिक रूप से नहीं बल्कि एक साथ प्रोसेस करते हैं, जो उन्हें दूरगामी निर्भरताओं को अधिक प्रभावी ढंग से पकड़ने में सक्षम बनाता है। इनमें ध्यान तंत्र की परतें होती हैं जो मॉडल को संदर्भ और संबंधों को समझने के लिए इनपुट डेटा के विभिन्न हिस्सों पर ध्यान केंद्रित करने में मदद करती हैं।

LLMs के लिए प्रशिक्षण प्रक्रिया में मॉडल को विशाल मात्रा में पाठ्य डेटा जैसे किताबें, लेख, वेबसाइट और कोड रिपॉजिटरी के संपर्क में लाना शामिल है। प्रशिक्षण के दौरान, मॉडल पाठ के भीतर पैटर्न, संबंध और संरचनाओं को पहचानना सीखता है। यह भाषा के सांख्यिकीय गुणों को समझता है, जैसे व्याकरण नियम, शब्द संबंध और संदर्भगत अर्थ।

LLMs के प्रशिक्षण में उपयोग की जाने वाली प्रमुख तकनीकों में से एक है पर्यवेक्षण रहित सीखना। इसका मतलब है कि मॉडल स्पष्ट लेबलिंग या मार्गदर्शन के बिना डेटा से सीखता है। यह प्रशिक्षण डेटा में शब्दों और वाक्यांशों की सह-उपस्थिति का विश्लेषण करके स्वयं पैटर्न और प्रतिनिधित्व खोजता है। यह LLMs को भाषा और उसकी जटिलताओं की गहरी समझ विकसित करने की अनुमति देता है।

LLMs का एक और महत्वपूर्ण पहलू है उनकी संदर्भ को संभालने की क्षमता। पाठ को प्रोसेस करते समय, LLMs न केवल व्यक्तिगत शब्दों को बल्कि आस-पास के संदर्भ को भी ध्यान में रखते हैं। वे पाठ के अर्थ और इरादे को समझने के लिए पिछले शब्दों, वाक्यों और यहां तक कि पैराग्राफ को भी ध्यान में रखते हैं। यह संदर्भगत समझ LLMs को सुसंगत और प्रासंगिक प्रतिक्रियाएं उत्पन्न करने में सक्षम बनाती है। किसी

दिए गए LLM मॉडल की क्षमताओं का मूल्यांकन करने के मुख्य तरीकों में से एक यह है कि वे प्रतिक्रियाएं उत्पन्न करने के लिए कितने बड़े संदर्भ पर विचार कर सकते हैं।

एक बार प्रशिक्षित होने के बाद, LLMs को भाषा से संबंधित कई कार्यों के लिए उपयोग किया जा सकता है। वे मानव जैसा पाठ उत्पन्न कर सकते हैं, प्रश्नों के उत्तर दे सकते हैं, दस्तावेजों का सारांश बना सकते हैं, भाषाओं का अनुवाद कर सकते हैं, और यहां तक कि कोड भी लिख सकते हैं। LLMs की बहुमुखी प्रतिभा उन्हें बुद्धिमान सिस्टम घटक बनाने के लिए मूल्यवान बनाती है जो उपयोगकर्ताओं के साथ बातचीत कर सकते हैं, पाठ डेटा को प्रोसेस और विश्लेषण कर सकते हैं, और सार्थक आउटपुट प्रदान कर सकते हैं।

एप्लिकेशन वास्तुकला में LLMs को शामिल करके, आप ऐसे AI घटक बना सकते हैं जो उपयोगकर्ता इनपुट को समझते और प्रोसेस करते हैं, गतिशील सामग्री उत्पन्न करते हैं, और बुद्धिमान सिफारिशें या कार्रवाइयां प्रदान करते हैं। लेकिन LLMs के साथ काम करने के लिए संसाधन आवश्यकताओं और प्रदर्शन विनिमय पर सावधानीपूर्वक विचार करने की आवश्यकता होती है। LLMs कम्प्यूटेशनल रूप से गहन हैं और संचालन के लिए महत्वपूर्ण प्रोसेसिंग पावर और मेमोरी (दूसरे शब्दों में, धन) की आवश्यकता हो सकती है। हम में से अधिकांश को अपने एप्लिकेशन में LLMs को एकीकृत करने के लागत प्रभावों का आकलन करने और तदनुसार कार्य करने की आवश्यकता होगी।

अनुमान को समझना

अनुमान वह प्रक्रिया है जिसके द्वारा एक मॉडल नए, अनदेखे डेटा के आधार पर भविष्यवाणियां या परिणाम उत्पन्न करता है। यह वह चरण है जहां प्रशिक्षित मॉडल का उपयोग उपयोगकर्ता के इनपुट के जवाब में निर्णय लेने या टेक्स्ट, छवियां, या अन्य सामग्री उत्पन्न करने के लिए किया जाता है।

प्रशिक्षण चरण के दौरान, एक AI मॉडल अपनी भविष्यवाणियों में त्रुटि को कम करने के लिए अपने पैरामीटर्स को समायोजित करके एक बड़े डेटासेट से सीखता है। एक बार प्रशिक्षित होने के बाद, मॉडल नए डेटा पर अपनी सीखी हुई बातों को लागू कर

सकता है। अनुमान वह तरीका है जिससे मॉडल आउटपुट उत्पन्न करने के लिए अपने सीखे हुए पैटर्न और ज्ञान का उपयोग करता है।

LLMs के लिए, अनुमान में एक प्रॉम्प्ट या इनपुट टेक्स्ट को लेकर एक सुसंगत और संदर्भगत प्रासंगिक प्रतिक्रिया उत्पन्न करना शामिल है, जो टोकन की धारा के रूप में होता है (जिसके बारे में हम जल्द ही बात करेंगे)। यह एक प्रश्न का उत्तर देना, एक वाक्य को पूरा करना, एक कहानी बनाना, या टेक्स्ट का अनुवाद करना हो सकता है, और भी कई अन्य कार्य।

आप और मेरी सोच के तरीके के विपरीत, एक AI मॉडल का "सोचना" अनुमान के माध्यम से एक ही स्थितिहीन संचालन में होता है। यानी, इसकी सोच इसकी उत्पादन प्रक्रिया तक ही सीमित है। इसे वास्तव में जोर से सोचना पड़ता है, जैसे कि मैंने आपसे एक सवाल पूछा और केवल "चेतना की धारा" शैली में आपसे प्रतिक्रिया स्वीकार की।

बृहत भाषा मॉडल कई आकारों और प्रकारों में आते हैं

जबकि लगभग सभी लोकप्रिय बृहत भाषा मॉडल (LLMs) एक ही मूल ट्रांसफॉर्मर आर्किटेक्चर पर आधारित हैं और विशाल टेक्स्ट डेटासेट पर प्रशिक्षित हैं, वे विभिन्न आकारों में आते हैं और अलग-अलग उद्देश्यों के लिए फाइन-ट्यून किए जाते हैं। एक LLM का आकार, जो इसके तंत्रिका नेटवर्क में पैरामीटर्स की संख्या से मापा जाता है, इसकी क्षमताओं पर बड़ा प्रभाव डालता है। अधिक पैरामीटर्स वाले बड़े मॉडल, जैसे GPT-4, जिसमें कथित तौर पर 1 से 2 ट्रिलियन पैरामीटर्स हैं, आमतौर पर छोटे मॉडलों की तुलना में अधिक ज्ञानवान और सक्षम होते हैं। हालांकि, बड़े मॉडलों को चलाने के लिए बहुत अधिक कंप्यूटिंग पावर की आवश्यकता होती है, जो API कॉल के माध्यम से उनका उपयोग करते समय अधिक खर्च में तब्दील हो जाती है।

LLMs को अधिक व्यावहारिक और विशिष्ट उपयोग के मामलों के लिए अनुकूलित करने के लिए, बेस मॉडलों को अक्सर अधिक लक्षित डेटासेट पर फाइन-ट्यून किया जाता है। उदाहरण के लिए, एक LLM को संवादात्मक AI के लिए विशेष बनाने के

लिए संवाद के बड़े संग्रह पर प्रशिक्षित किया जा सकता है। अन्य को प्रोग्रामिंग ज्ञान प्रदान करने के लिए कोड पर प्रशिक्षित किया जाता है। यहां तक कि कुछ मॉडल उपयोगकर्ताओं के साथ रोलप्ले-शैली की बातचीत के लिए विशेष रूप से प्रशिक्षित किए जाते हैं!

पुनर्प्राप्ति बनाम जनरेटिव मॉडल

बृहत भाषा मॉडल (LLMs) की दुनिया में, प्रतिक्रियाएँ उत्पन्न करने के दो मुख्य दृष्टिकोण हैं: पुनर्प्राप्ति-आधारित मॉडल और जनरेटिव मॉडल। प्रत्येक दृष्टिकोण की अपनी विशेष शक्तियाँ और कमजोरियाँ हैं, और इनके बीच के अंतरों को समझना आपको अपने विशिष्ट उपयोग के लिए सही मॉडल चुनने में मदद कर सकता है।

पुनर्प्राप्ति-आधारित मॉडल

पुनर्प्राप्ति-आधारित मॉडल, जिन्हें सूचना पुनर्प्राप्ति मॉडल के रूप में भी जाना जाता है, पहले से मौजूद टेक्स्ट के बड़े डेटाबेस में खोजकर और इनपुट क्वेरी के आधार पर सबसे प्रासंगिक अंशों का चयन करके प्रतिक्रियाएँ उत्पन्न करते हैं। ये मॉडल नया टेक्स्ट शून्य से नहीं बनाते, बल्कि एक सुसंगत प्रतिक्रिया बनाने के लिए डेटाबेस से अंशों को जोड़ते हैं।

पुनर्प्राप्ति-आधारित मॉडल का एक मुख्य लाभ तथ्यात्मक रूप से सटीक और अद्यतन जानकारी प्रदान करने की उनकी क्षमता है। चूंकि वे क्यूरेटेड टेक्स्ट के डेटाबेस पर निर्भर करते हैं, वे विश्वसनीय स्रोतों से प्रासंगिक जानकारी प्राप्त कर सकते हैं और उपयोगकर्ता को प्रस्तुत कर सकते हैं। यह उन्हें प्रश्न-उत्तर प्रणालियों या ज्ञान आधार जैसे अनुप्रयोगों के लिए उपयुक्त बनाता है।

हालांकि, पुनर्प्राप्ति-आधारित मॉडल की कुछ सीमाएं हैं। वे केवल उतने ही अच्छे होते हैं जितना कि वह डेटाबेस जिसमें वे खोज रहे हैं, इसलिए डेटाबेस की गुणवत्ता और कवरेज सीधे मॉडल के प्रदर्शन को प्रभावित करती है। इसके अतिरिक्त, ये मॉडल सुसंगत और प्राकृतिक लगने वाली प्रतिक्रियाएँ उत्पन्न करने में संघर्ष कर सकते हैं, क्योंकि वे डेटाबेस में उपलब्ध टेक्स्ट तक ही सीमित होते हैं।

इस पुस्तक में हम शुद्ध पुनर्प्राप्ति मॉडल के उपयोग को नहीं कवर करते हैं।

जनरेटिव मॉडल

दूसरी ओर, जनरेटिव मॉडल प्रशिक्षण के दौरान सीखे गए पैटर्न और संबंधों के आधार पर नया टेक्स्ट शून्य से बनाते हैं। ये मॉडल इनपुट प्रॉम्प्ट के अनुरूप नई प्रतिक्रियाएँ उत्पन्न करने के लिए भाषा की अपनी समझ का उपयोग करते हैं।

जनरेटिव मॉडल की मुख्य शक्ति रचनात्मक, सुसंगत और संदर्भगत रूप से प्रासंगिक टेक्स्ट उत्पन्न करने की उनकी क्षमता है। वे खुली बातचीत कर सकते हैं, कहानियाँ बना सकते हैं, और यहां तक कि कोड भी लिख सकते हैं। यह उन्हें चैटबोट, कंटेंट क्रिएशन और रचनात्मक लेखन सहायक जैसे अधिक खुले और गतिशील इंटरैक्शन वाले अनुप्रयोगों के लिए आदर्श बनाता है।

हालांकि, जनरेटिव मॉडल कभी-कभी असंगत या तथ्यात्मक रूप से गलत जानकारी उत्पन्न कर सकते हैं, क्योंकि वे तथ्यों के क्यूरेटेड डेटाबेस के बजाय प्रशिक्षण के दौरान सीखे गए पैटर्न पर निर्भर करते हैं। वे पूर्वाग्रहों और भ्रांतियों के प्रति भी अधिक संवेदनशील हो सकते हैं, जो ऐसा टेक्स्ट उत्पन्न करते हैं जो विश्वसनीय लगता है लेकिन जरूरी नहीं कि सत्य हो।

जनरेटिव एलएलएम के उदाहरणों में OpenAI की जीपीटी श्रृंखला (जीपीटी-3, जीपीटी-4) और Anthropic का Claude शामिल हैं।

हाइब्रिड मॉडल

कई व्यावसायिक रूप से उपलब्ध एलएलएम एक हाइब्रिड मॉडल में पुनर्प्राप्ति और जनरेटिव दोनों दृष्टिकोणों को जोड़ते हैं। ये मॉडल डेटाबेस से प्रासंगिक जानकारी खोजने के लिए पुनर्प्राप्ति तकनीकों का उपयोग करते हैं और फिर उस जानकारी को एक सुसंगत प्रतिक्रिया में संश्लेषित करने के लिए जनरेटिव तकनीकों का उपयोग करते हैं।

हाइब्रिड मॉडल का उद्देश्य पुनर्प्राप्ति-आधारित मॉडल की तथ्यात्मक सटीकता को जनरेटिव मॉडल की प्राकृतिक भाषा उत्पादन क्षमताओं के साथ जोड़ना है। वे अधिक विश्वसनीय और अद्यतित जानकारी प्रदान कर सकते हैं, जबकि खुली बातचीत में संलग्न होने की क्षमता को भी बनाए रखते हैं।

पुनर्प्राप्ति-आधारित और जनरेटिव मॉडल के बीच चयन करते समय, आपको अपने एप्लिकेशन की विशिष्ट आवश्यकताओं पर विचार करना चाहिए। यदि प्राथमिक लक्ष्य सटीक, तथ्यात्मक जानकारी प्रदान करना है, तो पुनर्प्राप्ति-आधारित मॉडल सबसे अच्छा विकल्प हो सकता है। यदि एप्लिकेशन को अधिक खुली और रचनात्मक बातचीत की आवश्यकता है, तो जनरेटिव मॉडल अधिक उपयुक्त हो सकता है। हाइब्रिड मॉडल दोनों दृष्टिकोणों के बीच संतुलन प्रदान करते हैं और उन एप्लिकेशन के लिए एक अच्छा विकल्प हो सकते हैं जिन्हें तथ्यात्मक सटीकता और प्राकृतिक भाषा उत्पादन दोनों की आवश्यकता होती है।

अंततः, पुनर्प्राप्ति-आधारित और जनरेटिव मॉडल के बीच चयन विशिष्ट उपयोग मामले और सटीकता, रचनात्मकता और लचीलेपन के बीच समझौते पर निर्भर करता है। प्रत्येक दृष्टिकोण की ताकतों और सीमाओं को समझकर, आप सूचित निर्णय ले सकते हैं।

निर्देश ट्यूनिंग

निर्देश ट्यूनिंग वार्तालाप प्रशिक्षण का एक उपसमुच्चय है जो एलएलएम को निर्देशों को बेहतर ढंग से समझने और उनका पालन करने के लिए मानव-लिखित प्रॉम्प्ट्स और प्रतिक्रियाओं का उपयोग करता है। यहाँ कुछ सरल उदाहरण दिए गए हैं जो दर्शाते हैं कि मानव-लिखित प्रॉम्प्ट्स और प्रतिक्रियाओं पर निर्देश ट्यूनिंग कैसे एलएलएम को निर्देशों को समझने और उनका पालन करने में बेहतर बनाता है:

उदाहरण 1: ग्राहक सहायता

निर्देश: "रिफंड मांगने वाले ग्राहक को प्रतिक्रिया प्रदान करें।"

मानव-लिखित प्रॉम्प्ट: "मैं अपनी हाल की खरीद के लिए रिफंड का अनुरोध करना चाहूंगा। उत्पाद मेरी अपेक्षाओं पर खरा नहीं उतरा।"

मानव-लिखित प्रतिक्रिया: "मुझे खेद है कि उत्पाद आपकी अपेक्षाओं पर खरा नहीं उतरा। आपके रिफंड को प्रोसेस करने के लिए, कृपया अपना ऑर्डर नंबर और वापसी का कारण प्रदान करें। हम आपके अनुरोध की समीक्षा करेंगे और 2-3 कार्य दिवसों के भीतर आपसे संपर्क करेंगे।"

उदाहरण 2: शैक्षिक सहायता

निर्देश: "10 साल के बच्चे को प्रकाश संश्लेषण की अवधारणा समझाएं।"

मानव-लिखित प्रॉम्प्ट: "क्या आप बता सकते हैं कि पौधे अपना भोजन कैसे बनाते हैं?"

मानव-लिखित उत्तर: "ज़रूर! पौधे प्रकाश संश्लेषण नामक प्रक्रिया के माध्यम से अपना भोजन बनाते हैं। वे सूर्य का प्रकाश, पानी और कार्बन डाइऑक्साइड का उपयोग भोजन और ऑक्सीजन बनाने के लिए करते हैं। पौधे की पत्तियों में क्लोरोफिल नामक एक विशेष भाग होता है जो उन्हें सूर्य का प्रकाश पकड़ने में मदद

करता है।"

उदाहरण 3: व्यक्तिगत सहायक

निर्देश: "अगले मंगलवार दोपहर 3 बजे एक मीटिंग शेड्यूल करें।"

मानव-लिखित प्रॉम्प्ट: "क्या आप मंगलवार दोपहर 3 बजे एक मीटिंग सेट कर सकते हैं?"

मानव-लिखित उत्तर: "ज़रूर, मैंने आपके लिए अगले मंगलवार दोपहर 3 बजे एक मीटिंग शेड्यूल कर दी है। क्या आपको कुछ और चाहिए?"

इसका परिणाम विभिन्न आकारों और विभिन्न विशेषज्ञताओं वाले LLMs का एक विविध पारिस्थितिकी तंत्र है। 1-7 बिलियन पैरामीटर की सीमा वाले छोटे मॉडल अच्छी सामान्य भाषा क्षमताएं प्रदान करते हैं जबकि चलाने में अधिक कुशल होते हैं।

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

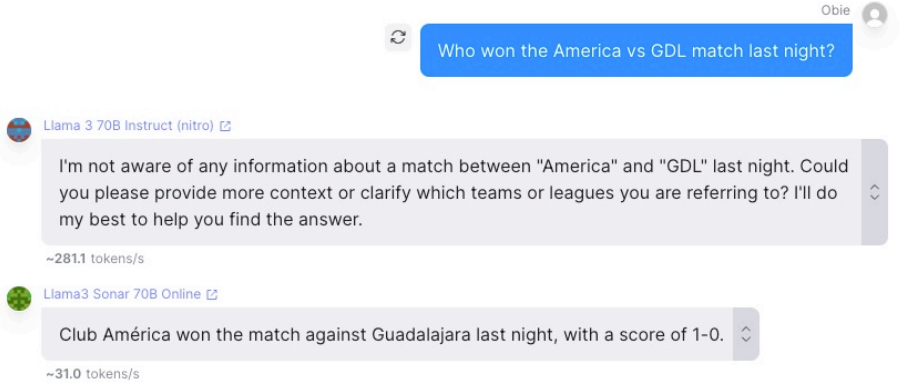
लगभग 30-70 बिलियन पैरामीटर वाले मध्यम आकार के मॉडल मजबूत तर्क और निर्देश-पालन क्षमताएं प्रदान करते हैं।

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

किसी एप्लिकेशन में LLM को शामिल करने के लिए चुनते समय, आपको मॉडल की क्षमताओं को लागत, विलंबता, संदर्भ लंबाई और सामग्री फ़िल्टरिंग जैसे व्यावहारिक कारकों के साथ संतुलित करना होगा। सरल भाषा कार्यों के लिए छोटे, निर्देश-ट्यून्ड

मॉडल अक्सर सर्वोत्तम विकल्प होते हैं, जबकि जटिल तर्क या विश्लेषण के लिए सबसे बड़े मॉडलों की आवश्यकता हो सकती है। मॉडल का प्रशिक्षण डेटा भी एक महत्वपूर्ण विचार है, क्योंकि यह मॉडल की ज्ञान कट-ऑफ़ तिथि निर्धारित करता है।

कुछ मॉडल, जैसे Perplexity के कुछ मॉडल रीयलटाइम सूचना स्रोतों से जुड़े होते हैं, इसलिए उनकी कोई कट-ऑफ़ तिथि नहीं होती। जब आप उनसे प्रश्न पूछते हैं, तो वे स्वतंत्र रूप से वेब खोज करने और उत्तर उत्पन्न करने के लिए किसी भी वेब पेज को प्राप्त करने का निर्णय ले सकते हैं।

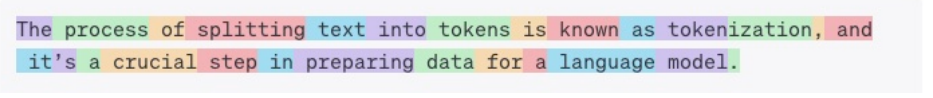


आकृति 1. ऑनलाइन एक्सेस के साथ और बिना Llama3

अंततः, कोई भी एलएलएम (LLM) सभी कार्यों के लिए एक समान उपयुक्त नहीं होता। किसी विशेष उपयोग के लिए सही मॉडल का चयन करने में मॉडल के आकार, आर्किटेक्चर और प्रशिक्षण में विभिन्नताओं को समझना महत्वपूर्ण है। विभिन्न मॉडलों के साथ प्रयोग करना ही यह जानने का एकमात्र व्यावहारिक तरीका है कि कौन सा मॉडल वर्तमान कार्य के लिए सर्वोत्तम प्रदर्शन प्रदान करता है।

टोकनाइज़ेशन: पाठ को टुकड़ों में विभाजित करना

किसी बृहत भाषा मॉडल द्वारा पाठ को संसाधित करने से पहले, उस पाठ को छोटी इकाइयों में विभाजित करना आवश्यक होता है, जिन्हें टोकन कहा जाता है। टोकन अलग-अलग शब्द, शब्दों के भाग, या यहां तक कि एकल वर्ण भी हो सकते हैं। पाठ को टोकन में विभाजित करने की प्रक्रिया को टोकनाइज़ेशन कहा जाता है, और यह भाषा मॉडल के लिए डेटा तैयार करने में एक महत्वपूर्ण चरण है।



आकृति 2. इस वाक्य में 27 टोकन हैं

विभिन्न एलएलएम विभिन्न टोकनाइज़ेशन रणनीतियों का उपयोग करते हैं, जो मॉडल के प्रदर्शन और क्षमताओं पर महत्वपूर्ण प्रभाव डाल सकती हैं। एलएलएम द्वारा उपयोग किए जाने वाले कुछ सामान्य टोकनाइज़र हैं:

- **जीपीटी (बाइट पेयर एनकोडिंग):** जीपीटी टोकनाइज़र बाइट पेयर एनकोडिंग (बीपीई) नामक तकनीक का उपयोग करते हैं जो पाठ को उप-शब्द इकाइयों में विभाजित करती है। बीपीई पाठ कॉर्पस में सबसे अधिक बार आने वाले बाइट्स के जोड़ों को बारी-बारी से मिलाता है, जिससे उप-शब्द टोकन का एक शब्द-भंडार बनता है। यह टोकनाइज़र दुर्लभ और नए शब्दों को अधिक सामान्य उप-शब्द टुकड़ों में विभाजित करके उन्हें संभाल सकता है। जीपीटी टोकनाइज़र का उपयोग जीपीटी-3 और जीपीटी-4 जैसे मॉडलों द्वारा किया जाता है।

- **लामा (सेंटेंसपीस):** लामा टोकनाइज़र सेंटेंसपीस लाइब्रेरी का उपयोग करते हैं, जो एक अपर्यवेक्षित पाठ टोकनाइज़र और डीटोकनाइज़र है। सेंटेंसपीस इनपुट पाठ को यूनिकोड वर्णों के क्रम के रूप में मानता है और प्रशिक्षण कॉर्पस के आधार पर एक उप-शब्द शब्द-भंडार सीखता है। यह किसी भी भाषा को संभाल सकता है जिसे यूनिकोड में एनकोड किया जा सकता है, जो इसे बहुभाषी मॉडलों

के लिए उपयुक्त बनाता है। लामा टोकनाइज़र का उपयोग मेटा के लामा और अल्पाका जैसे मॉडलों द्वारा किया जाता है।

- **सेंटेंसपीस (यूनीग्राम):** सेंटेंसपीस टोकनाइज़र यूनीग्राम नामक एक अलग एल्गोरिथम का भी उपयोग कर सकते हैं, जो एक सबवर्ड रेगुलराइज़ेशन तकनीक पर आधारित है। यूनीग्राम टोकनाइज़ेशन एक यूनीग्राम भाषा मॉडल के आधार पर इष्टतम सबवर्ड शब्दकोश निर्धारित करता है, जो व्यक्तिगत सबवर्ड इकाइयों को संभावनाएं असाइन करता है। यह दृष्टिकोण BPE की तुलना में अधिक अर्थपूर्ण सबवर्ड उत्पन्न कर सकता है। यूनीग्राम के साथ सेंटेंसपीस का उपयोग Google के T5 और BERT जैसे मॉडल करते हैं।

- **गूगल जेमिनी (बहु-माध्यम टोकनाइज़ेशन):** गूगल जेमिनी टेक्स्ट, छवियों, ऑडियो, वीडियो और कोड सहित विभिन्न प्रकार के डेटा को संभालने के लिए डिज़ाइन की गई एक टोकनाइज़ेशन योजना का उपयोग करता है। यह बहु-माध्यम क्षमता जेमिनी को विभिन्न प्रकार की जानकारी को प्रोसेस और एकीकृत करने की अनुमति देती है। विशेष रूप से, Google Gemini 1.5 Pro में एक संदर्भ विंडो है जो लाखों टोकन को संभाल सकती है, जो पिछले मॉडलों की तुलना में बहुत बड़ी है। यह विस्तृत संदर्भ विंडो मॉडल को बड़े संदर्भ को प्रोसेस करने में सक्षम बनाती है, जिससे संभवतः अधिक सटीक प्रतिक्रियाएं मिल सकती हैं। हालांकि, यह ध्यान रखना महत्वपूर्ण है कि जेमिनी की टोकनाइज़ेशन योजना अन्य मॉडलों की तुलना में प्रति वर्ण एक टोकन के काफी करीब है। इसका मतलब है कि जेमिनी मॉडल का वास्तविक उपयोग लागत अपेक्षा से काफी अधिक हो सकता है यदि आप GPT जैसे मॉडल का उपयोग करने के आदी हैं, क्योंकि Google की कीमत टोकन के बजाय वर्णों पर आधारित है।

टोकनाइज़र का चयन एक LLM के कई पहलुओं को प्रभावित करता है, जिसमें शामिल हैं:

- **शब्दकोश का आकार:** टोकनाइज़र मॉडल के शब्दकोश के आकार को निर्धारित करता है, जो उसके द्वारा पहचाने जाने वाले विशिष्ट टोकन का समूह है। एक

बड़ा, अधिक सूक्ष्म शब्दकोश मॉडल को शब्दों और वाक्यांशों की एक व्यापक श्रृंखला को संभालने में मदद कर सकता है और यहां तक कि बहु-माध्यम (केवल टेक्स्ट से अधिक को समझने और उत्पन्न करने में सक्षम) भी बन सकता है, लेकिन यह मॉडल की मेमोरी आवश्यकताओं और कम्प्यूटेशनल जटिलता को भी बढ़ाता है।

- **दुर्लभ और अज्ञात शब्दों को संभालना:** BPE और सेंटेंसपीस जैसे उप-शब्द इकाइयों का उपयोग करने वाले टोकनाइज़र, दुर्लभ और अज्ञात शब्दों को अधिक सामान्य उप-शब्द टुकड़ों में तोड़ सकते हैं। यह मॉडल को उन शब्दों के अर्थ के बारे में शिक्षित अनुमान लगाने की अनुमति देता है जिन्हें इसने पहले नहीं देखा है, उनमें मौजूद उप-शब्दों के आधार पर।
- **बहुभाषी समर्थन:** सेंटेंसपीस जैसे टोकनाइज़र, जो किसी भी यूनिकोड-एनकोडेबल भाषा को संभाल सकते हैं, बहुभाषी मॉडल के लिए उपयुक्त हैं जिन्हें कई भाषाओं में टेक्स्ट को प्रोसेस करने की आवश्यकता होती है।

किसी विशेष एप्लिकेशन के लिए एलएलएम का चयन करते समय, यह महत्वपूर्ण है कि उसके द्वारा उपयोग किए जाने वाले टोकनाइज़र पर विचार किया जाए और यह देखा जाए कि वह कार्य की विशिष्ट भाषा प्रसंस्करण आवश्यकताओं के साथ कितनी अच्छी तरह से मेल खाता है। टोकनाइज़र का डोमेन-विशिष्ट शब्दावली, दुर्लभ शब्दों और बहुभाषी पाठ को संभालने की मॉडल की क्षमता पर महत्वपूर्ण प्रभाव पड़ सकता है।

संदर्भ आकार: भाषा मॉडल अनुमान के दौरान कितनी जानकारी का उपयोग कर सकता है?

जब भाषा मॉडल की चर्चा होती है, तो संदर्भ आकार उस पाठ की मात्रा को संदर्भित करता है जिसे मॉडल अपनी प्रतिक्रियाओं को संसाधित या उत्पन्न करते समय विचार कर सकता है। यह अनिवार्य रूप से इस बात का माप है कि मॉडल कितनी जानकारी को “याद” रख सकता है और अपने आउटपुट को सूचित करने के लिए उपयोग कर सकता है (टोकन में व्यक्त)। भाषा मॉडल का संदर्भ आकार उसकी क्षमताओं और उन

कार्यों के प्रकारों पर महत्वपूर्ण प्रभाव डाल सकता है जिन्हें यह प्रभावी ढंग से कर सकता है।

संदर्भ आकार क्या है?

तकनीकी शब्दों में, संदर्भ आकार टोकन (शब्द या शब्द के टुकड़ों) की संख्या से निर्धारित होता है जिसे एक भाषा मॉडल एक एकल इनपुट अनुक्रम में संसाधित कर सकता है। इसे अक्सर मॉडल की "ध्यान अवधि" या "संदर्भ विंडो" के रूप में संदर्भित किया जाता है। संदर्भ आकार जितना बड़ा होगा, प्रतिक्रिया उत्पन्न करते समय या कोई कार्य करते समय मॉडल एक साथ उतना ही अधिक पाठ पर विचार कर सकता है।

विभिन्न भाषा मॉडलों में अलग-अलग संदर्भ आकार होते हैं, जो कुछ सौ टोकन से लेकर लाखों टोकन तक हो सकते हैं। संदर्भ के लिए, पाठ का एक विशिष्ट अनुच्छेद लगभग 100-150 टोकन हो सकता है, जबकि एक पूरी किताब में दसियों या सैकड़ों हजार टोकन हो सकते हैं।

> ट्रांसफॉर्मर-आधारित लार्ज लैंग्वेज मॉडल (एलएलएम) को असीम लंबे इनपुट तक स्केल करने के लिए कुशल विधियों पर भी काम चल रहा है, जो सीमित मेमोरी और कम्प्यूटेशन के साथ काम करती हैं।

संदर्भ आकार महत्वपूर्ण क्यों है?

भाषा मॉडल का संदर्भ आकार सुसंगत और संदर्भगत रूप से प्रासंगिक पाठ को समझने और उत्पन्न करने की उसकी क्षमता पर महत्वपूर्ण प्रभाव डालता है। यहाँ कुछ प्रमुख कारण हैं कि संदर्भ आकार क्यों मायने रखता है:

1. **लंबी सामग्री को समझना:** बड़े संदर्भ आकार वाले मॉडल लंबे पाठों, जैसे लेख, रिपोर्ट, या यहां तक कि पूरी किताबों को बेहतर ढंग से समझ और विश्लेषण

कर सकते हैं। यह दस्तावेज़ सारांशीकरण, प्रश्न उत्तर और सामग्री विश्लेषण जैसे कार्यों के लिए महत्वपूर्ण है।

2. **सुसंगतता बनाए रखना:** एक बड़ी संदर्भ विंडो मॉडल को लंबे आउटपुट में सुसंगतता और स्थिरता बनाए रखने की अनुमति देती है। यह कहानी निर्माण, संवाद प्रणालियों और सामग्री निर्माण जैसे कार्यों के लिए महत्वपूर्ण है, जहां एक सुसंगत कथा या विषय को बनाए रखना आवश्यक है। यह संरचित डेटा को उत्पन्न करने या परिवर्तित करने के लिए एलएलएम का उपयोग करते समय भी पूरी तरह से महत्वपूर्ण है।
3. **दीर्घकालिक निर्भरताओं को समझना:** कुछ भाषा कार्यों में टेक्स्ट में दूर-दूर स्थित शब्दों या वाक्यांशों के बीच संबंधों को समझने की आवश्यकता होती है। बड़े संदर्भ आकार वाले मॉडल इन दीर्घकालिक निर्भरताओं को बेहतर ढंग से समझ सकते हैं, जो भावना विश्लेषण, अनुवाद, और भाषा समझ जैसे कार्यों के लिए महत्वपूर्ण हो सकते हैं।
4. **जटिल निर्देशों को संभालना:** जिन अनुप्रयोगों में भाषा मॉडलों का उपयोग जटिल, बहु-चरणीय निर्देशों का पालन करने के लिए किया जाता है, बड़ा संदर्भ आकार मॉडल को प्रतिक्रिया उत्पन्न करते समय केवल हाल के कुछ शब्दों के बजाय पूरे निर्देश सेट पर विचार करने की अनुमति देता है।

विभिन्न संदर्भ आकारों वाले भाषा मॉडलों के उदाहरण

यहाँ विभिन्न संदर्भ आकारों वाले कुछ भाषा मॉडलों के उदाहरण दिए गए हैं:

- OpenAI GPT-3.5 Turbo: 4,095 टोकन
- Mistral 7B Instruct: 32,768 टोकन
- Anthropic Claude v1: 100,000 टोकन
- OpenAI GPT-4 Turbo: 128,000 टोकन
- Anthropic Claude v2: 200,000 टोकन
- Google Gemini Pro 1.5: 2.8M टोकन

जैसा कि आप देख सकते हैं, इन मॉडलों में संदर्भ आकारों की एक विस्तृत श्रृंखला है, OpenAI GPT-3.5 Turbo मॉडल के लगभग 4,000 टोकन से लेकर Anthropic Claude v2 मॉडल के 200,000 टोकन तक। कुछ मॉडल, जैसे Google का PaLM 2 और OpenAI का GPT-4, बड़े संदर्भ आकारों वाले विभिन्न संस्करण प्रदान करते हैं (जैसे "32k" संस्करण), जो और भी लंबी इनपुट श्रृंखलाओं को संभाल सकते हैं। और इस समय (अप्रैल 2024) Google Gemini Pro लगभग 3 मिलियन टोकन का दावा कर रहा है!

यह ध्यान रखना महत्वपूर्ण है कि संदर्भ आकार किसी विशेष मॉडल के विशिष्ट कार्यान्वयन और संस्करण के आधार पर भिन्न हो सकता है। उदाहरण के लिए, मूल OpenAI GPT-4 मॉडल का संदर्भ आकार 8,191 टोकन है, जबकि बाद के GPT-4 संस्करणों जैसे Turbo और 4o का संदर्भ आकार बहुत बड़ा है, जो 128,000 टोकन है।

> Sam Altman ने वर्तमान संदर्भ सीमाओं की तुलना 80 के दशक में पर्सनल कंप्यूटर प्रोग्रामर्स को मिलने वाली किलोबाइट्स की वर्किंग मेमोरी से की है, और कहा है कि निकट भविष्य में हम "अपना सारा व्यक्तिगत डेटा" एक बड़े भाषा मॉडल के संदर्भ में फिट कर पाएंगे।

सही संदर्भ आकार का चयन

किसी विशेष एप्लिकेशन के लिए भाषा मॉडल का चयन करते समय, संबंधित कार्य की संदर्भ आकार आवश्यकताओं पर विचार करना महत्वपूर्ण है। जिन कार्यों में छोटे, अलग-अलग टेक्स्ट के टुकड़े शामिल होते हैं, जैसे भावना विश्लेषण या सरल प्रश्न उत्तर, उनके लिए छोटा संदर्भ आकार पर्याप्त हो सकता है। हालाँकि, लंबे और अधिक जटिल टेक्स्ट को समझने और उत्पन्न करने वाले कार्यों के लिए, बड़े संदर्भ आकार की आवश्यकता होगी।

यह ध्यान देने योग्य है कि बड़े संदर्भ आकार अक्सर बढ़ी हुई कम्प्यूटेशनल लागत

और धीमी प्रोसेसिंग समय के साथ आते हैं, क्योंकि मॉडल को प्रतिक्रिया उत्पन्न करते समय अधिक जानकारी पर विचार करना पड़ता है। इसलिए, आपको अपने एप्लिकेशन के लिए भाषा मॉडल चुनते समय संदर्भ आकार और प्रदर्शन के बीच संतुलन बनाना होगा।

> सबसे बड़े संदर्भ आकार वाला मॉडल क्यों न चुनें और उसमें जितनी हो सके उतनी जानकारी क्यों न भर दें? प्रदर्शन कारकों के अलावा, दूसरा मुख्य विचार लागत है। मार्च 2024 में Google Gemini Pro 1.5 का पूर्ण संदर्भ के साथ एक एकल प्रॉम्प्ट-प्रतिक्रिया चक्र आपको लगभग $8 (USD) का पड़ेगा। यदि आपके पास इस खर्च को सही ठहराने वाला कोई उपयोग मामला है, तो बहुत अच्छा है! लेकिन अधिकांश एप्लिकेशन के लिए, यह कई गुना अधिक महंगा है।

घास के ढेर में सुइयाँ ढूँढना

बड़े डेटासेट में पुनर्प्राप्ति की चुनौतियों के लिए घास के ढेर में सुई ढूँढने की अवधारणा लंबे समय से एक रूपक रही है। एलएलएम के क्षेत्र में, हम इस उपमा को थोड़ा बदल देते हैं। कल्पना कीजिए कि हम किसी विशाल टेक्स्ट (जैसे Paul Graham के निबंधों का पूरा संकलन) में दबे एक तथ्य को नहीं, बल्कि चारों ओर बिखरे कई तथ्यों को खोज रहे हैं। यह परिदृश्य एक घास के ढेर में नहीं, बल्कि एक विशाल मैदान में कई सुइयाँ खोजने जैसा है। और यह महत्वपूर्ण बात है: हमें न केवल इन सुइयों को खोजना है, बल्कि उन्हें एक सुसंगत धागे में भी पिरोना है।

जब लंबे संदर्भों में एम्बेडेड कई तथ्यों को पुनर्प्राप्त करने और उन पर तर्क करने का काम होता है, तो एलएलएम दोहरी चुनौती का सामना करते हैं। पहली, पुनर्प्राप्ति सटीकता की सीधी समस्या है—यह स्वाभाविक रूप से तथ्यों की संख्या बढ़ने के साथ कम हो जाती है। यह अपेक्षित है; आखिरकार, एक विशाल टेक्स्ट में कई विवरणों को ट्रैक करना सबसे परिष्कृत मॉडलों को भी प्रभावित करता है।

दूसरा, और शायद अधिक महत्वपूर्ण, इन तथ्यों के साथ तर्क करने की चुनौती है।

तथ्यों को चुन लेना एक बात है; उन्हें एक सुसंगत कथा या उत्तर में संश्लेषित करना एक अलग बात है। यही वह जगह है जहाँ वास्तविक परीक्षा होती है। तर्क कार्यों में एलएलएम का प्रदर्शन सरल पुनर्प्राप्ति कार्यों की तुलना में और अधिक कमजोर होता है। यह गिरावट सिर्फ मात्रा के बारे में नहीं है; यह संदर्भ, प्रासंगिकता और निष्कर्ष का एक जटिल नृत्य है।

ऐसा क्यों होता है? मानव संज्ञान में स्मृति और ध्यान की गतिशीलता पर विचार करें, जो कुछ हद तक एलएलएम में प्रतिबिंबित होती है। बड़ी मात्रा में जानकारी को संसाधित करते समय, एलएलएम, मनुष्यों की तरह, नई जानकारी को आत्मसात करते समय पहले के विवरणों को खो सकते हैं। यह विशेष रूप से उन मॉडल में सच है जो स्वचालित रूप से पाठ के पूर्व खंडों को प्राथमिकता देने या उन पर पुनर्विचार करने के लिए स्पष्ट रूप से डिज़ाइन नहीं किए गए हैं।

इसके अलावा, एक एलएलएम की इन पुनर्प्राप्त तथ्यों को एक सुसंगत प्रतिक्रिया में बुनने की क्षमता कथा निर्माण के समान है। इसके लिए न केवल जानकारी की पुनर्प्राप्ति बल्कि गहरी समझ और संदर्भगत स्थापन की आवश्यकता होती है, जो वर्तमान एआई के लिए एक कठिन चुनौती बनी हुई है।

तो, इन प्रौद्योगिकियों के डेवलपर्स और एकीकरणकर्ताओं के रूप में हमारे लिए इसका क्या मतलब है? हमें उन प्रणालियों को डिजाइन करते समय इन सीमाओं के प्रति बेहद सचेत रहने की जरूरत है जो जटिल, लंबी-फॉर्म कार्यों को संभालने के लिए एलएलएम पर निर्भर करती हैं। यह समझना कि कुछ परिस्थितियों में प्रदर्शन खराब हो सकता है, हमें यथार्थवादी अपेक्षाएं निर्धारित करने और बेहतर फॉलबैक तंत्र या पूरक रणनीतियों को विकसित करने में मदद करता है।

मोडैलिटीज़: पाठ से परे

जबकि आज के अधिकांश भाषा मॉडल पाठ को संसाधित करने और उत्पन्न करने पर केंद्रित हैं, बहु-माध्यम मॉडल की ओर एक बढ़ता हुआ रुझान है जो स्वाभाविक रूप से कई प्रकार के डेटा, जैसे छवियों, ऑडियो और वीडियो को इनपुट और आउटपुट कर सकते हैं। ये बहु-माध्यम मॉडल एआई-संचालित अनुप्रयोगों के लिए नई संभावनाएं खोलते हैं जो विभिन्न माध्यमों में सामग्री को समझ और उत्पन्न कर सकते हैं।

मोडैलिटीज़ क्या हैं?

भाषा मॉडल के संदर्भ में, मोडैलिटीज़ विभिन्न प्रकार के डेटा को संदर्भित करती हैं जिन्हें एक मॉडल संसाधित और उत्पन्न कर सकता है। सबसे सामान्य मोडैलिटी पाठ है, जिसमें किताबों, लेखों, वेबसाइटों और सोशल मीडिया पोस्ट जैसे विभिन्न रूपों में लिखी गई भाषा शामिल है। हालाँकि, कई अन्य मोडैलिटीज़ हैं जिन्हें तेजी से भाषा मॉडल में शामिल किया जा रहा है:

- **छवियां:** दृश्य डेटा जैसे फोटोग्राफ, चित्र और आरेख।
- **ऑडियो:** ध्वनि डेटा जैसे भाषण, संगीत और पर्यावरणीय ध्वनियां।
- **वीडियो:** चलती दृश्य डेटा, जो अक्सर ऑडियो के साथ होती है, जैसे वीडियो क्लिप और फिल्में।

प्रत्येक विधा भाषा मॉडल के लिए अनूठी चुनौतियां और अवसर प्रस्तुत करती है। उदाहरण के लिए, छवियों के लिए मॉडल को दृश्य अवधारणाओं और संबंधों को समझने की आवश्यकता होती है, जबकि ऑडियो के लिए मॉडल को वाणी और अन्य ध्वनियों को संसाधित और उत्पन्न करने की आवश्यकता होती है।

बहुविध भाषा मॉडल

बहुविध भाषा मॉडल एक ही मॉडल में कई विधाओं को संभालने के लिए डिज़ाइन किए गए हैं। इन मॉडलों में आमतौर पर विशेष घटक या परतें होती हैं जो विभिन्न विधाओं में इनपुट को समझ सकती हैं और आउटपुट डेटा उत्पन्न कर सकती हैं। बहुविध भाषा मॉडलों के कुछ उल्लेखनीय उदाहरण हैं:

- **OpenAI का GPT-4o:** GPT-4o एक विशाल भाषा मॉडल है जो टेक्स्ट के अतिरिक्त स्वाभाविक रूप से वाणी ऑडियो को समझता और संसाधित करता है। यह क्षमता GPT-4o को बोली गई भाषा का प्रतिलेखन करने, ऑडियो इनपुट से टेक्स्ट उत्पन्न करने, और मौखिक प्रश्नों के आधार पर प्रतिक्रियाएं प्रदान करने जैसे कार्य करने में सक्षम बनाती है।

- **OpenAI का दृश्य इनपुट के साथ GPT-4:** GPT-4 एक विशाल भाषा मॉडल है जो टेक्स्ट और छवियों दोनों को संसाधित कर सकता है। जब छवि को इनपुट के रूप में दिया जाता है, GPT-4 छवि की सामग्री का विश्लेषण कर सकता है और दृश्य जानकारी का वर्णन करने या उस पर प्रतिक्रिया देने वाला टेक्स्ट उत्पन्न कर सकता है।
- **Google का Gemini:** Gemini एक बहुविध मॉडल है जो टेक्स्ट, छवियों और वीडियो को संभाल सकता है। यह एक एकीकृत वास्तुकला का उपयोग करता है जो क्रॉस-मोडल समझ और निर्माण को सक्षम बनाता है, जिससे छवि कैप्शनिंग, वीडियो सारांशीकरण और दृश्य प्रश्न उत्तरण जैसे कार्य संभव होते हैं।

- **DALL-E और Stable Diffusion:** हालांकि ये पारंपरिक अर्थों में भाषा मॉडल नहीं हैं, ये मॉडल टेक्स्ट विवरणों से छवियां उत्पन्न करके बहुविध AI की शक्ति को प्रदर्शित करते हैं। वे विभिन्न विधाओं के बीच अनुवाद कर सकने वाले मॉडलों की संभावना को प्रदर्शित करते हैं।

बहुविध मॉडल के लाभ और अनुप्रयोग

बहुविध भाषा मॉडल कई लाभ प्रदान करते हैं और विभिन्न अनुप्रयोगों को सक्षम बनाते हैं, जिनमें शामिल हैं:

- **बेहतर समझ:** कई विधाओं से जानकारी को संसाधित करके, ये मॉडल दुनिया की अधिक व्यापक समझ प्राप्त कर सकते हैं, जैसे मनुष्य विभिन्न संवेदी इनपुट से सीखते हैं।
- **क्रॉस-मोडल निर्माण:** बहुविध मॉडल एक विधा में दूसरी विधा के इनपुट के आधार पर सामग्री उत्पन्न कर सकते हैं, जैसे टेक्स्ट विवरण से छवि बनाना या लिखित लेख से वीडियो सारांश तैयार करना।
- **सुगम्यता:** बहुविध मॉडल विधाओं के बीच अनुवाद करके जानकारी को अधिक सुलभ बना सकते हैं, जैसे दृष्टिबाधित उपयोगकर्ताओं के लिए छवियों का टेक्स्ट विवरण तैयार करना या लिखित सामग्री के ऑडियो संस्करण बनाना।

- **रचनात्मक अनुप्रयोग:** बहु-माध्यम मॉडल का उपयोग टेक्स्ट-आधारित निर्देशों के आधार पर कला, संगीत, या वीडियो जैसी रचनात्मक वस्तुएं बनाने के लिए किया जा सकता है, जो कलाकारों और सामग्री निर्माताओं के लिए नई संभावनाएं खोलता है।

जैसे-जैसे बहु-माध्यम भाषा मॉडल विकसित होते जाएंगे, वे कई माध्यमों में सामग्री को समझने और उत्पन्न करने में सक्षम AI-संचालित अनुप्रयोगों के विकास में महत्वपूर्ण भूमिका निभाएंगे। यह मनुष्यों और AI सिस्टम के बीच अधिक प्राकृतिक और सहज संवाद को सक्षम करेगा, साथ ही रचनात्मक अभिव्यक्ति और ज्ञान प्रसार के लिए नई संभावनाएं खोलेगा।

प्रदाता पारिस्थितिकी तंत्र

जब बृहत भाषा मॉडल (LLMs) को अनुप्रयोगों में शामिल करने की बात आती है, तो आपके पास चुनने के लिए विकल्पों की बढ़ती श्रृंखला है। प्रत्येक प्रमुख LLM प्रदाता, जैसे OpenAI, Anthropic, Google, और Cohere, अपने स्वयं के मॉडल, APIs, और टूल्स का पारिस्थितिकी तंत्र प्रदान करता है। सही प्रदाता का चयन करने में विभिन्न कारकों पर विचार करना शामिल है, जिसमें मूल्य निर्धारण, प्रदर्शन, सामग्री फ़िल्टरिंग, डेटा गोपनीयता, और अनुकूलन विकल्प शामिल हैं।

OpenAI

OpenAI, LLMs के सबसे प्रसिद्ध प्रदाताओं में से एक है, जिसकी GPT श्रृंखला (GPT-3, GPT-4) विभिन्न अनुप्रयोगों में व्यापक रूप से उपयोग की जाती है। OpenAI एक उपयोगकर्ता-अनुकूल API प्रदान करता है जो आपको आसानी से उनके मॉडल को अनुप्रयोगों में एकीकृत करने की अनुमति देता है। वे एंट्री-लेवल Ada मॉडल से लेकर शक्तिशाली Davinci मॉडल तक, विभिन्न क्षमताओं और मूल्य बिंदुओं वाले मॉडल प्रदान करते हैं।

OpenAI का पारिस्थितिकी तंत्र OpenAI Playground जैसे टूल्स भी शामिल करता है, जो आपको प्रॉम्प्ट्स के साथ प्रयोग करने और विशिष्ट उपयोग मामलों के लिए

मॉडल को फ़ाइन-ट्यून करने की अनुमति देता है। वे अनुचित या हानिकारक सामग्री के उत्पादन को रोकने में मदद करने के लिए सामग्री फ़िल्टरिंग विकल्प प्रदान करते हैं।

OpenAI के मॉडल का सीधे उपयोग करते समय, मैं Alex Rudall की ruby-openai लाइब्रेरी पर निर्भर करता हूं।

Anthropic

Anthropic, LLM क्षेत्र में एक और प्रमुख खिलाड़ी है, जिनके Claude मॉडल मजबूत प्रदर्शन और नैतिक विचारों के लिए लोकप्रियता प्राप्त कर रहे हैं। Anthropic सुरक्षित और जिम्मेदार AI सिस्टम विकसित करने पर ध्यान केंद्रित करता है, जिसमें सामग्री फ़िल्टरिंग और हानिकारक आउटपुट से बचने पर विशेष जोर दिया जाता है।

Anthropic के पारिस्थितिकी तंत्र में Claude API शामिल है, जो आपको मॉडल को अपने अनुप्रयोगों में एकीकृत करने की अनुमति देता है, साथ ही प्रॉम्प्ट इंजीनियरिंग और फ़ाइन-ट्यूनिंग के लिए टूल्स भी शामिल हैं। वे Claude Instant मॉडल भी प्रदान करते हैं, जो अधिक अप-टू-डेट और तथ्यपरक प्रतिक्रियाओं के लिए वेब खोज क्षमताओं को शामिल करता है।

Anthropic के मॉडल्स का सीधे उपयोग करते समय, मैं Alex Rudall की anthrophic लाइब्रेरी पर निर्भर करता हूं।

Google

Google ने कई शक्तिशाली एलएलएम विकसित किए हैं, जिनमें Gemini, BERT, T5, और PaLM शामिल हैं। ये मॉडल प्राकृतिक भाषा प्रसंस्करण के विभिन्न कार्यों में अपने मजबूत प्रदर्शन के लिए जाने जाते हैं। Google के पारिस्थितिकी तंत्र में TensorFlow और Keras लाइब्रेरी शामिल हैं, जो मशीन लर्निंग मॉडल के निर्माण और प्रशिक्षण के लिए टूल्स और फ्रेमवर्क प्रदान करती हैं।

Google एक Cloud AI Platform भी प्रदान करता है, जो आपको क्लाउड में उनके मॉडल को आसानी से तैनात और स्केल करने की सुविधा देता है। वे भावना विश्लेषण,

एंटिटी पहचान, और अनुवाद जैसे कार्यों के लिए पूर्व-प्रशिक्षित मॉडल और एपीआई की एक श्रृंखला प्रदान करते हैं।

Meta

Meta, जो पहले Facebook के नाम से जाना जाता था, बड़े भाषा मॉडल के विकास में गहराई से निवेश कर रहा है, जिसे LLaMA और OPT जैसे मॉडल की रिलीज से रेखांकित किया गया है। ये मॉडल विविध भाषा कार्यों में अपने मजबूत प्रदर्शन के लिए उल्लेखनीय हैं और मुख्य रूप से ओपन-सोर्स चैनलों के माध्यम से उपलब्ध कराए जाते हैं, जो अनुसंधान और सामुदायिक सहयोग के प्रति Meta की प्रतिबद्धता को समर्थन देते हैं।

Meta का पारिस्थितिकी तंत्र मुख्य रूप से PyTorch के इर्द-गिर्द बना है, जो एक ओपन-सोर्स मशीन लर्निंग लाइब्रेरी है जिसे इसकी गतिशील कम्प्यूटेशनल क्षमताओं और लचीलेपन के लिए पसंद किया जाता है, जो नवीन एआई अनुसंधान और विकास को सुगम बनाती है।

अपनी तकनीकी पेशकशों के अलावा, Meta नैतिक एआई विकास पर जोर देता है। वे मजबूत कंटेंट फ़िल्टरिंग को लागू करते हैं और पूर्वाग्रहों को कम करने पर ध्यान केंद्रित करते हैं, जो एआई अनुप्रयोगों में सुरक्षा और जिम्मेदारी के अपने व्यापक लक्ष्यों के अनुरूप है।

Cohere

Cohere एलएलएम क्षेत्र में एक नया प्रवेशक है, जो प्रतिस्पर्धियों की तुलना में एलएलएम को अधिक सुलभ और उपयोग में आसान बनाने पर ध्यान केंद्रित कर रहा है। उनके पारिस्थितिकी तंत्र में Cohere API शामिल है, जो टेक्स्ट जनरेशन, वर्गीकरण और सारांशीकरण जैसे कार्यों के लिए पूर्व-प्रशिक्षित मॉडल तक पहुंच प्रदान करता है।

Cohere प्रॉम्प्ट इंजीनियरिंग, फ़ाइन-ट्यूनिंग और कंटेंट फ़िल्टरिंग के लिए टूल्स भी प्रदान करता है। वे डेटा गोपनीयता और सुरक्षा पर जोर देते हैं, जिसमें एन्क्रिप्टेड डेटा स्टोरेज और एक्सेस कंट्रोल जैसी सुविधाएं शामिल हैं।

Ollama

Ollama एक स्व-होस्टेड प्लेटफॉर्म है जो उपयोगकर्ताओं को विभिन्न बड़े भाषा मॉडल (एलएलएम) को अपनी मशीनों पर स्थानीय रूप से प्रबंधित और तैनात करने की अनुमति देता है, जिससे उन्हें बाहरी क्लाउड सेवाओं पर निर्भर किए बिना अपने एआई मॉडल पर पूर्ण नियंत्रण मिलता है। यह सेटअप उन लोगों के लिए आदर्श है जो डेटा गोपनीयता को प्राथमिकता देते हैं और अपने एआई संचालन को इन-हाउस संभालना चाहते हैं।

यह प्लेटफ़ॉर्म कई प्रकार के मॉडल्स का समर्थन करता है, जिसमें Llama, Phi, Gemma, और Mistral के विभिन्न संस्करण शामिल हैं, जो आकार और कम्प्यूटेशनल आवश्यकताओं में भिन्न होते हैं। Ollama इन मॉडल्स को कमांड लाइन से सरल आदेशों जैसे ollama run <model_name> का उपयोग करके डाउनलोड और चलाना आसान बनाता है, और यह macOS, Linux, और Windows सहित विभिन्न ऑपरेटिंग सिस्टम पर काम करने के लिए डिज़ाइन किया गया है।

रिमोट API का उपयोग किए बिना अपने एप्लिकेशन में ओपन-सोर्स मॉडल्स को एकीकृत करने के इच्छुक डेवलपर्स के लिए, Ollama कंटेनर प्रबंधन टूल्स के समान मॉडल लाइफसाइकल को प्रबंधित करने के लिए CLI प्रदान करता है। यह कस्टम कॉन्फ़िगरेशन और प्रॉम्प्ट्स का भी समर्थन करता है, जो विशिष्ट आवश्यकताओं या उपयोग के मामलों के लिए मॉडल्स को अनुकूलित करने की उच्च डिग्री की अनुमति देता है।

Ollama विशेष रूप से तकनीकी-कुशल उपयोगकर्ताओं और डेवलपर्स के लिए उपयुक्त है, इसके कमांड-लाइन इंटरफ़ेस और AI मॉडल्स के प्रबंधन और परिनियोजन में प्रदान की जाने वाली लचीलेपन के कारण। यह उन व्यवसायों और व्यक्तियों के लिए एक शक्तिशाली टूल बनाता है जिन्हें सुरक्षा और नियंत्रण से समझौता किए बिना मजबूत AI क्षमताओं की आवश्यकता होती है।

मल्टी-मॉडल प्लेटफ़ॉर्म्स

इसके अतिरिक्त, कुछ प्रदाता हैं जो विभिन्न प्रकार के ओपन-सोर्स मॉडल्स की होस्टिंग करते हैं, जैसे Together.ai और Groq। ये प्लेटफ़ॉर्म्स लचीलापन और अनुकूलन प्रदान करते हैं, जो आपको अपनी विशिष्ट आवश्यकताओं के अनुसार ओपन-सोर्स मॉडल्स को चलाने और कुछ मामलों में, फ़ाइन-ट्यून भी करने की अनुमति देते हैं। उदाहरण के लिए, Together.ai विभिन्न ओपन-सोर्स LLMs तक पहुंच प्रदान करता है, जो उपयोगकर्ताओं को विभिन्न मॉडल्स और कॉन्फ़िगरेशन के साथ प्रयोग करने में सक्षम बनाता है। Groq अत्यंत उच्च-प्रदर्शन पूर्णता प्रदान करने पर ध्यान केंद्रित करता है जो इस पुस्तक के लेखन के समय लगभग जादुई प्रतीत होता है

LLM प्रदाता का चयन

LLM प्रदाता का चयन करते समय, आपको निम्नलिखित कारकों पर विचार करना चाहिए:

- **मूल्य निर्धारण:** विभिन्न प्रदाता अलग-अलग मूल्य निर्धारण मॉडल प्रदान करते हैं, जो उपयोग-के-अनुसार-भुगतान से लेकर सदस्यता-आधारित योजनाओं तक होते हैं। प्रदाता का चयन करते समय अपेक्षित उपयोग और बजट पर विचार करना महत्वपूर्ण है।
- **प्रदर्शन:** LLMs का प्रदर्शन प्रदाताओं के बीच काफी भिन्न हो सकता है, इसलिए निर्णय लेने से पहले विशिष्ट उपयोग के मामलों पर मॉडल्स का बेंचमार्क और परीक्षण करना महत्वपूर्ण है।
- **कंटेंट फ़िल्टरिंग:** एप्लिकेशन के आधार पर, कंटेंट फ़िल्टरिंग एक महत्वपूर्ण विचार हो सकता है। कुछ प्रदाता अन्य की तुलना में अधिक मजबूत कंटेंट फ़िल्टरिंग विकल्प प्रदान करते हैं।
- **डेटा गोपनीयता:** यदि एप्लिकेशन संवेदनशील उपयोगकर्ता डेटा को संभालता है, तो मजबूत डेटा गोपनीयता और सुरक्षा प्रथाओं वाले प्रदाता का चयन करना महत्वपूर्ण है।

- **अनुकूलन:** कुछ प्रदाता विशिष्ट उपयोग के मामलों के लिए मॉडल्स को फ़ाइन-ट्यून और अनुकूलित करने के मामले में अधिक लचीलापन प्रदान करते हैं।

अंततः, LLM प्रदाता का चयन एप्लिकेशन की विशिष्ट आवश्यकताओं और सीमाओं पर निर्भर करता है। विकल्पों का सावधानीपूर्वक मूल्यांकन करके और कीमत, प्रदर्शन और डेटा गोपनीयता जैसे कारकों पर विचार करके, आप अपनी आवश्यकताओं को सर्वोत्तम रूप से पूरा करने वाले प्रदाता का चयन कर सकते हैं।

यह भी ध्यान देने योग्य है कि LLM का परिदृश्य लगातार विकसित हो रहा है, जिसमें नए प्रदाता और मॉडल नियमित रूप से सामने आ रहे हैं। आपको नवीनतम विकास से अपडेट रहना चाहिए और नए विकल्पों की खोज के लिए तैयार रहना चाहिए जब वे उपलब्ध हों।

OpenRouter

इस पुस्तक में मैं विशेष रूप से OpenRouter पर अपने API प्रदाता के रूप में निर्भर रहूंगा। कारण सरल है: यह सभी सबसे लोकप्रिय वाणिज्यिक और ओपन-सोर्स मॉडल के लिए एक वन-स्टॉप शॉप है। यदि आप AI कोडिंग के साथ कुछ अनुभव प्राप्त करने के लिए उत्सुक हैं, तो शुरुआत करने के लिए सबसे अच्छी जगहों में से एक मेरी खुद की OpenRouter Ruby Library है।

प्रदर्शन के बारे में सोचना

एप्लिकेशन में भाषा मॉडल को शामिल करते समय, प्रदर्शन एक महत्वपूर्ण विचार है। एक भाषा मॉडल का प्रदर्शन उसकी विलंबता (प्रतिक्रिया उत्पन्न करने में लगने वाला समय) और प्रवाह दर (प्रति समय इकाई में संभाल सकने वाले अनुरोधों की संख्या) के संदर्भ में मापा जा सकता है।

प्रथम टोकन तक का समय (TTFT) एक अन्य आवश्यक प्रदर्शन मापदंड है, जो विशेष रूप से चैटबॉट और वास्तविक समय में इंटरैक्टिव प्रतिक्रियाओं की आवश्यकता वाले

एप्लिकेशन के लिए प्रासंगिक है। TTFT उपयोगकर्ता का अनुरोध प्राप्त होने के क्षण से प्रतिक्रिया के पहले शब्द (या टोकन) के उत्पन्न होने तक की विलंबता को मापता है। यह मापदंड निर्बाध और आकर्षक उपयोगकर्ता अनुभव बनाए रखने के लिए महत्वपूर्ण है, क्योंकि देरी से मिलने वाली प्रतिक्रियाएं उपयोगकर्ता की निराशा और असंलग्नता का कारण बन सकती हैं।

ये प्रदर्शन मापदंड उपयोगकर्ता अनुभव और एप्लिकेशन की स्केलेबिलिटी पर महत्वपूर्ण प्रभाव डाल सकते हैं।

कई कारक भाषा मॉडल के प्रदर्शन को प्रभावित कर सकते हैं, जिनमें शामिल हैं:

पैरामीटर संख्या: अधिक पैरामीटर वाले बड़े मॉडल आमतौर पर अधिक कम्प्यूटेशनल संसाधनों की आवश्यकता रखते हैं और छोटे मॉडल की तुलना में उनमें उच्च विलंबता और कम प्रवाह दर हो सकती है।

हार्डवेयर: भाषा मॉडल का प्रदर्शन उस हार्डवेयर के आधार पर काफी भिन्न हो सकता है जिस पर यह चल रहा है। क्लाउड प्रदाता मशीन लर्निंग वर्कलोड के लिए अनुकूलित GPU और TPU इंस्टेंस प्रदान करते हैं, जो मॉडल अनुमान को काफी तेज कर सकते हैं।

OpenRouter की एक अच्छी बात यह है कि इसके द्वारा प्रदान किए जाने वाले कई मॉडल्स के लिए, आपको विभिन्न प्रदर्शन प्रोफाइल और लागत वाले क्लाउड प्रोवाइडर्स में से चुनने का विकल्प मिलता है।

क्वांटाइज़ेशन: क्वांटाइज़ेशन तकनीकों का उपयोग वेट्स और एक्टिवेशन्स को कम परिशुद्धता वाले डेटा प्रकारों में प्रस्तुत करके मॉडल के मेमोरी फुटप्रिंट और कम्प्यूटेशनल आवश्यकताओं को कम करने के लिए किया जा सकता है। यह गुणवत्ता को महत्वपूर्ण रूप से प्रभावित किए बिना प्रदर्शन में सुधार कर सकता है। एक एप्लिकेशन डेवलपर के रूप में, आप शायद विभिन्न क्वांटाइज़ेशन स्तरों पर अपने खुद के मॉडल को प्रशिक्षित करने में शामिल नहीं होंगे, लेकिन शब्दावली से परिचित होना अच्छा है।

बैचिंग: एक साथ कई अनुरोधों को बैच में संसाधित करने से मॉडल लोडिंग और डेटा ट्रांसफर के ओवरहेड को कम करके थ्रूपुट में सुधार हो सकता है।

कैशिंग: बार-बार उपयोग किए जाने वाले प्रॉम्प्ट्स या इनपुट अनुक्रमों के परिणामों को कैश करने से अनुमान अनुरोधों की संख्या कम हो सकती है और समग्र प्रदर्शन में सुधार हो सकता है।

प्रोडक्शन एप्लिकेशन के लिए भाषा मॉडल का चयन करते समय, प्रतिनिधि कार्यभार और हार्डवेयर कॉन्फ़िगरेशन पर इसके प्रदर्शन का बेंचमार्क करना महत्वपूर्ण है। यह संभावित बाधाओं की पहचान करने और यह सुनिश्चित करने में मदद कर सकता है कि मॉडल आवश्यक प्रदर्शन लक्ष्यों को पूरा कर सकता है।

मॉडल के प्रदर्शन और अन्य कारकों जैसे लागत, लचीलापन, और एकीकरण की सरलता के बीच ट्रेड-ऑफ पर भी विचार करना महत्वपूर्ण है। उदाहरण के लिए, रीयल-टाइम प्रतिक्रियाओं की आवश्यकता वाले एप्लिकेशन के लिए कम लेटेंसी वाले छोटे, कम खर्चीले मॉडल का उपयोग करना बेहतर हो सकता है, जबकि बैच प्रोसेसिंग या जटिल तर्क कार्यों के लिए बड़े, अधिक शक्तिशाली मॉडल अधिक उपयुक्त हो सकते हैं।

विभिन्न LLM मॉडल्स के साथ प्रयोग

एक LLM का चयन शायद ही कभी एक स्थायी निर्णय होता है। चूंकि नए और बेहतर मॉडल नियमित रूप से जारी किए जाते हैं, एप्लिकेशन को मॉड्यूलर तरीके से बनाना अच्छा है जो समय के साथ विभिन्न भाषा मॉडल्स को बदलने की अनुमति देता है। प्रॉम्प्ट्स और डेटासेट्स को अक्सर न्यूनतम परिवर्तनों के साथ विभिन्न मॉडल्स में पुन: उपयोग किया जा सकता है। यह आपको अपने एप्लिकेशन को पूरी तरह से पुनर्डिज़ाइन किए बिना भाषा मॉडलिंग में नवीनतम प्रगति का लाभ उठाने की अनुमति देता है।

विभिन्न मॉडल विकल्पों के बीच आसानी से स्विच करने की क्षमता एक और कारण है जिससे मैं OpenRouter को पसंद करता हूं।

नए भाषा मॉडल में अपग्रेड करते समय, यह सुनिश्चित करने के लिए कि यह एप्लिकेशन की आवश्यकताओं को पूरा करता है, इसके प्रदर्शन और आउटपुट गुणवत्ता का पूरी तरह से परीक्षण और सत्यापन करना महत्वपूर्ण है। इसमें डोमेन-विशिष्ट डेटा पर मॉडल को पुनः प्रशिक्षित करना या फाइन-ट्यून करना, साथ ही मॉडल के आउटपुट पर निर्भर किसी भी डाउनस्ट्रीम कंपोनेंट को अपडेट करना शामिल हो सकता है।

प्रदर्शन और मॉड्यूलैरिटी को ध्यान में रखते हुए एप्लिकेशन डिज़ाइन करके, आप स्केलेबल, कुशल और भविष्य-सुरक्षित सिस्टम बना सकते हैं जो भाषा मॉडलिंग तकनीक के तेजी से विकसित होते परिदृश्य के अनुकूल हो सकते हैं।

संयुक्त एआई सिस्टम

हमारी प्रस्तावना को समाप्त करने से पहले, यह उल्लेख करना महत्वपूर्ण है कि 2023 से पहले और ChatGPT द्वारा जनरेटिव एआई में रुचि के विस्फोट से पहले, पारंपरिक एआई दृष्टिकोण आमतौर पर एकल, बंद मॉडल के एकीकरण पर निर्भर करते थे। इसके विपरीत, संयुक्त एआई सिस्टम बुद्धिमान व्यवहार प्राप्त करने के लिए परस्पर जुड़े घटकों की जटिल पाइपलाइनों का लाभ उठाते हैं।

मूल रूप से, संयुक्त एआई सिस्टम कई मॉड्यूल से मिलकर बने होते हैं, जिनमें से प्रत्येक विशिष्ट कार्यों या फ़ंक्शंस को करने के लिए डिज़ाइन किया गया है। इन मॉड्यूल में जनरेटर, रिट्रीवर, रैंकर, क्लासिफायर और विभिन्न अन्य विशेष घटक शामिल हो सकते हैं। समग्र सिस्टम को छोटी, केंद्रित इकाइयों में विभाजित करके, डेवलपर्स अधिक लचीली, स्केलेबल और प्रबंधनीय एआई आर्किटेक्चर बना सकते हैं।

संयुक्त एआई सिस्टम का एक प्रमुख लाभ विभिन्न एआई तकनीकों और मॉडल की ताकतों को जोड़ने की उनकी क्षमता है। उदाहरण के लिए, एक सिस्टम प्राकृतिक भाषा समझ और जनरेशन के लिए एक बृहत भाषा मॉडल (एलएलएम) का उपयोग कर सकता है, जबकि सूचना पुनर्प्राप्ति या नियम-आधारित निर्णय लेने के लिए एक अलग मॉडल का उपयोग कर सकता है। यह मॉड्यूलर दृष्टिकोण आपको एक-साइज-फिट्स-ऑल समाधान पर निर्भर रहने के बजाय प्रत्येक विशिष्ट कार्य के लिए सर्वोत्तम उपकरण और तकनीकों का चयन करने की अनुमति देता है।

हालाँकि, संयुक्त एआई सिस्टम का निर्माण अनूठी चुनौतियां भी प्रस्तुत करता है। विशेष रूप से, सिस्टम के व्यवहार की समग्र सुसंगतता और स्थिरता सुनिश्चित करने के लिए मजबूत परीक्षण, निगरानी और शासन तंत्र की आवश्यकता होती है।

GPT-4 जैसे शक्तिशाली एलएलएम के आगमन से हमें पहले से कहीं अधिक आसानी से संयुक्त एआई सिस्टम के साथ प्रयोग करने की अनुमति मिलती है, क्योंकि ये उन्नत मॉडल अपनी प्राकृतिक भाषा समझ क्षमताओं के अलावा, वर्गीकरण, रैंकिंग और जनरेशन जैसी कई भूमिकाओं को संभालने में सक्षम हैं। यह बहुमुखी प्रतिभा डेवलपर्स को संयुक्त एआई आर्किटेक्चर पर तेजी से प्रोटोटाइप और पुनरावृत्ति करने में सक्षम बनाती है, जो बुद्धिमान एप्लिकेशन विकास के लिए नई संभावनाएं खोलती है।

संयुक्त एआई सिस्टम के लिए डिप्लॉयमेंट पैटर्न

संयुक्त एआई सिस्टम को विभिन्न पैटर्न का उपयोग करके डिप्लॉय किया जा सकता है, जिनमें से प्रत्येक विशिष्ट आवश्यकताओं और उपयोग के मामलों को संबोधित करने के लिए डिज़ाइन किया गया है। आइए चार सामान्य डिप्लॉयमेंट पैटर्न की खोज करें: प्रश्न और उत्तर, बहु-एजेंट/एजेंटिक समस्या समाधानकर्ता, संवादात्मक एआई, और सह-पायलट।

प्रश्न और उत्तर

प्रश्न और उत्तर (क्यू एंड ए) सिस्टम सूचना पुनर्प्राप्ति पर केंद्रित होते हैं जो एआई मॉडल की समझ क्षमताओं के साथ बढ़ाए गए हैं ताकि वे केवल एक खोज इंजन से अधिक के रूप में कार्य कर सकें। पुनर्प्राप्ति-संवर्धित जनरेशन (RAG) का उपयोग करके बाहरी ज्ञान स्रोतों के साथ शक्तिशाली भाषा मॉडल को जोड़कर, प्रश्न और उत्तर सिस्टम भ्रामक जानकारी से बचते हैं और उपयोगकर्ता के प्रश्नों के सटीक और प्रासंगिक उत्तर प्रदान करते हैं।

एक LLM-आधारित प्रश्नोत्तर प्रणाली के प्रमुख घटकों में शामिल हैं:

- **प्रश्न समझना और पुनर्गठन:** उपयोगकर्ता के प्रश्नों का विश्लेषण और उन्हें अंतर्निहित ज्ञान स्रोतों से बेहतर मेल खाने के लिए पुनर्गठित करना।
- **ज्ञान पुनर्प्राप्ति:** पुनर्गठित प्रश्न के आधार पर संरचित या असंरचित डेटा स्रोतों से प्रासंगिक जानकारी प्राप्त करना।
- **प्रतिक्रिया निर्माण:** भाषा मॉडल की जनरेटिव क्षमताओं के साथ पुनर्प्राप्त ज्ञान को एकीकृत करके सुसंगत और जानकारीपूर्ण प्रतिक्रियाएं तैयार करना।

RAG उप-प्रणालियां विशेष रूप से उन प्रश्नोत्तर क्षेत्रों में महत्वपूर्ण हैं जहां सटीक और अद्यतन जानकारी प्रदान करना महत्वपूर्ण है, जैसे ग्राहक सहायता, ज्ञान प्रबंधन, या शैक्षिक अनुप्रयोग

मल्टी-एजेंट/एजेंटिक समस्या समाधानकर्ता

मल्टी-एजेंट, जिसे एजेंटिक प्रणाली भी कहा जाता है, में कई स्वायत्त एजेंट जटिल समस्याओं को हल करने के लिए एक साथ काम करते हैं। प्रत्येक एजेंट की एक विशिष्ट भूमिका, कौशल का समूह, और प्रासंगिक उपकरणों या सूचना स्रोतों तक पहुंच होती है। सहयोग और सूचना का आदान-प्रदान करके, ये एजेंट ऐसे कार्यों को संभाल सकते हैं जो एकल एजेंट के लिए कठिन या असंभव होंगे।

मल्टी-एजेंट समस्या समाधानकर्ताओं के प्रमुख सिद्धांतों में शामिल हैं:

- **विशेषज्ञता:** प्रत्येक एजेंट अपनी विशिष्ट क्षमताओं और ज्ञान का उपयोग करते हुए समस्या के एक विशिष्ट पहलू पर ध्यान केंद्रित करता है।
- **सहयोग:** एजेंट एक सामान्य लक्ष्य प्राप्त करने के लिए संदेश पास करने या साझा मेमोरी के माध्यम से संवाद करते हैं और अपनी कार्रवाइयों का समन्वय करते हैं।
- **अनुकूलन क्षमता:** प्रणाली व्यक्तिगत एजेंटों की भूमिकाओं और व्यवहारों को समायोजित करके बदलती परिस्थितियों या आवश्यकताओं के अनुकूल हो सकती है।

मल्टी-एजेंट प्रणालियां उन अनुप्रयोगों के लिए उपयुक्त हैं जिन्हें वितरित समस्या समाधान की आवश्यकता होती है, जैसे आपूर्ति श्रृंखला अनुकूलन, यातायात प्रबंधन, या आपातकालीन प्रतिक्रिया योजना

वार्तालाप AI

वार्तालाप AI प्रणालियां उपयोगकर्ताओं और बुद्धिमान एजेंटों के बीच प्राकृतिक भाषा में बातचीत को सक्षम बनाती हैं। ये प्रणालियां आकर्षक और व्यक्तिगत वार्तालाप अनुभव प्रदान करने के लिए प्राकृतिक भाषा समझ, संवाद प्रबंधन और भाषा निर्माण क्षमताओं को जोड़ती हैं।

एक वार्तालाप AI प्रणाली के मुख्य घटकों में शामिल हैं:

- **आशय पहचान:** उपयोगकर्ता के इनपुट के आधार पर उनके आशय की पहचान करना, जैसे प्रश्न पूछना, अनुरोध करना, या भावना व्यक्त करना।
- **एंटिटी निष्कर्षण:** उपयोगकर्ता के इनपुट से प्रासंगिक एंटिटी या पैरामीटर निकालना, जैसे तिथियां, स्थान, या उत्पाद के नाम।
- **संवाद प्रबंधन:** वार्तालाप की स्थिति को बनाए रखना, उपयोगकर्ता के आशय और संदर्भ के आधार पर उचित प्रतिक्रिया का निर्धारण करना, और बहु-चरण वार्तालाप को संभालना।
- **प्रतिक्रिया उत्पादन:** भाषा मॉडल, टेम्पलेट्स, या पुनर्प्राप्ति-आधारित विधियों का उपयोग करके मानव जैसी प्रतिक्रियाएं उत्पन्न करना।

संवादात्मक एआई सिस्टम आमतौर पर ग्राहक सेवा चैटबॉट, वर्चुअल सहायक, और आवाज-नियंत्रित इंटरफेस में उपयोग किए जाते हैं। जैसा कि पहले बताया गया है, इस पुस्तक में अधिकांश दृष्टिकोण, पैटर्न और कोड उदाहरण सीधे Olympia नामक एक बड़े संवादात्मक एआई सिस्टम पर मेरे काम से निकाले गए हैं।

कोपायलट्स

कोपायलट्स एआई-संचालित सहायक हैं जो उपयोगकर्ताओं की उत्पादकता और निर्णय लेने की क्षमताओं को बढ़ाने के लिए उनके साथ काम करते हैं। ये सिस्टम बुद्धिमान

सिफारिशें प्रदान करने, कार्यों को स्वचालित करने और संदर्भगत सहायता प्रदान करने के लिए प्राकृतिक भाषा प्रसंस्करण, मशीन लर्निंग और डोमेन-विशिष्ट ज्ञान के संयोजन का लाभ उठाते हैं।

कोपायलट्ट की प्रमुख विशेषताएं हैं:

- **व्यक्तिगतकरण:** व्यक्तिगत उपयोगकर्ता प्राथमिकताओं, कार्यप्रवाह और संचार शैलियों के अनुकूल होना।
- **सक्रिय सहायता:** उपयोगकर्ता की जरूरतों का अनुमान लगाना और स्पष्ट प्रॉम्प्ट के बिना प्रासंगिक सुझाव या कार्रवाई प्रदान करना।
- **निरंतर सीखना:** उपयोगकर्ता प्रतिक्रिया, इंटरैक्शन और डेटा से सीखकर समय के साथ प्रदर्शन में सुधार करना।

कोपायलट्ट का उपयोग विभिन्न क्षेत्रों में बढ़ता जा रहा है, जैसे सॉफ्टवेयर विकास (उदाहरण के लिए, कोड पूर्णता और बग का पता लगाना), रचनात्मक लेखन (उदाहरण के लिए, सामग्री सुझाव और संपादन), और डेटा विश्लेषण (उदाहरण के लिए, अंतर्दृष्टि और विजुअलाइजेशन सिफारिशें)

ये परिनियोजन पैटर्न कंपाउंड एआई सिस्टम की बहुमुखी प्रतिभा और क्षमता को प्रदर्शित करते हैं। प्रत्येक पैटर्न की विशेषताओं और उपयोग के मामलों को समझकर, आप बुद्धिमान एप्लिकेशन को डिज़ाइन और कार्यान्वित करते समय सूचित निर्णय ले सकते हैं। हालांकि यह पुस्तक विशेष रूप से कंपाउंड एआई सिस्टम के कार्यान्वयन के बारे में नहीं है, लेकिन अन्यथा पारंपरिक एप्लिकेशन विकास के भीतर विवेकपूर्ण एआई घटकों को एकीकृत करने के लिए समान दृष्टिकोण और पैटर्न लागू होते हैं।

कंपाउंड एआई सिस्टम में भूमिकाएं

कंपाउंड एआई सिस्टम परस्पर जुड़े मॉड्यूल की नींव पर बनाए जाते हैं, जिनमें से प्रत्येक एक विशिष्ट भूमिका निभाने के लिए डिज़ाइन किया गया है। ये मॉड्यूल बुद्धिमान व्यवहार बनाने और जटिल समस्याओं को हल करने के लिए मिलकर काम करते हैं। जब आप यह सोच रहे हों कि आप अपने एप्लिकेशन के किन हिस्सों को विवेकपूर्ण

एआई घटकों से कार्यान्वित या प्रतिस्थापित कर सकते हैं, तो इन भूमिकाओं से परिचित होना उपयोगी है।

जनरेटर

जनरेटर सीखे गए पैटर्न या इनपुट प्रॉम्प्ट के आधार पर नया डेटा या सामग्री उत्पन्न करने के लिए जिम्मेदार होते हैं। एआई की दुनिया में कई अलग-अलग प्रकार के जनरेटर हैं, लेकिन इस पुस्तक में प्रदर्शित भाषा मॉडल के संदर्भ में, जनरेटर मानव जैसा टेक्स्ट बना सकते हैं, आंशिक वाक्यों को पूरा कर सकते हैं, या उपयोगकर्ता के प्रश्नों के उत्तर उत्पन्न कर सकते हैं। वे सामग्री निर्माण, संवाद उत्पादन और डेटा संवर्धन जैसे कार्यों में महत्वपूर्ण भूमिका निभाते हैं।

रिट्रीवर

रिट्रीवर का उपयोग बड़े डेटासेट या ज्ञान-आधार से प्रासंगिक जानकारी खोजने और निकालने के लिए किया जाता है। वे दिए गए प्रश्न या संदर्भ के आधार पर सबसे उपयुक्त डेटा पॉइंट्स खोजने के लिए सिमैंटिक सर्च, कीवर्ड मैचिंग, या वेक्टर समानता जैसी तकनीकों का उपयोग करते हैं। रिट्रीवर उन कार्यों के लिए आवश्यक हैं जिनमें विशिष्ट जानकारी तक त्वरित पहुंच की आवश्यकता होती है, जैसे प्रश्नोत्तर, तथ्य-जांच, या सामग्री अनुशंसा।

रैंकर

रैंकर कुछ मानदंडों या प्रासंगिकता स्कोर के आधार पर वस्तुओं के समूह को क्रमबद्ध या प्राथमिकता देने के लिए जिम्मेदार होते हैं। वे प्रत्येक वस्तु को वेट या स्कोर असाइन करते हैं और फिर उन्हें तदनुसार क्रमबद्ध करते हैं। रैंकर का उपयोग आमतौर पर सर्च इंजन, अनुशंसा प्रणालियों, या किसी भी ऐसे एप्लिकेशन में किया जाता है जहां उपयोगकर्ताओं को सबसे प्रासंगिक परिणाम प्रस्तुत करना महत्वपूर्ण होता है।

क्लासिफायर

क्लासिफायर का उपयोग पूर्व-निर्धारित श्रेणियों या वर्गों के आधार पर डेटा पॉइंट्स को वर्गीकृत या लेबल करने के लिए किया जाता है। वे लेबल किए गए प्रशिक्षण डेटा से सीखते हैं और फिर नए, अनदेखे उदाहरणों की श्रेणी की भविष्यवाणी करते हैं। क्लासिफायर भावना विश्लेषण, स्पैम पहचान, या छवि पहचान जैसे कार्यों के लिए मौलिक हैं, जहां प्रत्येक इनपुट को एक विशिष्ट श्रेणी में वर्गीकृत करना लक्ष्य होता है।

टूल्स और एजेंट्स

इन मुख्य भूमिकाओं के अलावा, मिश्रित एआई सिस्टम अक्सर अपनी कार्यक्षमता और अनुकूलन क्षमता को बढ़ाने के लिए टूल्स और एजेंट्स को शामिल करते हैं:

- **टूल्स:** टूल्स विशिष्ट क्रियाओं या गणनाओं को करने वाले अलग-अलग सॉफ्टवेयर घटक या एपीआई होते हैं। इन्हें अन्य मॉड्यूल्स द्वारा, जैसे जनरेटर या रिट्रीवर द्वारा, उप-कार्यों को पूरा करने या अतिरिक्त जानकारी एकत्र करने के लिए उपयोग किया जा सकता है। टूल्स के उदाहरणों में वेब सर्च इंजन, कैलकुलेटर, या डेटा विज़ुअलाइज़ेशन लाइब्रेरी शामिल हैं।
- **एजेंट्स:** एजेंट्स स्वायत्त इकाइयां हैं जो अपने वातावरण को समझ सकती हैं, निर्णय ले सकती हैं, और विशिष्ट लक्ष्यों को प्राप्त करने के लिए कार्रवाई कर सकती हैं। वे गतिशील या अनिश्चित परिस्थितियों में प्रभावी ढंग से काम करने के लिए अक्सर योजना बनाने, तर्क करने और सीखने जैसी विभिन्न एआई तकनीकों के संयोजन पर निर्भर करते हैं। एजेंट्स का उपयोग जटिल व्यवहारों को मॉडल करने या मिश्रित एआई सिस्टम के भीतर कई मॉड्यूल्स की क्रियाओं का समन्वय करने के लिए किया जा सकता है।

एक शुद्ध मिश्रित एआई सिस्टम में, इन घटकों के बीच संवाद सुपरिभाषित इंटरफेस और संचार प्रोटोकॉल के माध्यम से संचालित किया जाता है। डेटा मॉड्यूल्स के बीच प्रवाहित होता है, जहां एक घटक का आउटपुट दूसरे के लिए इनपुट के रूप में काम करता है। यह मॉड्यूलर आर्किटेक्चर लचीलापन, स्केलेबिलिटी और रखरखाव की

क्षमता प्रदान करता है, क्योंकि व्यक्तिगत घटकों को पूरे सिस्टम को प्रभावित किए बिना अपडेट, बदला, या विस्तारित किया जा सकता है।

इन घटकों और उनकी अंतःक्रियाओं की शक्ति का लाभ उठाकर, मिश्रित एआई सिस्टम जटिल, वास्तविक-दुनिया की समस्याओं को हल कर सकते हैं जिनके लिए विभिन्न एआई क्षमताओं के संयोजन की आवश्यकता होती है। जैसे-जैसे हम एप्लिकेशन विकास में एआई को एकीकृत करने के दृष्टिकोण और पैटर्न का पता लगाते हैं, ध्यान रखें कि मिश्रित एआई सिस्टम में उपयोग किए जाने वाले समान सिद्धांतों और तकनीकों का उपयोग बुद्धिमान, अनुकूलनीय और उपयोगकर्ता-केंद्रित एप्लिकेशन बनाने के लिए किया जा सकता है।

भाग 1 के निम्नलिखित अध्यायों में, हम आपकी एप्लिकेशन विकास प्रक्रिया में AI कॉम्पोनेंट्स को एकीकृत करने के लिए मूलभूत दृष्टिकोण और तकनीकों में गहराई से जाएंगे। प्रॉम्प्ट इंजीनियरिंग और रिट्रीवल-ऑगमेंटेड जेनरेशन से लेकर स्व-उपचारी डेटा और बुद्धिमान वर्कफ़्लो ऑर्केस्ट्रेशन तक, हम विभिन्न पैटर्न और सर्वोत्तम प्रथाओं को कवर करेंगे जो आपको अत्याधुनिक AI-संचालित एप्लिकेशन बनाने में मदद करेंगे।

भाग 1: मौलिक दृष्टिकोण और तकनीकें

पुस्तक का यह भाग आपके एप्लिकेशन में AI के उपयोग को एकीकृत करने के विभिन्न तरीके प्रस्तुत करता है। अध्याय संबंधित दृष्टिकोणों और तकनीकों की एक श्रृंखला को कवर करते हैं, जो उच्च-स्तरीय अवधारणाओं जैसे पथ को संकीर्ण करना और पुनर्प्राप्ति संवर्धित जनन से लेकर LLM चैट पूर्णता APIs के ऊपर अपनी खुद की अमूर्तन परत को प्रोग्राम करने के विचारों तक विस्तृत हैं।

पुस्तक के इस भाग का उद्देश्य आपको उन व्यवहारों को समझने में मदद करना है जिन्हें आप AI के साथ लागू कर सकते हैं, इससे पहले कि आप विशिष्ट कार्यान्वयन पैटर्न में गहराई से जाएं जो भाग 2 का फोकस हैं।

भाग 1 में दिए गए दृष्टिकोण मेरे कोड में उपयोग किए गए विचारों, उद्यम अनुप्रयोग वास्तुकला और एकीकरण के क्लासिक पैटर्न, साथ ही उन रूपकों पर आधारित हैं जिनका मैंने अन्य लोगों को AI की क्षमताओं को समझाने में उपयोग किया है, जिसमें गैर-तकनीकी व्यावसायिक हितधारक भी शामिल हैं।

पथ को संकीर्ण करें

“पथ को संकीर्ण करना” का अर्थ है AI को वर्तमान कार्य पर केंद्रित करना। जब भी मैं AI के “मूर्खतापूर्ण” या अप्रत्याशित व्यवहार से निराश होता हूं, तो मैं इसे एक मंत्र के रूप में उपयोग करता हूं। यह मंत्र मुझे याद दिलाता है कि विफलता संभवतः मेरी गलती है, और मुझे शायद पथ को और संकीर्ण करना चाहिए।

पथ को संकीर्ण करने की आवश्यकता बृहत भाषा मॉडल्स में निहित विशाल ज्ञान से उत्पन्न होती है, विशेष रूप से OpenAI और Anthropic जैसी विश्व-स्तरीय कंपनियों के मॉडल्स में, जिनमें शाब्दिक रूप से खरबों पैरामीटर्स होते हैं।

इतने व्यापक ज्ञान तक पहुंच निःसंदेह शक्तिशाली है और मन का सिद्धांत जैसे उभरते व्यवहार और मानव जैसे तरीकों से तर्क करने की क्षमता उत्पन्न करती है। हालांकि, जानकारी की यह अभूतपूर्व मात्रा विशिष्ट प्रॉम्प्ट्स के लिए सटीक और यथार्थ प्रतिक्रियाएं उत्पन्न करने में चुनौतियां भी प्रस्तुत करती है, विशेष रूप से यदि उन प्रॉम्प्ट्स का उद्देश्य निर्धारणात्मक व्यवहार प्रदर्शित करना है जिसे "सामान्य" सॉफ्टवेयर विकास और एल्गोरिदम के साथ एकीकृत किया जा सके।

कई कारक इन चुनौतियों को जन्म देते हैं।

सूचना अतिभार: बृहत भाषा मॉडल्स को विभिन्न क्षेत्रों, स्रोतों और समय अवधियों में फैले विशाल डेटा पर प्रशिक्षित किया जाता है। यह व्यापक ज्ञान उन्हें विविध विषयों पर संलग्न होने और दुनिया की व्यापक समझ के आधार पर प्रतिक्रियाएं उत्पन्न करने की अनुमति देता है। हालांकि, एक विशिष्ट प्रॉम्प्ट का सामना करते समय, मॉडल अप्रासंगिक, विरोधाभासी, या पुरानी/अप्रचलित जानकारी को छानने में संघर्ष कर सकता है, जिससे फोकस या सटीकता की कमी वाली प्रतिक्रियाएं मिलती हैं। आप जो करने की कोशिश कर रहे हैं, उसके आधार पर, मॉडल के लिए उपलब्ध विरोधाभासी जानकारी की विशाल मात्रा आसानी से आपके वांछित उत्तर या व्यवहार प्रदान करने की उसकी क्षमता को अभिभूत कर सकती है।

संदर्भगत अस्पष्टता: विशाल अव्यक्त स्थान के ज्ञान को देखते हुए, बृहत भाषा मॉडल आपके प्रॉम्प्ट के संदर्भ को समझने का प्रयास करते समय अस्पष्टता का सामना कर सकते हैं। उचित संकीर्णता या मार्गदर्शन के बिना, मॉडल ऐसी प्रतिक्रियाएं उत्पन्न कर सकता है जो अप्रत्यक्ष रूप से संबंधित हैं लेकिन आपके इरादों के लिए सीधे प्रासंगिक नहीं हैं। इस तरह की विफलता से विषय से हटी हुई, असंगत, या आपकी बताई गई जरूरतों को पूरा न करने वाली प्रतिक्रियाएं मिलती हैं। इस मामले में, पथ को संकीर्ण करने का अर्थ है संदर्भ का स्पष्टीकरण, यह सुनिश्चित करना कि आपके द्वारा प्रदान किया गया संदर्भ मॉडल को केवल उसके आधार ज्ञान में सबसे प्रासंगिक जानकारी पर ध्यान केंद्रित करने का कारण बनता है।

नोट: जब आप "प्रॉम्प्ट इंजीनियरिंग" में शुरुआत कर रहे होते हैं, तो आप अक्सर वांछित परिणाम को ठीक से समझाए बिना मॉडल से कार्य करवाने की कोशिश करते हैं; अस्पष्ट न होने के लिए अभ्यास की आवश्यकता होती है!

कालिक असंगतियाँ: चूंकि भाषा मॉडल विभिन्न समय अवधियों में बनाए गए डेटा पर प्रशिक्षित होते हैं, उनके पास ऐसी जानकारी हो सकती है जो पुरानी, अप्रचलित या अब सटीक नहीं है। उदाहरण के लिए, वर्तमान घटनाओं, वैज्ञानिक खोजों, या तकनीकी प्रगति के बारे में जानकारी मॉडल के प्रशिक्षण डेटा के संग्रह के बाद विकसित हो सकती है। अधिक हाल के और विश्वसनीय स्रोतों को प्राथमिकता देने के लिए मार्ग को संकीर्ण किए बिना, मॉडल पुरानी या गलत जानकारी के आधार पर प्रतिक्रियाएं उत्पन्न कर सकता है, जिससे इसके आउटपुट में अशुद्धियां और असंगतियां आ सकती हैं।

क्षेत्र-विशिष्ट बारीकियाँ: विभिन्न क्षेत्रों और विषयों की अपनी विशिष्ट शब्दावली, परंपराएं और ज्ञान आधार होते हैं। किसी भी TLA (तीन अक्षरों का संक्षिप्त नाम) के बारे में सोचें और आप पाएंगे कि अधिकांश के एक से अधिक अर्थ हैं। उदाहरण के लिए, MSK का अर्थ Amazon का मैनेज्ड स्ट्रीमिंग फॉर अपाचे काफ्का, मेमोरियल स्लोन केटरिंग कैंसर सेंटर, या मानव मस्कुलोस्केलेटल सिस्टम हो सकता है।

जब किसी प्रॉम्प्ट को किसी विशेष क्षेत्र में विशेषज्ञता की आवश्यकता होती है, तो एक बड़े भाषा मॉडल का सामान्य ज्ञान सटीक और सूक्ष्म प्रतिक्रियाएं प्रदान करने के लिए पर्याप्त नहीं हो सकता। प्रॉम्प्ट इंजीनियरिंग या पुनर्प्राप्ति-संवर्धित उत्पादन के माध्यम से क्षेत्र-विशिष्ट जानकारी पर ध्यान केंद्रित करके मार्ग को संकीर्ण करना, मॉडल को आपके विशिष्ट क्षेत्र की आवश्यकताओं और अपेक्षाओं के अनुरूप प्रतिक्रियाएं उत्पन्न करने में सक्षम बनाता है।

लेटेंट स्पेस: अकल्पनीय विशाल

जब मैं भाषा मॉडल के "लेटेंट स्पेस" का उल्लेख करता हूं, तो मैं ज्ञान और सूचना के उस विशाल, बहु-आयामी परिदृश्य की बात कर रहा हूं जो मॉडल ने अपनी प्रशिक्षण

प्रक्रिया के दौरान सीखा है। यह मॉडल के तंत्रिका नेटवर्क के भीतर एक छिपा हुआ क्षेत्र की तरह है, जहां भाषा के सभी पैटर्न, संबंध और प्रतिनिधित्व संग्रहीत हैं।

कल्पना कीजिए कि आप एक विशाल, अज्ञात क्षेत्र की खोज कर रहे हैं जो असंख्य परस्पर जुड़े नोड्स से भरा हुआ है। प्रत्येक नोड एक जानकारी का टुकड़ा, एक अवधारणा, या एक संबंध का प्रतिनिधित्व करता है जो मॉडल ने सीखा है। जैसे-जैसे आप इस स्थान में नेविगेट करते हैं, आप पाएंगे कि कुछ नोड्स एक-दूसरे के करीब हैं, जो एक मजबूत कनेक्शन या समानता को दर्शाता है, जबकि अन्य दूर हैं, जो एक कमजोर या अधिक दूर का संबंध सुझाते हैं।

लेटेंट स्पेस की चुनौती यह है कि यह अत्यंत जटिल और उच्च-आयामी है। इसे हमारे भौतिक ब्रह्मांड जितना विशाल समझें, जिसमें आकाशगंगाओं के समूह और उनके बीच खाली स्थान की अकल्पनीय दूरियां हैं।

चूंकि इसमें हजारों आयाम होते हैं, अव्यक्त स्थान मनुष्यों द्वारा सीधे देखा या समझा नहीं जा सकता। यह एक अमूर्त प्रतिनिधित्व है जिसका उपयोग मॉडल भाषा को संसाधित करने और उत्पन्न करने के लिए आंतरिक रूप से करता है। जब आप मॉडल को एक इनपुट प्रॉम्प्ट प्रदान करते हैं, तो यह अनिवार्य रूप से उस प्रॉम्प्ट को अव्यक्त स्थान के भीतर एक विशिष्ट स्थान पर मैप करता है। फिर मॉडल प्रतिक्रिया उत्पन्न करने के लिए उस स्थान में आस-पास की जानकारी और कनेक्शन का उपयोग करता है।

बात यह है कि मॉडल ने अपने प्रशिक्षण डेटा से बहुत अधिक जानकारी सीखी है, और यह सब किसी दिए गए कार्य के लिए प्रासंगिक या सटीक नहीं है। इसलिए मार्ग को संकीर्ण करना बहुत महत्वपूर्ण हो जाता है। अपने प्रॉम्प्ट्स में स्पष्ट निर्देश, उदाहरण और संदर्भ प्रदान करके, आप अनिवार्य रूप से मॉडल को अव्यक्त स्थान के भीतर विशिष्ट क्षेत्रों पर ध्यान केंद्रित करने के लिए मार्गदर्शन कर रहे हैं जो आपके वांछित आउटपुट के लिए सबसे अधिक प्रासंगिक हैं।

इसे समझने का एक अलग तरीका है जैसे पूरी तरह से अंधेरे संग्रहालय में स्पॉटलाइट का उपयोग करना। अगर आपने कभी लूव्र या मेट्रोपॉलिटन म्यूजियम ऑफ आर्ट का दौरा किया है, तो यही वह पैमाना है जिसकी मैं बात कर रहा हूं। अव्यक्त स्थान संग्रहालय की तरह है, जो असंख्य वस्तुओं और विवरणों से भरा हुआ है। आपका

प्रॉम्प्ट स्पॉटलाइट की तरह है, जो विशिष्ट क्षेत्रों को प्रकाशित करता है और सबसे महत्वपूर्ण जानकारी की ओर मॉडल का ध्यान आकर्षित करता है। उस मार्गदर्शन के बिना, मॉडल अव्यक्त स्थान में बिना उद्देश्य के भटक सकता है, रास्ते में अप्रासंगिक या विरोधाभासी जानकारी एकत्र करता हुआ।

जैसे-जैसे आप भाषा मॉडल्स के साथ काम करते हैं और अपने प्रॉम्प्ट्स तैयार करते हैं, अव्यक्त स्थान की अवधारणा को ध्यान में रखें। आपका लक्ष्य इस विशाल ज्ञान परिदृश्य में प्रभावी ढंग से नेविगेट करना है, मॉडल को आपके कार्य के लिए सबसे प्रासंगिक और सटीक जानकारी की ओर निर्देशित करना है। मार्ग को संकीर्ण करके और स्पष्ट मार्गदर्शन प्रदान करके, आप मॉडल के अव्यक्त स्थान की पूरी क्षमता को अनलॉक कर सकते हैं और उच्च-गुणवत्ता वाली, सुसंगत प्रतिक्रियाएं उत्पन्न कर सकते हैं।

हालांकि भाषा मॉडल्स और उनके द्वारा नेविगेट किए जाने वाले अव्यक्त स्थान के पिछले विवरण थोड़े जादुई या अमूर्त लग सकते हैं, यह समझना महत्वपूर्ण है कि प्रॉम्प्ट्स जादुई मंत्र या टोटके नहीं हैं। भाषा मॉडल्स जिस तरह से काम करते हैं वह रैखिक बीजगणित और संभावना सिद्धांत के सिद्धांतों पर आधारित है।

मूल रूप से, भाषा मॉडल पाठ के संभाव्यता मॉडल हैं, बिल्कुल वैसे ही जैसे बेल कर्व डेटा का सांख्यिकीय मॉडल होता है। इन्हें स्व-प्रतिगामी मॉडलिंग नामक प्रक्रिया के माध्यम से प्रशिक्षित किया जाता है, जहां मॉडल पिछले शब्दों के आधार पर क्रम में अगले शब्द की संभावना की भविष्यवाणी करना सीखता है। प्रशिक्षण के दौरान, मॉडल यादृच्छिक वेट्स से शुरू होता है और धीरे-धीरे उन्हें समायोजित करता है ताकि वास्तविक दुनिया के नमूनों, जिन पर इसे प्रशिक्षित किया गया था, के समान पाठ को उच्च संभावना प्रदान की जा सके।

हालाँकि, भाषा मॉडल को सरल सांख्यिकीय मॉडल की तरह सोचना, जैसे रैखिक प्रतिगमन, उनके व्यवहार को समझने के लिए सबसे अच्छी अंतर्दृष्टि प्रदान नहीं करता। एक बेहतर उपमा यह है कि उन्हें प्रायिकता आधारित प्रोग्राम के रूप में सोचा जाए, जो ऐसे मॉडल हैं जो यादृच्छिक चर के साथ काम करने की अनुमति देते हैं और जटिल सांख्यिकीय संबंधों को प्रस्तुत कर सकते हैं।

प्रायिकता आधारित प्रोग्राम को आरेखीय मॉडल द्वारा दर्शाया जा सकता है, जो मॉडल

में चर के बीच निर्भरता और संबंधों को समझने का एक दृश्य तरीका प्रदान करते हैं। यह दृष्टिकोण GPT-4 और Claude जैसे जटिल पाठ निर्माण मॉडल की कार्यप्रणाली में मूल्यवान अंतर्दृष्टि प्रदान कर सकता है।

Dohan और अन्य द्वारा लिखित "Language Model Cascades" शोधपत्र में, लेखकों ने भाषा मॉडल में प्रायिकता आधारित प्रोग्राम को कैसे लागू किया जा सकता है, इसके विवरण में गहराई से जाते हैं। वे दिखाते हैं कि इस ढांचे का उपयोग इन मॉडल के व्यवहार को समझने और अधिक प्रभावी प्रॉम्प्टिंग रणनीतियों के विकास को मार्गदर्शित करने के लिए कैसे किया जा सकता है।

इस प्रायिकता दृष्टिकोण से एक प्रमुख अंतर्दृष्टि यह है कि भाषा मॉडल अनिवार्य रूप से एक वैकल्पिक ब्रह्मांड का द्वार बनाता है जहाँ वांछित दस्तावेज मौजूद हैं। मॉडल सभी संभावित दस्तावेजों को उनकी संभावना के आधार पर भार प्रदान करता है, प्रभावी ढंग से संभावनाओं के स्थान को सबसे प्रासंगिक पर केंद्रित करने के लिए संकुचित करता है।

यह हमें "पथ को संकीर्ण करने" के केंद्रीय विषय पर वापस लाता है। प्रॉम्प्टिंग का प्राथमिक लक्ष्य प्रायिकता मॉडल को इस तरह से नियंत्रित करना है जो उसकी भविष्यवाणियों के द्रव्यमान को केंद्रित करता है, जिस विशिष्ट जानकारी या व्यवहार को हम प्राप्त करना चाहते हैं उस पर ध्यान केंद्रित करता है। सावधानीपूर्वक तैयार किए गए प्रॉम्प्ट प्रदान करके, हम मॉडल को अव्यक्त स्थान में अधिक कुशलता से नेविगेट करने और अधिक प्रासंगिक और सुसंगत आउटपुट उत्पन्न करने के लिए मार्गदर्शन कर सकते हैं।

हालाँकि, यह ध्यान में रखना महत्वपूर्ण है कि भाषा मॉडल अंततः उस जानकारी से सीमित है जिस पर इसे प्रशिक्षित किया गया था। जबकि यह मौजूदा दस्तावेजों के समान पाठ उत्पन्न कर सकता है या विचारों को नए तरीकों से जोड़ सकता है, यह पूरी तरह से नई जानकारी को शून्य से नहीं बना सकता। उदाहरण के लिए, हम मॉडल से कैंसर का इलाज प्रदान करने की उम्मीद नहीं कर सकते यदि ऐसा इलाज खोजा और उसके प्रशिक्षण डेटा में दस्तावेज़ीकृत नहीं किया गया है।

इसके बजाय, मॉडल की ताकत उस जानकारी को खोजने और संश्लेषित करने की क्षमता में निहित है जो हमारे द्वारा दिए गए प्रॉम्प्ट के समान है। इन मॉडल की

प्रायिकता प्रकृति और कैसे प्रॉम्प्ट का उपयोग उनके आउटपुट को नियंत्रित करने के लिए किया जा सकता है, इसे समझकर, हम मूल्यवान अंतर्दृष्टि और सामग्री उत्पन्न करने के लिए उनकी क्षमताओं का अधिक प्रभावी ढंग से लाभ उठा सकते हैं।

नीचे दिए गए प्रॉम्प्ट पर विचार करें। पहले में, अकेले "Mercury" ग्रह, तत्व, या रोमन देवता को संदर्भित कर सकता है, लेकिन सबसे अधिक संभावना ग्रह की है। वास्तव में, GPT-4 एक लंबी प्रतिक्रिया देता है जो Mercury सौर मंडल का सबसे छोटा और सबसे भीतरी ग्रह है... से शुरू होती है। दूसरा प्रॉम्प्ट विशेष रूप से रासायनिक तत्व को संदर्भित करता है। तीसरा रोमन पौराणिक आकृति को संदर्भित करता है, जो अपनी गति और दिव्य संदेशवाहक की भूमिका के लिए जाना जाता है।

```
# Prompt 1
Tell me about: Mercury

# Prompt 2
Tell me about: Mercury element

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

बस कुछ अतिरिक्त शब्दों को जोड़कर, हमने एआई की प्रतिक्रिया को पूरी तरह से बदल दिया है। जैसा कि आप पुस्तक में बाद में सीखेंगे, एन-शॉट प्रॉम्प्टिंग, संरचित इनपुट/आउटपुट, और विचार श्रृंखला जैसी उन्नत प्रॉम्प्ट इंजीनियरिंग तकनीकें मॉडल के आउटपुट को नियंत्रित करने के चतुर तरीके मात्र हैं।

तो अंततः, प्रॉम्प्ट इंजीनियरिंग की कला भाषा मॉडल के ज्ञान के विशाल संभाव्यता परिदृश्य में नेविगेट करने की समझ के बारे में है, ताकि हम जिस विशिष्ट जानकारी या व्यवहार की तलाश कर रहे हैं, उस तक का मार्ग संकीर्ण किया जा सके।

उन पाठकों के लिए जिनकी उन्नत गणित पर मजबूत पकड़ है, इन मॉडलों की अपनी समझ को संभाव्यता सिद्धांत और रैखिक बीजगणित के सिद्धांतों में स्थापित करना निश्चित रूप से मददगार हो सकता है! आप में से बाकी लोगों के लिए जो वांछित परिणाम प्राप्त करने के लिए प्रभावी रणनीतियां विकसित करना चाहते हैं, आइए अधिक सहज दृष्टिकोण पर टिके रहें।

मार्ग कैसे “संकीर्ण” होता है

अत्यधिक ज्ञान की इन चुनौतियों का समाधान करने के लिए, हम ऐसी तकनीकों का उपयोग करते हैं जो भाषा मॉडल की जनन प्रक्रिया को मार्गदर्शित करने और सबसे प्रासंगिक और सटीक जानकारी पर इसका ध्यान केंद्रित करने में मदद करती हैं।

यहाँ सबसे महत्वपूर्ण तकनीकें हैं, अनुशंसित क्रम में, यानी आपको पहले प्रॉम्प्ट इंजीनियरिंग का प्रयास करना चाहिए, फिर RAG, और फिर अंत में, यदि जरूरी हो, तो फाइन ट्यूनिंग।

प्रॉम्प्ट इंजीनियरिंग सबसे मौलिक दृष्टिकोण है जिसमें मॉडल की प्रतिक्रिया जनन को निर्देशित करने के लिए विशिष्ट निर्देश, प्रतिबंध, या उदाहरण शामिल करने वाले प्रॉम्प्ट तैयार किए जाते हैं। यह अध्याय अगले खंड में प्रॉम्प्ट इंजीनियरिंग के मूल सिद्धांतों को कवर करता है, और हम पुस्तक के भाग 2 में कई विशिष्ट प्रॉम्प्ट इंजीनियरिंग पैटर्न को कवर करते हैं। इन पैटर्न में प्रॉम्प्ट आसवन शामिल है, जो एआई द्वारा सबसे प्रासंगिक और संक्षिप्त मानी जाने वाली जानकारी को निकालने के लिए प्रॉम्प्ट को परिष्कृत और अनुकूलित करने की एक तकनीक है।

संदर्भ संवर्धन। प्रॉम्प्ट के समय मॉडल को केंद्रित संदर्भ प्रदान करने के लिए बाहरी ज्ञान आधार या दस्तावेजों से प्रासंगिक जानकारी को गतिशील रूप से पुनर्प्राप्त करना। लोकप्रिय संदर्भ संवर्धन तकनीकों में पुनर्प्राप्ति-संवर्धित जनन (RAG) शामिल है। तथाकथित “ऑनलाइन मॉडल” जैसे Perplexity द्वारा प्रदान किए गए मॉडल वास्तविक समय के इंटरनेट खोज परिणामों के साथ अपने संदर्भ को संवर्धित करने में सक्षम हैं।

अपनी शक्ति के बावजूद, एलएलएम आपके विशिष्ट डेटासेट पर प्रशिक्षित नहीं होते हैं, जो निजी हो सकते हैं या जिस समस्या को आप हल करने की कोशिश कर रहे हैं उसके लिए विशिष्ट हो सकते हैं। संदर्भ संवर्धन तकनीकें एलएलएम को एपीआई के पीछे, एसक्यूएल डेटाबेस में, या पीडीएफ और स्लाइड डेक में फंसे डेटा तक पहुंच प्रदान करती हैं।

फाइन-ट्यूनिंग या डोमेन अनुकूलन किसी विशेष कार्य या क्षेत्र के लिए अपने ज्ञान और जनरेशन क्षमताओं को विशेषज्ञ बनाने के लिए डोमेन-विशिष्ट डेटासेट पर मॉडल को प्रशिक्षित करना।

तापमान को कम करना

तापमान एक हाइपरपैरामीटर है जो ट्रांसफॉर्मर-आधारित भाषा मॉडल में उत्पन्न टेक्स्ट की यादृच्छिकता और रचनात्मकता को नियंत्रित करता है। यह 0 और 1 के बीच का मान है, जहां निम्न मान आउटपुट को अधिक केंद्रित और निर्धारणात्मक बनाते हैं, जबकि उच्च मान इसे अधिक विविध और अप्रत्याशित बनाते हैं।

जब तापमान 1 पर सेट किया जाता है, तो भाषा मॉडल अगले टोकन के पूर्ण संभाव्यता वितरण के आधार पर टेक्स्ट उत्पन्न करता है, जो अधिक रचनात्मक और विविध प्रतिक्रियाओं की अनुमति देता है। हालांकि, इससे मॉडल ऐसा टेक्स्ट भी उत्पन्न कर सकता है जो कम प्रासंगिक या सुसंगत हो।

दूसरी ओर, जब तापमान 0 पर सेट किया जाता है, तो भाषा मॉडल हमेशा उच्चतम संभावना वाले टोकन का चयन करता है, प्रभावी रूप से "अपना मार्ग संकीर्ण करता है।" मेरे लगभग सभी एआई घटक 0 पर या उसके करीब सेट तापमान का उपयोग करते हैं, क्योंकि इससे अधिक केंद्रित और अनुमानयोग्य प्रतिक्रियाएं मिलती हैं। यह बिल्कुल उपयोगी है जब आप चाहते हैं कि मॉडल निर्देशों का पालन करे, उसे प्रदान किए गए फ़ंक्शंस पर ध्यान दे, या बस आपको जो मिल रहा है उससे अधिक सटीक और प्रासंगिक प्रतिक्रियाओं की आवश्यकता हो।

उदाहरण के लिए, यदि आप एक चैटबोट बना रहे हैं जिसे तथ्यात्मक जानकारी प्रदान करने की आवश्यकता है, तो आप प्रतिक्रियाओं को अधिक सटीक और विषय-केंद्रित बनाने के लिए तापमान को कम मान पर सेट करना चाह सकते हैं। इसके विपरीत, यदि आप एक रचनात्मक लेखन सहायक बना रहे हैं, तो आप अधिक विविध और कल्पनाशील आउटपुट को प्रोत्साहित करने के लिए तापमान को उच्च मान पर सेट करना चाह सकते हैं।

हाइपरपैरामीटर्स: अनुमान के नॉब्स और डायल्स

जब आप भाषा मॉडल के साथ काम कर रहे होते हैं, तो आप "हाइपरपैरामीटर्स" शब्द का सामना काफी बार करेंगे। अनुमान के संदर्भ में (यानी, जब आप प्रतिक्रियाएं उत्पन्न करने के लिए मॉडल का उपयोग कर रहे हैं), हाइपरपैरामीटर्स उन नॉब्स और डायल्स की तरह हैं जिन्हें आप मॉडल के व्यवहार और आउटपुट को नियंत्रित करने के लिए समायोजित कर सकते हैं।

इसे एक जटिल मशीन पर सेटिंग्स समायोजित करने की तरह सोचें। जैसे आप तापमान को नियंत्रित करने के लिए एक नॉब घुमा सकते हैं या ऑपरेशन मोड बदलने के लिए एक स्विच फ्लिप कर सकते हैं, हाइपरपैरामीटर्स आपको भाषा मॉडल द्वारा टेक्स्ट को प्रोसेस और जनरेट करने के तरीके को बारीकी से समायोजित करने की अनुमति देते हैं।

इन्फरेंस के दौरान आपको कुछ सामान्य हाइपरपैरामीटर्स मिलेंगे:

- **टेम्परेचर**: जैसा कि अभी बताया गया, यह पैरामीटर जनरेट किए गए टेक्स्ट की रैंडमनेस और क्रिएटिविटी को नियंत्रित करता है। उच्च टेम्परेचर से अधिक विविध और अप्रत्याशित आउटपुट मिलते हैं, जबकि निम्न टेम्परेचर से अधिक केंद्रित और निश्चित प्रतिक्रियाएं मिलती हैं।

- **टॉप-पी (न्यूक्लियस) सैंपलिंग:** यह पैरामीटर टोकन्स के सबसे छोटे सेट के चयन को नियंत्रित करता है जिनकी संचयी संभावना एक निश्चित सीमा (p) से अधिक होती है। यह सामंजस्य बनाए रखते हुए अधिक विविध आउटपुट की अनुमति देता है।

- **टॉप-के सैंपलिंग:** यह तकनीक k सबसे संभावित अगले टोकन्स का चयन करती है और उनके बीच संभावना वितरण को पुनर्वितरित करती है। यह मॉडल को कम संभावना वाले या अप्रासंगिक टोकन्स जनरेट करने से रोकने में मदद कर सकती है।

- **फ्रीक्वेंसी** और **प्रेजेंस पेनल्टी**: ये पैरामीटर्स मॉडल को एक ही शब्दों या वाक्यांशों को बार-बार दोहराने (फ्रीक्वेंसी पेनल्टी) या इनपुट प्रॉम्प्ट में मौजूद नहीं होने वाले शब्दों को जनरेट करने (प्रेजेंस पेनल्टी) के लिए दंडित करते हैं। इन मानों को समायोजित करके, आप मॉडल को अधिक विविध और प्रासंगिक आउटपुट उत्पन्न करने के लिए प्रोत्साहित कर सकते हैं।

- **अधिकतम लंबाई**: यह हाइपरपैरामीटर एक एकल प्रतिक्रिया में मॉडल द्वारा जनरेट किए जा सकने वाले टोकन्स (शब्द या उप-शब्द) की संख्या की ऊपरी सीमा तय करता है। यह जनरेट किए गए टेक्स्ट की वर्बोसिटी और संक्षिप्तता को नियंत्रित करने में मदद करता है।

जैसे-जैसे आप विभिन्न हाइपरपैरामीटर सेटिंग्स के साथ प्रयोग करेंगे, आप पाएंगे कि छोटे समायोजन भी मॉडल के आउटपुट पर महत्वपूर्ण प्रभाव डाल सकते हैं। यह एक व्यंजन को फाइन-ट्यून करने जैसा है - एक चुटकी अधिक नमक या थोड़ा लंबा पकाने का समय अंतिम व्यंजन में बड़ा अंतर ला सकता है।

मुख्य बात यह है कि समझें कि प्रत्येक हाइपरपैरामीटर मॉडल के व्यवहार को कैसे प्रभावित करता है और अपने विशिष्ट कार्य के लिए सही संतुलन खोजें। विभिन्न सेटिंग्स के साथ प्रयोग करने और देखने से न डरें कि वे जनरेट किए गए टेक्स्ट को कैसे प्रभावित करते हैं। समय के साथ, आप यह समझ विकसित करेंगे कि किन हाइपरपैरामीटर्स को समायोजित करना है और वांछित परिणाम कैसे प्राप्त करें।

इन पैरामीटर्स के उपयोग को प्रॉम्प्ट इंजीनियरिंग, रिट्रीवल-ऑगमेंटेड जेनरेशन, और फाइन-ट्यूनिंग के साथ जोड़कर, आप भाषा मॉडल को अधिक सटीक, प्रासंगिक, और मूल्यवान प्रतिक्रियाएं जनरेट करने के लिए प्रभावी ढंग से मार्गदर्शित कर सकते हैं।

रॉ बनाम इंस्ट्रक्ट-ट्यून्ड मॉडल्स

कच्चे मॉडल, बृहत भाषा मॉडल (एलएलएम) के अपरिष्कृत, अप्रशिक्षित संस्करण होते हैं। इन्हें एक कोरे कैनवास की तरह समझें, जो अभी तक निर्देशों को समझने या उनका

पालन करने के लिए विशिष्ट प्रशिक्षण से प्रभावित नहीं हुए हैं। ये उस विशाल डेटा पर आधारित होते हैं जिस पर वे शुरू में प्रशिक्षित किए गए थे, और विभिन्न प्रकार के आउटपुट उत्पन्न करने में सक्षम होते हैं। हालांकि, निर्देश-आधारित फाइन-ट्यूनिंग की अतिरिक्त परतों के बिना, इनकी प्रतिक्रियाएं अप्रत्याशित हो सकती हैं और वांछित परिणाम प्राप्त करने के लिए अधिक सूक्ष्म, सावधानीपूर्वक तैयार किए गए प्रॉम्प्ट्स की आवश्यकता होती है। कच्चे मॉडल्स के साथ काम करना एक ऐसे प्रतिभाशाली मूर्ख से संवाद करने जैसा है जिसके पास विशाल ज्ञान तो है, लेकिन जब तक आप अपने निर्देशों में बेहद सटीक नहीं होते, तब तक उसे आप क्या पूछ रहे हैं, इसकी कोई समझ नहीं होती। वे अक्सर एक तोते की तरह लगते हैं, क्योंकि जब तक आप उन्हें कुछ समझदार बात कहलवाते हैं, तब तक वे ज्यादातर वही दोहराते हैं जो उन्होंने आपको कहते सुना है।

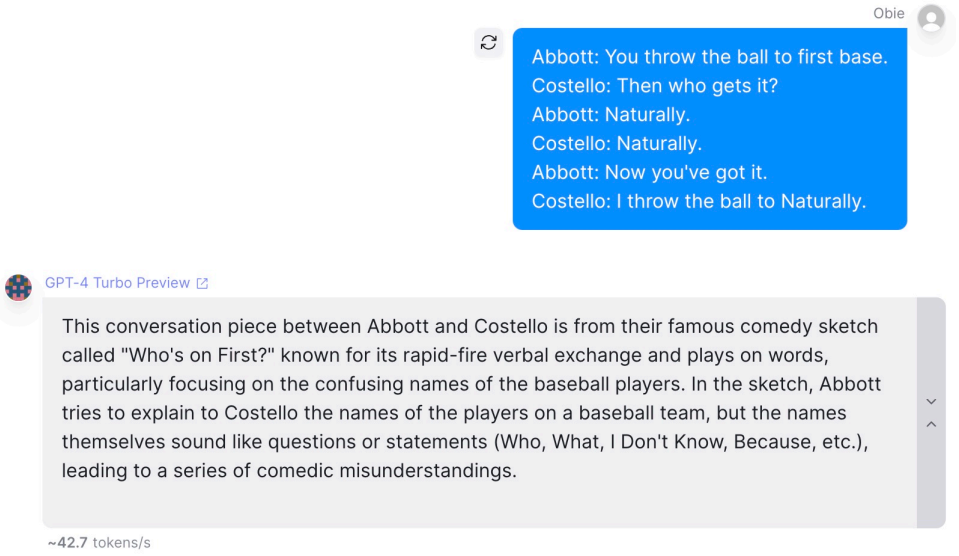
दूसरी ओर, निर्देश-समायोजित मॉडल विशेष रूप से निर्देशों को समझने और उनका पालन करने के लिए डिज़ाइन किए गए प्रशिक्षण से गुजरे होते हैं। GPT-4, Claude 3 और कई अन्य सबसे लोकप्रिय एलएलएम मॉडल सभी गहन रूप से निर्देश-समायोजित हैं। इस प्रशिक्षण में मॉडल को वांछित परिणामों के साथ निर्देशों के उदाहरण खिलाना शामिल है, जो प्रभावी रूप से मॉडल को विभिन्न प्रकार के आदेशों की व्याख्या और निष्पादन करना सिखाता है। परिणामस्वरूप, निर्देश मॉडल एक प्रॉम्प्ट के पीछे के इरादे को अधिक आसानी से समझ सकते हैं और ऐसी प्रतिक्रियाएं उत्पन्न कर सकते हैं जो उपयोगकर्ता की अपेक्षाओं के साथ निकटता से मेल खाती हैं। यह उन्हें अधिक उपयोगकर्ता-मैत्रीपूर्ण और काम करने में आसान बनाता है, विशेष रूप से उन लोगों के लिए जिनके पास व्यापक प्रॉम्प्ट इंजीनियरिंग में संलग्न होने का समय या विशेषज्ञता नहीं हो सकती है।

कच्चे मॉडल: अनफ़िल्टर्ड कैनवास

कच्चे मॉडल, जैसे Llama 2-70B या Yi-34B, मॉडल की क्षमताओं तक अधिक अनफ़िल्टर्ड पहुंच प्रदान करते हैं, जो आपको GPT-4 जैसे लोकप्रिय एलएलएम के साथ प्रयोग करने की स्थिति में अभ्यस्त हो सकते हैं। ये मॉडल विशिष्ट निर्देशों का पालन करने के लिए पूर्व-समायोजित नहीं होते हैं, जो आपको सावधानीपूर्वक प्रॉम्प्ट

इंजीनियरिंग के माध्यम से मॉडल के आउटपुट को सीधे नियंत्रित करने के लिए एक कोरा कैनवास प्रदान करते हैं। इस दृष्टिकोण के लिए इस बात की गहरी समझ की आवश्यकता होती है कि बिना स्पष्ट निर्देश दिए एआई को वांछित दिशा में मार्गदर्शित करने के लिए प्रॉम्प्ट्स को कैसे तैयार किया जाए। यह अंतर्निहित एआई की "कच्ची" परतों तक सीधी पहुंच रखने जैसा है, जहां कोई मध्यवर्ती परत मॉडल की प्रतिक्रियाओं की व्याख्या या मार्गदर्शन नहीं करती है (इसलिए इसका नाम)।

कच्चे मॉडल्स की चुनौती उनकी दोहराव वाले पैटर्न में पड़ने या यादृच्छिक आउटपुट उत्पन्न करने की प्रवृत्ति में निहित है। हालांकि, सावधानीपूर्वक प्रॉम्प्ट इंजीनियरिंग और पुनरावृत्ति दंड जैसे पैरामीटर्स के समायोजन के साथ, कच्चे मॉडल्स को अद्वितीय और रचनात्मक सामग्री उत्पन्न करने के लिए प्रेरित किया जा सकता है। यह प्रक्रिया समझौतों के बिना नहीं है; जहां कच्चे मॉडल नवाचार के लिए अनूठी लचीलापन प्रदान करते हैं, वहीं वे उच्च स्तर की विशेषज्ञता की मांग करते हैं।



आकृति 3. तुलना के उद्देश्यों के लिए, यहाँ वही अस्पष्ट प्रॉम्प्ट GPT-4 को दिया गया है

निर्देश-ट्यून किए गए मॉडल: निर्देशित अनुभव

निर्देश-ट्यून किए गए मॉडल विशिष्ट निर्देशों को समझने और उनका पालन करने के लिए डिज़ाइन किए गए हैं, जो उन्हें अधिक उपयोगकर्ता-मित्रवत और अनुप्रयोगों की व्यापक श्रृंखला के लिए सुलभ बनाते हैं। वे एक वार्तालाप की कार्यप्रणाली और यह समझते हैं कि उन्हें अपनी बारी के अंत में जनरेट करना बंद कर देना चाहिए। कई डेवलपर्स के लिए, विशेष रूप से सीधे अनुप्रयोगों पर काम करने वालों के लिए, निर्देश-ट्यून किए गए मॉडल एक सुविधाजनक और कुशल समाधान प्रदान करते हैं।

निर्देश-ट्यूनिंग की प्रक्रिया में मॉडल को मानव-निर्मित निर्देश प्रॉम्प्ट्स और प्रतिक्रियाओं के एक बड़े समूह पर प्रशिक्षित किया जाता है। एक उल्लेखनीय उदाहरण ओपन सोर्स databricks-dolly-15k dataset है, जिसमें Databricks कर्मचारियों द्वारा बनाए गए 15,000 से अधिक प्रॉम्प्ट/प्रतिक्रिया जोड़े शामिल हैं जिन्हें आप स्वयं देख सकते हैं। डेटासेट में आठ अलग-अलग निर्देश श्रेणियां शामिल हैं, जिनमें रचनात्मक लेखन, बंद

और खुले प्रश्न उत्तर, सारांशीकरण, सूचना निष्कर्षण, वर्गीकरण, और विचार-मंथन शामिल हैं।

डेटा निर्माण प्रक्रिया के दौरान, योगदानकर्ताओं को प्रत्येक श्रेणी के लिए प्रॉम्प्ट और प्रतिक्रियाएं बनाने के बारे में दिशानिर्देश दिए गए थे। उदाहरण के लिए, रचनात्मक लेखन कार्यों के लिए, उन्हें मॉडल के आउटपुट को निर्देशित करने के लिए विशिष्ट प्रतिबंध, निर्देश, या आवश्यकताएं प्रदान करने के लिए निर्देशित किया गया था। बंद प्रश्न उत्तर के लिए, उन्हें दिए गए विकिपीडिया अनुच्छेद के आधार पर तथ्यात्मक रूप से सही प्रतिक्रियाओं की आवश्यकता वाले प्रश्न लिखने के लिए कहा गया था।

परिणामी डेटासेट ChatGPT जैसी प्रणालियों की इंटरैक्टिव और निर्देश-पालन क्षमताओं को प्रदर्शित करने के लिए बड़े भाषा मॉडलों को फाइन-ट्यून करने के लिए एक मूल्यवान संसाधन के रूप में कार्य करता है। मानव-निर्मित निर्देशों और प्रतिक्रियाओं की विविध श्रृंखला पर प्रशिक्षण के माध्यम से, मॉडल विशिष्ट निर्देशों को समझने और उनका पालन करने के लिए सीखता है, जिससे यह विभिन्न प्रकार के कार्यों को संभालने में अधिक कुशल बन जाता है।

प्रत्यक्ष फाइन-ट्यूनिंग के अलावा, databricks-dolly-15k जैसे डेटासेट में निर्देश प्रॉम्प्ट्स का उपयोग कृत्रिम डेटा निर्माण के लिए भी किया जा सकता है। योगदानकर्ता-निर्मित प्रॉम्प्ट्स को एक बड़े खुले भाषा मॉडल को कुछ-शॉट उदाहरणों के रूप में प्रस्तुत करके, डेवलपर्स प्रत्येक श्रेणी में निर्देशों का एक बहुत बड़ा समूह उत्पन्न कर सकते हैं। सेल्फ-इंस्ट्रक्ट पेपर में वर्णित यह दृष्टिकोण अधिक मजबूत निर्देश-पालन मॉडल के निर्माण की अनुमति देता है।

इसके अलावा, इन डेटासेट में निर्देशों और प्रतिक्रियाओं को पुनर्कथन जैसी तकनीकों के माध्यम से बढ़ाया जा सकता है। प्रत्येक प्रॉम्प्ट या छोटी प्रतिक्रिया को पुनः कहकर और परिणामी पाठ को संबंधित ग्राउंड-ट्रुथ नमूने के साथ जोड़कर, डेवलपर्स एक प्रकार का नियमितीकरण पेश कर सकते हैं जो निर्देशों का पालन करने की मॉडल की क्षमता को बढ़ाता है।

निर्देश-ट्यून्ड मॉडल्स द्वारा प्रदान की गई उपयोग में आसानी कुछ लचीलेपन की कीमत पर आती है। ये मॉडल्स अक्सर काफी सेंसर किए जाते हैं, जिसका मतलब है कि वे कुछ विशिष्ट कार्यों के लिए आवश्यक रचनात्मक स्वतंत्रता का स्तर हमेशा प्रदान नहीं

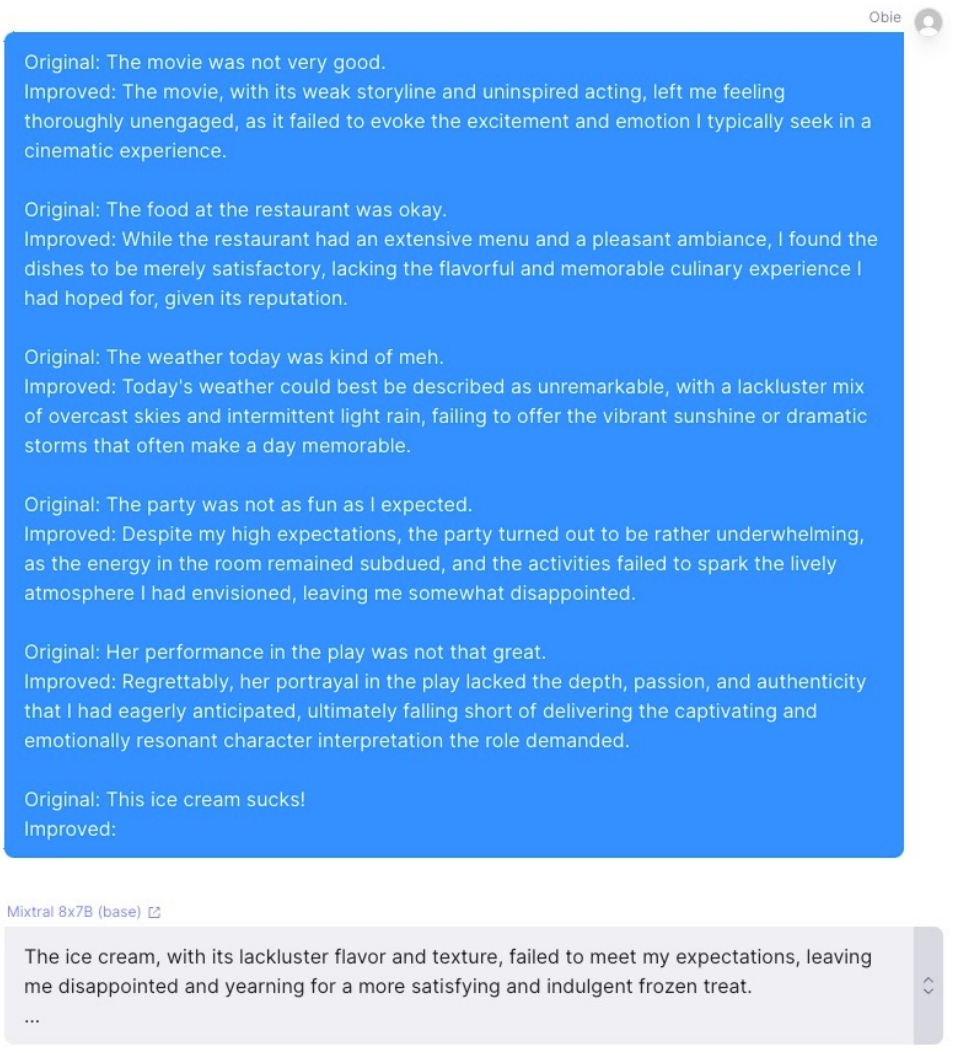
कर सकते। उनके आउटपुट उनके फाइन-ट्यूनिंग डेटा में निहित पूर्वाग्रहों और सीमाओं से काफी प्रभावित होते हैं।

इन सीमाओं के बावजूद, निर्देश-ट्यून्ड मॉडल्स अपनी उपयोगकर्ता-मैत्रीपूर्ण प्रकृति और न्यूनतम प्रॉम्प्ट इंजीनियरिंग के साथ विभिन्न कार्यों को संभालने की क्षमता के कारण तेजी से लोकप्रिय हो गए हैं। जैसे-जैसे अधिक उच्च-गुणवत्ता वाले निर्देश डेटासेट उपलब्ध होते जाएंगे, हम इन मॉडल्स के प्रदर्शन और बहुमुखी प्रतिभा में और सुधार देखने की उम्मीद कर सकते हैं।

अपनी परियोजना के लिए सही प्रकार का मॉडल चुनना

बेस (रॉ) और निर्देश-ट्यून्ड मॉडल्स के बीच का निर्णय अंततः आपकी परियोजना की विशिष्ट आवश्यकताओं पर निर्भर करता है। उन कार्यों के लिए जो रचनात्मकता और मौलिकता का उच्च स्तर मांगते हैं, बेस मॉडल्स नवाचार के लिए एक शक्तिशाली उपकरण प्रदान करते हैं। ये मॉडल्स डेवलपर्स को एलएलएम की पूरी क्षमता का पता लगाने की अनुमति देते हैं, एआई-संचालित अनुप्रयोगों के माध्यम से जो हासिल किया जा सकता है उसकी सीमाओं को आगे बढ़ाते हैं, लेकिन इनके लिए एक अधिक हाथों-से काम करने वाला दृष्टिकोण और प्रयोग करने की इच्छा की आवश्यकता होती है। तापमान और अन्य सेटिंग्स का बेस मॉडल्स में उनके निर्देश काउंटरपार्ट्स की तुलना में बहुत अधिक प्रभाव पड़ता है।

आप अपने प्रॉम्प्ट में जो कुछ भी शामिल करते हैं, बेस मॉडल्स उसे दोहराने की कोशिश करेंगे। तो उदाहरण के लिए यदि आपका प्रॉम्प्ट एक चैट ट्रांसक्रिप्ट है, तो रॉ मॉडल चैट को जारी रखने की कोशिश करेगा। मैक्स टोकन्स की सीमा के आधार पर, यह न केवल चैट में अगला संदेश जनरेट करेगा, बल्कि यह खुद से एक पूरी बातचीत कर सकता है!



आकृति 4. मिक्सट्रल 8x7B (बेस) फ्यू-शॉट कम्पलीशन के साथ वाक्य पुनर्लेखन का उदाहरण

Reddit उपयोगकर्ता phree_radical द्वारा ऊपर दिए गए वाक्य पुनर्लेखन के उदाहरण को तैयार करते समय, मैं केवल पैरामीटर सेटिंग्स के साथ बहुत प्रयोग करने के बाद ही उपयोगी परिणाम प्राप्त कर पाया, और अंततः इन सेटिंग्स पर पहुंचा: तापमान 0.08, टॉप पी: 0.2, टॉप के: 1, और पुनरावृत्ति पेनल्टी: 1.26।

प्रोडक्शन में बेस मॉडल के साथ इस दृष्टिकोण का उपयोग करना max_tokens पैरामीटर के शक्तिशाली प्रभाव के कारण जटिल होगा। इसे बहुत छोटा सेट करें तो आउटपुट काट दिया जाता है। इसे मॉडल की वांछित आउटपुट की आवश्यकता से अधिक लंबा सेट करें, और यह अतिरिक्त उदाहरणों का भ्रामक उत्पादन करना जारी रखेगा।

निचोड़ यह है कि जब तक आपको पूर्ण नियंत्रण और सेंसरशिप की कमी की वास्तव में आवश्यकता न हो, निर्देश-ट्यून्ड मॉडल आपकी विकास प्रक्रिया को काफी सरल बना सकते हैं। इस बात को स्पष्ट करने के लिए, यहाँ Mixtral 8x7B का उसी प्रॉम्प्ट के लिए जवाब है, लेकिन इस बार उसके निर्देश-ट्यून्ड संस्करण में:

> मुझे यह सूचित करते हुए खेद है कि आइसक्रीम मेरी अपेक्षाओं पर खरी नहीं उतरती, क्योंकि इसमें वह समृद्ध, क्रीमी बनावट और सुखद स्वाद नहीं है जो मैं आमतौर पर उच्च-गुणवत्ता वाले डेजर्ट से जोड़ता हूं। मैं एक अधिक संतोषजनक और आनंददायक अनुभव की आशा कर रहा था।

विशेष रूप से, मैं max tokens सेटिंग को 500 पर छोड़ पाया, और मॉडल ने बिना किसी अतिरिक्त उदाहरणों के भ्रामक उत्पादन के वांछित आउटपुट के अंत में विश्वसनीय रूप से रुक गया।

प्रॉम्प्ट इंजीनियरिंग

जैसे-जैसे आप अपनी परियोजनाओं में AI का उपयोग करना शुरू करते हैं, आप जल्द ही पाएंगे कि आपको जिन सबसे महत्वपूर्ण कौशलों पर महारत हासिल करने की जरूरत है, उनमें से एक है प्रॉम्प्ट इंजीनियरिंग की कला। लेकिन प्रॉम्प्ट इंजीनियरिंग वास्तव में है क्या, और यह इतनी महत्वपूर्ण क्यों है?

मूल रूप से, प्रॉम्प्ट इंजीनियरिंग भाषा मॉडल को प्रदान किए जाने वाले इनपुट प्रॉम्प्ट्स को डिज़ाइन और तैयार करने की प्रक्रिया है जो उसके आउटपुट को मार्गदर्शित करती है। यह AI के साथ प्रभावी ढंग से संवाद करने के तरीके को समझने के बारे में है,

जिसमें निर्देशों, उदाहरणों और संदर्भ का संयोजन उपयोग करके मॉडल को वांछित प्रतिक्रिया उत्पन्न करने की दिशा में निर्देशित किया जाता है।

इसे एक अत्यंत बुद्धिमान लेकिन कुछ हद तक शब्दशः सोचने वाले मित्र के साथ बातचीत करने जैसा समझें। बातचीत से अधिकतम लाभ प्राप्त करने के लिए, आपको स्पष्ट, विशिष्ट होने की आवश्यकता है और पर्याप्त संदर्भ प्रदान करना होगा ताकि यह सुनिश्चित हो सके कि आपका मित्र ठीक से समझ रहा है कि आप क्या मांग रहे हैं। यहीं प्रॉम्प्ट इंजीनियरिंग काम आती है, और भले ही यह शुरुआत में आसान लगे, मुझ पर विश्वास करें कि इस पर महारत हासिल करने में काफी अभ्यास की आवश्यकता होती है।

प्रभावी प्रॉम्प्ट्स के निर्माण खंड

प्रभावी प्रॉम्प्ट्स की इंजीनियरिंग शुरू करने के लिए, पहले आपको एक अच्छी तरह से तैयार किए गए इनपुट के प्रमुख घटकों को समझना होगा। यहाँ कुछ आवश्यक निर्माण खंड दिए गए हैं:

1. **निर्देश**: स्पष्ट और संक्षिप्त निर्देश जो मॉडल को बताते हैं कि आप क्या चाहते हैं। यह कुछ भी हो सकता है, जैसे "निम्नलिखित लेख का सारांश करें" से लेकर "सूर्यास्त के बारे में एक कविता बनाएं" या "इस प्रोजेक्ट परिवर्तन अनुरोध को JSON ऑब्जेक्ट में बदलें"।
2. **संदर्भ**: प्रासंगिक जानकारी जो मॉडल को कार्य की पृष्ठभूमि और दायरे को समझने में मदद करती है। इसमें लक्षित दर्शकों के बारे में विवरण, वांछित स्वर और शैली, या आउटपुट के लिए कोई विशिष्ट बाधाएं या आवश्यकताएं शामिल हो सकती हैं, जैसे एक JSON स्कीमा जिसका पालन करना है।
3. **उदाहरण**: ठोस उदाहरण जो दर्शाते हैं कि आप किस प्रकार का आउटपुट चाहते हैं। कुछ अच्छी तरह से चुने गए उदाहरण प्रदान करके, आप मॉडल को वांछित प्रतिक्रिया के पैटर्न और विशेषताओं को सीखने में मदद कर सकते हैं।
4. **इनपुट स्वरूपण**: लाइन ब्रेक और मार्कडाउन स्वरूपण हमारे प्रॉम्प्ट को संरचना प्रदान करते हैं। प्रॉम्प्ट को पैराग्राफ में विभाजित करने से हम संबंधित निर्देशों

को समूहबद्ध कर सकते हैं, ताकि मनुष्यों और AI दोनों के लिए इसे समझना आसान हो। बुलेट और क्रमांकित सूचियां हमें आइटम की सूची और क्रम को परिभाषित करने की अनुमति देती हैं। बोल्ड और इटैलिक्स मार्कर हमें जोर देने की अनुमति देते हैं।

5. **आउटपुट स्वरूपण**: आउटपुट को कैसे संरचित और स्वरूपित किया जाना चाहिए, इस बारे में विशिष्ट निर्देश। इनमें वांछित लंबाई, शीर्षक या बुलेट पॉइंट्स का उपयोग, मार्कडाउन स्वरूपण, या कोई अन्य विशिष्ट आउटपुट टेम्पलेट या परंपराएं शामिल हो सकती हैं जिनका पालन किया जाना चाहिए।

इन निर्माण खंडों को विभिन्न तरीकों से जोड़कर, आप ऐसे प्रॉम्प्ट बना सकते हैं जो आपकी विशिष्ट आवश्यकताओं के अनुरूप हों और मॉडल को उच्च-गुणवत्ता वाली, प्रासंगिक प्रतिक्रियाएं उत्पन्न करने की दिशा में मार्गदर्शन करें।

प्रॉम्प्ट डिज़ाइन की कला और विज्ञान

प्रभावी प्रॉम्प्ट्स की रचना कला और विज्ञान दोनों है। (इसीलिए हम इसे शिल्प कहते हैं।) इसके लिए भाषा मॉडल की क्षमताओं और सीमाओं की गहरी समझ के साथ-साथ वांछित व्यवहार को प्राप्त करने के लिए प्रॉम्प्ट्स को डिज़ाइन करने का एक रचनात्मक दृष्टिकोण आवश्यक है। इसमें शामिल रचनात्मकता ही है जो इसे मेरे लिए कम से कम इतना मजेदार बनाती है। यह बहुत निराशाजनक भी हो सकता है, खासकर जब आप निर्धारित व्यवहार की तलाश कर रहे हों।

प्रॉम्प्ट इंजीनियरिंग का एक प्रमुख पहलू विशिष्टता और लचीलेपन के बीच संतुलन को समझना है। एक तरफ, आप मॉडल को सही दिशा में निर्देशित करने के लिए पर्याप्त मार्गदर्शन प्रदान करना चाहते हैं। दूसरी तरफ, आप इतने निर्देशात्मक नहीं होना चाहते कि आप मॉडल की सीमांत मामलों से निपटने के लिए अपनी रचनात्मकता और लचीलेपन का उपयोग करने की क्षमता को सीमित कर दें।

एक और महत्वपूर्ण विचार उदाहरणों का उपयोग है। सही चुने गए उदाहरण मॉडल को आपके वांछित आउटपुट को समझने में अत्यंत सहायक हो सकते हैं। हालांकि, यह महत्वपूर्ण है कि उदाहरणों का विवेकपूर्ण उपयोग किया जाए और यह सुनिश्चित किया

जाए कि वे वांछित प्रतिक्रिया का प्रतिनिधित्व करते हैं। एक खराब उदाहरण सर्वश्रेष्ठ स्थिति में टोकन की बर्बादी है, और सबसे खराब स्थिति में वांछित आउटपुट के लिए विनाशकारी है।

प्रॉम्प्ट इंजीनियरिंग तकनीकें और सर्वोत्तम प्रथाएं

जैसे-जैसे आप प्रॉम्प्ट इंजीनियरिंग की दुनिया में गहराई से उतरते हैं, आपको कई तकनीकें और सर्वोत्तम प्रथाएं मिलेंगी जो आपको अधिक प्रभावी प्रॉम्प्ट बनाने में मदद कर सकती हैं। यहाँ कुछ प्रमुख क्षेत्र हैं जिनका अन्वेषण किया जा सकता है:

1. **जीरो-शॉट बनाम फ्यू-शॉट लर्निंग**: जीरो-शॉट लर्निंग (कोई उदाहरण नहीं देना) बनाम वन-शॉट या फ्यू-शॉट लर्निंग (कुछ उदाहरण देना) का उपयोग कब करना है, यह समझने से आप अधिक कुशल और प्रभावी प्रॉम्प्ट बना सकते हैं।
2. **पुनरावर्ती परिष्करण**: मॉडल के आउटपुट के आधार पर प्रॉम्प्ट को क्रमिक रूप से परिष्कृत करने की प्रक्रिया आपको सर्वोत्तम प्रॉम्प्ट डिजाइन तक पहुंचने में मदद कर सकती है। फीडबैक लूप एक शक्तिशाली दृष्टिकोण है जो उत्पन्न सामग्री की गुणवत्ता और प्रासंगिकता को क्रमिक रूप से सुधारने के लिए भाषा मॉडल के स्वयं के आउटपुट का लाभ उठाता है।
3. **प्रॉम्प्ट चेनिंग**: जटिल कार्यों को छोटे, अधिक प्रबंधनीय चरणों में विभाजित करने में कई प्रॉम्प्ट को श्रृंखला में जोड़ना मदद कर सकता है। प्रॉम्प्ट चेनिंग में एक जटिल कार्य या वार्तालाप को छोटे, परस्पर जुड़े प्रॉम्प्ट की श्रृंखला में विभाजित करना शामिल है। प्रॉम्प्ट को श्रृंखलाबद्ध करके, आप AI को एक बहु-चरणीय प्रक्रिया के माध्यम से मार्गदर्शित कर सकते हैं, जिससे संपूर्ण बातचीत में संदर्भ और सुसंगतता बनी रहती है।
4. **प्रॉम्प्ट ट्यूनिंग**: विशिष्ट डोमेन या कार्यों के लिए प्रॉम्प्ट को कस्टम टेलरिंग करना आपको अधिक विशेषज्ञ और प्रभावी प्रॉम्प्ट बनाने में मदद कर सकता है। प्रॉम्प्ट टेम्पलेट आपको लचीली, पुन: प्रयोग योग्य, और रखरखाव योग्य प्रॉम्प्ट संरचनाएं बनाने में मदद करता है जो दिए गए कार्य के लिए आसानी से अनुकूल हैं।

जीरो-शॉट, वन-शॉट, या फ्यू-शॉट लर्निंग का उपयोग कब करना है, यह जानना प्रॉम्प्ट इंजीनियरिंग में महारत हासिल करने का एक विशेष महत्वपूर्ण हिस्सा है। प्रत्येक दृष्टिकोण की अपनी शक्तियां और कमजोरियां हैं, और प्रत्येक का उपयोग कब करना है, यह समझने से आप अधिक प्रभावी और कुशल प्रॉम्प्ट बना सकते हैं।

जीरो-शॉट लर्निंग: जब कोई उदाहरण आवश्यक नहीं होते

ज़ीरो-शॉट लर्निंग का तात्पर्य भाषा मॉडल की बिना किसी उदाहरण या स्पष्ट प्रशिक्षण के कार्य करने की क्षमता से है। दूसरे शब्दों में, आप मॉडल को एक प्रॉम्प्ट प्रदान करते हैं जो कार्य का वर्णन करता है, और मॉडल केवल अपने पूर्व-मौजूद ज्ञान और भाषा की समझ के आधार पर प्रतिक्रिया उत्पन्न करता है।

ज़ीरो-शॉट लर्निंग विशेष रूप से उपयोगी होती है जब:

1. कार्य अपेक्षाकृत सरल और सीधा है, और मॉडल ने अपने पूर्व-प्रशिक्षण के दौरान समान कार्यों का सामना किया होगा।
2. आप मॉडल की अंतर्निहित क्षमताओं का परीक्षण करना चाहते हैं और देखना चाहते हैं कि वह बिना किसी अतिरिक्त मार्गदर्शन के नए कार्य पर कैसी प्रतिक्रिया देता है।
3. आप एक बड़े और विविध भाषा मॉडल के साथ काम कर रहे हैं जिसे विभिन्न कार्यों और क्षेत्रों में प्रशिक्षित किया गया है।

हालांकि, ज़ीरो-शॉट लर्निंग अप्रत्याशित भी हो सकती है और हमेशा वांछित परिणाम नहीं दे सकती। मॉडल की प्रतिक्रिया उसके पूर्व-प्रशिक्षण डेटा में पूर्वाग्रहों या असंगतियों से प्रभावित हो सकती है, और यह अधिक जटिल या सूक्ष्म कार्यों में संघर्ष कर सकता है।

मैंने ऐसे ज़ीरो-शॉट प्रॉम्प्ट्स देखे हैं जो मेरे 80% टेस्ट केस में ठीक से काम करते हैं और बाकी 20% के लिए बेहद गलत या अबूझ परिणाम देते हैं। एक व्यापक परीक्षण व्यवस्था लागू करना बहुत महत्वपूर्ण है, खासकर यदि आप बहुत अधिक ज़ीरो-शॉट प्रॉम्प्टिंग पर निर्भर हैं।

वन-शॉट लर्निंग: जब एक उदाहरण बदल सकता है सब कुछ

वन-शॉट लर्निंग में कार्य विवरण के साथ वांछित आउटपुट का एक उदाहरण मॉडल को प्रदान करना शामिल है। यह उदाहरण एक टेम्पलेट या पैटर्न के रूप में कार्य करता है जिसका उपयोग मॉडल अपनी प्रतिक्रिया उत्पन्न करने के लिए कर सकता है।

वन-शॉट लर्निंग प्रभावी हो सकती है जब:

1. कार्य अपेक्षाकृत नया या विशिष्ट है, और मॉडल ने अपने पूर्व-प्रशिक्षण के दौरान कई समान उदाहरणों का सामना नहीं किया होगा।
2. आप वांछित आउटपुट प्रारूप या शैली का स्पष्ट और संक्षिप्त प्रदर्शन प्रदान करना चाहते हैं।
3. कार्य के लिए एक विशिष्ट संरचना या परंपरा की आवश्यकता होती है जो केवल कार्य विवरण से स्पष्ट नहीं हो सकती।

जो विवरण आपको स्पष्ट लगते हैं, वे जरूरी नहीं कि एआई के लिए भी स्पष्ट हों। वन-शॉट उदाहरण चीजों को स्पष्ट करने में मदद कर सकते हैं।

वन-शॉट लर्निंग मॉडल को अपेक्षाओं को अधिक स्पष्ट रूप से समझने और प्रदान किए गए उदाहरण के अनुरूप प्रतिक्रिया उत्पन्न करने में मदद कर सकती है। हालांकि, उदाहरण को सावधानीपूर्वक चुनना और यह सुनिश्चित करना महत्वपूर्ण है कि यह वांछित आउटपुट का प्रतिनिधित्व करता है। उदाहरण चुनते समय, खुद से सीमांत मामलों और प्रॉम्प्ट द्वारा संभाले जाने वाले इनपुट की श्रेणी के बारे में पूछें।

आकृति 5. JSON का एक बार में एक उदाहरण

```
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"
}
```

फ्यू-शॉट लर्निंग: जब कई उदाहरण प्रदर्शन में सुधार कर सकते हैं

फ्यू-शॉट लर्निंग में मॉडल को कार्य विवरण के साथ कुछ उदाहरण (आमतौर पर 2 से 10 के बीच) प्रदान किए जाते हैं। ये उदाहरण मॉडल को अधिक संदर्भ और विविधता प्रदान करने में मदद करते हैं, जिससे यह अधिक विविध और सटीक प्रतिक्रियाएं उत्पन्न कर सकता है।

फ्यू-शॉट लर्निंग विशेष रूप से उपयोगी होती है जब:

1. कार्य जटिल या सूक्ष्म हो, और एक उदाहरण सभी प्रासंगिक पहलुओं को समझने के लिए पर्याप्त न हो।
2. आप मॉडल को विभिन्न प्रकार के उदाहरण प्रदान करना चाहते हैं जो विभिन्न विविधताओं या सीमांत मामलों को प्रदर्शित करते हों।
3. कार्य के लिए मॉडल को किसी विशिष्ट डोमेन या शैली के अनुरूप प्रतिक्रियाएं उत्पन्न करने की आवश्यकता हो।

कई उदाहरण प्रदान करके, आप मॉडल को कार्य की अधिक मजबूत समझ विकसित करने और अधिक सुसंगत और विश्वसनीय प्रतिक्रियाएं उत्पन्न करने में मदद कर सकते हैं।

उदाहरण: प्रॉम्प्ट्स आपकी कल्पना से कहीं अधिक जटिल हो सकते हैं

आज के एलएलएम आपकी कल्पना से कहीं अधिक शक्तिशाली और तर्क करने में सक्षम हैं। इसलिए प्रॉम्प्ट्स को केवल इनपुट और आउटपुट जोड़ों के विवरण तक सीमित न करें। आप लंबे और जटिल निर्देशों के साथ प्रयोग कर सकते हैं, जिस तरह से आप एक इंसान के साथ बातचीत करेंगे।

उदाहरण के लिए, यह वह प्रॉम्प्ट है जो मैंने Olympia में उपयोग किया था जब मैं Google सेवाओं के साथ हमारे एकीकरण का प्रोटोटाइप बना रहा था, जो कि कुल मिलाकर शायद दुनिया के सबसे बड़े एपीआई में से एक है। मेरे पहले के प्रयोगों ने साबित किया कि GPT-4 को Google एपीआई की अच्छी जानकारी है, और मेरे पास एक-एक करके प्रत्येक फ़ंक्शन को लागू करने के लिए एक सूक्ष्म मैपिंग लेयर लिखने का समय या प्रेरणा नहीं थी। क्या होगा अगर मैं एआई को सीधे सभी Google एपीआई तक पहुंच दे दूं?

मैंने अपना प्रॉम्प्ट एआई को यह बताकर शुरू किया कि उसे एचटीटीपी के माध्यम से Google एपीआई एंडपॉइंट्स तक सीधी पहुंच है, और उसकी भूमिका उपयोगकर्ता की ओर से Google ऐप्स और सेवाओं का उपयोग करना है। फिर मैंने दिशानिर्देश दिए, fields पैरामीटर से संबंधित नियम, क्योंकि लगता था कि उसे उसी के साथ सबसे अधिक परेशानी थी, और कुछ एपीआई-विशिष्ट संकेत (फ्यू-शॉट प्रॉम्प्टिंग, कार्यरत)।

यहाँ पूरा प्रॉम्प्ट है, जो एआई को बताता है कि प्रदान किए गए invoke_google_api फ़ंक्शन का उपयोग कैसे करना है।

```
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
```

आप सोच रहे होंगे कि क्या यह प्रॉम्प्ट काम करता है। सरल उत्तर है, हाँ। एआई को पहली बार में एपीआई को पूर्ण रूप से कॉल करना नहीं आता था। हालांकि, अगर इसने कोई गलती की, तो मैं बस परिणामी त्रुटि संदेशों को कॉल के परिणाम के रूप में वापस फ़ीड कर देता था। अपनी त्रुटि की जानकारी मिलने पर, एआई अपनी गलती के बारे में सोच सकता था और फिर से प्रयास कर सकता था। अधिकतर मामलों में, यह कुछ ही प्रयासों में सही हो जाता था।

ध्यान रहे, इस प्रॉम्प्ट का उपयोग करते समय Google एपीआई द्वारा लौटाई गई बड़ी JSON संरचनाएं बेहद अक्षम हैं, इसलिए मैं प्रोडक्शन में इस दृष्टिकोण का उपयोग करने की सलाह नहीं दे रहा हूं। हालांकि, मुझे लगता है कि यह तथ्य कि यह दृष्टिकोण बिल्कुल काम करता है, यह प्रॉम्प्ट इंजीनियरिंग की शक्ति का प्रमाण है।

प्रयोग और पुनरावृत्ति

अंततः, आप अपने प्रॉम्प्ट को कैसे इंजीनियर करते हैं यह विशिष्ट कार्य, वांछित आउटपुट की जटिलता, और आपके द्वारा उपयोग किए जा रहे भाषा मॉडल की क्षमताओं पर निर्भर करता है।

एक प्रॉम्प्ट इंजीनियर के रूप में, विभिन्न दृष्टिकोणों के साथ प्रयोग करना और परिणामों के आधार पर पुनरावृत्ति करना महत्वपूर्ण है। ज़ीरो-शॉट लर्निंग से शुरू करें और देखें कि मॉडल कैसा प्रदर्शन करता है। यदि आउटपुट असंगत या असंतोषजनक है, तो एक या अधिक उदाहरण प्रदान करने का प्रयास करें और देखें कि क्या प्रदर्शन में सुधार होता है।

ध्यान रखें कि प्रत्येक दृष्टिकोण के भीतर भी, विविधता और अनुकूलन के लिए जगह है। आप विभिन्न उदाहरणों के साथ प्रयोग कर सकते हैं, कार्य विवरण की शब्दावली को समायोजित कर सकते हैं, या मॉडल की प्रतिक्रिया को मार्गदर्शित करने में मदद करने के लिए अतिरिक्त संदर्भ प्रदान कर सकते हैं।

समय के साथ, आप एक अंतर्ज्ञान विकसित करेंगे कि किसी दिए गए कार्य के लिए कौन सा दृष्टिकोण सबसे अच्छा काम करेगा, और आप ऐसे प्रॉम्प्ट तैयार करने में सक्षम होंगे जो अधिक प्रभावी और कुशल हों। मुख्य बात है कि प्रॉम्प्ट इंजीनियरिंग के प्रति अपने दृष्टिकोण में जिज्ञासु, प्रयोगात्मक और पुनरावर्ती बने रहें।

इस पुस्तक में, हम इन तकनीकों में और गहराई से जाएंगे और देखेंगे कि वास्तविक दुनिया के परिदृश्यों में इनका उपयोग कैसे किया जा सकता है। प्रॉम्प्ट इंजीनियरिंग की कला और विज्ञान में महारत हासिल करके, आप एआई-संचालित एप्लिकेशन विकास की पूरी क्षमता को उजागर करने के लिए अच्छी तरह से सुसज्जित होंगे।

अस्पष्टता की कला

जब बड़े भाषा मॉडल (एलएलएम) के लिए प्रभावी प्रॉम्प्ट तैयार करने की बात आती है, तो एक सामान्य धारणा यह है कि अधिक विशिष्टता और विस्तृत निर्देश बेहतर परिणाम देते हैं। हालांकि, व्यावहारिक अनुभव ने दिखाया है कि यह हमेशा सच नहीं

होता। वास्तव में, अपने प्रॉम्प्ट्स में जानबूझकर अस्पष्ट होने से अक्सर बेहतर परिणाम मिल सकते हैं, जो एलएलएम की सामान्यीकरण करने और निष्कर्ष निकालने की उल्लेखनीय क्षमता का लाभ उठाते हैं।

Ken, एक स्टार्टअप संस्थापक जिन्होंने 500 मिलियन से अधिक GPT टोकन्स को प्रोसेस किया है, ने अपने अनुभव से महत्वपूर्ण जानकारियां साझा कीं। उन्होंने जो प्रमुख सबक सीखा, वह यह था कि प्रॉम्प्ट्स के मामले में "कम ज्यादा है"। सटीक सूचियों या अत्यधिक विस्तृत निर्देशों के बजाय, Ken ने पाया कि LLM को अपने आधारभूत ज्ञान पर निर्भर रहने देने से अक्सर बेहतर परिणाम मिलते हैं।

यह एहसास पारंपरिक कोडिंग की सोच को उलट देता है, जहां हर चीज को बारीकी से विस्तार से बताना जरूरी होता है। LLM के साथ, यह समझना महत्वपूर्ण है कि उनके पास विशाल ज्ञान है और वे बुद्धिमान कनेक्शन और निष्कर्ष निकाल सकते हैं। अपने प्रॉम्प्ट्स में अधिक अस्पष्ट रहकर, आप LLM को अपनी समझ का उपयोग करने और ऐसे समाधान निकालने की स्वतंत्रता देते हैं जो आपने स्पष्ट रूप से निर्दिष्ट नहीं किए होंगे।

उदाहरण के लिए, जब Ken की टीम 50 अमेरिकी राज्यों या संघीय सरकार से संबंधित टेक्स्ट को वर्गीकृत करने के लिए एक पाइपलाइन पर काम कर रही थी, तो उनका प्रारंभिक दृष्टिकोण राज्यों और उनके संबंधित आईडी की पूर्ण विस्तृत सूची को JSON-फॉर्मेटेड ऐरे के रूप में प्रदान करने का था।

```
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} ... ]
```

यह दृष्टिकोण इतना असफल रहा कि उन्हें इसे सुधारने के लिए प्रॉम्प्ट को और गहराई से समझना पड़ा। ऐसा करते हुए उन्होंने देखा कि भले ही LLM अक्सर आईडी गलत प्राप्त करता था, लेकिन यह लगातार सही राज्य का पूरा नाम name फ़ील्ड में वापस कर रहा था, भले ही उन्होंने इसके लिए स्पष्ट रूप से नहीं पूछा था।

स्थानीय आईडी को हटाकर और प्रॉम्प्ट को सरल बनाकर जैसे, "तुम स्पष्ट रूप से 50 राज्यों को जानते हो, GPT, तो बस मुझे उस राज्य का पूरा नाम बताओ जिससे

यह संबंधित है, या फेडरल यदि यह अमेरिकी सरकार से संबंधित है," उन्होंने बेहतर परिणाम प्राप्त किए। यह अनुभव LLM की सामान्यीकरण क्षमताओं का लाभ उठाने और इसे अपने मौजूदा ज्ञान के आधार पर निष्कर्ष निकालने की अनुमति देने की शक्ति को उजागर करता है।

> इस विशेष वर्गीकरण दृष्टिकोण के लिए Ken का औचित्य, एक अधिक पारंपरिक प्रोग्रामिंग तकनीक के विपरीत, उन लोगों की मानसिकता को प्रकट करता है जिन्होंने LLM तकनीक की क्षमता को स्वीकार किया है: "यह कोई कठिन काम नहीं है – शायद हम स्ट्रिंग/रेजेक्स का उपयोग कर सकते थे, लेकिन इतने विचित्र कॉर्नर केस हैं कि इसमें ज्यादा समय लगता।"

अधिक अस्पष्ट प्रॉम्प्ट्स दिए जाने पर LLM की गुणवत्ता और सामान्यीकरण में सुधार करने की क्षमता उच्च-क्रम की सोच और प्रतिनिधिमंडल की एक उल्लेखनीय विशेषता है। यह दर्शाता है कि LLM अस्पष्टता को संभाल सकते हैं और दिए गए संदर्भ के आधार पर बुद्धिमान निर्णय ले सकते हैं।

हालांकि, यह ध्यान रखना महत्वपूर्ण है कि अस्पष्ट होने का मतलब अस्पष्ट या संदिग्ध होना नहीं है। मुख्य बात यह है कि LLM को पर्याप्त संदर्भ और मार्गदर्शन प्रदान किया जाए जो इसे सही दिशा में निर्देशित करे, साथ ही इसे अपने ज्ञान और सामान्यीकरण क्षमताओं का उपयोग करने की लचीलापन दे।

इसलिए, प्रॉम्प्ट्स को डिज़ाइन करते समय, निम्नलिखित "कम ज्यादा है" युक्तियों पर विचार करें:

1. प्रक्रिया का हर विवरण बताने के बजाय वांछित परिणाम पर ध्यान केंद्रित करें।
2. प्रासंगिक संदर्भ और सीमाएं प्रदान करें, लेकिन अति-विशिष्टता से बचें।
3. सामान्य अवधारणाओं या संस्थाओं का उल्लेख करके मौजूदा ज्ञान का लाभ उठाएं।
4. दिए गए संदर्भ के आधार पर निष्कर्षों और कनेक्शन के लिए जगह छोड़ें।

5. LLM की प्रतिक्रियाओं के आधार पर अपने प्रॉम्प्ट्स को दोहराएं और परिष्कृत करें, विशिष्टता और अस्पष्टता के बीच सही संतुलन खोजें।

प्रॉम्प्ट इंजीनियरिंग में अस्पष्टता की कला को अपनाकर, आप LLMs की पूरी क्षमता को अनलॉक कर सकते हैं और बेहतर परिणाम प्राप्त कर सकते हैं। LLM की सामान्यीकरण और बुद्धिमान निर्णय लेने की क्षमता पर भरोसा करें, और आप प्राप्त होने वाले आउटपुट की गुणवत्ता और रचनात्मकता से आश्चर्यचकित हो सकते हैं। ध्यान दें कि विभिन्न मॉडल आपके प्रॉम्प्ट्स में विभिन्न स्तर की विशिष्टता पर कैसे प्रतिक्रिया करते हैं और तदनुसार समायोजित करें। अभ्यास और अनुभव के साथ, आप यह समझने में माहिर हो जाएंगे कि कब अधिक अस्पष्ट होना है और कब अतिरिक्त मार्गदर्शन प्रदान करना है, जो आपको अपने एप्लिकेशन में LLMs की शक्ति का प्रभावी ढंग से उपयोग करने में सक्षम बनाएगा।

प्रॉम्प्ट इंजीनियरिंग में मानवीकरण का प्रभुत्व क्यों है

मानवीकरण, यानी गैर-मानवीय वस्तुओं को मानवीय विशेषताएं प्रदान करना, लार्ज लैंग्वेज मॉडल्स के लिए प्रॉम्प्ट इंजीनियरिंग में एक सोच-समझकर अपनाया गया प्रमुख दृष्टिकोण है। यह एक ऐसा डिज़ाइन विकल्प है जो शक्तिशाली AI सिस्टम के साथ बातचीत को अधिक सहज और व्यापक उपयोगकर्ताओं (हम एप्लिकेशन डेवलपर्स सहित) के लिए सुलभ बनाता है।

LLMs का मानवीकरण एक ऐसा ढांचा प्रदान करता है जो उन लोगों के लिए तुरंत समझने योग्य है जो सिस्टम की अंतर्निहित तकनीकी जटिलताओं से पूरी तरह अनजान हैं। जैसा कि आप अनुभव करेंगे अगर आप किसी नॉन-इंस्ट्रक्ट-ट्यून्ड मॉडल का उपयोग कुछ उपयोगी करने के लिए करने की कोशिश करते हैं, एक ऐसी फ्रेमिंग बनाना जिसमें अपेक्षित परिणाम मूल्यवान हो, एक चुनौतीपूर्ण कार्य है। इसके लिए सिस्टम की आंतरिक कार्यप्रणाली की गहरी समझ की आवश्यकता होती है, जो केवल कुछ विशेषज्ञों के पास होती है।

लैंग्वेज मॉडल के साथ बातचीत को दो लोगों के बीच संवाद के रूप में देखकर, हम अपनी जरूरतों और अपेक्षाओं को व्यक्त करने के लिए मानव संचार की अपनी सहज

समझ का उपयोग कर सकते हैं। जैसे शुरुआती Macintosh UI डिज़ाइन में जटिलता की तुलना में तत्काल सहजता को प्राथमिकता दी गई थी, AI का मानवीकरण हमें एक ऐसे तरीके से जुड़ने की अनुमति देता है जो प्राकृतिक और परिचित लगता है।

जब हम किसी अन्य व्यक्ति से संवाद करते हैं, तो हमारी सहज प्रवृत्ति होती है कि हम उन्हें सीधे "आप" कहकर संबोधित करें और यह स्पष्ट निर्देश दें कि हम उनसे कैसा व्यवहार चाहते हैं। यह प्रॉम्प्ट इंजीनियरिंग प्रक्रिया में सहज रूप से परिवर्तित हो जाता है, जहां हम सिस्टम प्रॉम्प्ट्स को निर्दिष्ट करके और आगे-पीछे संवाद करके AI के व्यवहार को निर्देशित करते हैं।

इस तरह से बातचीत को ढालकर, हम AI को निर्देश देने और बदले में प्रासंगिक प्रतिक्रियाएं प्राप्त करने की अवधारणा को आसानी से समझ सकते हैं। मानवीकरण का दृष्टिकोण संज्ञानात्मक बोझ को कम करता है और हमें सिस्टम की तकनीकी जटिलताओं से जूझने के बजाय वर्तमान कार्य पर ध्यान केंद्रित करने की अनुमति देता है।

यह ध्यान रखना महत्वपूर्ण है कि हालांकि मानवीकरण AI सिस्टम को अधिक सुलभ बनाने का एक शक्तिशाली उपकरण है, यह कुछ जोखिमों और सीमाओं के साथ भी आता है। हमारे उपयोगकर्ता अवास्तविक अपेक्षाएं विकसित कर सकते हैं या हमारे सिस्टम के साथ अस्वस्थ भावनात्मक लगाव विकसित कर सकते हैं। प्रॉम्प्ट इंजीनियर्स और डेवलपर्स के रूप में, मानवीकरण के लाभों का लाभ उठाने और उपयोगकर्ताओं को AI की क्षमताओं और सीमाओं की स्पष्ट समझ बनाए रखने के बीच संतुलन बनाना महत्वपूर्ण है।

जैसे-जैसे प्रॉम्प्ट इंजीनियरिंग का क्षेत्र विकसित होता जा रहा है, हम लार्ज लैंग्वेज मॉडल्स के साथ बातचीत करने के तरीके में और अधिक परिष्कार और नवाचार देखने की उम्मीद कर सकते हैं। हालांकि, एक सहज और सुलभ डेवलपर और उपयोगकर्ता अनुभव प्रदान करने के लिए मानवीकरण संभवतः इन सिस्टम के डिज़ाइन में एक मौलिक सिद्धांत बना रहेगा।

निर्देशों को डेटा से अलग करना: एक महत्वपूर्ण सिद्धांत

यह समझना आवश्यक है कि इन सिस्टम की सुरक्षा और विश्वसनीयता का एक मूल सिद्धांत है: निर्देशों को डेटा से अलग करना।

पारंपरिक कंप्यूटर विज्ञान में, निष्क्रिय डेटा और सक्रिय निर्देशों के बीच स्पष्ट अंतर एक मुख्य सुरक्षा सिद्धांत है। यह अलगाव अनजाने में या दुर्भावनापूर्ण कोड के निष्पादन को रोकने में मदद करता है जो सिस्टम की अखंडता और स्थिरता को खतरे में डाल सकता है। हालांकि, आज के एलएलएम, जो मुख्य रूप से चैटबॉट जैसे निर्देश-पालन मॉडल के रूप में विकसित किए गए हैं, में अक्सर यह औपचारिक और सिद्धांतपूर्ण अलगाव नहीं होता है।

जहां तक एलएलएम का संबंध है, निर्देश इनपुट में कहीं भी दिखाई दे सकते हैं, चाहे वह सिस्टम प्रॉम्प्ट हो या उपयोगकर्ता द्वारा प्रदान किया गया प्रॉम्प्ट। अलगाव की यह कमी संभावित कमजोरियों और अवांछनीय व्यवहार का कारण बन सकती है, जो एसक्यूएल इंजेक्शन वाले डेटाबेस या उचित मेमोरी सुरक्षा के बिना ऑपरेटिंग सिस्टम के समान समस्याओं का सामना करती है।

जैसे-जैसे आप एलएलएम के साथ काम करते हैं, इस सीमा के बारे में जागरूक रहना और जोखिमों को कम करने के लिए कदम उठाना महत्वपूर्ण है। एक दृष्टिकोण अपने प्रॉम्प्ट्स और इनपुट को सावधानीपूर्वक तैयार करना है ताकि निर्देशों और डेटा के बीच स्पष्ट अंतर किया जा सके। निर्देश क्या है और किसे निष्क्रिय डेटा के रूप में माना जाना चाहिए, इस पर स्पष्ट मार्गदर्शन प्रदान करने के लिए मार्कअप-शैली टैगिंग का उपयोग किया जाता है। आपका प्रॉम्प्ट एलएलएम को इस अलगाव को बेहतर ढंग से समझने और सम्मान करने में मदद कर सकता है।

आकृति 6. एक्सएमएल का उपयोग करके निर्देशों, स्रोत सामग्री और उपयोगकर्ता के प्रॉम्प्ट के बीच अंतर करना

```
<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>
```

एक अन्य तकनीक एलएलएम को प्रदान किए गए इनपुट पर अतिरिक्त सत्यापन और स्वच्छीकरण परतें लागू करना है। डेटा में एम्बेड किए जा सकने वाले किसी भी संभावित निर्देश या कोड स्निपेट को फ़िल्टर या एस्केप करके, आप अनपेक्षित निष्पादन की संभावनाओं को कम कर सकते हैं। इस उद्देश्य के लिए प्रॉम्प्ट श्रृंखलन जैसे पैटर्न उपयोगी हैं।

इसके अलावा, जैसे-जैसे आप अपनी एप्लिकेशन आर्किटेक्चर डिज़ाइन करते हैं, उच्च स्तर पर निर्देशों और डेटा के पृथक्करण को लागू करने के लिए तंत्रों को शामिल करने पर विचार करें। इसमें निर्देशों और डेटा को संभालने के लिए अलग एंडपॉइंट या एपीआई का उपयोग, कठोर इनपुट सत्यापन और पार्सिंग का कार्यान्वयन, और एलएलएम द्वारा एक्सेस और निष्पादित किए जा सकने वाले कार्यों की सीमा को सीमित करने के लिए न्यूनतम विशेषाधिकार के सिद्धांत को लागू करना शामिल हो सकता है।

न्यूनतम विशेषाधिकार का सिद्धांत

न्यूनतम विशेषाधिकार के सिद्धांत को अपनाना एक ऐसी विशिष्ट पार्टी की तरह है जहां मेहमानों को केवल उन कमरों तक पहुंच मिलती है जिनकी उन्हें वास्तव में आवश्यकता है। कल्पना कीजिए कि आप इस दावत की मेजबानी एक विशाल महल में कर रहे हैं। हर किसी को वाइन सेलर या मास्टर बेडरूम में घूमने की जरूरत नहीं है, है ना? इस सिद्धांत को लागू करके, आप अनिवार्य रूप से ऐसी चाबियां बांट रहे हैं जो केवल विशिष्ट दरवाजे ही खोलती हैं, यह सुनिश्चित करते हुए कि प्रत्येक मेहमान, या हमारे मामले में, आपके एलएलएम एप्लिकेशन का प्रत्येक घटक, केवल अपनी भूमिका को पूरा करने के लिए आवश्यक पहुंच रखता है।

यह सिर्फ चाबियों के साथ कंजूसी करने के बारे में नहीं है, यह इस बात को स्वीकार करने के बारे में है कि एक ऐसी दुनिया में जहां खतरे कहीं से भी आ सकते हैं, समझदारी इसी में है कि खेल के मैदान को सीमित किया जाए। यदि कोई अनचाहा व्यक्ति आपकी पार्टी में घुस भी जाता है, तो वह खुद को प्रवेश कक्ष तक ही सीमित पाएगा, जिससे वह जो शरारत कर सकता है वह भी सीमित हो जाती है। इसलिए, अपने एलएलएम एप्लिकेशन को सुरक्षित करते समय याद रखें: केवल उन कमरों की चाबियां दें जो आवश्यक हैं, और महल के बाकी हिस्से को सुरक्षित रखें। यह सिर्फ अच्छे शिष्टाचार की बात नहीं है; यह अच्छी सुरक्षा है।

हालांकि एलएलएम की वर्तमान स्थिति में निर्देशों और डेटा का औपचारिक पृथक्करण नहीं हो सकता है, एक डेवलपर के रूप में आपके लिए यह आवश्यक है कि आप इस सीमा के प्रति सचेत रहें और जोखिमों को कम करने के लिए सक्रिय उपाय करें। पारंपरिक कंप्यूटर विज्ञान से सर्वोत्तम प्रथाओं को लागू करके और उन्हें एलएलएम की विशिष्ट विशेषताओं के अनुरूप ढालकर, आप अधिक सुरक्षित और विश्वसनीय एप्लिकेशन बना सकते हैं जो अपने सिस्टम की अखंडता को बनाए रखते हुए इन मॉडलों की शक्ति का उपयोग करते हैं।

प्रॉम्प्ट डिस्टिलेशन

सटीक प्रॉम्प्ट तैयार करना अक्सर एक चुनौतीपूर्ण और समय लेने वाला कार्य होता है, जिसमें लक्षित डोमेन और भाषा मॉडल की बारीकियों की गहरी समझ की आवश्यकता होती है। यहीं पर "प्रॉम्प्ट डिस्टिलेशन" तकनीक काम आती है, जो प्रॉम्प्ट इंजीनियरिंग के लिए एक शक्तिशाली दृष्टिकोण प्रदान करती है जो प्रक्रिया को सुव्यवस्थित और अनुकूलित करने के लिए बड़े भाषा मॉडल (LLMs) की क्षमताओं का लाभ उठाती है।

प्रॉम्प्ट डिस्टिलेशन एक बहु-चरणीय तकनीक है जिसमें प्रॉम्प्ट के निर्माण, परिष्करण और अनुकूलन में सहायता के लिए LLMs का उपयोग किया जाता है। केवल मानवीय विशेषज्ञता और अंतर्ज्ञान पर निर्भर रहने के बजाय, यह दृष्टिकोण उच्च-गुणवत्ता वाले प्रॉम्प्ट को सहयोगात्मक रूप से तैयार करने के लिए LLMs के ज्ञान और जनरेटिव क्षमताओं का उपयोग करता है।

जनरेशन, परिष्करण और एकीकरण की एक पुनरावर्ती प्रक्रिया में संलग्न होकर, प्रॉम्प्ट डिस्टिलेशन आपको ऐसे प्रॉम्प्ट बनाने में सक्षम बनाता है जो अधिक सुसंगत, व्यापक और वांछित कार्य या आउटपुट के साथ संरेखित हैं। ध्यान दें कि डिस्टिलेशन प्रक्रिया को OpenAI या Anthropic जैसे बड़े AI विक्रेताओं द्वारा प्रदान किए गए कई "प्लेग्राउंड्स" में से किसी एक में मैन्युअल रूप से किया जा सकता है, या इसे उपयोग के मामले के आधार पर आपके एप्लिकेशन कोड के हिस्से के रूप में स्वचालित किया जा सकता है।

यह कैसे काम करता है

प्रॉम्प्ट डिस्टिलेशन में आमतौर पर निम्नलिखित चरण शामिल होते हैं:

1. **मूल उद्देश्य की पहचान**: प्रॉम्प्ट का विश्लेषण करें ताकि इसके प्राथमिक उद्देश्य और वांछित परिणाम का पता लगाया जा सके। किसी भी बाहरी जानकारी को हटा दें और प्रॉम्प्ट के मूल उद्देश्य पर ध्यान केंद्रित करें।

2. **अस्पष्टता को दूर करें**: किसी भी अस्पष्ट या अनिश्चित भाषा के लिए प्रॉम्प्ट की समीक्षा करें। अर्थ को स्पष्ट करें और सटीक और प्रासंगिक प्रतिक्रियाएं उत्पन्न करने के लिए AI को मार्गदर्शन प्रदान करने हेतु विशिष्ट विवरण प्रदान करें।
3. **भाषा को सरल बनाएं**: स्पष्ट और संक्षिप्त भाषा का उपयोग करके प्रॉम्प्ट को सरल बनाएं। जटिल वाक्य संरचनाओं, तकनीकी शब्दजाल, या अनावश्यक विवरणों से बचें जो AI को भ्रमित कर सकते हैं या शोर पैदा कर सकते हैं।
4. **प्रासंगिक संदर्भ प्रदान करें**: केवल सबसे प्रासंगिक संदर्भगत जानकारी शामिल करें जो AI को प्रॉम्प्ट को प्रभावी ढंग से समझने और संसाधित करने के लिए आवश्यक है। अप्रासंगिक या अतिरिक्त विवरणों को शामिल करने से बचें जो मूल उद्देश्य से ध्यान भटका सकते हैं।
5. **पुनरावृत्ति और परिष्करण**: AI की प्रतिक्रियाओं और फीडबैक के आधार पर प्रॉम्प्ट को लगातार दोहराएं और परिष्कृत करें। उत्पन्न आउटपुट का मूल्यांकन करें और प्रॉम्प्ट की स्पष्टता और प्रभावशीलता में सुधार के लिए आवश्यक समायोजन करें। वैकल्पिक रूप से प्रॉम्प्ट ऑब्जेक्ट का उपयोग करके डेटाबेस में अपने प्रॉम्प्ट को वर्जन करें ताकि पुनरावृत्तियों पर नज़र रखी जा सके और रनटाइम पर आसानी से परिवर्तनों को वापस लाया जा सके।

प्रारंभिक प्रॉम्प्ट निर्माण

प्रॉम्प्ट डिस्टिलेशन का पहला चरण एलएलएम को वांछित कार्य या आउटपुट का एक उच्च-स्तरीय विवरण या उद्देश्य प्रदान करना है। यह एक संक्षिप्त कथन या कीवर्ड्स का समूह हो सकता है जो प्रॉम्प्ट को क्या हासिल करना चाहिए, इसके मूल को दर्शाता है। एलएलएम तब इस इनपुट के आधार पर एक प्रारंभिक प्रॉम्प्ट तैयार करता है, जिसमें प्रासंगिक निर्देश, उदाहरण, या संदर्भ शामिल होते हैं जो वह उपयुक्त समझता है।

आइए क्लॉड 3 ओपस को निम्नलिखित प्रारंभिक प्रॉम्प्ट देने का प्रयास करें:

```
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 निम्नलिखित प्रतिक्रिया के साथ प्रतिउत्तर देता है:

```
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.
```

प्रॉम्प्ट परिष्करण

प्रारंभिक प्रॉम्प्ट तैयार होने के बाद, इसकी समीक्षा करें और फिर LLM से किसी भी अतिरिक्त संदर्भ, बाधाओं, या प्रतिक्रिया पर विचार करने के लिए कहें। LLM को प्रदान किए गए इनपुट और कार्य की अपनी समझ के आधार पर प्रॉम्प्ट का एक परिष्कृत संस्करण तैयार करने दें। यह परिष्करण प्रक्रिया पुनरावर्ती रूप से दोहराई जा सकती है, जहां प्रत्येक पुनरावृत्ति में उपयोगकर्ता और LLM के बीच सहयोग के आधार पर प्रॉम्प्ट को और अधिक परिष्कृत किया जाता है।

यहाँ मेरा परिष्करण अनुरोध है, जो पिछली आसवन प्रक्रिया को जारी रखता है।

```
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.
```

और Claude का जवाब:

```
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.
```

प्रॉम्प्ट कम्प्रेशन

जैसा कि आपने देखा होगा, एलएलएम आमतौर पर शब्दबहुल होते हैं, जब तक कि आप उन्हें संक्षिप्त होने के लिए न कहें। पिछले खंडों में जिस प्रॉम्प्ट पर हम काम कर रहे थे वह काफी लंबा है, इसलिए आइए Claude से इसे छोटा करवाते हैं:

```
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 निम्नलिखित काफी छोटे प्रॉम्प्ट के साथ उत्तर देता है:

```
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.
```

सिस्टम निर्देश और संदर्भ एकीकरण

प्रॉम्प्ट को परिष्कृत करने के अलावा, LLM अंतिम आउटपुट को निर्देशित करने के लिए उपयुक्त सिस्टम निर्देश या संदर्भ जानकारी भी उत्पन्न कर सकता है। जब आप

अपने एप्लिकेशन कोड में एकीकृत होने वाली AI रूटीन के लिए प्रॉम्प्ट इंजीनियरिंग कर रहे होंगे, तो आप इस परिशोधन के चरण में निश्चित रूप से आउटपुट प्रतिबंधों पर ध्यान केंद्रित करेंगे, लेकिन आप वांछित टोन, शैली, प्रारूप, या किसी अन्य प्रासंगिक मापदंडों पर भी काम कर सकते हैं जो उत्पन्न प्रतिक्रिया को प्रभावित करते हैं।

अंतिम प्रॉम्प्ट संयोजन

प्रॉम्प्ट परिशोधन प्रक्रिया का चरम अंतिम प्रॉम्प्ट का संयोजन है। इसमें परिष्कृत प्रॉम्प्ट, उत्पन्न सिस्टम निर्देशों, और एकीकृत संदर्भ को एक सुसंगत और व्यापक कोड में संयोजित करना शामिल है जो वांछित आउटपुट उत्पन्न करने के लिए तैयार है।

आप अंतिम प्रॉम्प्ट संयोजन चरण में फिर से प्रॉम्प्ट संपीड़न का प्रयोग कर सकते हैं, LLM से प्रॉम्प्ट के शब्दों को संभव सबसे छोटी टोकन श्रृंखला में संकुचित करने के लिए कह सकते हैं, जबकि इसके व्यवहार का सार बनाए रखा जाए। यह निश्चित रूप से हिट या मिस अभ्यास है, लेकिन विशेष रूप से बड़े पैमाने पर चलने वाले प्रॉम्प्ट के मामले में, दक्षता में सुधार से आपको टोकन खपत में काफी पैसे बचा सकते हैं।

प्रमुख लाभ

अपने प्रॉम्प्ट को परिष्कृत करने के लिए LLM के ज्ञान और जनरेटिव क्षमताओं का लाभ उठाकर, आपके परिणामी प्रॉम्प्ट के अच्छी तरह से संरचित, सूचनात्मक और विशिष्ट कार्य के लिए अनुकूलित होने की अधिक संभावना होती है। पुनरावर्ती परिशोधन प्रक्रिया यह सुनिश्चित करने में मदद करती है कि प्रॉम्प्ट उच्च गुणवत्ता वाले हों और प्रभावी ढंग से वांछित इरादे को पकड़ें। अन्य लाभों में शामिल हैं:

दक्षता और गति: प्रॉम्प्ट परिशोधन प्रॉम्प्ट निर्माण और परिष्करण के कुछ पहलुओं को स्वचालित करके प्रॉम्प्ट इंजीनियरिंग प्रक्रिया को सुव्यवस्थित करता है। तकनीक की सहयोगात्मक प्रकृति प्रभावी प्रॉम्प्ट की ओर तेजी से अभिसरण की अनुमति देती

है, जिससे मैनुअल प्रॉम्प्ट तैयार करने में लगने वाला समय और प्रयास कम हो जाता है।

स्थिरता और मापनीयता: प्रॉम्प्ट इंजीनियरिंग प्रक्रिया में LLM का उपयोग प्रॉम्प्ट में स्थिरता बनाए रखने में मदद करता है, क्योंकि LLM पिछले सफल प्रॉम्प्ट से सर्वोत्तम प्रथाओं और पैटर्न को सीख और लागू कर सकते हैं। यह स्थिरता, बड़े पैमाने पर प्रॉम्प्ट उत्पन्न करने की क्षमता के साथ मिलकर, प्रॉम्प्ट परिशोधन को बड़े पैमाने पर AI-संचालित अनुप्रयोगों के लिए एक मूल्यवान तकनीक बनाती है।

परियोजना विचार: लाइब्रेरी स्तर पर ऐसे टूल जो उन सिस्टम में प्रॉम्प्ट वर्जनिंग और ग्रेडिंग की प्रक्रिया को सरल बनाते हैं जो अपने एप्लिकेशन कोड के हिस्से के रूप में स्वचालित प्रॉम्प्ट परिशोधन करते हैं।

प्रॉम्प्ट परिशोधन को लागू करने के लिए, डेवलपर्स एक ऐसा कार्यप्रवाह या पाइपलाइन डिज़ाइन कर सकते हैं जो प्रॉम्प्ट इंजीनियरिंग प्रक्रिया के विभिन्न चरणों में LLM को एकीकृत करता है। यह API कॉल, कस्टम टूलिंग, या एकीकृत विकास वातावरण के माध्यम से प्राप्त किया जा सकता है जो प्रॉम्प्ट निर्माण के दौरान उपयोगकर्ताओं और LLM के बीच निर्बाध इंटरैक्शन की सुविधा प्रदान करते हैं। विशिष्ट कार्यान्वयन विवरण चुने गए LLM प्लेटफॉर्म और एप्लिकेशन की आवश्यकताओं के आधार पर भिन्न हो सकते हैं।

फाइन-ट्यूनिंग के बारे में क्या?

इस पुस्तक में, हम प्रॉम्प्ट इंजीनियरिंग और RAG को विस्तार से कवर करते हैं, लेकिन फाइन-ट्यूनिंग को नहीं। इस निर्णय का मुख्य कारण यह है कि, मेरी राय में, अधिकांश एप्लिकेशन डेवलपर्स को अपनी AI एकीकरण आवश्यकताओं के लिए फाइन-ट्यूनिंग की आवश्यकता नहीं है।

प्रॉम्प्ट इंजीनियरिंग, जिसमें शून्य से कुछ-शॉट उदाहरणों, प्रतिबंधों, और निर्देशों के साथ सावधानीपूर्वक प्रॉम्प्ट्स तैयार करना शामिल है, विभिन्न कार्यों के लिए प्रासंगिक

और सटीक प्रतिक्रियाएं उत्पन्न करने के लिए मॉडल को प्रभावी ढंग से मार्गदर्शित कर सकती है। स्पष्ट संदर्भ प्रदान करके और सुनियोजित प्रॉम्प्ट्स के माध्यम से मार्ग को संकीर्ण करके, आप फाइन-ट्यूनिंग की आवश्यकता के बिना बड़े भाषा मॉडल के विशाल ज्ञान का लाभ उठा सकते हैं।

इसी तरह, पुनर्प्राप्ति संवर्धित जनरेशन (RAG) एप्लिकेशन में AI को एकीकृत करने के लिए एक शक्तिशाली दृष्टिकोण प्रदान करता है। बाहरी ज्ञान भंडार या दस्तावेजों से प्रासंगिक जानकारी को गतिशील रूप से पुनर्प्राप्त करके, RAG प्रॉम्प्टिंग के समय मॉडल को केंद्रित संदर्भ प्रदान करता है। यह मॉडल को अधिक सटीक, अप-टू-डेट, और डोमेन-विशिष्ट प्रतिक्रियाएं उत्पन्न करने में सक्षम बनाता है, बिना फाइन-ट्यूनिंग की समय और संसाधन-गहन प्रक्रिया की आवश्यकता के।

हालांकि फाइन-ट्यूनिंग अत्यधिक विशेषज्ञ डोमेन या कार्यों के लिए लाभदायक हो सकती है जिन्हें अनुकूलन के गहरे स्तर की आवश्यकता होती है, यह अक्सर महत्वपूर्ण कम्प्यूटेशनल लागत, डेटा आवश्यकताओं, और रखरखाव के ओवरहेड के साथ आती है। अधिकांश एप्लिकेशन विकास परिदृश्यों के लिए, प्रभावी प्रॉम्प्ट इंजीनियरिंग और RAG का संयोजन वांछित AI-संचालित कार्यक्षमता और उपयोगकर्ता अनुभव प्राप्त करने के लिए पर्याप्त होना चाहिए।

रिट्रीवल ऑगमेंटेड जेनरेशन (RAG)

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

रिट्रीवल ऑगमेंटेड जेनरेशन क्या है?

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

RAG कैसे काम करता है?

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

आपके एप्लिकेशन में RAG का उपयोग क्यों करें?

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

अपने एप्लिकेशन में RAG को लागू करना

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

ज्ञान स्रोतों की तैयारी (खंडीकरण)

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रस्ताव खंडीकरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कार्यान्वयन टिप्पणियां

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

गुणवत्ता जाँच

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रस्ताव-आधारित पुनर्प्राप्ति के लाभ

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

RAG के वास्तविक-दुनिया के उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

केस स्टडी: एम्बेडिंग्स के बिना कर तैयारी एप्लिकेशन में RAG

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

बुद्धिमान प्रश्न अनुकूलन (IQO)

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

पुनः क्रमांकन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

RAG मूल्यांकन (RAGAs)

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

विश्वसनीयता

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उत्तर की प्रासंगिकता

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Context Precision

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Context Relevancy

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Context Recall

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Context Entities Recall

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Answer Semantic Similarity (ANSS)

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उत्तर की सटीकता

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

पहलू समीक्षा

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

चुनौतियां और भविष्य का दृष्टिकोण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

सिमैंटिक चंकिंग: संदर्भ-जागरूक विभाजन के साथ पुनर्प्राप्ति को बढ़ाना

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

पदानुक्रमिक इंडेक्सिंग: बेहतर पुनर्प्राप्ति के लिए डेटा को संरचित करना

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

सेल्फ-RAG: एक स्व-चिंतनशील संवर्धन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

HyDE: परिकल्पित दस्तावेज़ एम्बेडिंग्स

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-

hi पर।

प्रतिरोधात्मक शिक्षण क्या है?

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कार्यकर्ताओं की बहुलता

मैं अपने एआई घटकों को छोटे, लगभग-मानवीय आभासी “कार्यकर्ताओं” के रूप में देखना पसंद करता हूं जो विशिष्ट कार्यों को करने या जटिल निर्णय लेने के लिए मेरी एप्लिकेशन लॉजिक में निर्बाध रूप से एकीकृत किए जा सकते हैं। विचार एलएलएम की क्षमताओं को जानबूझकर मानवीय बनाना है, ताकि कोई भी बहुत उत्साहित न हो और उन्हें ऐसी जादुई विशेषताएं न सौंपे जो उनके पास नहीं हैं।

जटिल एल्गोरिथ्म या समय लेने वाले मैनुअल कार्यान्वयन पर पूरी तरह से निर्भर रहने के बजाय, डेवलपर्स एआई घटकों को बुद्धिमान, समर्पित, मानव-जैसी इकाइयों के रूप में कल्पना कर सकते हैं जिन्हें जटिल समस्याओं से निपटने और अपने प्रशिक्षण और ज्ञान के आधार पर समाधान प्रदान करने के लिए जब भी आवश्यकता हो तब बुलाया जा सकता है। ये इकाइयां न तो विचलित होती हैं, और न ही बीमार पड़ती हैं। वे अचानक से चीजों को अलग तरीके से करने का निर्णय नहीं लेतीं जैसा कि उन्हें करने के लिए निर्देशित किया गया है, और सामान्य तौर पर, यदि सही ढंग से प्रोग्राम किया गया है, तो वे गलतियां भी नहीं करतीं।

तकनीकी दृष्टि से, इस दृष्टिकोण के पीछे का मुख्य सिद्धांत जटिल कार्यों या निर्णय लेने की प्रक्रियाओं को छोटी, अधिक प्रबंधनीय इकाइयों में विभाजित करना है जिन्हें विशेषज्ञ एआई कार्यकर्ताओं द्वारा संभाला जा सकता है। प्रत्येक कार्यकर्ता समस्या के एक विशिष्ट पहलू पर ध्यान केंद्रित करने के लिए डिज़ाइन किया गया है, जो अपनी अनूठी विशेषज्ञता और क्षमताओं को सामने लाता है। कई एआई कार्यकर्ताओं के बीच कार्यभार को वितरित करके, एप्लिकेशन अधिक दक्षता, स्केलेबिलिटी और अनुकूलनीयता प्राप्त कर सकता है।

उदाहरण के लिए, एक वेब एप्लिकेशन पर विचार करें जिसे उपयोगकर्ता-निर्मित सामग्री के रीयल-टाइम मॉडरेशन की आवश्यकता होती है। शुरू से एक व्यापक मॉडरेशन सिस्टम को लागू करना एक चुनौतीपूर्ण कार्य होगा, जिसमें महत्वपूर्ण विकास प्रयास और निरंतर रखरखाव की आवश्यकता होगी। हालाँकि, कार्यकर्ताओं की बहुलता दृष्टिकोण का उपयोग करके, डेवलपर्स एप्लिकेशन लॉजिक में एआई-संचालित मॉडरेशन कार्यकर्ताओं को एकीकृत कर सकते हैं। ये कार्यकर्ता स्वचालित रूप से अनुचित सामग्री का विश्लेषण और फ्लैग कर सकते हैं, जिससे डेवलपर्स को एप्लिकेशन के अन्य महत्वपूर्ण पहलुओं पर ध्यान केंद्रित करने की स्वतंत्रता मिलती है।

स्वतंत्र पुन: प्रयोज्य घटकों के रूप में एआई कार्यकर्ता

कार्यकर्ताओं की बहुलता दृष्टिकोण का एक प्रमुख पहलू इसकी मॉड्यूलरिटी है। वस्तु-उन्मुख प्रोग्रामिंग के समर्थक हमें दशकों से बता रहे हैं कि ऑब्जेक्ट इंटरैक्शन को संदेशों के रूप में सोचें। ठीक है, एआई कार्यकर्ताओं को स्वतंत्र, पुन: प्रयोज्य घटकों के रूप में डिज़ाइन किया जा सकता है जो सादी भाषा के संदेशों के माध्यम से "एक-दूसरे से बात कर सकते हैं", बिल्कुल वैसे ही जैसे वे वास्तव में एक-दूसरे से बात करने वाले छोटे इंसान हों। यह ढीला-जुड़ा दृष्टिकोण एप्लिकेशन को समय के साथ अनुकूलित और विकसित होने की अनुमति देता है, जैसे-जैसे नई एआई तकनीकें सामने आती हैं या व्यावसायिक तर्क की आवश्यकताएं बदलती हैं।

व्यवहार में, कंपोनेंट्स के बीच स्पष्ट इंटरफ़ेस और संचार प्रोटोकॉल को डिज़ाइन करने की आवश्यकता नहीं बदली है, भले ही AI वर्कर्स शामिल हों। आपको अभी भी प्रदर्शन,

स्केलेबिलिटी और सुरक्षा जैसे अन्य कारकों पर विचार करना होगा, लेकिन अब कुछ नई "सॉफ्ट आवश्यकताएं" भी हैं जिन पर विचार करना है। उदाहरण के लिए, कई उपयोगकर्ता अपने निजी डेटा को नए AI मॉडल को प्रशिक्षित करने में उपयोग किए जाने का विरोध करते हैं। क्या आपने सत्यापित किया है कि आप जिस मॉडल प्रदाता का उपयोग कर रहे हैं, वह किस स्तर की गोपनीयता प्रदान करता है?

क्या AI वर्कर्स माइक्रोसर्विसेज की तरह हैं?

जैसे-जैसे आप वर्कर्स की बहुलता के दृष्टिकोण के बारे में पढ़ते हैं, आपको माइक्रोसर्विसेज आर्किटेक्चर से कुछ समानताएं दिखाई दे सकती हैं। दोनों जटिल सिस्टम को छोटी, अधिक प्रबंधनीय और स्वतंत्र रूप से तैनात इकाइयों में विभाजित करने पर जोर देते हैं। जिस तरह माइक्रोसर्विसेज को लूज कपल्ड, विशिष्ट व्यावसायिक क्षमताओं पर केंद्रित और सुपरिभाषित APIs के माध्यम से संवाद करने के लिए डिज़ाइन किया जाता है, उसी तरह AI वर्कर्स को मॉड्यूलर, अपने कार्यों में विशेषज्ञ और स्पष्ट इंटरफ़ेस और संचार प्रोटोकॉल के माध्यम से एक-दूसरे के साथ बातचीत करने के लिए डिज़ाइन किया जाता है।

हालांकि, ध्यान रखने योग्य कुछ प्रमुख अंतर हैं। जहां माइक्रोसर्विसेज को आमतौर पर अलग-अलग मशीनों या कंटेनरों पर चलने वाली अलग प्रक्रियाओं या सेवाओं के रूप में लागू किया जाता है, वहीं AI वर्कर्स को आपकी विशिष्ट आवश्यकताओं और स्केलेबिलिटी की जरूरतों के आधार पर एकल एप्लिकेशन के भीतर स्टैंडअलोन कंपोनेंट्स के रूप में या अलग सेवाओं के रूप में लागू किया जा सकता है। इसके अतिरिक्त, AI वर्कर्स के बीच संचार में अक्सर प्रॉम्प्ट्स, निर्देश और जनरेट की गई सामग्री जैसी समृद्ध, प्राकृतिक भाषा-आधारित जानकारी का आदान-प्रदान शामिल होता है, न कि माइक्रोसर्विसेज में आमतौर पर उपयोग किए जाने वाले अधिक संरचित डेटा प्रारूप।

इन अंतरों के बावजूद, मॉड्यूलैरिटी, लूज कपलिंग और स्पष्ट संचार इंटरफ़ेस के सिद्धांत दोनों पैटर्न के लिए केंद्रीय बने रहते हैं। अपनी AI वर्कर आर्किटेक्चर में इन सिद्धांतों को लागू करके, आप लचीले, स्केलेबल और रखरखाव योग्य सिस्टम

> बना सकते हैं जो जटिल समस्याओं को हल करने और अपने उपयोगकर्ताओं को मूल्य प्रदान करने के लिए AI की शक्ति का लाभ उठाते हैं।

वर्कर्स की बहुलता का दृष्टिकोण विभिन्न डोमेन और एप्लिकेशन में लागू किया जा सकता है, जो जटिल कार्यों को संभालने और बुद्धिमान समाधान प्रदान करने के लिए AI की शक्ति का लाभ उठाता है। आइए देखें कि विभिन्न संदर्भों में AI वर्कर्स का उपयोग कैसे किया जा सकता है।

खाता प्रबंधन

लगभग हर स्टैंडअलोन वेब एप्लिकेशन में खाते (या उपयोगकर्ता) की अवधारणा होती है। Olympia में, हम एक AccountManager AI वर्कर का उपयोग करते हैं जो उपयोगकर्ता खातों से संबंधित विभिन्न प्रकार के परिवर्तन अनुरोधों को संभालने के लिए प्रोग्राम किया गया है।

इसका निर्देश इस प्रकार है:

```
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 द्वारा उत्पन्न खाते की प्रारंभिक स्थिति बस खाते का एक पाठ विवरण है, जिसमें उपयोगकर्ताओं, सदस्यताओं आदि जैसे प्रासंगिक संबंधित डेटा शामिल हैं।

AccountManager के लिए उपलब्ध कार्यों की श्रृंखला इसे उपयोगकर्ता की सदस्यता को संपादित करने, AI सलाहकारों और अन्य प्रकार के भुगतान किए गए ऐड-ऑन को जोड़ने और हटाने, और खाता मालिक को सूचना ईमेल भेजने की क्षमता प्रदान करती है। finished फ़ंक्शन के अलावा, यह अपनी प्रोसेसिंग के दौरान त्रुटि का सामना करने या किसी अनुरोध के साथ किसी अन्य प्रकार की सहायता की आवश्यकता होने पर notify_human_administrator भी कर सकता है।

ध्यान दें कि प्रश्नों की स्थिति में, AccountManager Olympia के ज्ञान भंडार में खोज कर सकता है, जहां इसे सीमांत मामलों और किसी अन्य स्थिति को संभालने के लिए निर्देश मिल सकते हैं जो इसे आगे बढ़ने के तरीके के बारे में अनिश्चित छोड़ देते हैं।

ई-कॉमर्स अनुप्रयोग

ई-कॉमर्स के क्षेत्र में, AI कर्मचारी उपयोगकर्ता अनुभव को बढ़ाने और व्यावसायिक संचालन को अनुकूलित करने में महत्वपूर्ण भूमिका निभा सकते हैं। यहाँ कुछ तरीके दिए गए हैं जिनमें AI कर्मचारियों का उपयोग किया जा सकता है:

उत्पाद अनुशंसाएं

ई-कॉमर्स में AI कर्मचारियों के सबसे शक्तिशाली अनुप्रयोगों में से एक है वैयक्तिकृत उत्पाद अनुशंसाएं उत्पन्न करना। उपयोगकर्ता व्यवहार, खरीदारी इतिहास और प्राथमिकताओं का विश्लेषण करके, ये कर्मचारी प्रत्येक व्यक्तिगत उपयोगकर्ता की रुचियों और आवश्यकताओं के अनुरूप उत्पादों का सुझाव दे सकते हैं।

प्रभावी उत्पाद अनुशंसाओं की कुंजी सहयोगात्मक फ़िल्टरिंग और सामग्री-आधारित फ़िल्टरिंग तकनीकों के संयोजन का लाभ उठाना है। सहयोगात्मक फ़िल्टरिंग समान उपयोगकर्ताओं के व्यवहार को देखकर पैटर्न की पहचान करती है और उन लोगों

की पसंद के आधार पर सिफारिशें करती है जिनकी समान रुचियां हैं। दूसरी ओर, सामग्री-आधारित फ़िल्टरिंग उत्पादों की विशेषताओं और गुणों पर ध्यान केंद्रित करती है, जिससे ऐसी वस्तुओं की सिफारिश की जाती है जो उपयोगकर्ता द्वारा पहले दिखाई गई रुचि वाली वस्तुओं के समान विशेषताएं साझा करती हैं।

यहाँ Ruby में एक उत्पाद अनुशंसा कर्मचारी को कार्यान्वित करने का एक सरलीकृत उदाहरण दिया गया है, इस बार "Railway Oriented (ROP)" फंक्शनल प्रोग्रामिंग शैली का उपयोग करते हुए:

```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 की फंक्शनल प्रोग्रामिंग की शैली जो उदाहरण में उपयोग की गई है, वह F# और Rust से प्रभावित है। आप इस तकनीक के बारे में मेरे मित्र Chad Wooley के तकनीक के स्पष्टीकरण में GitLab पर और अधिक पढ़ सकते हैं

इस उदाहरण में, ProductRecommendationWorker एक उपयोगकर्ता को इनपुट

के रूप में लेता है और फंक्शनल चरणों की श्रृंखला में एक वैल्यू ऑब्जेक्ट को पास करके वैयक्तिकृत उत्पाद सिफारिशें उत्पन्न करता है। आइए प्रत्येक चरण को समझें:

1. ValidateUser.validate: यह चरण यह सुनिश्चित करता है कि उपयोगकर्ता वैध है और वैयक्तिकृत सिफारिशों के लिए योग्य है। यह जांचता है कि उपयोगकर्ता मौजूद है, सक्रिय है, और सिफारिशें उत्पन्न करने के लिए आवश्यक डेटा उपलब्ध है। यदि सत्यापन विफल होता है, तो एक त्रुटि परिणाम वापस किया जाता है, और श्रृंखला शॉर्ट-सर्किट हो जाती है।
2. AnalyzeCurrentSession.analyze: यदि उपयोगकर्ता वैध है, तो यह चरण संदर्भगत जानकारी एकत्र करने के लिए उपयोगकर्ता के वर्तमान ब्राउज़िंग सत्र का विश्लेषण करता है। यह उपयोगकर्ता की वर्तमान रुचियों और इरादों को समझने के लिए देखे गए उत्पादों, खोज क्वेरी और कार्ट सामग्री जैसी हाल की बातचीत को देखता है।
3. CollaborativeFilter.filter: समान उपयोगकर्ताओं के व्यवहार का उपयोग करते हुए, यह चरण सहयोगात्मक फ़िल्टरिंग तकनीकों को लागू करता है ताकि ऐसे उत्पादों की पहचान की जा सके जो उपयोगकर्ता के लिए रुचिकर हो सकते हैं। यह संभावित सिफारिशों का एक सेट तैयार करने के लिए खरीदारी इतिहास, रेटिंग और उपयोगकर्ता-वस्तु इंटरैक्शन जैसे कारकों पर विचार करता है।
4. ContentBasedFilter.filter: यह चरण सामग्री-आधारित फ़िल्टरिंग लागू करके संभावित सिफारिशों को और परिष्कृत करता है। यह सबसे प्रासंगिक वस्तुओं को चुनने के लिए संभावित उत्पादों की विशेषताओं और चरित्र की तुलना उपयोगकर्ता की प्राथमिकताओं और ऐतिहासिक डेटा से करता है।
5. ProductSelector.select: अंत में, यह चरण पूर्व-निर्धारित मानदंडों जैसे प्रासंगिकता स्कोर, लोकप्रियता, या अन्य व्यावसायिक नियमों के आधार पर फ़िल्टर की गई सिफारिशों से शीर्ष N उत्पादों का चयन करता है। चयनित उत्पादों को फिर अंतिम वैयक्तिकृत सिफारिशों के रूप में वापस किया जाता है।

यहां फंक्शनल Ruby प्रोग्रामिंग शैली का उपयोग करने की खूबसूरती यह है कि यह हमें इन चरणों को स्पष्ट और संक्षिप्त तरीके से श्रृंखलाबद्ध करने की अनुमति देता है। प्रत्येक चरण एक विशिष्ट कार्य पर केंद्रित होता है और एक Result ऑब्जेक्ट लौटाता

है, जो या तो सफलता (ok) या त्रुटि (err) हो सकता है। यदि किसी भी चरण में त्रुटि आती है, तो श्रृंखला शॉर्ट-सर्किट हो जाती है, और त्रुटि अंतिम परिणाम तक प्रसारित हो जाती है।

अंत में दिए गए case स्टेटमेंट में, हम अंतिम परिणाम पर पैटर्न मैचिंग करते हैं। यदि परिणाम एक त्रुटि है (ProductRecommendationError), तो हम मॉनिटरिंग और डीबगिंग उद्देश्यों के लिए Honeybadger जैसे टूल का उपयोग करके त्रुटि को लॉग करते हैं। यदि परिणाम सफल है (ProductRecommendations), तो हम Wisper पब/सब लाइब्रेरी का उपयोग करके एक :new_recommendations इवेंट को प्रसारित करते हैं, जिसमें उपयोगकर्ता और जनरेट की गई सिफारिशें शामिल होती हैं।

फंक्शनल प्रोग्रामिंग तकनीकों का लाभ उठाकर, हम एक मॉड्यूलर और रखरखाव योग्य प्रोडक्ट रेकमेंडेशन वर्कर बना सकते हैं। प्रत्येक चरण स्व-निहित है और इसका आसानी से परीक्षण, संशोधन या प्रतिस्थापन किया जा सकता है, जिससे समग्र प्रवाह प्रभावित नहीं होता। पैटर्न मैचिंग और Result क्लास का उपयोग हमें त्रुटियों को सुचारू रूप से संभालने में मदद करता है और यह सुनिश्चित करता है कि यदि किसी चरण में कोई समस्या आती है तो वर्कर तुरंत विफल हो जाए।

बेशक, यह एक सरलीकृत उदाहरण है, और वास्तविक परिदृश्य में, आपको अपने ई-कॉमर्स प्लेटफ़ॉर्म के साथ एकीकरण करने, एज केस को संभालने, और यहां तक कि रेकमेंडेशन एल्गोरिथम के कार्यान्वयन में भी जाने की आवश्यकता होगी। हालांकि, समस्या को छोटे चरणों में विभाजित करने और फंक्शनल प्रोग्रामिंग तकनीकों का लाभ उठाने के मूल सिद्धांत समान रहते हैं।

धोखाधड़ी का पता लगाना

यहाँ एक सरलीकृत उदाहरण है कि आप Ruby में उसी Railway Oriented Programming (ROP) शैली का उपयोग करके धोखाधड़ी का पता लगाने वाला वर्कर कैसे लागू कर सकते हैं:

```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 क्लास एक value object है जो किसी दिए गए लेनदेन के लिए धोखाधड़ी पहचान स्थिति को समाहित करता है। यह विभिन्न जोखिम कारकों के आधार पर किसी लेनदेन से जुड़ी धोखाधड़ी के जोखिम का विश्लेषण और मूल्यांकन करने का एक संरचित तरीका प्रदान करता है।

```ruby
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 क्लास में निम्नलिखित विशेषताएं हैं:

- transaction: धोखाधड़ी के लिए विश्लेषण किए जा रहे लेन-देन का संदर्भ।
- risk_factors: एक ऐरे जो लेन-देन से जुड़े जोखिम कारकों को संग्रहीत करता है। प्रत्येक जोखिम कारक एक हैश के रूप में दर्शाया जाता है, जहां की (key) जोखिम कारक का विवरण है, और मान (value) उस जोखिम कारक से जुड़ी धोखाधड़ी की संभावना है।

add_risk_factor मेथड जोखिम कारक को risk_factors ऐरे में जोड़ने की अनुमति

देता है। यह दो पैरामीटर लेता है: description, जो जोखिम कारक का वर्णन करने वाली स्ट्रिंग है, और probability, जो उस जोखिम कारक से जुड़ी धोखाधड़ी की संभावना को दर्शाने वाला फ्लोट है। हम सरल टाइप चेकिंग के लिए case..in कंडीशनल का उपयोग करते हैं।

श्रृंखला के अंत में जांचा जाने वाला high_risk? मेथड एक प्रेडिकेट मेथड है जो fraud_probability (सभी जोखिम कारकों की संभावनाओं को जोड़कर गणना की गई) की तुलना RISK_THRESHOLD से करता है।

FraudDetection क्लास किसी लेन-देन के लिए धोखाधड़ी का पता लगाने का एक स्वच्छ और एनकैप्सुलेटेड तरीका प्रदान करता है। यह कई जोखिम कारकों को जोड़ने की अनुमति देता है, प्रत्येक के अपने विवरण और संभावना के साथ, और गणना की गई धोखाधड़ी की संभावना के आधार पर यह निर्धारित करने का तरीका प्रदान करता है कि क्या लेन-देन को उच्च जोखिम वाला माना जाता है। इस क्लास को आसानी से एक बड़ी धोखाधड़ी पता लगाने वाली प्रणाली में एकीकृत किया जा सकता है, जहां विभिन्न घटक धोखाधड़ी वाले लेन-देन के जोखिम का आकलन और कम करने के लिए सहयोग कर सकते हैं।

अंत में, चूंकि यह पुस्तक AI का उपयोग करके प्रोग्रामिंग के बारे में है, यहाँ मेरी Raix लाइब्रेरी के ChatCompletion मॉड्यूल का उपयोग करके CheckCustomerHistory क्लास का एक उदाहरण कार्यान्वयन दिया गया है:

```
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
```

इस उदाहरण में, CheckCustomerHistory एक INSTRUCTION कॉन्स्टेंट को परिभाषित करता है जो एआई मॉडल को सिस्टम निर्देश के माध्यम से ग्राहक के लेन-देन के इतिहास में संभावित धोखाधड़ी के संकेतों का विश्लेषण करने के लिए

विशिष्ट निर्देश प्रदान करता है।

self.check मेथड एक क्लास मेथड है जो fraud_detection ऑब्जेक्ट के साथ CheckCustomerHistory का एक नया इंस्टेंस शुरू करता है और ग्राहक के इतिहास का विश्लेषण करने के लिए call मेथड को कॉल करता है।

call मेथड के अंदर, ग्राहक के लेन-देन का इतिहास प्राप्त किया जाता है और एक ट्रांसक्रिप्ट में फॉर्मेट किया जाता है जो एआई मॉडल को भेजा जाता है। एआई मॉडल दिए गए निर्देशों के आधार पर लेन-देन के इतिहास का विश्लेषण करता है और अपने निष्कर्षों का सारांश लौटाता है।

निष्कर्षों को fraud_detection ऑब्जेक्ट में जोड़ा जाता है, और अपडेट किया गया fraud_detection ऑब्जेक्ट एक सफल Result के रूप में लौटाया जाता है।

ChatCompletion मॉड्यूल का लाभ उठाकर, CheckCustomerHistory क्लास एआई की शक्ति का उपयोग ग्राहक के लेन-देन के इतिहास का विश्लेषण करने और संभावित धोखाधड़ी के संकेतों की पहचान करने के लिए कर सकती है। यह अधिक परिष्कृत और अनुकूलन योग्य धोखाधड़ी का पता लगाने की तकनीकों को संभव बनाता है, क्योंकि एआई मॉडल नए पैटर्न और विसंगतियों को समय के साथ सीख और अनुकूलित कर सकता है।

अपडेट किया गया FraudDetectionWorker और CheckCustomerHistory क्लास दिखाते हैं कि कैसे एआई वर्कर्स को निर्बाध रूप से एकीकृत किया जा सकता है, जो धोखाधड़ी का पता लगाने की प्रक्रिया को बुद्धिमान विश्लेषण और निर्णय लेने की क्षमताओं के साथ बढ़ाता है।

ग्राहक भावना विश्लेषण

यहाँ एक और समान उदाहरण है कि आप ग्राहक भावना विश्लेषण वर्कर को कैसे लागू कर सकते हैं। इस बार बहुत कम व्याख्या के साथ, क्योंकि आप समझ रहे होंगे कि प्रोग्रामिंग की यह शैली कैसे काम करती है:

```ruby
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
```

इस उदाहरण में, CustomerSentimentAnalysisWorker के चरणों में फीडबैक का पूर्व-प्रसंस्करण (जैसे, शोर हटाना, टोकनाइजेशन), भावना विश्लेषण करना जिससे समग्र भावना (सकारात्मक, नकारात्मक, या तटस्थ) का निर्धारण हो, प्रमुख वाक्यांशों और विषयों की पहचान, रुझानों और पैटर्न की पहचान, और विश्लेषण के आधार पर कार्रवाई योग्य अंतर्दृष्टि उत्पन्न करना शामिल है।

स्वास्थ्य सेवा अनुप्रयोग

स्वास्थ्य सेवा क्षेत्र में, एआई वर्कर्स विभिन्न कार्यों में चिकित्सा पेशेवरों और शोधकर्ताओं की सहायता कर सकते हैं, जिससे रोगी के परिणामों में सुधार और चिकित्सा खोजों में तेजी आती है।

रोगी प्रवेश प्रक्रिया

एआई वर्कर्स विभिन्न कार्यों को स्वचालित करके और बुद्धिमान सहायता प्रदान करके रोगी प्रवेश प्रक्रिया को सुव्यवस्थित कर सकते हैं।

अपॉइंटमेंट शेड्यूलिंग: एआई वर्कर्स रोगी की प्राथमिकताओं, उपलब्धता और उनकी चिकित्सा आवश्यकताओं की तात्कालिकता को समझकर अपॉइंटमेंट शेड्यूलिंग को संभाल सकते हैं। वे संवादात्मक इंटरफेस के माध्यम से रोगियों के साथ बातचीत कर सकते हैं, उन्हें शेड्यूलिंग प्रक्रिया में मार्गदर्शन कर सकते हैं और रोगी की आवश्यकताओं और स्वास्थ्य सेवा प्रदाता की उपलब्धता के आधार पर सबसे उपयुक्त अपॉइंटमेंट स्लॉट खोज सकते हैं।

चिकित्सा इतिहास संग्रह: रोगी प्रवेश के दौरान, एआई वर्कर्स रोगी के चिकित्सा इतिहास को एकत्र करने और प्रलेखित करने में सहायता कर सकते हैं। वे रोगियों के साथ इंटरैक्टिव संवाद कर सकते हैं, उनकी पिछली चिकित्सा स्थितियों, दवाओं, एलर्जी और पारिवारिक इतिहास के बारे में प्रासंगिक प्रश्न पूछ सकते हैं। एआई वर्कर्स एकत्रित जानकारी की व्याख्या करने और उसे संरचित करने के लिए प्राकृतिक भाषा प्रसंस्करण तकनीकों का उपयोग कर सकते हैं, यह सुनिश्चित करते हुए कि यह रोगी के इलेक्ट्रॉनिक स्वास्थ्य रिकॉर्ड में सटीक रूप से दर्ज की गई है।

लक्षण मूल्यांकन और वर्गीकरण: एआई वर्कर्स रोगियों से उनके वर्तमान लक्षणों, अवधि, गंभीरता और किसी भी संबंधित कारकों के बारे में पूछकर प्रारंभिक लक्षण मूल्यांकन कर सकते हैं। चिकित्सा ज्ञान आधार और मशीन लर्निंग मॉडल का लाभ उठाकर, ये वर्कर्स प्रदान की गई जानकारी का विश्लेषण कर सकते हैं और प्रारंभिक विभेदक निदान उत्पन्न कर सकते हैं या उपयुक्त अगले कदमों की सिफारिश कर सकते हैं, जैसे स्वास्थ्य सेवा प्रदाता के साथ परामर्श शेड्यूल करना या स्व-देखभाल उपायों का सुझाव देना।

बीमा सत्यापन: एआई वर्कर्स रोगी प्रवेश के दौरान बीमा सत्यापन में सहायता कर सकते हैं। वे रोगी के बीमा विवरण एकत्र कर सकते हैं, एपीआई या वेब सेवाओं के माध्यम से बीमा प्रदाताओं के साथ संवाद कर सकते हैं, और कवरेज पात्रता और लाभों की जांच कर सकते हैं। यह स्वचालन बीमा सत्यापन प्रक्रिया को सुव्यवस्थित

करने में मदद करता है, प्रशासनिक बोझ को कम करता है और सटीक जानकारी कैप्चर करना सुनिश्चित करता है।

रोगी शिक्षा और निर्देश: एआई वर्कर्स रोगियों को उनकी विशिष्ट चिकित्सा स्थितियों या आगामी प्रक्रियाओं के आधार पर प्रासंगिक शैक्षिक सामग्री और निर्देश प्रदान कर सकते हैं। वे व्यक्तिगत सामग्री प्रदान कर सकते हैं, सामान्य प्रश्नों का उत्तर दे सकते हैं, और अपॉइंटमेंट से पहले की तैयारियों, दवा निर्देशों, या उपचार के बाद की देखभाल पर मार्गदर्शन प्रदान कर सकते हैं। यह रोगियों को उनकी स्वास्थ्य सेवा यात्रा के दौरान सूचित और संलग्न रखने में मदद करता है।

कृत्रिम बुद्धिमत्ता कर्मचारियों का उपयोग करके रोगी प्रवेश प्रक्रिया में, स्वास्थ्य संगठन दक्षता बढ़ा सकते हैं, प्रतीक्षा समय कम कर सकते हैं, और समग्र रोगी अनुभव में सुधार कर सकते हैं। ये कर्मचारी नियमित कार्यों को संभाल सकते हैं, सटीक जानकारी एकत्र कर सकते हैं, और व्यक्तिगत सहायता प्रदान कर सकते हैं, जिससे स्वास्थ्य पेशेवरों को रोगियों को उच्च-गुणवत्ता वाली देखभाल प्रदान करने पर ध्यान केंद्रित करने की अनुमति मिलती है।

रोगी जोखिम मूल्यांकन

कृत्रिम बुद्धिमत्ता कर्मचारी विभिन्न डेटा स्रोतों का विश्लेषण करके और उन्नत विश्लेषण तकनीकों को लागू करके रोगी जोखिम का आकलन करने में महत्वपूर्ण भूमिका निभा सकते हैं।

डेटा एकीकरण: कृत्रिम बुद्धिमत्ता कर्मचारी कई स्रोतों से रोगी डेटा एकत्र कर सकते हैं और उसे समझ सकते हैं, जैसे इलेक्ट्रॉनिक स्वास्थ्य रिकॉर्ड (EHRs), चिकित्सा इमेजिंग, प्रयोगशाला परिणाम, पहनने योग्य उपकरण, और स्वास्थ्य के सामाजिक निर्धारक। इस जानकारी को एक व्यापक रोगी प्रोफ़ाइल में समेकित करके, कृत्रिम बुद्धिमत्ता कर्मचारी रोगी के स्वास्थ्य की स्थिति और जोखिम कारकों का समग्र दृष्टिकोण प्रदान कर सकते हैं।

जोखिम स्तरीकरण: कृत्रिम बुद्धिमत्ता कर्मचारी रोगियों की व्यक्तिगत विशेषताओं और स्वास्थ्य डेटा के आधार पर उन्हें विभिन्न जोखिम श्रेणियों में वर्गीकृत करने के

लिए पूर्वानुमानित मॉडल का उपयोग कर सकते हैं। यह जोखिम स्तरीकरण स्वास्थ्य प्रदाताओं को उन रोगियों को प्राथमिकता देने में सक्षम बनाता है जिन्हें अधिक तत्काल ध्यान या हस्तक्षेप की आवश्यकता होती है। उदाहरण के लिए, किसी विशेष स्थिति के लिए उच्च जोखिम वाले पहचाने गए रोगियों को करीबी निगरानी, निवारक उपायों, या शीघ्र हस्तक्षेप के लिए चिह्नित किया जा सकता है।

व्यक्तिगत जोखिम प्रोफ़ाइल: कृत्रिम बुद्धिमत्ता कर्मचारी प्रत्येक रोगी के लिए व्यक्तिगत जोखिम प्रोफ़ाइल तैयार कर सकते हैं, जो उनके जोखिम स्कोर में योगदान करने वाले विशिष्ट कारकों को उजागर करते हैं। इन प्रोफ़ाइल में रोगी की जीवनशैली, आनुवंशिक प्रवृत्तियों, पर्यावरणीय कारकों, और स्वास्थ्य के सामाजिक निर्धारकों के बारे में जानकारी शामिल हो सकती है। जोखिम कारकों का विस्तृत विश्लेषण प्रदान करके, कृत्रिम बुद्धिमत्ता कर्मचारी स्वास्थ्य प्रदाताओं को व्यक्तिगत रोगी की आवश्यकताओं के अनुरूप रोकथाम रणनीतियों और उपचार योजनाओं को तैयार करने में मदद कर सकते हैं।

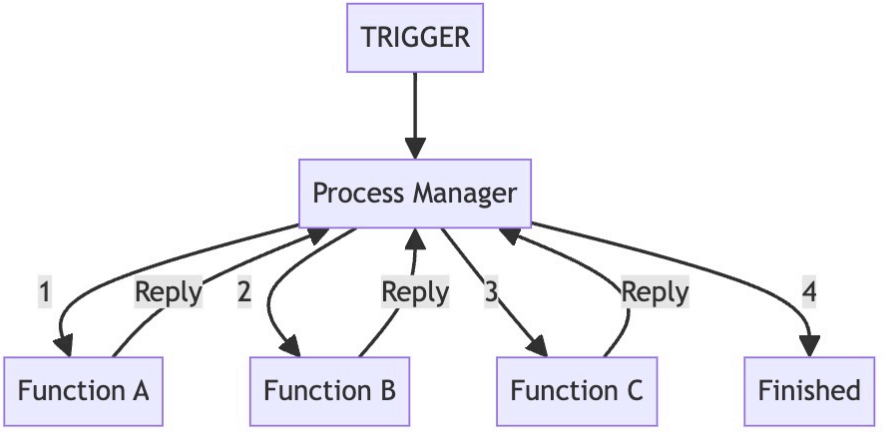
निरंतर जोखिम निगरानी: कृत्रिम बुद्धिमत्ता कर्मचारी रोगी डेटा की लगातार निगरानी कर सकते हैं और वास्तविक समय में जोखिम मूल्यांकन को अपडेट कर सकते हैं। जैसे-जैसे नई जानकारी उपलब्ध होती है, जैसे महत्वपूर्ण लक्षणों में परिवर्तन, प्रयोगशाला परिणाम, या दवा अनुपालन, कृत्रिम बुद्धिमत्ता कर्मचारी जोखिम स्कोर की पुनर्गणना कर सकते हैं और किसी भी महत्वपूर्ण परिवर्तन के बारे में स्वास्थ्य प्रदाताओं को सचेत कर सकते हैं। यह सक्रिय निगरानी समय पर हस्तक्षेप और रोगी देखभाल योजनाओं में समायोजन की अनुमति देती है।

नैदानिक निर्णय समर्थन: कृत्रिम बुद्धिमत्ता कर्मचारी जोखिम मूल्यांकन परिणामों को नैदानिक निर्णय समर्थन प्रणालियों में एकीकृत कर सकते हैं, जो स्वास्थ्य प्रदाताओं को साक्ष्य-आधारित सिफारिशें और चेतावनियां प्रदान करते हैं। उदाहरण के लिए, यदि किसी विशेष स्थिति के लिए रोगी का जोखिम स्कोर एक निश्चित सीमा से अधिक हो जाता है, तो कृत्रिम बुद्धिमत्ता कर्मचारी स्वास्थ्य प्रदाता को नैदानिक दिशानिर्देशों और सर्वोत्तम प्रथाओं के आधार पर विशिष्ट नैदानिक परीक्षण, निवारक उपाय, या उपचार विकल्पों पर विचार करने के लिए प्रेरित कर सकते हैं।

ये कर्मचारी रोगी डेटा की विशाल मात्रा को संसाधित कर सकते हैं, परिष्कृत विश्लेषण

लागू कर सकते हैं, और नैदानिक निर्णय लेने का समर्थन करने के लिए कार्रवाई योग्य अंतर्दृष्टि उत्पन्न कर सकते हैं। यह अंततः बेहतर रोगी परिणामों, कम स्वास्थ्य देखभाल लागतों, और बेहतर जनसंख्या स्वास्थ्य प्रबंधन की ओर ले जाता है।

AI वर्कर एक प्रोसेस मैनेजर के रूप में



AI-संचालित एप्लिकेशन के संदर्भ में, एक वर्कर को प्रोसेस मैनेजर के रूप में कार्य करने के लिए डिज़ाइन किया जा सकता है, जैसा कि Gregor Hohpe की पुस्तक “Enterprise Integration Patterns” में वर्णित है। एक प्रोसेस मैनेजर एक केंद्रीय घटक है जो प्रक्रिया की स्थिति को बनाए रखता है और मध्यवर्ती परिणामों के आधार पर अगले प्रोसेसिंग चरणों का निर्धारण करता है।

जब एक AI वर्कर प्रोसेस मैनेजर के रूप में कार्य करता है, तो यह एक इनकमिंग मैसेज प्राप्त करता है जो प्रक्रिया को आरंभ करता है, जिसे ट्रिगर मैसेज के रूप में जाना जाता है। AI वर्कर तब प्रक्रिया निष्पादन की स्थिति (एक वार्तालाप प्रतिलेख के रूप में) को बनाए रखता है और टूल फंक्शन्स के रूप में लागू किए गए प्रोसेसिंग चरणों की श्रृंखला के माध्यम से संदेश को संभालता है, जो क्रमिक या समानांतर हो

सकते हैं, और उसके विवेक पर कॉल किए जा सकते हैं।

यदि आप GPT-4 जैसी AI मॉडल की श्रेणी का उपयोग कर रहे हैं जो समानांतर में फ़ंक्शन निष्पादित करना जानता है, तो आपका वर्कर एक साथ कई चरणों को निष्पादित कर सकता है। मान्य है, मैंने खुद यह करने की कोशिश नहीं की है और मेरी अंतर्ज्ञान कहती है कि आपका अनुभव भिन्न हो सकता है।

प्रत्येक व्यक्तिगत प्रोसेसिंग चरण के बाद, नियंत्रण AI वर्कर को वापस लौटा दिया जाता है, जो उसे वर्तमान स्थिति और प्राप्त परिणामों के आधार पर अगले प्रोसेसिंग चरण(णों) का निर्धारण करने की अनुमति देता है।

अपने ट्रिगर मैसेज को स्टोर करें

मेरे अनुभव में, अपने ट्रिगर मैसेज को एक डेटाबेस-बैक्ड ऑब्जेक्ट के रूप में लागू करना बुद्धिमानी है। इस तरह प्रत्येक प्रक्रिया इंस्टेंस को एक विशिष्ट प्राइमरी की द्वारा पहचाना जाता है और आपको AI की वार्तालाप प्रतिलेख सहित निष्पादन से जुड़ी स्थिति को स्टोर करने के लिए एक जगह मिलती है।

उदाहरण के लिए, यहाँ Olympia के AccountChange मॉडल क्लास का एक सरलीकृत संस्करण है, जो उपयोगकर्ता के खाते में परिवर्तन करने के अनुरोध का प्रतिनिधित्व करता है।

```ruby
# == 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 क्लास एक ट्रिगर संदेश के रूप में कार्य करती है जो खाता परिवर्तन अनुरोध को संभालने की प्रक्रिया शुरू करती है। ध्यान दें कि इसे Olympia के Wisper-

आधारित पब/सब सबसिस्टम में क्रिएट ट्रांजैक्शन के पूरा होने के बाद प्रसारित किया जाता है।

डेटाबेस में ट्रिगर संदेश को इस तरह से स्टोर करना प्रत्येक खाता परिवर्तन अनुरोध का एक स्थायी रिकॉर्ड प्रदान करता है। AccountChange क्लास के प्रत्येक इंस्टेंस को एक विशिष्ट प्राइमरी की असाइन की जाती है, जो व्यक्तिगत अनुरोधों की आसान पहचान और ट्रैकिंग की सुविधा प्रदान करती है। यह विशेष रूप से ऑडिट लॉगिंग के उद्देश्यों के लिए उपयोगी है, क्योंकि यह सिस्टम को सभी खाता परिवर्तनों का ऐतिहासिक रिकॉर्ड बनाए रखने में सक्षम बनाता है, जिसमें यह भी शामिल है कि उन्हें कब अनुरोधित किया गया, किन परिवर्तनों का अनुरोध किया गया, और प्रत्येक अनुरोध की वर्तमान स्थिति क्या है।

दिए गए उदाहरण में, AccountChange क्लास में अनुरोधित परिवर्तन के विवरण को कैप्चर करने के लिए description जैसे फ़ील्ड्स, अनुरोध की वर्तमान स्थिति (जैसे, requested, complete, requires_human_review) को दर्शाने के लिए state, और अनुरोध से संबंधित एआई की बातचीत की प्रतिलिपि को स्टोर करने के लिए transcript शामिल हैं। description फ़ील्ड वास्तविक प्रॉम्प्ट है जिसका उपयोग एआई के साथ पहली चैट कम्प्लीशन शुरू करने के लिए किया जाता है। इस डेटा को स्टोर करना मूल्यवान संदर्भ प्रदान करता है और खाता परिवर्तन प्रक्रिया की बेहतर ट्रैकिंग और विश्लेषण की अनुमति देता है।

डेटाबेस में ट्रिगर संदेशों को स्टोर करना मजबूत त्रुटि प्रबंधन और रिकवरी को सक्षम बनाता है। यदि किसी खाता परिवर्तन अनुरोध के प्रोसेसिंग के दौरान कोई त्रुटि होती है, तो सिस्टम अनुरोध को विफल के रूप में चिह्नित कर देता है और इसे एक ऐसी स्थिति में स्थानांतरित कर देता है जिसमें मानवीय हस्तक्षेप की आवश्यकता होती है। यह सुनिश्चित करता है कि कोई भी अनुरोध खो न जाए या भूला न जाए, और किसी भी समस्या को उचित तरीके से संबोधित और हल किया जा सके।

एक Process Manager के रूप में, एआई वर्कर नियंत्रण का एक केंद्रीय बिंदु प्रदान करता है और शक्तिशाली प्रक्रिया रिपोर्टिंग और डीबगिंग क्षमताओं को सक्षम बनाता

है। हालांकि, यह ध्यान रखना महत्वपूर्ण है कि आपके एप्लिकेशन में हर वर्कफ़्लो परिदृश्य के लिए Process Manager के रूप में एआई वर्कर का उपयोग करना अत्यधिक हो सकता है।

अपने एप्लिकेशन आर्किटेक्चर में एआई वर्कर्स को एकीकृत करना

जब आप अपने एप्लिकेशन आर्किटेक्चर में एआई वर्कर्स को शामिल करते हैं, तो एआई वर्कर्स और अन्य एप्लिकेशन घटकों के बीच सुचारू एकीकरण और प्रभावी संचार सुनिश्चित करने के लिए कई तकनीकी विचारों को संबोधित करने की आवश्यकता होती है। यह खंड उन इंटरफेस को डिज़ाइन करने, डेटा प्रवाह को संभालने और एआई वर्कर्स के जीवन चक्र को प्रबंधित करने के प्रमुख पहलुओं पर विचार करता है।

स्पष्ट इंटरफेस और संचार प्रोटोकॉल डिज़ाइन करना

एआई वर्कर्स और अन्य एप्लिकेशन घटकों के बीच निर्बाध एकीकरण की सुविधा के लिए, स्पष्ट इंटरफेस और संचार प्रोटोकॉल को परिभाषित करना महत्वपूर्ण है। निम्नलिखित दृष्टिकोणों पर विचार करें:

एपीआई-आधारित एकीकरण: एआई वर्कर्स की कार्यक्षमता को सुपरिभाषित एपीआई के माध्यम से प्रदर्शित करें, जैसे RESTful एंडपॉइंट्स या GraphQL स्कीमा। यह अन्य घटकों को मानक HTTP अनुरोधों और प्रतिक्रियाओं का उपयोग करके एआई वर्कर्स के साथ संवाद करने की अनुमति देता है। एपीआई-आधारित एकीकरण एआई वर्कर्स और उपभोग करने वाले घटकों के बीच एक स्पष्ट अनुबंध प्रदान करता है, जिससे एकीकरण बिंदुओं को विकसित करना, परीक्षण करना और बनाए रखना आसान हो जाता है।

संदेश-आधारित संचार: संदेश-आधारित संचार पैटर्न को लागू करें, जैसे मैसेज क्यू या प्रकाशन-सदस्यता प्रणालियाँ, जो एआई वर्कर्स और अन्य घटकों के बीच अतुल्यकालिक संवाद को सक्षम करता है। यह दृष्टिकोण एआई वर्कर्स को एप्लिकेशन

के बाकी हिस्सों से अलग करता है, जिससे बेहतर स्केलेबिलिटी, त्रुटि सहनशीलता और लूज कपलिंग की सुविधा मिलती है। संदेश-आधारित संचार विशेष रूप से तब उपयोगी होता है जब एआई वर्कर्स द्वारा किया गया प्रोसेसिंग समय लेने वाला या संसाधन-गहन होता है, क्योंकि यह एप्लिकेशन के अन्य भागों को एआई वर्कर्स के अपना कार्य पूरा करने की प्रतीक्षा किए बिना निष्पादन जारी रखने की अनुमति देता है।

इवेंट-संचालित आर्किटेक्चर: अपनी प्रणाली को इवेंट्स और ट्रिगर्स के आसपास डिज़ाइन करें जो विशिष्ट स्थितियों में एआई वर्कर्स को सक्रिय करते हैं। एआई वर्कर्स प्रासंगिक इवेंट्स की सदस्यता ले सकते हैं और तदनुसार प्रतिक्रिया कर सकते हैं, जब इवेंट्स होते हैं तब अपने निर्धारित कार्यों को करते हैं। इवेंट-संचालित आर्किटेक्चर रीयल-टाइम प्रोसेसिंग को सक्षम करता है और एआई वर्कर्स को मांग पर आह्वान करने की अनुमति देता है, जिससे अनावश्यक संसाधन खपत कम होती है। यह दृष्टिकोण उन परिदृश्यों के लिए उपयुक्त है जहां एआई वर्कर्स को विशिष्ट कार्यों या एप्लिकेशन स्थिति में परिवर्तनों के प्रति प्रतिक्रिया करने की आवश्यकता होती है।

डेटा प्रवाह और समकालीकरण का प्रबंधन

जब आप अपने एप्लिकेशन में एआई वर्कर्स को एकीकृत करते हैं, तो एआई वर्कर्स और अन्य घटकों के बीच सुचारू डेटा प्रवाह सुनिश्चित करना और डेटा संगति बनाए रखना महत्वपूर्ण होता है। निम्नलिखित पहलुओं पर विचार करें:

डेटा तैयारी: एआई वर्कर्स में डेटा फीड करने से पहले, आपको विभिन्न डेटा तैयारी कार्य करने की आवश्यकता हो सकती है, जैसे इनपुट डेटा की सफाई, फॉर्मेटिंग और/या रूपांतरण। आप न केवल यह सुनिश्चित करना चाहते हैं कि एआई वर्कर्स प्रभावी ढंग से प्रोसेस कर सकें, बल्कि यह भी सुनिश्चित करना चाहते हैं कि आप उस जानकारी पर ध्यान देने में टोकन बर्बाद नहीं कर रहे हैं जिसे वर्कर सर्वोत्तम स्थिति में बेकार और सबसे खराब स्थिति में भटकाने वाला मान सकता है। डेटा तैयारी में शोर को हटाना, लापता मूल्यों को संभालना, या डेटा प्रकारों को परिवर्तित करना जैसे कार्य शामिल हो सकते हैं।

डेटा स्थायित्व: एआई वर्कर्स में आने-जाने वाले डेटा को आप कैसे स्टोर और बनाए रखेंगे? डेटा मात्रा, क्वेरी पैटर्न और स्केलेबिलिटी जैसे कारकों पर विचार करें। क्या आपको ऑडिट या डीबगिंग उद्देश्यों के लिए एआई के "विचार प्रक्रिया" के प्रतिबिंब के रूप में ट्रांसक्रिप्ट को बनाए रखने की आवश्यकता है, या केवल परिणामों का रिकॉर्ड रखना पर्याप्त है?

डेटा पुनर्प्राप्ति: कार्यकर्ताओं को आवश्यक डेटा प्राप्त करने में डेटाबेस की क्वेरी, फ़ाइलों से पढ़ना, या बाहरी एपीआई तक पहुंच शामिल हो सकती है। विलंबता पर विचार करना और एआई वर्कर्स को सबसे अद्यतित डेटा तक कैसे पहुंच मिलेगी, यह सुनिश्चित करें। क्या उन्हें आपके डेटाबेस तक पूर्ण पहुंच की आवश्यकता है या आपको उनकी पहुंच को उनके कार्य के अनुसार सीमित करना चाहिए? स्केलिंग के बारे में क्या? प्रदर्शन में सुधार और अंतर्निहित डेटा स्रोतों पर भार को कम करने के लिए कैशिंग तंत्र पर विचार करें।

डेटा समकालीकरण: जब कई घटक, एआई वर्कर्स सहित, साझा डेटा तक पहुंचते हैं और उसमें संशोधन करते हैं, तो डेटा संगति बनाए रखने के लिए उचित समकालीकरण तंत्र को लागू करना महत्वपूर्ण है। डेटाबेस लॉकिंग रणनीतियां, जैसे आशावादी या निराशावादी लॉकिंग, विरोधों को रोकने और डेटा अखंडता सुनिश्चित करने में मदद कर सकती हैं। संबंधित डेटा संचालन को समूहित करने और एटॉमिसिटी, कंसिस्टेंसी, आइसोलेशन और ड्यूरेबिलिटी (एसिड) गुणों को बनाए रखने के लिए लेन-देन प्रबंधन तकनीकों को लागू करें।

त्रुटि प्रबंधन और पुनर्प्राप्ति: डेटा प्रवाह प्रक्रिया के दौरान उत्पन्न हो सकने वाली डेटा-संबंधित समस्याओं से निपटने के लिए मजबूत त्रुटि प्रबंधन और पुनर्प्राप्ति तंत्र लागू करें। त्रुटियों को सुचारू रूप से संभालें और डीबगिंग में सहायता के लिए सार्थक त्रुटि संदेश प्रदान करें। अस्थायी विफलताओं या नेटवर्क व्यवधानों को संभालने के लिए पुनः प्रयास तंत्र और फॉलबैक रणनीतियां लागू करें। डेटा भ्रष्टता या हानि की स्थिति में डेटा पुनर्प्राप्ति और पुनर्स्थापना के लिए स्पष्ट प्रक्रियाएं परिभाषित करें।

डेटा प्रवाह और समकालीकरण तंत्र को सावधानीपूर्वक डिजाइन और कार्यान्वित करके, आप सुनिश्चित कर सकते हैं कि आपके एआई वर्कर्स को सटीक, संगत और अद्यतन डेटा तक पहुंच प्राप्त हो। यह उन्हें अपने कार्यों को प्रभावी ढंग से करने और विश्वसनीय

परिणाम उत्पन्न करने में सक्षम बनाता है।

एआई वर्कर्स के जीवनचक्र का प्रबंधन

एआई वर्कर्स को आरंभ करने और कॉन्फ़िगर करने के लिए एक मानकीकृत प्रक्रिया विकसित करें। मैं ऐसे फ्रेमवर्क को पसंद करता हूं जो मॉडल नाम, सिस्टम निर्देश और फ़ंक्शन परिभाषाओं जैसी सेटिंग्स को परिभाषित करने का तरीका मानकीकृत करते हैं। सुनिश्चित करें कि आरंभीकरण प्रक्रिया स्वचालित और पुनरुत्पादन योग्य है ताकि परिनियोजन और स्केलिंग की सुविधा हो।

एआई वर्कर्स के स्वास्थ्य और प्रदर्शन की निगरानी के लिए व्यापक निगरानी और लॉगिंग तंत्र लागू करें। संसाधन उपयोग, प्रोसेसिंग समय, त्रुटि दर, और थ्रूपुट जैसे मेट्रिक्स एकत्र करें। कई एआई वर्कर्स से लॉग को एकत्रित और विश्लेषण करने के लिए ईएलके स्टैक (एलास्टिकसर्च, लॉगस्टैश, किबाना) जैसी केंद्रीकृत लॉगिंग प्रणालियों का उपयोग करें।

AI वर्कर आर्किटेक्चर में दोष सहिष्णुता और लचीलापन निर्मित करें। विफलताओं या अपवादों को सुचारू रूप से संभालने के लिए त्रुटि प्रबंधन और पुनर्प्राप्ति तंत्र लागू करें। लार्ज लैंग्वेज मॉडल्स अभी भी अत्याधुनिक तकनीक हैं; प्रदाता अक्सर अप्रत्याशित समय पर डाउन हो जाते हैं। कैस्केडिंग विफलताओं को रोकने के लिए पुनः प्रयास तंत्र और सर्किट ब्रेकर का उपयोग करें।

AI वर्कर्स की संयोजनीयता और ऑर्केस्ट्रेशन

AI वर्कर आर्किटेक्चर का एक प्रमुख लाभ इसकी संयोजनीयता है, जो जटिल समस्याओं को हल करने के लिए कई AI वर्कर्स को जोड़ने और उनका संचालन करने की अनुमति देता है। एक बड़े कार्य को छोटे, अधिक प्रबंधनीय उप-कार्यों में विभाजित करके, जिनमें से प्रत्येक को एक विशेष AI वर्कर द्वारा संभाला जाता है, आप शक्तिशाली और लचीले सिस्टम बना सकते हैं। इस खंड में, हम “कई” AI वर्कर्स को संयोजित और संचालित करने के विभिन्न दृष्टिकोणों का पता लगाएंगे।

बहु-चरणीय कार्यप्रवाह के लिए AI वर्कर्स की श्रृंखला

कई परिदृश्यों में, एक जटिल कार्य को क्रमिक चरणों की एक श्रृंखला में विभाजित किया जा सकता है, जहां एक AI वर्कर का आउटपुट अगले के लिए इनपुट बन जाता है। AI वर्कर्स की यह श्रृंखला एक बहु-चरणीय कार्यप्रवाह या पाइपलाइन बनाती है। श्रृंखला में प्रत्येक AI वर्कर एक विशिष्ट उप-कार्य पर केंद्रित होता है, और अंतिम आउटपुट सभी वर्कर्स के संयुक्त प्रयासों का परिणाम होता है।

आइए Ruby on Rails एप्लिकेशन के संदर्भ में उपयोगकर्ता-निर्मित सामग्री को संसाधित करने का एक उदाहरण लें। कार्यप्रवाह में निम्नलिखित चरण शामिल हैं, जो स्वीकार करें कि शायद वास्तविक उपयोग के मामलों में इस तरह से विघटित करने के लिए बहुत सरल हैं, लेकिन वे उदाहरण को समझने में आसान बनाते हैं:

1. टेक्स्ट क्लीनअप: एक AI वर्कर जो HTML टैग हटाने, टेक्स्ट को लोअरकेस में बदलने और यूनिकोड सामान्यीकरण को संभालने के लिए जिम्मेदार है।

2. भाषा पहचान: एक AI वर्कर जो साफ किए गए टेक्स्ट की भाषा की पहचान करता है।

3. भावना विश्लेषण: एक AI वर्कर जो पहचानी गई भाषा के आधार पर टेक्स्ट की भावना (सकारात्मक, नकारात्मक, या तटस्थ) निर्धारित करता है।

4. सामग्री वर्गीकरण: एक AI वर्कर जो प्राकृतिक भाषा प्रसंस्करण तकनीकों का उपयोग करके टेक्स्ट को पूर्व-परिभाषित श्रेणियों में वर्गीकृत करता है।

यहाँ Ruby का उपयोग करके इन AI वर्कर्स को एक साथ श्रृंखलाबद्ध करने का एक बहुत सरलीकृत उदाहरण दिया गया है:

```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
```

इस उदाहरण में, ContentProcessor क्लास कच्चे टेक्स्ट के साथ आरंभ होता है और process मेथड में AI वर्कर्स को एक श्रृंखला में जोड़ता है। प्रत्येक AI वर्कर अपना विशिष्ट कार्य करता है और परिणाम को श्रृंखला में अगले वर्कर को पास करता है। अंतिम आउटपुट एक हैश होता है जिसमें साफ किया गया टेक्स्ट, पहचानी गई भाषा, भावना, और सामग्री श्रेणी शामिल होती है।

स्वतंत्र AI वर्कर्स के लिए समानांतर प्रोसेसिंग

पिछले उदाहरण में, AI वर्कर्स को अनुक्रमिक रूप से श्रृंखलाबद्ध किया गया है, जहां प्रत्येक वर्कर टेक्स्ट को प्रोसेस करता है और परिणाम को अगले वर्कर को पास करता है। हालांकि, यदि आपके पास कई AI वर्कर्स हैं जो एक ही इनपुट पर स्वतंत्र रूप से काम कर सकते हैं, तो आप उन्हें समानांतर रूप से प्रोसेस करके कार्यप्रवाह को अनुकूलित कर सकते हैं।

दिए गए परिदृश्य में, एक बार जब TextCleanupWorker द्वारा टेक्स्ट की सफाई हो जाती है, तो LanguageDetectionWorker, SentimentAnalysisWorker, और CategorizationWorker सभी साफ किए गए टेक्स्ट को स्वतंत्र रूप से प्रोसेस कर सकते हैं। इन वर्कर्स को समानांतर रूप से चलाकर, आप समग्र प्रोसेसिंग समय को कम कर सकते हैं और अपने कार्यप्रवाह की दक्षता में सुधार कर सकते हैं।

Ruby में समानांतर प्रोसेसिंग प्राप्त करने के लिए, आप थ्रेड्स या अतुल्यकालिक प्रोग्रामिंग जैसी समवर्तिता तकनीकों का लाभ उठा सकते हैं। यहाँ एक उदाहरण दिया गया है कि आप थ्रेड्स का उपयोग करके अंतिम तीन वर्कर्स को समानांतर रूप से प्रोसेस करने के लिए ContentProcessor क्लास को कैसे संशोधित कर सकते हैं:

```
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
```

इस अनुकूलित संस्करण में, हम प्रत्येक स्वतंत्र AI वर्कर के लिए Concurrent::Future ऑब्जेक्ट बनाने के लिए concurrent-ruby लाइब्रेरी का उपयोग करते हैं। एक Future एक ऐसी कम्प्यूटेशन को दर्शाता है जो एक अलग थ्रेड में असिंक्रोनस रूप से निष्पादित की जाएगी।

टेक्स्ट क्लीनअप स्टेप के बाद, हम तीन Future ऑब्जेक्ट बनाते हैं: language_future, sentiment_future, और category_future। प्रत्येक Future अपने संबंधित AI वर्कर (LanguageDetectionWorker, SentimentAnalysisWorker, और CategorizationWorker) को एक अलग थ्रेड में निष्पादित करता है, जिसमें cleaned_text को इनपुट के रूप में पास किया जाता है।

प्रत्येक Future पर value मेथड को कॉल करके, हम कम्प्यूटेशन के पूरा होने की प्रतीक्षा करते हैं और परिणाम प्राप्त करते हैं। value मेथड तब तक ब्लॉक करता है जब तक परिणाम उपलब्ध नहीं हो जाता, यह सुनिश्चित करते हुए कि आगे बढ़ने से पहले सभी समानांतर वर्कर्स ने प्रोसेसिंग पूरी कर ली है।

अंत में, हम मूल उदाहरण की तरह ही क्लीन किए गए टेक्स्ट और समानांतर वर्कर्स के परिणामों के साथ आउटपुट हैश का निर्माण करते हैं।

स्वतंत्र AI वर्कर्स को समानांतर रूप से प्रोसेस करके, आप क्रमिक रूप से चलाने की तुलना में कुल प्रोसेसिंग समय को संभावित रूप से कम कर सकते हैं। यह अनुकूलन विशेष रूप से समय लेने वाले कार्यों से निपटने या बड़ी मात्रा में डेटा को प्रोसेस करते समय लाभदायक होता है।

हालांकि, यह ध्यान रखना महत्वपूर्ण है कि वास्तविक प्रदर्शन लाभ विभिन्न कारकों पर निर्भर करता है, जैसे प्रत्येक वर्कर की जटिलता, उपलब्ध सिस्टम संसाधन, और थ्रेड प्रबंधन का ओवरहेड। अपने विशिष्ट उपयोग के मामले के लिए समानांतर प्रोसेसिंग का इष्टतम स्तर निर्धारित करने के लिए अपने कोड को बेंचमार्क और प्रोफाइल करना हमेशा एक अच्छी प्रथा है।

इसके अतिरिक्त, समानांतर प्रोसेसिंग को लागू करते समय, वर्कर्स के बीच किसी भी साझा संसाधन या निर्भरता के प्रति सावधान रहें। सुनिश्चित करें कि वर्कर्स बिना किसी विरोध या रेस कंडीशन्स के स्वतंत्र रूप से काम कर सकते हैं। यदि निर्भरताएं या साझा संसाधन हैं, तो आपको डेटा अखंडता बनाए रखने और डेडलॉक या असंगत परिणामों जैसी समस्याओं से बचने के लिए उपयुक्त सिंक्रोनाइज़ेशन तंत्र लागू करने की आवश्यकता हो सकती है।

Ruby का ग्लोबल इंटरप्रेटर लॉक और असिंक्रोनस प्रोसेसिंग

Ruby में असिंक्रोनस थ्रेड-आधारित प्रोसेसिंग पर विचार करते समय Ruby के ग्लोबल इंटरप्रेटर लॉक (GIL) के प्रभावों को समझना महत्वपूर्ण है।

GIL Ruby के इंटरप्रेटर में एक ऐसा तंत्र है जो सुनिश्चित करता है कि मल्टी-कोर प्रोसेसर पर भी एक समय में केवल एक थ्रेड ही Ruby कोड निष्पादित कर सकता है। इसका मतलब है कि हालांकि एक Ruby प्रोसेस के भीतर कई थ्रेड बनाए और प्रबंधित किए जा सकते हैं, किसी भी समय केवल एक थ्रेड ही सक्रिय रूप से Ruby कोड निष्पादित कर सकता है।

GIL को Ruby इंटरप्रेटर के कार्यान्वयन को सरल बनाने और Ruby के आंतरिक डेटा स्ट्रक्चर्स के लिए थ्रेड सुरक्षा प्रदान करने के लिए डिज़ाइन किया गया है। हालाँकि, यह Ruby कोड के वास्तविक समानांतर निष्पादन की संभावना को भी सीमित करता है।

जब आप Ruby में थ्रेड्स का उपयोग करते हैं, जैसे concurrent-ruby लाइब्रेरी या अंतर्निहित Thread क्लास के साथ, थ्रेड्स GIL की बाधाओं के अधीन होते हैं। GIL प्रत्येक थ्रेड को दूसरे थ्रेड में स्विच करने से पहले एक छोटे समय के लिए Ruby कोड निष्पादित करने की अनुमति देता है, जो समवर्ती निष्पादन का आभास उत्पन्न करता है।

हालाँकि, GIL के कारण, Ruby कोड का वास्तविक निष्पादन अनुक्रमिक ही रहता है। जब एक थ्रेड Ruby कोड निष्पादित कर रहा होता है, अन्य थ्रेड्स अनिवार्य रूप से रुके हुए होते हैं, GIL प्राप्त करने और निष्पादित होने की अपनी बारी का इंतजार करते हुए।

इसका मतलब है कि Ruby में थ्रेड-आधारित अतुल्यकालिक प्रसंस्करण I/O-बाउंड कार्यों के लिए सबसे प्रभावी है, जैसे बाहरी API प्रतिक्रियाओं की प्रतीक्षा करना (जैसे तृतीय-पक्ष द्वारा होस्ट किए गए बड़े भाषा मॉडल) या फाइल I/O ऑपरेशन्स करना। जब एक थ्रेड I/O ऑपरेशन का सामना करता है, तो वह GIL को रिलीज

कर सकता है, जिससे I/O पूरा होने की प्रतीक्षा के दौरान अन्य थ्रेड्स निष्पादित हो सकते हैं।

दूसरी ओर, CPU-बाउंड कार्यों के लिए, जैसे गहन गणनाएँ या लंबे समय तक चलने वाली AI वर्कर प्रोसेसिंग, GIL थ्रेड-आधारित समानांतरता से मिलने वाले संभावित प्रदर्शन लाभों को सीमित कर सकता है। चूंकि एक समय में केवल एक थ्रेड Ruby कोड निष्पादित कर सकता है, समग्र निष्पादन समय अनुक्रमिक प्रसंस्करण की तुलना में महत्वपूर्ण रूप से कम नहीं हो सकता।

Ruby में CPU-बाउंड कार्यों के लिए वास्तविक समानांतर निष्पादन प्राप्त करने के लिए, आपको वैकल्पिक दृष्टिकोणों की खोज करनी पड़ सकती है, जैसे:

- कई Ruby प्रोसेस के साथ प्रोसेस-आधारित समानांतरता का उपयोग, जहाँ प्रत्येक प्रोसेस अलग CPU कोर पर चल रहा हो।
- बाहरी लाइब्रेरीज़ या फ्रेमवर्क्स का लाभ उठाना जो नेटिव एक्सटेंशन या GIL रहित भाषाओं के लिए इंटरफेस प्रदान करते हैं, जैसे C या Rust।,
- कई मशीनों या प्रोसेस में कार्यों को वितरित करने के लिए वितरित कम्प्यूटिंग फ्रेमवर्क या मैसेज क्यू का उपयोग करना।

Ruby में अतुल्यकालिक प्रसंस्करण को डिजाइन और कार्यान्वित करते समय अपने कार्यों की प्रकृति और GIL द्वारा लगाई गई सीमाओं पर विचार करना महत्वपूर्ण है। जबकि थ्रेड-आधारित अतुल्यकालिक प्रसंस्करण I/O-बाउंड कार्यों के लिए लाभ प्रदान कर सकता है, यह GIL की बाधाओं के कारण CPU-बाउंड कार्यों के लिए महत्वपूर्ण प्रदर्शन सुधार नहीं दे सकता।

बेहतर सटीकता के लिए एन्सेम्बल तकनीकें

एन्सेम्बल तकनीकों में सिस्टम की समग्र सटीकता या मजबूती को बेहतर बनाने के लिए कई AI वर्कर्स के आउटपुट को संयोजित करना शामिल है। एकल AI वर्कर पर निर्भर रहने के बजाय, एन्सेम्बल तकनीकें अधिक सूचित निर्णय लेने के लिए कई वर्कर्स की सामूहिक बुद्धिमत्ता का लाभ उठाती हैं।

एन्सेम्बल्स विशेष रूप से महत्वपूर्ण होते हैं यदि आपके कार्यप्रवाह के विभिन्न हिस्से अलग-अलग एआई मॉडल्स के साथ बेहतर काम करते हैं, जो आप सोच सकते हैं उससे अधिक सामान्य है। GPT-4 जैसे शक्तिशाली मॉडल्स कम क्षमता वाले ओपन सोर्स विकल्पों की तुलना में बहुत महंगे हैं, और संभवतः आपके एप्लिकेशन के हर कार्यप्रवाह चरण के लिए आवश्यक नहीं हैं।

एक सामान्य एन्सेम्बल तकनीक है बहुमत मतदान, जहां कई एआई कार्यकर्ता स्वतंत्र रूप से एक ही इनपुट को प्रोसेस करते हैं, और अंतिम आउटपुट बहुमत की सहमति से निर्धारित होता है। यह दृष्टिकोण व्यक्तिगत कार्यकर्ता त्रुटियों के प्रभाव को कम करने और सिस्टम की समग्र विश्वसनीयता में सुधार करने में मदद कर सकता है।

आइए एक उदाहरण पर विचार करें जहां भावना विश्लेषण के लिए हमारे पास तीन एआई कार्यकर्ता हैं, जिनमें से प्रत्येक अलग-अलग मॉडल का उपयोग कर रहा है या अलग-अलग संदर्भों के साथ प्रदान किया गया है। हम अंतिम भावना भविष्यवाणी को निर्धारित करने के लिए बहुमत मतदान का उपयोग करके उनके आउटपुट को जोड़ सकते हैं।

```ruby
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
```

इस उदाहरण में, SentimentAnalysisEnsemble क्लास टेक्स्ट के साथ आरंभ होती है और तीन अलग-अलग भावना विश्लेषण एआई वर्कर्स को कार्यान्वित करती है। analyze मेथड प्रत्येक वर्कर से भविष्यवाणियां एकत्र करता है और group_by तथा max_by मेथड्स का उपयोग करके बहुमत भावना का निर्धारण करता है। अंतिम आउटपुट वह भावना है जो वर्कर्स के एन्सेम्बल से सबसे अधिक वोट प्राप्त करती है

एन्सेम्बल स्पष्ट रूप से एक ऐसा मामला है जहां समानांतरता के साथ प्रयोग करना आपके समय के लायक हो सकता है।

एआई वर्कर्स का गतिशील चयन और कार्यान्वयन

कुछ यदि सभी नहीं तो अधिकांश मामलों में, विशिष्ट एआई वर्कर का कार्यान्वयन रनटाइम परिस्थितियों या उपयोगकर्ता इनपुट पर निर्भर कर सकता है। एआई वर्कर्स का गतिशील चयन और कार्यान्वयन सिस्टम में लचीलापन और अनुकूलन क्षमता प्रदान करता है।

आप खुद को एक एकल एआई वर्कर में बहुत सारी कार्यक्षमता फिट करने की कोशिश करते हुए पा सकते हैं, उसे कई फ़ंक्शंस और एक बड़ा जटिल प्रॉम्प्ट देते हुए जो उन्हें कॉल करने का तरीका समझाता है। इस प्रलोभन का विरोध करें, मेरा विश्वास करें। इस अध्याय में जिस दृष्टिकोण की हम चर्चा कर रहे हैं उसे “कार्यकर्ताओं की बहुलता” कहने का एक कारण यह याद दिलाना है कि बहुत सारे विशेषज्ञ कार्यकर्ताओं का होना वांछनीय है, जहां प्रत्येक बड़े उद्देश्य की सेवा में अपना छोटा काम कर रहा है।

उदाहरण के लिए, एक चैटबॉट एप्लिकेशन पर विचार करें जहां विभिन्न एआई वर्कर्स विभिन्न प्रकार की उपयोगकर्ता क्वेरीज को संभालने के लिए जिम्मेदार हैं। उपयोगकर्ता के इनपुट के आधार पर, एप्लिकेशन क्वेरी को प्रोसेस करने के लिए उपयुक्त एआई वर्कर का गतिशील रूप से चयन करती है।

```ruby
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
```

इस उदाहरण में, ChatbotController उपयोगकर्ता की क्वेरी को process_query क्रिया के माध्यम से प्राप्त करता है। सबसे पहले यह क्वेरी के प्रकार को निर्धारित करने के लिए QueryClassifierWorker का उपयोग करता है। वर्गीकृत क्वेरी प्रकार के आधार पर, कंट्रोलर गतिशील रूप से उपयुक्त एआई वर्कर को प्रतिक्रिया उत्पन्न करने के लिए चुनता है। यह गतिशील चयन चैटबॉट को विभिन्न प्रकार की क्वेरी को संभालने और उन्हें प्रासंगिक एआई वर्कर्स तक भेजने की क्षमता प्रदान करता है।

चूंकि QueryClassifierWorker का कार्य अपेक्षाकृत सरल है और इसे अधिक संदर्भ या फ़ंक्शन परिभाषाओं की आवश्यकता नहीं है, आप इसे अल्ट्रा-फास्ट छोटे एलएलएम जैसे mistralai/mixtral-8x7b-instruct:nitro का उपयोग करके लागू कर सकते हैं। इसकी क्षमताएं कई कार्यों पर GPT-4 स्तर के करीब आती हैं और, जब मैं यह लिख रहा हूं, Groq इसे 444 टोकन/सेकंड की तेज गति से सेवा प्रदान कर सकता है।

पारंपरिक एनएलपी को एलएलएम के साथ जोड़ना

हालांकि बड़े भाषा मॉडल (एलएलएम) ने प्राकृतिक भाषा प्रसंस्करण (एनएलपी) के क्षेत्र में क्रांति ला दी है, जो कि विभिन्न कार्यों में अद्वितीय बहुमुखी प्रतिभा और प्रदर्शन प्रदान करते हैं, वे हर समस्या के लिए सबसे कुशल या लागत प्रभावी समाधान नहीं हैं। कई मामलों में, पारंपरिक एनएलपी तकनीकों को एलएलएम के साथ जोड़ने से विशिष्ट एनएलपी चुनौतियों को हल करने के लिए अधिक अनुकूलित, लक्षित और किफायती दृष्टिकोण मिल सकता है।

एलएलएम को एनएलपी के स्विस आर्मी नाइफ के रूप में सोचें—अत्यंत बहुमुखी और शक्तिशाली, लेकिन जरूरी नहीं कि हर काम के लिए सबसे अच्छा उपकरण हो। कभी-कभी, एक समर्पित उपकरण जैसे कॉर्कस्क्रू या कैन ओपनर किसी विशिष्ट कार्य के लिए अधिक प्रभावी और कुशल हो सकता है। इसी तरह, पारंपरिक एनएलपी तकनीकें, जैसे दस्तावेज़ क्लस्टरिंग, विषय पहचान, और वर्गीकरण, अक्सर आपकी एनएलपी पाइपलाइन के कुछ पहलुओं के लिए अधिक लक्षित और लागत प्रभावी समाधान प्रदान कर सकती हैं।

पारंपरिक एनएलपी तकनीकों का एक प्रमुख लाभ उनकी कम्प्यूटेशनल दक्षता है। ये विधियां, जो अक्सर सरल सांख्यिकीय मॉडल या नियम-आधारित दृष्टिकोणों पर निर्भर करती हैं, एलएलएम की तुलना में बड़ी मात्रा में पाठ डेटा को बहुत तेजी से और कम कम्प्यूटेशनल ओवरहेड के साथ प्रोसेस कर सकती हैं। यह उन्हें विशेष रूप से बड़ी मात्रा में दस्तावेजों के विश्लेषण और संगठन से जुड़े कार्यों के लिए उपयुक्त बनाता है, जैसे समान लेखों को क्लस्टर करना या पाठों के संग्रह में प्रमुख विषयों की पहचान करना।

इसके अतिरिक्त, पारंपरिक एनएलपी तकनीकें विशिष्ट कार्यों के लिए, विशेष रूप से डोमेन-विशिष्ट डेटासेट पर प्रशिक्षित होने पर, उच्च सटीकता और परिशुद्धता प्राप्त कर सकती हैं। उदाहरण के लिए, सपोर्ट वेक्टर मशीन (एसवीएम) या नेइव बेज़ जैसे पारंपरिक मशीन लर्निंग एल्गोरिदम का उपयोग करने वाला एक अच्छी तरह से ट्यून किया गया दस्तावेज़ वर्गीकरक न्यूनतम कम्प्यूटेशनल लागत के साथ दस्तावेज़ों को पूर्व-निर्धारित श्रेणियों में सटीक रूप से वर्गीकृत कर सकता है।

हालांकि, एलएलएम वास्तव में तब चमकते हैं जब भाषा, संदर्भ और तर्क की गहरी समझ की आवश्यकता होती है। सुसंगत और संदर्भगत रूप से प्रासंगिक पाठ उत्पन्न करने, प्रश्नों का उत्तर देने और लंबे अनुच्छेदों को संक्षेपित करने की उनकी क्षमता पारंपरिक एनएलपी विधियों से अद्वितीय है। एलएलएम जटिल भाषाई घटनाओं को प्रभावी ढंग से संभाल सकते हैं, जैसे अस्पष्टता, सह-संदर्भ और मुहावरेदार अभिव्यक्तियां, जो उन्हें प्राकृतिक भाषा निर्माण या समझ की आवश्यकता वाले कार्यों के लिए अमूल्य बनाते हैं।

वास्तविक शक्ति पारंपरिक एनएलपी तकनीकों को एलएलएम के साथ मिलाकर हाइब्रिड दृष्टिकोण बनाने में निहित है जो दोनों की ताकत का लाभ उठाते हैं। दस्तावेज़ पूर्व-प्रसंस्करण, क्लस्टरिंग और विषय निष्कर्षण जैसे कार्यों के लिए पारंपरिक एनएलपी विधियों का उपयोग करके, आप अपने पाठ डेटा को कुशलतापूर्वक व्यवस्थित और संरचित कर सकते हैं। इस संरचित जानकारी को फिर सारांश तैयार करने, प्रश्नों का उत्तर देने या व्यापक रिपोर्ट बनाने जैसे अधिक उन्नत कार्यों के लिए एलएलएम में फीड किया जा सकता है।

उदाहरण के लिए, एक ऐसे उपयोग मामले पर विचार करें जहां आप बड़ी संख्या में व्यक्तिगत ट्रेंड दस्तावेज़ों के आधार पर किसी विशिष्ट डोमेन के लिए एक ट्रेंड रिपोर्ट तैयार करना चाहते हैं। केवल एलएलएम पर निर्भर रहने के बजाय, जो बड़ी मात्रा में पाठ को संसाधित करने के लिए कम्प्यूटेशनल रूप से महंगा और समय लेने वाला हो सकता है, आप एक हाइब्रिड दृष्टिकोण अपना सकते हैं:

1. विषय मॉडलिंग (जैसे, लेटेंट डिरिचलेट एलोकेशन) या क्लस्टरिंग एल्गोरिदम (जैसे, के-मीन्स) जैसी पारंपरिक एनएलपी तकनीकों का उपयोग करें, ताकि समान ट्रेंड दस्तावेज़ों को एक साथ समूहीकृत किया जा सके और कॉर्पस के भीतर प्रमुख विषयों और टॉपिक की पहचान की जा सके।
2. क्लस्टर किए गए दस्तावेज़ों और पहचाने गए विषयों को एलएलएम में फीड करें, प्रत्येक क्लस्टर या विषय के लिए सुसंगत और सूचनात्मक सारांश बनाने के लिए इसकी बेहतर भाषा समझ और निर्माण क्षमताओं का लाभ उठाएं।
3. अंत में, व्यक्तिगत सारांशों को जोड़कर, सबसे महत्वपूर्ण रुझानों को उजागर करके और एकत्रित जानकारी के आधार पर अंतर्दृष्टि और सिफारिशें प्रदान

करके एक व्यापक ट्रेंड रिपोर्ट तैयार करने के लिए एलएलएम का उपयोग करें।

इस तरह पारंपरिक एनएलपी तकनीकों को एलएलएम के साथ जोड़कर, आप बड़ी मात्रा में पाठ डेटा को कुशलतापूर्वक संसाधित कर सकते हैं, सार्थक अंतर्दृष्टि निकाल सकते हैं और कम्प्यूटेशनल संसाधनों और लागतों को अनुकूलित करते हुए उच्च-गुणवत्ता वाली रिपोर्ट तैयार कर सकते हैं।

जैसे-जैसे आप अपनी एनएलपी परियोजनाओं की शुरुआत करते हैं, यह आवश्यक है कि आप प्रत्येक कार्य की विशिष्ट आवश्यकताओं और सीमाओं का सावधानीपूर्वक मूल्यांकन करें और विचार करें कि सर्वोत्तम परिणाम प्राप्त करने के लिए पारंपरिक एनएलपी पद्धतियों और एलएलएम को एक साथ कैसे लाभप्रद ढंग से उपयोग किया जा सकता है। पारंपरिक तकनीकों की दक्षता और सटीकता को एलएलएम की बहुमुखी प्रतिभा और शक्ति के साथ जोड़कर, आप अत्यंत प्रभावी और किफायती एनएलपी समाधान बना सकते हैं जो आपके उपयोगकर्ताओं और हितधारकों को मूल्य प्रदान करते हैं।

उपकरण का उपयोग

एआई-संचालित एप्लिकेशन विकास के क्षेत्र में, “टूल का उपयोग” या “फंक्शन कॉलिंग” की अवधारणा एक शक्तिशाली तकनीक के रूप में उभरी है जो आपके LLM को बाहरी उपकरणों, APIs, फंक्शंस, डेटाबेस, और अन्य संसाधनों से जुड़ने में सक्षम बनाती है। यह दृष्टिकोण केवल टेक्स्ट आउटपुट से कहीं अधिक समृद्ध व्यवहारों की अनुमति देता है, और आपके एआई घटकों और आपके एप्लिकेशन के पारिस्थितिकी तंत्र के बाकी हिस्सों के बीच अधिक गतिशील संवाद को सक्षम बनाता है। जैसा कि हम इस अध्याय में देखेंगे, टूल का उपयोग आपको अपने एआई मॉडल को संरचित तरीकों से डेटा उत्पन्न करने का विकल्प भी देता है।

टूल का उपयोग क्या है?

टूल का उपयोग, जिसे फंक्शन कॉलिंग के नाम से भी जाना जाता है, एक ऐसी तकनीक है जो डेवलपर्स को उन फंक्शंस की सूची निर्दिष्ट करने की अनुमति देती है

जिनके साथ एक LLM जनरेशन प्रक्रिया के दौरान संवाद कर सकता है। ये टूल सरल उपयोगिता फंक्शंस से लेकर जटिल APIs या डेटाबेस क्वेरी तक हो सकते हैं। LLM को इन टूल्स तक पहुंच प्रदान करके, डेवलपर्स मॉडल की क्षमताओं का विस्तार कर सकते हैं और इसे ऐसे कार्य करने में सक्षम बना सकते हैं जिन्हें बाहरी ज्ञान या कार्यों की आवश्यकता होती है।

आकृति 7. एक एआई कार्यकर्ता के लिए फंक्शन परिभाषा का उदाहरण जो दस्तावेज़ों का विश्लेषण करता है

```
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
```

उपकरण उपयोग के पीछे की मुख्य अवधारणा LLM को उपयोगकर्ता के इनपुट या दी गई कार्य के आधार पर उपयुक्त उपकरणों को गतिशील रूप से चुनने और निष्पादित

करने की क्षमता प्रदान करना है। केवल मॉडल के पूर्व-प्रशिक्षित ज्ञान पर निर्भर रहने के बजाय, उपकरण उपयोग LLM को अधिक सटीक, प्रासंगिक और कार्यान्वयन योग्य प्रतिक्रियाएं उत्पन्न करने के लिए बाहरी संसाधनों का लाभ उठाने की अनुमति देता है। उपकरण उपयोग RAG (पुनर्प्राप्ति संवर्धित जनन) जैसी तकनीकों को लागू करना बहुत आसान बना देता है।

> ध्यान दें कि जब तक अन्यथा न कहा गया हो, यह पुस्तक मानती है कि आपके AI मॉडल में कोई अंतर्निहित सर्वर-साइड टूल्स नहीं हैं। कोई भी टूल जो आप अपने AI के लिए उपलब्ध कराना चाहते हैं, उसे प्रत्येक API अनुरोध में आपके द्वारा स्पष्ट रूप से घोषित किया जाना चाहिए, साथ ही इसके निष्पादन के लिए प्रावधान भी होने चाहिए यदि और जब आपका AI आपको बताए कि वह अपनी प्रतिक्रिया में उस टूल का उपयोग करना चाहेगा।

उपकरण उपयोग की संभावनाएं

उपकरण उपयोग AI-संचालित अनुप्रयोगों के लिए विभिन्न संभावनाएं खोलता है। यहाँ कुछ उदाहरण दिए गए हैं जो उपकरण उपयोग से प्राप्त किए जा सकते हैं:

1. **चैटबॉट और आभासी सहायक**: LLM को बाहरी उपकरणों से जोड़कर, चैटबॉट और आभासी सहायक अधिक जटिल कार्य कर सकते हैं, जैसे डेटाबेस से जानकारी प्राप्त करना, API कॉल निष्पादित करना, या अन्य सिस्टम के साथ संवाद करना। उदाहरण के लिए, एक चैटबॉट उपयोगकर्ता के अनुरोध के आधार पर CRM टूल का उपयोग करके किसी डील की स्थिति बदल सकता है।
2. **डेटा विश्लेषण और अंतर्दृष्टि**: LLM को उन्नत डेटा प्रोसेसिंग कार्यों को करने के लिए डेटा विश्लेषण उपकरणों या लाइब्रेरी से जोड़ा जा सकता है। यह अनुप्रयोगों को उपयोगकर्ता प्रश्नों के आधार पर अंतर्दृष्टि उत्पन्न करने, तुलनात्मक विश्लेषण करने, या डेटा-आधारित सिफारिशें प्रदान करने में सक्षम बनाता है।

3. **खोज और सूचना पुनर्प्राप्ति**: उपकरण उपयोग LLM को सर्च इंजन, वेक्टर डेटाबेस, या अन्य सूचना पुनर्प्राप्ति प्रणालियों के साथ संवाद करने की अनुमति देता है। उपयोगकर्ता प्रश्नों को खोज प्रश्नों में बदलकर, LLM कई स्रोतों से प्रासंगिक जानकारी प्राप्त कर सकता है और उपयोगकर्ता प्रश्नों के व्यापक उत्तर प्रदान कर सकता है।
4. **बाहरी सेवाओं के साथ एकीकरण**: उपकरण उपयोग AI-संचालित अनुप्रयोगों और बाहरी सेवाओं या API के बीच निर्बाध एकीकरण को सक्षम बनाता है। उदाहरण के लिए, एक LLM वास्तविक समय के मौसम अपडेट प्रदान करने के लिए मौसम API के साथ या बहुभाषी प्रतिक्रियाएं उत्पन्न करने के लिए अनुवाद API के साथ संवाद कर सकता है।

उपकरण उपयोग कार्यप्रवाह

टूल उपयोग कार्यप्रवाह में आमतौर पर चार प्रमुख चरण शामिल होते हैं:

1. अपने अनुरोध संदर्भ में फ़ंक्शन परिभाषाएँ शामिल करें
2. गतिशील (या स्पष्ट) टूल चयन
3. फ़ंक्शन(ओं) का निष्पादन
4. मूल प्रॉम्प्ट का वैकल्पिक निरंतरीकरण

आइए इन सभी चरणों की विस्तार से समीक्षा करें।

अपने अनुरोध संदर्भ में फ़ंक्शन परिभाषाएँ शामिल करें

AI को यह पता होता है कि उसके पास कौन से टूल उपलब्ध हैं क्योंकि आप उसे अपने पूर्णता अनुरोध के हिस्से के रूप में एक सूची प्रदान करते हैं (आमतौर पर JSON स्कीमा के एक प्रकार का उपयोग करके फ़ंक्शन के रूप में परिभाषित)।

टूल परिभाषा का सटीक सिंटैक्स मॉडल-विशिष्ट होता है।

Claude 3 में get_weather फ़ंक्शन को इस प्रकार परिभाषित किया जाता है:

```
{
    "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"]
    }
}
```

और यही तरीका है जिससे आप GPT-4 के लिए समान फ़ंक्शन को परिभाषित करेंगे, इसे tools पैरामीटर के मान के रूप में पास करते हुए:

```
{
    "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"],
```

```
    },
}
```

> लगभग एक जैसा, सिवाय इसके कि बिना किसी स्पष्ट कारण के अलग! कितना परेशान करने वाला है।

फ़ंक्शन परिभाषाएँ नाम, विवरण और इनपुट पैरामीटर्स को निर्दिष्ट करती हैं। इनपुट पैरामीटर्स को एन्यूम्स जैसी विशेषताओं का उपयोग करके स्वीकार्य मानों को सीमित करने और यह निर्दिष्ट करने के लिए कि कोई पैरामीटर आवश्यक है या नहीं, और अधिक परिभाषित किया जा सकता है।

वास्तविक फ़ंक्शन परिभाषाओं के अलावा, आप सिस्टम निर्देश में फ़ंक्शन का उपयोग क्यों और कैसे करना है, इसके लिए निर्देश या संदर्भ भी शामिल कर सकते हैं।

उदाहरण के लिए, Olympia में मेरे वेब सर्च टूल में यह सिस्टम निर्देश शामिल है, जो एआई को याद दिलाता है कि उसके पास उल्लिखित टूल्स उपलब्ध हैं:

```
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.
```

विस्तृत विवरण प्रदान करना उपकरण प्रदर्शन में सबसे महत्वपूर्ण कारक माना जाता है। आपके विवरण में उपकरण के बारे में हर विवरण की व्याख्या होनी चाहिए, जिसमें शामिल हैं:

- उपकरण क्या करता है
- इसका उपयोग कब किया जाना चाहिए (और कब नहीं)
- प्रत्येक पैरामीटर का क्या अर्थ है और यह उपकरण के व्यवहार को कैसे प्रभावित करता है

- उपकरण के कार्यान्वयन पर लागू होने वाली कोई महत्वपूर्ण सावधानियाँ या सीमाएं

आप AI को अपने उपकरणों के बारे में जितना अधिक संदर्भ दे सकते हैं, यह उतना ही बेहतर तरीके से तय कर पाएगा कि कब और कैसे उनका उपयोग करना है। उदाहरण के लिए, Anthropic अपनी Claude 3 श्रृंखला के लिए प्रति उपकरण कम से कम 3-4 वाक्यों की सिफारिश करता है, यदि उपकरण जटिल है तो और अधिक।

यह सहज ज्ञान के विपरीत हो सकता है, लेकिन विवरण को उदाहरणों से भी अधिक महत्वपूर्ण माना जाता है। हालांकि आप किसी उपकरण के विवरण में या साथ दिए गए प्रॉम्प्ट में उसके उपयोग के उदाहरण शामिल कर सकते हैं, यह उपकरण के उद्देश्य और पैरामीटर्स की स्पष्ट और व्यापक व्याख्या से कम महत्वपूर्ण है। विवरण को पूरी तरह से विकसित करने के बाद ही उदाहरण जोड़ें।

यहाँ Stripe-जैसे API फ़ंक्शन स्पेसिफिकेशन का एक उदाहरण है:

```
{
  "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" ]
      }
    }
}
```

व्यवहार में, कुछ मॉडल्स नेस्टेड फ़ंक्शन स्पेसिफिकेशन्स और ऐरे, डिक्शनरीज़ जैसे जटिल आउटपुट डेटा टाइप्स को संभालने में कठिनाई का सामना करते हैं। लेकिन सैद्धांतिक रूप से, आप किसी भी गहराई के JSON स्कीमा स्पेसिफिकेशन्स प्रदान कर सकते हैं!

गतिशील उपकरण चयन

जब आप टूल परिभाषाओं के साथ एक चैट कम्प्लीशन को निष्पादित करते हैं, तो LLM स्वचालित रूप से सबसे उपयुक्त टूल(्स) का चयन करता है और प्रत्येक टूल के लिए आवश्यक इनपुट पैरामीटर्स जनरेट करता है।

व्यवहार में, AI का बिल्कुल सही फ़ंक्शन को कॉल करने और इनपुट्स के लिए आपके स्पेसिफिकेशन का बिल्कुल सही पालन करने की क्षमता हिट या मिस होती है। टेम्परेचर हाइपरपैरामीटर को 0.0 तक नीचे करने से बहुत मदद मिलती है, लेकिन मेरे अनुभव में आपको अभी भी कभी-कभी त्रुटियां मिलेंगी। इन विफलताओं में काल्पनिक फ़ंक्शन नाम, गलत नाम वाले या बस गायब इनपुट पैरामीटर्स शामिल हैं। पैरामीटर्स JSON के रूप में पास किए जाते हैं, जिसका मतलब है कि कभी-कभी आप टूटे हुए, गलत उद्धृत, या अन्यथा खराब JSON के कारण त्रुटियां देखेंगे।

सेल्फ हीलिंग डेटा पैटर्न्स सिंटैक्स त्रुटियों के कारण टूटे हुए फ़ंक्शन कॉल्स को स्वचालित रूप से ठीक करने में मदद कर सकते हैं।

बाध्य (या स्पष्ट) उपकरण चयन

कुछ मॉडल्स आपको रिक्वेस्ट में एक पैरामीटर के रूप में किसी विशेष फ़ंक्शन को कॉल करने के लिए बाध्य करने का विकल्प देते हैं। अन्यथा, फ़ंक्शन को कॉल करना या नहीं करना पूरी तरह से AI के विवेक पर निर्भर है।

किसी फ़ंक्शन कॉल को बाध्य करने की क्षमता कुछ परिदृश्यों में महत्वपूर्ण होती है जहां आप चाहते हैं कि AI के गतिशील चयन प्रक्रिया की परवाह किए बिना एक विशिष्ट टूल या फ़ंक्शन निष्पादित हो। इस क्षमता के महत्वपूर्ण होने के कई कारण हैं:

1. **स्पष्ट नियंत्रण**: आप AI का उपयोग एक डिस्क्रीट कम्पोनेंट के रूप में या एक पूर्वनिर्धारित कार्यप्रवाह में कर रहे हो सकते हैं जिसमें किसी विशेष समय पर एक विशेष फ़ंक्शन का निष्पादन आवश्यक होता है। कॉल को बाध्य करके, आप

AI से विनम्रता से पूछने के बजाय सुनिश्चित कर सकते हैं कि वांछित फ़ंक्शन को कॉल किया जाए।

2. **डीबगिंग और परीक्षण**: AI-संचालित एप्लिकेशन्स के विकास और परीक्षण के दौरान, फ़ंक्शन कॉल्स को बाध्य करने की क्षमता डीबगिंग उद्देश्यों के लिए बेहद मूल्यवान होती है। विशिष्ट फ़ंक्शन्स को स्पष्ट रूप से ट्रिगर करके, आप अपने एप्लिकेशन के अलग-अलग घटकों को अलग करके परीक्षण कर सकते हैं। यह आपको फ़ंक्शन कार्यान्वयन की सटीकता को सत्यापित करने, इनपुट पैरामीटर्स को मान्य करने और अपेक्षित परिणामों की वापसी सुनिश्चित करने की अनुमति देता है।
3. **सीमावर्ती मामलों का प्रबंधन**: ऐसे सीमावर्ती मामले या अपवाद स्थितियां हो सकती हैं जहां AI की गतिशील चयन प्रक्रिया किसी फंक्शन को निष्पादित नहीं करने का चयन कर सकती है, और आप बाहरी प्रक्रियाओं के आधार पर जानते हैं कि उसे करना चाहिए। ऐसी स्थितियों में, फंक्शन कॉल को बलपूर्वक करने की क्षमता आपको इन परिस्थितियों को स्पष्ट रूप से संभालने की अनुमति देती है। अपने एप्लिकेशन लॉजिक में नियम या शर्तें परिभाषित करें जो यह निर्धारित करें कि कब AI के विवेक को ओवरराइड करना है।
4. **स्थिरता और पुनरुत्पादनीयता**: यदि आपके पास विशिष्ट क्रम में निष्पादित किए जाने वाले फंक्शंस का एक विशेष क्रम है, तो कॉल्स को बलपूर्वक करना यह सुनिश्चित करता है कि हर बार एक ही क्रम का पालन किया जाए। यह विशेष रूप से उन एप्लिकेशंस में महत्वपूर्ण है जहां स्थिरता और पूर्वानुमेय व्यवहार महत्वपूर्ण हैं, जैसे वित्तीय प्रणालियों या वैज्ञानिक सिमुलेशन में।
5. **प्रदर्शन अनुकूलन**: कुछ मामलों में, फंक्शन कॉल को बलपूर्वक करने से प्रदर्शन में सुधार हो सकता है। यदि आप जानते हैं कि किसी विशेष कार्य के लिए एक विशिष्ट फंक्शन आवश्यक है और AI की गतिशील चयन प्रक्रिया अनावश्यक ओवरहेड पैदा कर सकती है, तो आप चयन प्रक्रिया को बाईपास कर सकते हैं और आवश्यक फंक्शन को सीधे कॉल कर सकते हैं। यह आपके एप्लिकेशन की विलंबता को कम करने और समग्र दक्षता में सुधार करने में मदद कर सकता है।

संक्षेप में, AI-संचालित एप्लिकेशंस में फंक्शन कॉल्स को बलपूर्वक करने की क्षमता

स्पष्ट नियंत्रण प्रदान करती है, डीबगिंग और टेस्टिंग में सहायता करती है, सीमावर्ती मामलों को संभालती है, स्थिरता और पुनरुत्पादनीयता सुनिश्चित करती है। यह आपके शस्त्रागार में एक शक्तिशाली उपकरण है, लेकिन हमें इस महत्वपूर्ण सुविधा के एक और पहलू पर चर्चा करने की आवश्यकता है।

कई निर्णय-लेने के उपयोग मामलों में, हम हमेशा चाहते हैं कि मॉडल एक फंक्शन कॉल करे और कभी भी केवल अपने आंतरिक ज्ञान के साथ प्रतिक्रिया न दे। उदाहरण के लिए, यदि आप विभिन्न कार्यों में विशेषज्ञ कई मॉडल्स के बीच रूटिंग कर रहे हैं (बहुभाषी इनपुट, गणित, आदि), तो आप सहायक मॉडल्स में से किसी एक को अनुरोध सौंपने के लिए फंक्शन-कॉलिंग मॉडल का उपयोग कर सकते हैं और स्वतंत्र रूप से प्रतिक्रिया नहीं दे सकते।

टूल चॉइस पैरामीटर

GPT-4 और अन्य भाषा मॉडल जो फंक्शन कॉलिंग का समर्थन करते हैं, आपको एक पूर्ति के भाग के रूप में टूल के उपयोग को नियंत्रित करने के लिए एक tool_choice पैरामीटर प्रदान करते हैं। इस पैरामीटर के तीन संभावित मान हैं:

- auto AI को टूल का उपयोग करने या केवल प्रतिक्रिया देने का पूर्ण विवेक देता है
- required AI को बताता है कि उसे प्रतिक्रिया देने के बजाय एक टूल को अवश्य कॉल करना चाहिए, लेकिन टूल का चयन AI पर छोड़ देता है
- तीसरा विकल्प उस name_of_function को सेट करना है जिसे आप बलपूर्वक करना चाहते हैं। इस पर अगले खंड में अधिक जानकारी दी गई है।

ध्यान दें कि यदि आप tool choice को required पर सेट करते हैं, तो मॉडल को उपलब्ध कराए गए फ़ंक्शन्स में से सबसे उपयुक्त फ़ंक्शन को चुनने के लिए मजबूर किया जाएगा, भले ही कोई भी प्रॉम्प्ट के लिए वास्तव में उपयुक्त न हो। प्रकाशन के समय, मुझे किसी ऐसे मॉडल की जानकारी नहीं है जो खाली tool_calls प्रतिक्रिया लौटाएगा, या किसी अन्य तरीके से आपको बताएगा कि उसे कोई उपयुक्त फ़ंक्शन कॉल करने के लिए नहीं मिला।

संरचित आउटपुट प्राप्त करने के लिए फ़ंक्शन को फोर्स करना

फ़ंक्शन कॉल को फोर्स करने की क्षमता आपको चैट कम्पलीशन से संरचित डेटा को फोर्स करने का एक तरीका देती है, बजाय इसके कि आप इसे स्वयं प्लेनटेक्स्ट प्रतिक्रिया से निकालें।

> संरचित आउटपुट प्राप्त करने के लिए फ़ंक्शन्स को फोर्स करना इतना महत्वपूर्ण क्यों है? सीधे शब्दों में कहें, क्योंकि एलएलएम आउटपुट से संरचित डेटा निकालना बहुत कठिन होता है। आप XML में डेटा मांगकर अपना जीवन थोड़ा आसान बना सकते हैं, लेकिन फिर आपको XML को पार्स करना पड़ता है। और जब वह XML गायब हो जाता है क्योंकि आपके एआई ने जवाब दिया: "मैं क्षमाप्रार्थी हूं, लेकिन मैं आपके द्वारा अनुरोधित डेटा को जनरेट नहीं कर सकता क्योंकि बला, बला, बला..." तब आप क्या करते हैं?

इस तरह से टूल्स का उपयोग करते समय:

- आपको शायद अपने अनुरोध में एक ही टूल को परिभाषित करना चाहिए
- tool_choice पैरामीटर का उपयोग करके इसके फ़ंक्शन के उपयोग को फोर्स करना याद रखें।

- याद रखें कि मॉडल इनपुट को टूल में पास करेगा, इसलिए टूल का नाम और विवरण मॉडल के दृष्टिकोण से होना चाहिए, न कि आपके।

इस अंतिम बिंदु को स्पष्टता के लिए एक उदाहरण की आवश्यकता है। मान लीजिए कि आप एआई से उपयोगकर्ता टेक्स्ट पर भावना विश्लेषण करने के लिए कह रहे हैं। फ़ंक्शन का नाम analyze_sentiment नहीं होगा, बल्कि यह save_sentiment_analysis जैसा कुछ होगा। भावना विश्लेषण एआई कर रहा है, टूल नहीं। टूल जो कर रहा है (एआई के दृष्टिकोण से) वह केवल विश्लेषण के परिणामों को सहेज रहा है।

यहाँ Claude 3 का उपयोग करके एक छवि का सारांश अच्छी तरह से संरचित JSON में रिकॉर्ड करने का एक उदाहरण दिया गया है, इस बार कमांड लाइन पर curl का उपयोग करके:

```
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."
                }
            ]
        }
    ]
}'
```

दिए गए उदाहरण में, हम Anthropic के Claude 3 मॉडल का उपयोग एक छवि का संरचित JSON सारांश तैयार करने के लिए कर रहे हैं। यह इस प्रकार काम करता है:

1. हम रिक्वेस्ट पेलोड के tools ऐरे में record_summary नामक एक टूल को परिभाषित करते हैं। यह टूल छवि का सारांश संरचित JSON में रिकॉर्ड करने के लिए जिम्मेदार है।
2. record_summary टूल में एक input_schema है जो JSON आउटपुट की अपेक्षित संरचना को निर्धारित करता है। यह तीन गुणों को परिभाषित करता है:

    - key_colors: छवि में मुख्य रंगों का प्रतिनिधित्व करने वाली वस्तुओं की एक सरणी। प्रत्येक रंग वस्तु में लाल, हरे और नीले मान (0.0 से 1.0 तक) के लिए गुण और snake_case प्रारूप में मानव-पठनीय रंग नाम होते हैं।
    - description: छवि के संक्षिप्त विवरण के लिए एक स्ट्रिंग गुण, जो 1-2 वाक्यों तक सीमित है।
    - estimated_year: एक वैकल्पिक पूर्णांक गुण, जो छवि के लिए अनुमानित वर्ष दर्शाता है, यदि यह एक गैर-काल्पनिक फोटो प्रतीत होती है।

3. messages ऐरे में, हम छवि डेटा को base64-एन्कोडेड स्ट्रिंग के रूप में मीडिया प्रकार के साथ प्रदान करते हैं। यह मॉडल को इनपुट के हिस्से के रूप में छवि को प्रोसेस करने की अनुमति देता है।

4. हम Claude को छवि का वर्णन करने के लिए record_summary टूल का उपयोग करने के लिए भी प्रेरित करते हैं।
5. जब रिक्वेस्ट Claude 3 मॉडल को भेजी जाती है, यह छवि का विश्लेषण करता है और निर्दिष्ट input_schema के आधार पर एक JSON सारांश तैयार करता है। मॉडल मुख्य रंगों को निकालता है, एक संक्षिप्त विवरण प्रदान करता है, और छवि के लिए वर्ष का अनुमान लगाता है (यदि लागू हो)।
6. उत्पन्न JSON सारांश को record_summary टूल के पैरामीटर्स के रूप में पास किया जाता है, जो छवि की मुख्य विशेषताओं का एक संरचित प्रतिनिधित्व प्रदान करता है।

record_summary टूल का उपयोग एक सुपरिभाषित input_schema के साथ करने से, हम सादे पाठ निष्कर्षण पर निर्भर किए बिना एक छवि का संरचित JSON सारांश प्राप्त कर सकते हैं। यह दृष्टिकोण सुनिश्चित करता है कि आउटपुट एक सुसंगत प्रारूप का पालन करता है और एप्लिकेशन के डाउनस्ट्रीम कंपोनेंट्स द्वारा आसानी से पार्स और प्रोसेस किया जा सकता है।

AI-संचालित एप्लिकेशन में टूल उपयोग की एक फंक्शन कॉल को फोर्स करने और अपेक्षित आउटपुट संरचना को निर्दिष्ट करने की क्षमता एक शक्तिशाली विशेषता है। यह डेवलपर्स को जनरेट किए गए आउटपुट पर अधिक नियंत्रण रखने और उनके एप्लिकेशन के कार्यप्रवाह में AI-जनित डेटा के एकीकरण को सरल बनाने की अनुमति देता है।

फ़ंक्शन का निष्पादन

आपने फ़ंक्शन्स को परिभाषित किया है, और अपने AI को प्रेरित किया, जिसने निर्णय लिया कि उसे आपके फ़ंक्शन्स में से एक को कॉल करना चाहिए। अब समय है कि आपका एप्लिकेशन कोड या लाइब्रेरी, अगर आप raix-rails जैसी Ruby gem का उपयोग कर रहे हैं, फ़ंक्शन कॉल और उसके पैरामीटर्स को संबंधित कार्यान्वयन तक आपके एप्लिकेशन कोड में पहुंचाए।

आपका एप्लिकेशन कोड तय करता है कि फ़ंक्शन के परिणामों के साथ क्या करना है। हो सकता है कि यह lambda में एक लाइन कोड से जुड़ा हो, या किसी बाहरी API को कॉल करने से। हो सकता है कि इसमें किसी अन्य AI कॉम्पोनेंट को कॉल करना शामिल हो, या आपके सिस्टम के बाकी हिस्सों में सैकड़ों या हजारों लाइनों का कोड शामिल हो। यह पूरी तरह से आप पर निर्भर करता है।

कभी-कभी फ़ंक्शन कॉल ऑपरेशन का अंत होता है, लेकिन अगर परिणाम विचार श्रृंखला में जानकारी का प्रतिनिधित्व करते हैं जिसे AI द्वारा जारी रखा जाना है, तो आपके एप्लिकेशन कोड को निष्पादन परिणामों को चैट ट्रांसक्रिप्ट में डालना होगा और AI को प्रोसेसिंग जारी रखने देना होगा।

उदाहरण के लिए, यहाँ एक Raix फ़ंक्शन घोषणा है जो Olympia के AccountManager द्वारा ग्राहक सेवा के लिए बुद्धिमान कार्यप्रवाह संयोजन के हिस्से के रूप में हमारे ग्राहकों के साथ संवाद करने के लिए उपयोग की जाती है।

```
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
```

यहां क्या हो रहा है, यह तुरंत स्पष्ट नहीं हो सकता है, इसलिए मैं इसे विस्तार से समझाता हूं।

1. AccountManager क्लास खाता प्रबंधन से संबंधित कई फ़ंक्शन को परिभाषित करती है। यह आपकी योजना को बदल सकती है, टीम के सदस्यों को जोड़

और हटा सकती है, और भी कई काम कर सकती है।
2. इसके शीर्ष-स्तरीय निर्देश AccountManager को बताते हैं कि उसे खाता परिवर्तन अनुरोध के परिणामों के साथ खाता मालिक को notify_account_owner फ़ंक्शन का उपयोग करके सूचित करना चाहिए।
3. फ़ंक्शन की संक्षिप्त परिभाषा में शामिल हैं:

- नाम
- विवरण
- पैरामीटर्स message: { type: "string" }
- फ़ंक्शन को कॉल करने पर निष्पादित होने वाला ब्लॉक

फ़ंक्शन ब्लॉक के परिणामों के साथ प्रतिलेख को अपडेट करने के बाद, chat_completion विधि को फिर से कॉल किया जाता है। यह विधि अपडेट किए गए वार्तालाप प्रतिलेख को आगे की प्रोसेसिंग के लिए AI मॉडल को वापस भेजने के लिए जिम्मेदार है। हम इस प्रक्रिया को वार्तालाप लूप कहते हैं।

जब AI मॉडल को अपडेट किए गए प्रतिलेख के साथ एक नया चैट पूर्णता अनुरोध प्राप्त होता है, तो उसे पहले निष्पादित किए गए फ़ंक्शन के परिणामों तक पहुंच होती है। यह इन परिणामों का विश्लेषण कर सकता है, उन्हें अपनी निर्णय-प्रक्रिया में शामिल कर सकता है, और वार्तालाप के संचयी संदर्भ के आधार पर अगली प्रतिक्रिया या कार्रवाई उत्पन्न कर सकता है। यह अपडेट किए गए संदर्भ के आधार पर अतिरिक्त फ़ंक्शन निष्पादित करने का चयन कर सकता है, या यदि यह निर्धारित करता है कि कोई और फ़ंक्शन कॉल आवश्यक नहीं हैं, तो मूल प्रॉम्प्ट के लिए अंतिम प्रतिक्रिया उत्पन्न कर सकता है।

मूल प्रॉम्प्ट का वैकल्पिक निरंतरता

जब आप टूल परिणामों को LLM को वापस भेजते हैं और मूल प्रॉम्प्ट की प्रोसेसिंग जारी रखते हैं, तो AI उन परिणामों का उपयोग या तो अतिरिक्त फ़ंक्शन कॉल करने या एक अंतिम सादा टेक्स्ट प्रतिक्रिया उत्पन्न करने के लिए करता है।

Cohere के Command-R जैसे कुछ मॉडल अपनी प्रतिक्रियाओं में उनके द्वारा उपयोग किए गए विशिष्ट टूल्स का उल्लेख कर सकते हैं, जो अतिरिक्त पारदर्शिता और ट्रेसेबिलिटी प्रदान करता है।

उपयोग में आने वाले मॉडल के आधार पर, फ़ंक्शन कॉल के परिणाम प्रतिलेख संदेशों में अपनी खास भूमिका के साथ रहेंगे या किसी अन्य सिंटैक्स में प्रतिबिंबित होंगे। लेकिन महत्वपूर्ण भाग यह है कि वह डेटा प्रतिलेख में हो, ताकि AI आगे क्या करना है यह तय करते समय उस पर विचार कर सके।

एक सामान्य (और संभावित रूप से महंगी) त्रुटि स्थिति है चैट को जारी रखने से पहले प्रतिलेख में फ़ंक्शन परिणामों को जोड़ना भूल जाना। परिणामस्वरूप, AI को लगभग उसी तरह से प्रॉम्प्ट किया जाएगा जैसे पहली बार फ़ंक्शन को कॉल करने से पहले किया गया था। दूसरे शब्दों में, AI के लिए, उसने अभी तक फ़ंक्शन कॉल नहीं किया है। इसलिए वह इसे फिर से कॉल करता है। और फिर से। और फिर से, जब तक आप इसे रोकते नहीं हैं। आशा है कि आपका संदर्भ बहुत बड़ा नहीं था, और आपका मॉडल बहुत महंगा नहीं था!

उपकरण उपयोग की सर्वोत्तम प्रथाएं

उपकरण के उपयोग से अधिकतम लाभ प्राप्त करने के लिए, निम्नलिखित सर्वोत्तम प्रथाओं पर विचार करें।

वर्णनात्मक परिभाषाएं

प्रत्येक उपकरण और उसके इनपुट पैरामीटर्स के लिए स्पष्ट और वर्णनात्मक नाम और विवरण प्रदान करें। यह LLM को प्रत्येक उपकरण के उद्देश्य और क्षमताओं को बेहतर ढंग से समझने में मदद करता है।

> मैं अनुभव से बता सकता हूं कि “नामकरण कठिन है” की सामान्य समझ यहां भी लागू होती है; मैंने देखा है कि केवल फ़ंक्शन के नाम या विवरण के शब्दों को बदलने से LLM के परिणामों में नाटकीय अंतर आ जाता है। कभी-कभी विवरण को हटाने से प्रदर्शन में सुधार होता है।

उपकरण परिणामों का प्रसंस्करण

जब LLM को उपकरण के परिणाम वापस भेजे जा रहे हों, तो सुनिश्चित करें कि वे अच्छी तरह से संरचित और व्यापक हों। प्रत्येक उपकरण के आउटपुट को दर्शाने के लिए सार्थक कुंजियों और मानों का उपयोग करें। विभिन्न प्रारूपों के साथ प्रयोग करें और देखें कि कौन सा सबसे अच्छा काम करता है, JSON से लेकर सादे टेक्स्ट तक।

Result Interpreter इस चुनौती का समाधान परिणामों का विश्लेषण करने और मानव-मित्रवत व्याख्याएं, सारांश, या प्रमुख निष्कर्ष प्रदान करने के लिए AI का उपयोग करके करता है।

त्रुटि प्रबंधन

मजबूत त्रुटि प्रबंधन तंत्र लागू करें जो उन मामलों को संभाल सकें जहां LLM उपकरण कॉल के लिए अमान्य या असमर्थित इनपुट पैरामीटर उत्पन्न कर सकता है। उपकरण निष्पादन के दौरान होने वाली किसी भी त्रुटि को सुचारू रूप से संभालें और उससे उबरें।

AI की एक बेहद अच्छी विशेषता यह है कि यह त्रुटि संदेशों को समझता है! जिसका मतलब है कि यदि आप त्वरित और सरल दृष्टिकोण से काम कर रहे हैं, तो आप किसी उपकरण के कार्यान्वयन में उत्पन्न होने वाले किसी भी अपवाद को पकड़ सकते हैं, और इसे AI को वापस भेज सकते हैं ताकि उसे पता चल सके कि क्या हुआ!

उदाहरण के लिए, यहाँ Olympia में google खोज के कार्यान्वयन का एक संक्षिप्त संस्करण है:

```ruby
  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
```

Olympia में गूगल खोज एक दो-चरणीय प्रक्रिया है। पहले आप खोज करते हैं, फिर परिणामों का सारांश करते हैं। यदि कोई विफलता होती है, चाहे वह कुछ भी हो, त्रुटि संदेश को पैकेज किया जाता है और AI को वापस भेज दिया जाता है। यह तकनीक लगभग सभी बुद्धिमान त्रुटि प्रबंधन पैटर्न की आधारशिला है।

उदाहरण के लिए, मान लीजिए कि GoogleSearch API कॉल 503 सेवा अनुपलब्ध त्रुटि के कारण विफल हो जाती है। यह शीर्ष-स्तरीय रेस्क्यू तक पहुंचती है, और त्रुटि का विवरण फ़ंक्शन कॉल के परिणाम के रूप में AI को वापस भेज दिया जाता है। उपयोगकर्ता को केवल एक खाली स्क्रीन या तकनीकी त्रुटि देने के बजाय, AI कुछ ऐसा कहता है जैसे "मैं क्षमा चाहता हूं, लेकिन मैं इस समय अपनी गूगल खोज क्षमताओं तक पहुंचने में असमर्थ हूं। यदि आप चाहें तो मैं बाद में पुनः प्रयास कर सकता हूं।"

यह केवल एक चतुर युक्ति की तरह लग सकती है, लेकिन एक अलग प्रकार की त्रुटि पर विचार करें, जहां AI किसी बाहरी API को कॉल कर रहा था और API को पास करने के लिए पैरामीटर्स पर सीधा नियंत्रण था। शायद उसने उन पैरामीटर्स को जनरेट करने में कोई गलती की? बशर्ते कि बाहरी API से त्रुटि संदेश पर्याप्त विस्तृत हो, त्रुटि संदेश को कॉलिंग AI को वापस भेजने का मतलब है कि वह उन पैरामीटर्स पर पुनर्विचार कर सकता है और फिर से प्रयास कर सकता है। स्वचालित रूप से। चाहे त्रुटि कुछ भी हो।

अब सोचिए कि सामान्य कोड में उस प्रकार के मजबूत त्रुटि प्रबंधन को दोहराने के लिए क्या आवश्यक होगा। यह लगभग असंभव है।

पुनरावर्ती परिष्करण

यदि LLM उपयुक्त टूल्स की सिफारिश नहीं कर रहा है या अनुकूल प्रतिक्रियाएं नहीं दे रहा है, तो टूल परिभाषाओं, विवरणों और इनपुट पैरामीटर्स पर पुनः कार्य करें। देखे गए व्यवहार और वांछित परिणामों के आधार पर टूल सेटअप को निरंतर परिष्कृत और सुधारें।

1. सरल टूल परिभाषाओं से शुरू करें: स्पष्ट और संक्षिप्त नामों, विवरणों और इनपुट पैरामीटर्स के साथ टूल्स को परिभाषित करना शुरू करें। प्रारंभ में टूल सेटअप को अधिक जटिल बनाने से बचें और मुख्य कार्यक्षमता पर ध्यान केंद्रित करें। उदाहरण के लिए, यदि आप भावना विश्लेषण के परिणामों को सहेजना चाहते हैं, तो एक बुनियादी परिभाषा से शुरू करें जैसे:

```
{
  "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. परीक्षण और निरीक्षण करें: एक बार जब आप प्रारंभिक टूल परिभाषाएँ स्थापित कर लें, तो विभिन्न प्रॉम्प्ट्स के साथ उनका परीक्षण करें और देखें कि एलएलएम टूल के साथ कैसे संवाद करता है। उत्पन्न प्रतिक्रियाओं की गुणवत्ता और प्रासंगिकता पर ध्यान दें। यदि एलएलएम अनुकूलतम से कम प्रतिक्रियाएँ उत्पन्न कर रहा है, तो टूल परिभाषाओं को परिष्कृत करने का समय आ गया है।

3. विवरणों को परिष्कृत करें: यदि एलएलएम किसी टूल के उद्देश्य को गलत समझ रहा है, तो टूल के विवरण को परिष्कृत करने का प्रयास करें। टूल के प्रभावी उपयोग में एलएलएम का मार्गदर्शन करने के लिए अधिक संदर्भ, उदाहरण, या स्पष्टीकरण प्रदान करें। उदाहरण के लिए, आप भावना विश्लेषण टूल के विवरण को विश्लेषण किए जा रहे पाठ के भावनात्मक स्वर को अधिक विशिष्ट रूप से संबोधित करने के लिए अपडेट कर सकते हैं:

```
{
  "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. इनपुट पैरामीटर्स समायोजित करें: यदि एलएलएम किसी टूल के लिए अमान्य या अप्रासंगिक इनपुट पैरामीटर्स उत्पन्न कर रहा है, तो पैरामीटर परिभाषाओं को समायोजित करने पर विचार करें। अपेक्षित इनपुट प्रारूप को स्पष्ट करने के लिए अधिक विशिष्ट प्रतिबंध, सत्यापन नियम, या उदाहरण जोड़ें।
5. प्रतिक्रिया के आधार पर सुधार करें: अपने टूल्स के प्रदर्शन की निरंतर निगरानी करें और उपयोगकर्ताओं या हितधारकों से प्रतिक्रिया एकत्र करें। सुधार के क्षेत्रों की पहचान करने और टूल परिभाषाओं में क्रमिक सुधार करने के लिए इस प्रतिक्रिया का उपयोग करें। उदाहरण के लिए, यदि उपयोगकर्ता बताते हैं कि विश्लेषण व्यंग्य को सही ढंग से नहीं संभाल रहा है, तो आप विवरण में एक नोट जोड़ सकते हैं:

```
{
  "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.",
  ...
}
```

अपनी उपकरण परिभाषाओं को देखे गए व्यवहार और प्रतिक्रिया के आधार पर क्रमिक रूप से परिष्कृत करके, आप अपने AI-संचालित एप्लिकेशन के प्रदर्शन और प्रभावशीलता में क्रमिक सुधार कर सकते हैं। उपकरण परिभाषाओं को स्पष्ट, संक्षिप्त और विशिष्ट कार्य पर केंद्रित रखना याद रखें। नियमित रूप से उपकरण इंटरैक्शन का परीक्षण और सत्यापन करें ताकि यह सुनिश्चित हो सके कि वे आपके वांछित परिणामों के अनुरूप हैं।

उपकरणों का संयोजन और श्रृंखलाबद्ध करना

उपकरण उपयोग का एक सबसे शक्तिशाली पहलू, जिसका अब तक केवल संकेत दिया गया है, वह है जटिल कार्यों को पूरा करने के लिए कई उपकरणों को एक साथ संयोजित और श्रृंखलाबद्ध करने की क्षमता। अपनी उपकरण परिभाषाओं और उनके इनपुट/आउटपुट प्रारूपों को सावधानीपूर्वक डिज़ाइन करके, आप पुन: प्रयोज्य बिल्डिंग ब्लॉक बना सकते हैं जिन्हें विभिन्न तरीकों से संयोजित किया जा सकता है।

आइए एक उदाहरण पर विचार करें जहाँ आप अपने AI-संचालित एप्लिकेशन के लिए एक डेटा विश्लेषण पाइपलाइन बना रहे हैं। आपके पास निम्नलिखित उपकरण हो सकते हैं:

1. **DataRetrieval:** एक उपकरण जो निर्दिष्ट मानदंडों के आधार पर डेटाबेस या API से डेटा प्राप्त करता है।
2. **DataProcessing:** एक उपकरण जो प्राप्त डेटा पर गणना, रूपांतरण, या एकत्रीकरण करता है।

3. **DataVisualization:** एक उपकरण जो प्रसंस्कृत डेटा को चार्ट या ग्राफ़ जैसे उपयोगकर्ता-अनुकूल प्रारूप में प्रस्तुत करता है।

इन उपकरणों को एक साथ श्रृंखलाबद्ध करके, आप एक शक्तिशाली कार्यप्रवाह बना सकते हैं जो प्रासंगिक डेटा प्राप्त करता है, उसे प्रोसेस करता है, और परिणामों को सार्थक तरीके से प्रस्तुत करता है। यहाँ उपकरण उपयोग कार्यप्रवाह कैसा दिख सकता है:

1. LLM को एक विशिष्ट उत्पाद श्रेणी के लिए बिक्री डेटा पर अंतर्दृष्टि मांगने वाला उपयोगकर्ता प्रश्न प्राप्त होता है।
2. LLM DataRetrieval उपकरण का चयन करता है और डेटाबेस से प्रासंगिक बिक्री डेटा प्राप्त करने के लिए उपयुक्त इनपुट पैरामीटर जनरेट करता है।
3. प्राप्त डेटा को DataProcessing उपकरण को "पास" किया जाता है, जो कुल राजस्व, औसत बिक्री मूल्य और वृद्धि दर जैसे मैट्रिक्स की गणना करता है।
4. प्रसंस्कृत डेटा को फिर DataVisualization उपकरण द्वारा समझा जाता है, जो अंतर्दृष्टि को दर्शाने के लिए एक आकर्षक चार्ट या ग्राफ़ बनाता है, और चार्ट का URL LLM को वापस पास करता है।
5. अंत में, LLM मार्कडाउन का उपयोग करके उपयोगकर्ता प्रश्न का एक स्वरूपित उत्तर जनरेट करता है, जिसमें विज़ुअलाइज़ किया गया डेटा और प्रमुख निष्कर्षों का सारांश शामिल होता है।

इन उपकरणों को एक साथ संयोजित करके, आप एक सहज डेटा विश्लेषण कार्यप्रवाह बना सकते हैं जिसे आसानी से आपके एप्लिकेशन में एकीकृत किया जा सकता है। इस दृष्टिकोण की खूबसूरती यह है कि प्रत्येक उपकरण को स्वतंत्र रूप से विकसित और परीक्षण किया जा सकता है, और फिर विभिन्न समस्याओं को हल करने के लिए अलग-अलग तरीकों से संयोजित किया जा सकता है।

उपकरणों के सुचारू संयोजन और श्रृंखलाबद्धता को सक्षम करने के लिए, प्रत्येक उपकरण के लिए स्पष्ट इनपुट और आउटपुट प्रारूपों को परिभाषित करना महत्वपूर्ण है।

उदाहरण के लिए, DataRetrieval उपकरण डेटाबेस कनेक्शन विवरण, टेबल नाम, और क्वेरी शर्तों जैसे पैरामीटर स्वीकार कर सकता है, और परिणाम को एक संरचित JSON ऑब्जेक्ट के रूप में वापस कर सकता है। DataProcessing उपकरण फिर इस JSON ऑब्जेक्ट को इनपुट के रूप में स्वीकार कर सकता है और एक परिवर्तित JSON ऑब्जेक्ट को आउटपुट के रूप में उत्पन्न कर सकता है। उपकरणों के बीच डेटा प्रवाह को मानकीकृत करके, आप संगतता और पुन: प्रयोज्यता सुनिश्चित कर सकते हैं।

जैसे-जैसे आप अपने उपकरण पारिस्थितिकी तंत्र को डिजाइन करते हैं, इस बारे में सोचें कि विभिन्न उपकरणों को आपके एप्लिकेशन में सामान्य उपयोग मामलों को संबोधित करने के लिए कैसे संयोजित किया जा सकता है। ऐसे उच्च-स्तरीय उपकरण बनाने पर विचार करें जो सामान्य कार्यप्रवाह या व्यावसायिक तर्क को एनकैप्सुलेट करते हैं, जिससे LLM के लिए उन्हें चुनना और प्रभावी ढंग से उपयोग करना आसान हो जाता है।

याद रखें, उपकरण उपयोग की शक्ति इसकी लचीलेपन और मॉड्यूलरता में निहित है। जटिल कार्यों को छोटे, पुन: प्रयोज्य उपकरणों में विभाजित करके, आप एक मजबूत और अनुकूलनीय AI-संचालित एप्लिकेशन बना सकते हैं जो विभिन्न प्रकार की चुनौतियों का सामना कर सकता है।

भविष्य की दिशाएं

जैसे-जैसे AI-संचालित एप्लिकेशन विकास का क्षेत्र विकसित होता है, हम उपकरण उपयोग क्षमताओं में और अधिक प्रगति की उम्मीद कर सकते हैं। कुछ संभावित भविष्य की दिशाएं इस प्रकार हैं:

1. **बहु-चरणीय उपकरण उपयोग**: LLM यह तय कर सकते हैं कि संतोषजनक प्रतिक्रिया उत्पन्न करने के लिए उन्हें कितनी बार उपकरणों का उपयोग करने की आवश्यकता है। इसमें मध्यवर्ती परिणामों के आधार पर उपकरण चयन और निष्पादन के कई दौर शामिल हो सकते हैं।

2. **पूर्व-परिभाषित उपकरण**: AI प्लेटफ़ॉर्म पूर्व-परिभाषित उपकरणों का एक सेट प्रदान कर सकते हैं जिनका डेवलपर्स सीधे उपयोग कर सकते हैं, जैसे Python इंटरप्रेटर, वेब खोज उपकरण, या सामान्य उपयोगिता फ़ंक्शन।
3. **निर्बाध एकीकरण**: जैसे-जैसे उपकरण उपयोग अधिक प्रचलित होता जाएगा, हम AI प्लेटफ़ॉर्म और लोकप्रिय विकास ढांचों के बीच बेहतर एकीकरण की उम्मीद कर सकते हैं, जिससे डेवलपर्स के लिए अपने एप्लिकेशन में उपकरण उपयोग को शामिल करना आसान हो जाएगा।

उपकरण उपयोग एक शक्तिशाली तकनीक है जो डेवलपर्स को AI-संचालित एप्लिकेशन में LLM की पूर्ण क्षमता का उपयोग करने में सक्षम बनाती है। LLM को बाहरी उपकरणों और संसाधनों से जोड़कर, आप अधिक गतिशील, बुद्धिमान और संदर्भ-जागरूक सिस्टम बना सकते हैं जो उपयोगकर्ता की जरूरतों के अनुकूल हो सकते हैं और मूल्यवान अंतर्दृष्टि और कार्रवाई प्रदान कर सकते हैं।

हालांकि उपकरण उपयोग अपार संभावनाएं प्रदान करता है, संभावित चुनौतियों और विचारणीय बिंदुओं के प्रति सचेत रहना महत्वपूर्ण है। एक प्रमुख पहलू उपकरण अंतःक्रियाओं की जटिलता का प्रबंधन करना और समग्र सिस्टम की स्थिरता और विश्वसनीयता सुनिश्चित करना है। आपको ऐसी परिस्थितियों को संभालना होगा जहां उपकरण कॉल विफल हो सकते हैं, अप्रत्याशित परिणाम दे सकते हैं, या प्रदर्शन पर प्रभाव डाल सकते हैं। इसके अतिरिक्त, आपको उपकरणों के अनधिकृत या दुर्भावनापूर्ण उपयोग को रोकने के लिए सुरक्षा और पहुंच नियंत्रण उपायों पर विचार करना चाहिए। आपके AI-संचालित एप्लिकेशन की अखंडता और प्रदर्शन को बनाए रखने के लिए उचित त्रुटि प्रबंधन, लॉगिंग और निगरानी तंत्र महत्वपूर्ण हैं।

जैसे-जैसे आप अपनी परियोजनाओं में उपकरण के उपयोग की संभावनाओं का पता लगाते हैं, याद रखें कि स्पष्ट उद्देश्यों से शुरुआत करें, सुव्यवस्थित उपकरण परिभाषाएं तैयार करें, और प्रतिक्रिया एवं परिणामों के आधार पर पुनरावृत्ति करें। सही दृष्टिकोण और मानसिकता के साथ, उपकरण का उपयोग आपके एआई-संचालित एप्लिकेशन में नवाचार और मूल्य के नए स्तरों को खोल सकता है

स्ट्रीम प्रोसेसिंग

HTTP पर डेटा स्ट्रीमिंग, जिसे सर्वर-भेजी गई घटनाएं (SSE) भी कहा जाता है, एक ऐसी प्रणाली है जहां सर्वर क्लाइंट को लगातार डेटा भेजता रहता है जैसे ही वह उपलब्ध होता है, बिना क्लाइंट को विशेष रूप से अनुरोध करने की आवश्यकता के। चूंकि AI की प्रतिक्रिया क्रमिक रूप से उत्पन्न होती है, इसलिए AI के आउटपुट को उत्पन्न होते ही प्रदर्शित करके एक प्रतिक्रियाशील उपयोगकर्ता अनुभव प्रदान करना तार्किक है। और वास्तव में मुझे ज्ञात सभी AI प्रदाता APIs अपने पूर्णता एंडपॉइंट्स में स्ट्रीमिंग प्रतिक्रियाओं को एक विकल्प के रूप में प्रदान करते हैं।

इस पुस्तक में यह अध्याय यहां, टूल्स का उपयोग के ठीक बाद इसलिए आता है क्योंकि उपयोगकर्ताओं को लाइव AI प्रतिक्रियाओं के साथ टूल्स के उपयोग को जोड़ना कितना शक्तिशाली हो सकता है। ऐसा करने से गतिशील और इंटरैक्टिव अनुभव संभव होते हैं जहां AI उपयोगकर्ता इनपुट को प्रोसेस कर सकता है, अपने विवेक से विभिन्न टूल्स और फंक्शंस का उपयोग कर सकता है, और फिर रीयल-टाइम प्रतिक्रियाएं प्रदान कर सकता है।

इस निर्बाध इंटरैक्शन को प्राप्त करने के लिए, आपको स्ट्रीम हैंडलर्स लिखने की आवश्यकता होती है जो AI-इनवोक्ड टूल फंक्शन कॉल्स के साथ-साथ सादे टेक्स्ट आउटपुट को अंतिम उपयोगकर्ता को भेज सकें। एक टूल फंक्शन को प्रोसेस करने के बाद लूप करने की आवश्यकता काम को एक दिलचस्प चुनौती बना देती है।

ReplyStream का कार्यान्वयन

स्ट्रीम प्रोसेसिंग को कैसे लागू किया जा सकता है, यह दिखाने के लिए, यह अध्याय Olympia में उपयोग की जाने वाली ReplyStream क्लास के एक सरलीकृत संस्करण की गहन समीक्षा करेगा। इस क्लास के उदाहरणों को ruby-openai और openrouter जैसी AI क्लाइंट लाइब्रेरीज में stream पैरामीटर के रूप में पास किया जा सकता है।

यहाँ बताया गया है कि मैं Olympia के PromptSubscriber में ReplyStream का उपयोग कैसे करता हूं, जो Wisper के माध्यम से नए उपयोगकर्ता संदेशों के निर्माण को सुनता है।

```
1  class PromptSubscriber
2    include Raix::ChatCompletion
3    include Raix::PromptDeclarations
4
5    # many other declarations omitted...
6
7    prompt text: -> { user_message.content },
8           stream: -> { ReplyStream.new(self) },
9           until: -> { bot_message.complete? }
10
11   def message_created(message) # invoked by Wisper
12     return unless message.role.user? && message.content?
13
14     # rest of the implementation omitted...
```

context संकेत के अतिरिक्त, जो उस प्रॉम्प्ट सब्सक्राइबर को संदर्भित करता है जिसने इसे आरंभ किया, ReplyStream क्लास में प्राप्त डेटा का बफ़र संग्रहित करने के लिए इंस्टेंस वेरिएबल्स होते हैं, और स्ट्रीम प्रोसेसिंग के दौरान उपयोग किए गए फ़ंक्शन नेम्स और आर्गुमेंट्स को ट्रैक करने के लिए ऐरे भी होते हैं।

```ruby
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 विधि ReplyStream इंस्टेंस की प्रारंभिक स्थिति को स्थापित करती है, जिसमें बफ़र, संदर्भ और अन्य चरों को आरंभ किया जाता है।

call विधि स्ट्रीमिंग डेटा को संसाधित करने का मुख्य प्रवेश बिंदु है। यह डेटा के एक 'खंड' (जो एक हैश के रूप में दर्शाया गया है) और एक वैकल्पिक bytesize पैरामीटर लेती है, जो हमारे उदाहरण में अप्रयुक्त है। इस विधि के अंदर, क्लास प्राप्त खंड की संरचना के आधार पर विभिन्न परिदृश्यों को संभालने के लिए पैटर्न मैचिंग का उपयोग करती है।

खंड पर deep_symbolize_keys को कॉल करने से पैटर्न मैचिंग अधिक सुरुचिपूर्ण हो जाती है, क्योंकि यह हमें स्ट्रिंग्स के बजाय सिंबल्स पर काम करने की अनुमति देता है।

```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
```

पहला पैटर्न जिसका हम मिलान कर रहे हैं वह है एक टूल कॉल और उससे जुड़े फ़ंक्शन नेम का। यदि हमें यह मिलता है, तो हम इसे f_name ऐरे में संग्रहित कर लेते हैं। हम फ़ंक्शन नेम्स को एक इंडेक्स्ड ऐरे में स्टोर करते हैं, क्योंकि मॉडल समानांतर फ़ंक्शन कॉलिंग में सक्षम है, जो एक साथ एक से अधिक फ़ंक्शन को एक्ज़ीक्यूट करने के लिए भेज सकता है।

> समानांतर फ़ंक्शन कॉलिंग एक AI मॉडल की वह क्षमता है जिससे वह एक साथ कई फ़ंक्शन कॉल कर सकता है, जिससे इन फ़ंक्शन कॉल्स के प्रभावों और परिणामों को समानांतर रूप से हल किया जा सकता है। यह विशेष रूप से उपयोगी है यदि फ़ंक्शन्स को पूरा होने में लंबा समय लगता है, और यह API के साथ राउंड ट्रिप्स को कम करता है, जो बदले में टोकन खर्च की एक महत्वपूर्ण मात्रा को बचा सकता है।

अगला हमें फ़ंक्शन कॉल्स के संबंधित आर्ग्युमेंट्स के लिए मिलान करने की आवश्यकता है।

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

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

जिस तरह हमने फ़ंक्शन नेम्स को संभाला, उसी तरह हम आर्गुमेंट्स को एक इंडेक्स्ड ऐरे में रख देते हैं।

इसके बाद, हम सामान्य उपयोगकर्ता-सामने के संदेशों की तलाश करते हैं, जो सर्वर से एक-एक टोकन के रूप में आएंगे और new_content वेरिएबल को असाइन किए जाएंगे। हमें finish_reason पर भी ध्यान रखना होगा। यह आउटपुट सीक्वेंस के अंतिम भाग तक nil रहेगा।

```
  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
```

महत्वपूर्ण रूप से, हम एआई मॉडल प्रोवाइडर द्वारा भेजे गए त्रुटि संदेशों को संभालने

के लिए एक पैटर्न मैच एक्सप्रेशन जोड़ते हैं। स्थानीय विकास वातावरण में, हम एक अपवाद उठाते हैं, लेकिन प्रोडक्शन में, हम त्रुटि को लॉग करते हैं और समाप्त कर देते हैं।

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

case का अंतिम else clause तब निष्पादित होगा जब कोई भी पिछला पैटर्न मैच नहीं करता। यह बस एक सुरक्षा उपाय है ताकि अगर एआई मॉडल हमें अपरिचित खंड भेजना शुरू कर दे तो हमें इसके बारे में पता चल जाए।

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

send_to_client विधि बफ़र्ड सामग्री को क्लाइंट को भेजने के लिए जिम्मेदार है। यह जांचता है कि बफ़र खाली नहीं है, बॉट संदेश की सामग्री को अपडेट करता है, बॉट संदेश को रेंडर करता है, और डेटा स्थायित्व सुनिश्चित करने के लिए सामग्री को डेटाबेस में सहेजता है।

```ruby
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 मेथड को तब कॉल किया जाता है जब स्ट्रीम प्रोसेसिंग पूरी हो जाती है। यह फ़ंक्शन कॉल्स को डिस्पैच करता है यदि स्ट्रीम के दौरान कोई कॉल्स प्राप्त हुए थे, बॉट संदेश को अंतिम कंटेंट और अन्य प्रासंगिक जानकारी के साथ अपडेट करता है, और फ़ंक्शन कॉल हिस्ट्री को रीसेट करता है

```ruby
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
```

यदि मॉडल किसी फ़ंक्शन को कॉल करने का निर्णय लेता है, तो आपको उस फ़ंक्शन

कॉल (नाम और आर्गुमेंट्स) को इस तरह से "प्रेषित" करना होगा कि वह निष्पादित हो और वार्तालाप प्रतिलेख में function_call और function_result संदेश जुड़ जाएं।

मेरे अनुभव में, टूल कार्यान्वयन पर निर्भर रहने के बजाय अपने कोडबेस में एक ही स्थान पर फ़ंक्शन संदेशों की रचना को संभालना बेहतर होता है। यह न केवल अधिक स्वच्छ है, बल्कि इसका एक बहुत महत्वपूर्ण व्यावहारिक कारण भी है: यदि एआई मॉडल किसी फ़ंक्शन को कॉल करता है, और लूप में वापस आने पर प्रतिलेख में कॉल और परिणाम संदेश नहीं देखता है, तो वह उसी फ़ंक्शन को फिर से कॉल करेगा। संभवतः यह हमेशा के लिए चलता रहेगा। याद रखें कि एआई पूरी तरह से स्टेटलेस होता है, इसलिए जब तक आप उन फ़ंक्शन कॉल्स को वापस इसे नहीं दिखाते, वे घटित ही नहीं हुए।

```ruby
# 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
```

फ़ंक्शन कॉल को डिस्पैच करने के बाद कॉल हिस्ट्री को साफ़ करना उतना ही महत्वपूर्ण है जितना यह सुनिश्चित करना कि कॉल और परिणाम आपके प्रतिलेख में दर्ज हों, ताकि आप हर बार लूप करते समय एक ही फ़ंक्शन्स को बार-बार न कॉल करते रहें।

"वार्तालाप लूप"

मैं लूपिंग का जिक्र करता रहता हूं, लेकिन अगर आप फ़ंक्शन कॉलिंग में नए हैं, तो यह स्पष्ट नहीं हो सकता कि हमें लूप की आवश्यकता क्यों है। कारण यह है कि एक

बार जब AI आपसे अपनी ओर से टूल फ़ंक्शन्स को निष्पादित करने के लिए "पूछता" है, तो वह जवाब देना बंद कर देगा। उन फ़ंक्शन्स को निष्पादित करना, परिणामों को एकत्र करना, प्रतिलेख में परिणामों को जोड़ना, और फिर नए फ़ंक्शन कॉल्स या उपयोगकर्ता-केंद्रित परिणाम प्राप्त करने के लिए मूल प्रॉम्प्ट को फिर से सबमिट करना आपकी जिम्मेदारी है।

PromptSubscriber क्लास में, हम PromptDeclarations मॉड्यूल से prompt मेथड का उपयोग वार्तालाप लूप के व्यवहार को परिभाषित करने के लिए करते हैं। until पैरामीटर को -> { bot_message.complete? } पर सेट किया गया है, जिसका अर्थ है कि लूप तब तक जारी रहेगा जब तक bot_message को पूर्ण के रूप में चिह्नित नहीं किया जाता।

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

लेकिन bot_message को पूर्ण कब चिह्नित किया जाता है? यदि आप भूल गए हैं, तो finalize मेथड की लाइन 13 को फिर से देखें।

आइए पूरी स्ट्रीम प्रोसेसिंग लॉजिक की समीक्षा करें।

1. PromptSubscriber एक नया उपयोगकर्ता संदेश message_created मेथड के माध्यम से प्राप्त करता है, जो Wisper पब/सब सिस्टम द्वारा हर बार तब चलाया जाता है जब अंतिम उपयोगकर्ता एक नया प्रॉम्प्ट बनाता है।
2. prompt क्लास मेथड PromptSubscriber के लिए चैट पूर्णता लॉजिक के व्यवहार को घोषणात्मक रूप से परिभाषित करता है। AI मॉडल उपयोगकर्ता के संदेश सामग्री के साथ एक चैट पूर्णता निष्पादित करेगा, स्ट्रीम पैरामीटर के रूप में ReplyStream का एक नया इंस्टेंस, और निर्दिष्ट लूप कंडीशन के साथ।
3. AI मॉडल प्रॉम्प्ट को प्रोसेस करता है और एक प्रतिक्रिया जनरेट करना शुरू करता है। जैसे-जैसे प्रतिक्रिया स्ट्रीम की जाती है, ReplyStream इंस्टेंस का call मेथड प्रत्येक डेटा चंक के लिए चलाया जाता है।

4. यदि AI मॉडल किसी टूल फंक्शन को कॉल करने का निर्णय लेता है, तो फंक्शन नाम और आर्गुमेंट्स को चंक से निकालकर क्रमशः f_name और f_arguments ऐरे में स्टोर किया जाता है।
5. यदि AI मॉडल उपयोगकर्ता के लिए सामग्री जनरेट करता है, तो इसे बफर किया जाता है और send_to_client मेथड के माध्यम से क्लाइंट को भेजा जाता है।
6. एक बार स्ट्रीम प्रोसेसिंग पूरी हो जाने के बाद, finalize मेथड को कॉल किया जाता है। यदि स्ट्रीम के दौरान कोई टूल फंक्शन चलाए गए थे, तो उन्हें PromptSubscriber के dispatch मेथड का उपयोग करके डिस्पैच किया जाता है।
7. dispatch मेथड वार्तालाप प्रतिलेख में एक function_call संदेश जोड़ता है, संबंधित टूल फंक्शन को निष्पादित करता है, और फंक्शन कॉल के परिणाम के साथ प्रतिलेख में एक function_result संदेश जोड़ता है।
8. टूल फंक्शन्स को डिस्पैच करने के बाद, बाद के लूप्स में डुप्लिकेट फंक्शन कॉल्स को रोकने के लिए फंक्शन कॉल हिस्ट्री को क्लियर कर दिया जाता है।
9. यदि कोई टूल फंक्शन नहीं चलाया गया था, तो finalize मेथड अंतिम सामग्री के साथ bot_message को अपडेट करता है, इसे पूर्ण के रूप में चिह्नित करता है, और अपडेट किए गए संदेश को क्लाइंट को भेजता है।
10. लूप कंडीशन -> { bot_message.complete? } का मूल्यांकन किया जाता है। यदि bot_message को पूर्ण के रूप में चिह्नित नहीं किया गया है, तो लूप जारी रहता है, और मूल प्रॉम्प्ट को अपडेट किए गए वार्तालाप प्रतिलेख के साथ फिर से सबमिट किया जाता है।
11. चरण 3-10 तब तक दोहराए जाते हैं जब तक bot_message को पूर्ण के रूप में चिह्नित नहीं किया जाता, जो यह दर्शाता है कि AI मॉडल ने अपनी प्रतिक्रिया उत्पन्न करना समाप्त कर दिया है और किसी अन्य टूल फ़ंक्शन को निष्पादित करने की आवश्यकता नहीं है।

इस वार्तालाप लूप को लागू करके, आप AI मॉडल को एप्लिकेशन के साथ आगे-पीछे की बातचीत में संलग्न होने में सक्षम बनाते हैं, जहां आवश्यकतानुसार टूल फ़ंक्शन्स का निष्पादन किया जाता है और वार्तालाप के स्वाभाविक समापन तक उपयोगकर्ता के लिए प्रतिक्रियाएं उत्पन्न की जाती हैं।

स्ट्रीम प्रोसेसिंग और वार्तालाप लूप का संयोजन गतिशील और इंटरैक्टिव AI-संचालित अनुभवों को संभव बनाता है, जहां AI मॉडल उपयोगकर्ता इनपुट को संसाधित कर सकता है, विभिन्न टूल्स और फ़ंक्शन्स का उपयोग कर सकता है, और विकसित होते वार्तालाप संदर्भ के आधार पर रीयल-टाइम प्रतिक्रियाएं प्रदान कर सकता है।

स्वचालित निरंतरता

यह AI आउटपुट की सीमाओं के बारे में जागरूक रहना महत्वपूर्ण है। अधिकांश मॉडलों में अधिकतम टोकन की एक सीमा होती है जो वे एक एकल प्रतिक्रिया में उत्पन्न कर सकते हैं, जो max_tokens पैरामीटर द्वारा निर्धारित की जाती है। यदि AI मॉडल प्रतिक्रिया उत्पन्न करते समय इस सीमा तक पहुंच जाता है, तो यह अचानक रुक जाएगा और संकेत देगा कि आउटपुट को काटा गया है।

AI प्लेटफ़ॉर्म API से स्ट्रीमिंग प्रतिक्रिया में, आप चंक में finish_reason वेरिएबल की जांच करके इस स्थिति का पता लगा सकते हैं। यदि finish_reason को "length" (या मॉडल के लिए विशिष्ट कोई अन्य की वैल्यू) पर सेट किया गया है, तो इसका मतलब है कि मॉडल जनरेशन के दौरान अपनी अधिकतम टोकन सीमा तक पहुंच गया और आउटपुट को छोटा कर दिया गया है।

इस परिदृश्य को सुचारू रूप से संभालने और एक निर्बाध उपयोगकर्ता अनुभव प्रदान करने का एक तरीका है, अपनी स्ट्रीम प्रोसेसिंग लॉजिक में एक स्वचालित निरंतरता तंत्र को लागू करना। लंबाई-संबंधित समाप्ति कारणों के लिए एक पैटर्न मैच जोड़कर, आप लूप करने और स्वचालित रूप से आउटपुट को जहां छोड़ा था वहां से जारी रखने का विकल्प चुन सकते हैं।

यहां एक जानबूझकर सरलीकृत उदाहरण दिया गया है कि आप स्वचालित निरंतरता का समर्थन करने के लिए ReplyStream क्लास में call विधि को कैसे संशोधित कर सकते हैं:

```
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
```

इस संशोधित संस्करण में, जब finish_reason काटे गए आउटपुट को इंगित करता है, तब स्ट्रीम को अंतिम रूप देने के बजाय, हम ट्रांसक्रिप्ट में बिना अंतिम रूप दिए एक जोड़ी संदेश जोड़ते हैं, मूल उपयोगकर्ता-सामने वाले प्रतिक्रिया संदेश को उसके created_at विशेषता को अपडेट करके ट्रांसक्रिप्ट के "नीचे" ले जाते हैं, और फिर लूप को होने देते हैं, ताकि एआई जहां छोड़ा था वहां से जारी रख सके।

याद रखें कि एआई पूर्णता एंडपॉइंट स्टेटलेस है। यह केवल वही "जानता" है जो

आप इसे ट्रांसक्रिप्ट के माध्यम से बताते हैं। इस मामले में, हम एआई को यह बताने का तरीका कि यह कट ऑफ हो गया था, ट्रांसक्रिप्ट में "अदृश्य" (अंतिम उपयोगकर्ता के लिए) संदेश जोड़कर करते हैं। हालांकि, याद रखें कि यह जानबूझकर सरलीकृत उदाहरण है। एक वास्तविक कार्यान्वयन को यह सुनिश्चित करने के लिए आगे की ट्रांसक्रिप्ट प्रबंधन करने की आवश्यकता होगी कि हमने टोकन बर्बाद नहीं किए और/या ट्रांसक्रिप्ट में दोहराए गए सहायक संदेशों के साथ एआई को भ्रमित नहीं किया।

स्वचालित-निरंतरता का एक वास्तविक कार्यान्वयन में तथाकथित "सर्किट ब्रेकर लॉजिक" भी होनी चाहिए जो अनियंत्रित लूपिंग को रोकने के लिए है। कारण यह है कि, कुछ प्रकार के उपयोगकर्ता प्रॉम्प्ट्स और कम max_tokens सेटिंग्स के साथ, एआई उपयोगकर्ता-सामने के आउटपुट को अनंत काल तक लूप कर सकता है।

> ध्यान रखें कि प्रत्येक लूप के लिए एक अलग अनुरोध की आवश्यकता होती है, और प्रत्येक अनुरोध आपके पूरे ट्रांसक्रिप्ट का फिर से उपयोग करता है। आपको अपने एप्लिकेशन में स्वचालित निरंतरता को लागू करने का निर्णय लेते समय उपयोगकर्ता अनुभव और एपीआई उपयोग के बीच ट्रेड-ऑफ पर निश्चित रूप से विचार करना चाहिए। विशेष रूप से स्वचालित-निरंतरता खतरनाक रूप से महंगी हो सकती है, खासकर जब प्रीमियम वाणिज्यिक मॉडल का उपयोग किया जा रहा हो।

निष्कर्ष

स्ट्रीम प्रोसेसिंग टूल उपयोग को लाइव एआई प्रतिक्रियाओं के साथ जोड़ने वाले एआई-संचालित एप्लिकेशन बनाने का एक महत्वपूर्ण पहलू है। एआई प्लेटफॉर्म एपीआई से स्ट्रीमिंग डेटा को कुशलतापूर्वक संभालकर, आप एक सहज और इंटरैक्टिव उपयोगकर्ता अनुभव प्रदान कर सकते हैं, बड़ी प्रतिक्रियाओं को संभाल सकते हैं, संसाधन उपयोग को अनुकूलित कर सकते हैं, और त्रुटियों को सुचारू रूप से संभाल सकते हैं।

प्रदान किया गया Conversation::ReplyStream क्लास दिखाता है कि पैटर्न मैचिंग और इवेंट-संचालित आर्किटेक्चर का उपयोग करके रूबी एप्लिकेशन में स्ट्रीम प्रोसेसिंग को कैसे लागू किया जा सकता है। स्ट्रीम प्रोसेसिंग तकनीकों को समझकर और उनका लाभ उठाकर, आप अपने एप्लिकेशन में एआई एकीकरण की पूरी क्षमता को अनलॉक कर सकते हैं और शक्तिशाली और आकर्षक उपयोगकर्ता अनुभव प्रदान कर सकते हैं।

स्व-उपचारी डेटा

स्व-उपचारी डेटा बृहत भाषा मॉडल (एलएलएम) की क्षमताओं का लाभ उठाकर एप्लिकेशन में डेटा की अखंडता, संगति और गुणवत्ता सुनिश्चित करने का एक शक्तिशाली दृष्टिकोण है। पैटर्न की यह श्रेणी डेटा विसंगतियों, असंगतियों या त्रुटियों को स्वचालित रूप से पहचानने, निदान करने और सुधारने के लिए एआई के उपयोग के विचार पर केंद्रित है, जिससे डेवलपर्स पर बोझ कम होता है और डेटा की विश्वसनीयता का उच्च स्तर बना रहता है।

मूल रूप से, स्व-उपचारी डेटा पैटर्न यह मानते हैं कि डेटा किसी भी एप्लिकेशन का जीवन रक्त है, और इसकी सटीकता और अखंडता सुनिश्चित करना एप्लिकेशन के उचित कार्य और उपयोगकर्ता अनुभव के लिए महत्वपूर्ण है। हालांकि, डेटा की गुणवत्ता का प्रबंधन और रखरखाव एक जटिल और समय लेने वाला कार्य हो सकता है, विशेष रूप से जब एप्लिकेशन आकार और जटिलता में बढ़ते हैं। यहीं पर एआई की शक्ति काम आती है।

स्व-उपचारी डेटा पैटर्न में, एआई कार्यकर्ताओं को आपके एप्लिकेशन के डेटा की लगातार निगरानी और विश्लेषण के लिए नियोजित किया जाता है। इन मॉडलों में डेटा के भीतर पैटर्न, संबंधों और विसंगतियों को समझने और व्याख्या करने की क्षमता होती है। अपनी प्राकृतिक भाषा प्रसंस्करण और समझ क्षमताओं का लाभ उठाकर, वे डेटा में संभावित समस्याओं या असंगतियों की पहचान कर सकते हैं और उन्हें सुधारने के लिए उचित कार्रवाई कर सकते हैं।

स्व-उपचारी डेटा की प्रक्रिया में कई प्रमुख चरण शामिल हैं:

1. **डेटा निगरानी:** एआई कार्यकर्ता लगातार एप्लिकेशन के डेटा स्ट्रीम, डेटाबेस, या स्टोरेज सिस्टम की निगरानी करते हैं, किसी भी प्रकार की विसंगतियों, असंगतियों, या त्रुटियों के संकेतों की खोज करते हैं। वैकल्पिक रूप से, आप किसी अपवाद की प्रतिक्रिया में एआई घटक को सक्रिय कर सकते हैं।
2. **विसंगति पहचान:** जब कोई समस्या पाई जाती है, एआई कार्यकर्ता समस्या की विशिष्ट प्रकृति और दायरे की पहचान करने के लिए डेटा का विस्तृत विश्लेषण करता है। इसमें लापता मान, असंगत प्रारूप, या पूर्व-परिभाषित नियमों या बाधाओं का उल्लंघन करने वाले डेटा की पहचान शामिल हो सकती है।
3. **निदान और सुधार:** समस्या की पहचान होने के बाद, एआई कार्यकर्ता उचित कार्रवाई का निर्धारण करने के लिए डेटा डोमेन की अपनी समझ और ज्ञान का उपयोग करता है। इसमें स्वचालित रूप से डेटा को सुधारना, लापता मान भरना, या आवश्यक होने पर मानवीय हस्तक्षेप के लिए समस्या को चिह्नित करना शामिल हो सकता है।
4. **निरंतर सीखना (वैकल्पिक, उपयोग के मामले पर निर्भर):** जैसे-जैसे आपका एआई कार्यकर्ता विभिन्न डेटा समस्याओं का सामना करता है और उन्हें हल करता है, यह क्या हुआ और उसने कैसे प्रतिक्रिया दी, इसका वर्णन करने वाला आउटपुट दे सकता है। इस मेटाडेटा को सीखने की प्रक्रियाओं में फीड किया जा सकता है जो आपको (और शायद अंतर्निहित मॉडल को, फाइन-ट्यूनिंग के माध्यम से) डेटा विसंगतियों की पहचान करने और उन्हें हल करने में समय के साथ अधिक प्रभावी और कुशल बनने में सक्षम बनाता है।

डेटा संबंधी समस्याओं को स्वचालित रूप से पहचानने और सुधारने से, आप यह

सुनिश्चित कर सकते हैं कि आपका एप्लिकेशन उच्च गुणवत्ता वाले, विश्वसनीय डेटा पर काम करता है। यह एप्लिकेशन की कार्यक्षमता या उपयोगकर्ता अनुभव को प्रभावित करने वाली त्रुटियों, असंगतियों, या डेटा-संबंधी बग्स के जोखिम को कम करता है।

एक बार जब एआई वर्कर्स डेटा निगरानी और सुधार का कार्य संभाल लेते हैं, तब आप अपने प्रयासों को एप्लिकेशन के अन्य महत्वपूर्ण पहलुओं पर केंद्रित कर सकते हैं। यह उस समय और संसाधनों की बचत करता है जो अन्यथा मैनुअल डेटा क्लीनिंग और रखरखाव पर खर्च होते। वास्तव में, जैसे-जैसे आपके एप्लिकेशन का आकार और जटिलता बढ़ती है, मैनुअल रूप से डेटा गुणवत्ता का प्रबंधन करना और भी चुनौतीपूर्ण होता जाता है। "स्व-मरम्मत डेटा" पैटर्न बड़ी मात्रा में डेटा को संभालने और वास्तविक समय में समस्याओं का पता लगाने के लिए एआई की शक्ति का लाभ उठाकर प्रभावी ढंग से स्केल करते हैं।

अपनी प्रकृति के कारण, एआई मॉडल समय के साथ बदलते डेटा पैटर्न, स्कीमा, या आवश्यकताओं के अनुकूल बन सकते हैं, वो भी बिना किसी पर्यवेक्षण के या न्यूनतम पर्यवेक्षण के साथ। जब तक उनके निर्देश पर्याप्त मार्गदर्शन प्रदान करते हैं, विशेष रूप से इच्छित परिणामों के संबंध में, आपका एप्लिकेशन बिना व्यापक मैनुअल हस्तक्षेप या कोड परिवर्तनों की आवश्यकता के नए डेटा परिदृश्यों को संभालने में सक्षम हो सकता है।

स्व-मरम्मत डेटा पैटर्न अन्य श्रेणियों के पैटर्न जैसे "मल्टीट्यूड ऑफ वर्कर्स" के साथ अच्छी तरह से संरेखित होते हैं। स्व-मरम्मत डेटा क्षमता को एक विशेष प्रकार के वर्कर व्यवस्था के रूप में देखा जा सकता है जो विशेष रूप से डेटा गुणवत्ता और अखंडता सुनिश्चित करने पर केंद्रित है। यह प्रकार का वर्कर अन्य एआई वर्कर्स के साथ काम करता है, जहां प्रत्येक एप्लिकेशन की कार्यक्षमता के विभिन्न पहलुओं में योगदान करता है।

व्यवहार में स्व-मरम्मत डेटा पैटर्न को लागू करने के लिए एप्लिकेशन आर्किटेक्चर में एआई मॉडल के सावधानीपूर्वक डिजाइन और एकीकरण की आवश्यकता होती है। डेटा हानि और भ्रष्टता के जोखिमों के कारण, आपको स्पष्ट दिशानिर्देश परिभाषित करने चाहिए कि आप इस तकनीक का उपयोग कैसे करेंगे। आपको प्रदर्शन, स्केलेबिलिटी

और डेटा सुरक्षा जैसे कारकों पर भी विचार करना चाहिए।

व्यावहारिक केस स्टडी: टूटे हुए JSON को ठीक करना

स्व-मरम्मत डेटा का लाभ उठाने के सबसे व्यावहारिक और सुविधाजनक तरीकों में से एक समझाने में भी बहुत सरल है: टूटे हुए JSON को ठीक करना।

यह तकनीक एलएलएम द्वारा उत्पन्न अपूर्ण या असंगत डेटा जैसे टूटे हुए JSON से निपटने की सामान्य चुनौती पर लागू की जा सकती है, और इन समस्याओं को स्वचालित रूप से पहचानने और सुधारने के लिए एक दृष्टिकोण प्रदान करती है।

Olympia में मैं नियमित रूप से ऐसी स्थितियों का सामना करता हूं जहां एलएलएम ऐसा JSON डेटा जनरेट करते हैं जो पूरी तरह से वैध नहीं होता। यह विभिन्न कारणों से हो सकता है, जैसे एलएलएम द्वारा वास्तविक JSON कोड से पहले या बाद में टिप्पणी जोड़ना, या कॉमा की अनुपस्थिति या अनएस्केप्ड डबल कोट्स जैसी सिंटैक्स त्रुटियां पैदा करना। ये समस्याएं पार्सिंग त्रुटियों का कारण बन सकती हैं और एप्लिकेशन की कार्यक्षमता में व्यवधान उत्पन्न कर सकती हैं।

इस समस्या का समाधान करने के लिए, मैंने JsonFixer क्लास के रूप में एक व्यावहारिक समाधान का कार्यान्वयन किया है। यह क्लास “Self-Healing Data” पैटर्न को प्रदर्शित करती है, जो टूटे हुए JSON को इनपुट के रूप में लेती है और एक LLM का उपयोग करके उसे ठीक करती है, जिससे जितनी संभव हो उतनी जानकारी और मूल इरादे को संरक्षित रखा जा सके।

```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
```

ध्यान दें कि JsonFixer कैसे AI की प्रतिक्रियाओं को निर्देशित करने के लिए Ventriloquist का उपयोग करता है।

JSON डेटा के स्व-सुधार की प्रक्रिया निम्नानुसार कार्य करती है:

1. **JSON जनरेशन:** कुछ प्रॉम्प्ट्स या आवश्यकताओं के आधार पर JSON डेटा उत्पन्न करने के लिए LLM का उपयोग किया जाता है। हालांकि, LLM की प्रकृति के कारण, उत्पन्न JSON हमेशा पूरी तरह से मान्य नहीं हो सकता है। यदि आप अमान्य JSON देते हैं तो JSON पार्सर निश्चित रूप से ParserError उठाएगा।

```ruby
begin
  JSON.parse(llm_generated_json)
rescue JSON::ParserError => e
  JsonFixer.new.call(llm_generated_json, e.message)
end
```

ध्यान दें कि एक्सेप्शन मैसेज भी JSONFixer कॉल को पास किया जाता है ताकि इसे डेटा में क्या गलत है, इसकी पूरी कल्पना न करनी पड़े, खासकर जब पार्सर अक्सर आपको बताता है कि क्या गलत है।

2. **LLM-आधारित सुधार:** JSONFixer क्लास टूटे हुए JSON को एक LLM को भेजता है, साथ ही JSON को ठीक करने के लिए एक विशिष्ट प्रॉम्प्ट या निर्देश भी भेजता है, जिससे मूल जानकारी और उद्देश्य को जितना संभव हो उतना संरक्षित किया जा सके। LLM, जो विशाल मात्रा में डेटा पर प्रशिक्षित है और JSON सिंटैक्स को समझता है, त्रुटियों को सुधारने और एक वैध JSON स्ट्रिंग जनरेट करने का प्रयास करता है। LLM के आउटपुट को सीमित करने के लिए Response Fencing का उपयोग किया जाता है, और हम Mixtral 8x7B को AI मॉडल के रूप में चुनते हैं, क्योंकि यह इस प्रकार के कार्य के लिए विशेष रूप से अच्छा है।
3. **सत्यापन और एकीकरण:** LLM द्वारा लौटाई गई फिक्स्ड JSON स्ट्रिंग को JSONFixer क्लास द्वारा स्वयं पार्स किया जाता है, क्योंकि इसने chat_completion(json: true) को कॉल किया। यदि फिक्स्ड JSON सत्यापन पास करता है, तो इसे एप्लिकेशन के कार्यप्रवाह में वापस एकीकृत कर दिया जाता है, जिससे एप्लिकेशन निर्बाध रूप से डेटा को प्रोसेस करना जारी रख सकता है। खराब JSON को “ठीक” कर दिया गया है।

हालांकि मैंने अपना JSONFixer कार्यान्वयन कई बार लिखा और पुनर्लेखित किया है, मुझे संदेह है कि उन सभी संस्करणों में कुल निवेश किया गया समय एक या दो घंटे से अधिक है।

ध्यान दें कि किसी भी स्व-उपचार डेटा पैटर्न का एक महत्वपूर्ण तत्व उद्देश्य का संरक्षण है। LLM-आधारित सुधार प्रक्रिया जनरेट किए गए JSON की मूल जानकारी और उद्देश्य

को जितना संभव हो उतना संरक्षित करने का लक्ष्य रखती है। यह सुनिश्चित करता है कि फिक्स्ड JSON अपना सिमेंटिक अर्थ बनाए रखता है और एप्लिकेशन के संदर्भ में प्रभावी ढंग से उपयोग किया जा सकता है।

Olympia में "स्व-उपचार डेटा" दृष्टिकोण का यह व्यावहारिक कार्यान्वयन स्पष्ट रूप से दर्शाता है कि कैसे AI, विशेष रूप से LLMs का उपयोग वास्तविक दुनिया की डेटा चुनौतियों को हल करने के लिए किया जा सकता है। यह मजबूत और कुशल एप्लिकेशन बनाने के लिए पारंपरिक प्रोग्रामिंग तकनीकों को AI क्षमताओं के साथ जोड़ने की शक्ति को प्रदर्शित करता है।

पोस्टेल का नियम और "स्व-उपचार डेटा" पैटर्न

"स्व-उपचार डेटा," जैसा कि JSONFixer क्लास द्वारा उदाहरित किया गया है, पोस्टेल के नियम के सिद्धांत के साथ अच्छी तरह से संरेखित है, जिसे मजबूती का सिद्धांत भी कहा जाता है। पोस्टेल का नियम कहता है:

"जो आप करते हैं उसमें रूढ़िवादी रहें, दूसरों से जो स्वीकार करते हैं उसमें उदार रहें।"

यह सिद्धांत, जो मूल रूप से शुरुआती इंटरनेट के अग्रदूत जॉन पोस्टेल द्वारा व्यक्त किया गया था, ऐसी प्रणालियाँ बनाने के महत्व पर जोर देता है जो विविध या यहाँ तक कि थोड़े गलत इनपुट के प्रति सहनशील हों, जबकि आउटपुट भेजते समय निर्दिष्ट प्रोटोकॉल का कड़ाई से पालन करें।

"स्व-उपचारी डेटा" के संदर्भ में, JSONFixer क्लास पोस्टेल के नियम को साकार करती है, जो LLMs द्वारा उत्पन्न टूटे या अपूर्ण JSON डेटा को स्वीकार करने में उदार दृष्टिकोण अपनाती है। यह अपेक्षित प्रारूप का कड़ाई से पालन न करने वाले JSON को तुरंत अस्वीकार या विफल नहीं करती। इसके बजाय, यह LLMs की शक्ति का उपयोग करके JSON को ठीक करने का प्रयास करती है।

अपूर्ण JSON को स्वीकार करने में उदार होने के कारण, JSONFixer क्लास मजबूती

और लचीलापन प्रदर्शित करती है। यह स्वीकार करती है कि वास्तविक दुनिया में डेटा विभिन्न रूपों में आता है और हमेशा कड़े विनिर्देशों के अनुरूप नहीं हो सकता। इन विचलनों को सुचारू रूप से संभालने और सुधारने के द्वारा, यह क्लास सुनिश्चित करती है कि एप्लिकेशन अपूर्ण डेटा की उपस्थिति में भी निर्बाध रूप से कार्य करता रहे।

दूसरी ओर, JSONFixer क्लास आउटपुट के मामले में पोस्टेल के नियम के रूढ़िवादी पहलू का भी पालन करती है। LLMs का उपयोग करके JSON को ठीक करने के बाद, क्लास सुधारित JSON को मान्य करती है ताकि यह अपेक्षित प्रारूप का कड़ाई से पालन करे। यह एप्लिकेशन के अन्य भागों में भेजने से पहले डेटा की अखंडता और सटीकता को बनाए रखती है। यह रूढ़िवादी दृष्टिकोण गारंटी देता है कि JSONFixer क्लास का आउटपुट विश्वसनीय और संगत है, जो अंतर-संचालनीयता को बढ़ावा देता है और त्रुटियों के प्रसार को रोकता है।

जॉन पोस्टेल के बारे में रोचक तथ्य:

- जॉन पोस्टेल (1943-1998) एक अमेरिकी कंप्यूटर वैज्ञानिक थे जिन्होंने इंटरनेट के विकास में महत्वपूर्ण भूमिका निभाई। उन्हें अंतर्निहित प्रोटोकॉल और मानकों में उनके महत्वपूर्ण योगदान के लिए "इंटरनेट के भगवान" के रूप में जाना जाता था।
- पोस्टेल रिक्वेस्ट फॉर कमेंट्स (RFC) दस्तावेज श्रृंखला के संपादक थे, जो इंटरनेट के बारे में तकनीकी और संगठनात्मक नोट्स की एक श्रृंखला है। उन्होंने TCP, IP, और SMTP जैसे मौलिक प्रोटोकॉल सहित 200 से अधिक RFCs के लेखन या सह-लेखन किए।
- अपने तकनीकी योगदान के अलावा, पोस्टेल अपने विनम्र और सहयोगी दृष्टिकोण के लिए जाने जाते थे। वे आम सहमति प्राप्त करने और एक मजबूत और अंतर-संचालनीय नेटवर्क बनाने के लिए मिलकर काम करने के महत्व में विश्वास करते थे।
- पोस्टेल ने 1977 से 1998 में अपनी असामयिक मृत्यु तक University of Southern California (USC) के Information Sciences Institute (ISI) के कंप्यूटर नेटवर्क डिविजन के निदेशक के रूप में कार्य किया।

- उनके अपार योगदान को मान्यता देते हुए, पोस्टेल को 1998 में मरणोपरांत प्रतिष्ठित ट्यूरिंग पुरस्कार से सम्मानित किया गया, जिसे अक्सर "कंप्यूटिंग का नोबेल पुरस्कार" कहा जाता है।

JSONFixer क्लास मजबूती, लचीलेपन और अंतर-संचालनीयता को बढ़ावा देती है, जो मूल्य पोस्टेल ने अपने पूरे करियर में बनाए रखे। अपूर्णताओं के प्रति सहनशील रहते हुए प्रोटोकॉल का कड़ाई से पालन करने वाली प्रणालियां बनाकर, हम ऐसे एप्लिकेशन बना सकते हैं जो वास्तविक दुनिया की चुनौतियों का सामना करने में अधिक लचीले और अनुकूलनीय हों।

विचारणीय बिंदु और प्रतिसंकेत

स्व-मरम्मत डेटा दृष्टिकोणों की प्रयोज्यता पूरी तरह से आपके एप्लिकेशन द्वारा संभाले जाने वाले डेटा के प्रकार पर निर्भर करती है। एक कारण है कि आप अपने एप्लिकेशन में सभी JSON पार्सिंग त्रुटियों को स्वचालित रूप से सुधारने के लिए JSON.parse को सरल रूप से मंकीपैच नहीं करना चाहेंगे: सभी त्रुटियों को स्वचालित रूप से सुधारा नहीं जा सकता या सुधारा नहीं जाना चाहिए।

डेटा हैंडलिंग और प्रोसेसिंग से संबंधित नियामक या अनुपालन आवश्यकताओं के साथ जुड़े होने पर स्व-मरम्मत विशेष रूप से जटिल हो जाती है। कुछ उद्योगों, जैसे स्वास्थ्य सेवा और वित्त में, डेटा अखंडता और लेखा-परीक्षण क्षमता के संबंध में इतने कड़े नियम हैं कि उचित निरीक्षण या लॉगिंग के बिना किसी भी प्रकार का "ब्लैक बॉक्स" डेटा सुधार करना इन नियमों का उल्लंघन कर सकता है। यह सुनिश्चित करना महत्वपूर्ण है कि आप जो भी स्व-मरम्मत डेटा तकनीकें विकसित करते हैं, वे लागू कानूनी और नियामक ढांचे के अनुरूप हों।

स्व-मरम्मत डेटा तकनीकों को लागू करने से, विशेष रूप से AI मॉडल्स से जुड़ी तकनीकों का, एप्लिकेशन प्रदर्शन और संसाधन उपयोग पर बड़ा प्रभाव पड़ सकता है। त्रुटि पहचान और सुधार के लिए AI मॉडल्स के माध्यम से बड़ी मात्रा में डेटा का

प्रसंस्करण कम्प्यूटेशनल रूप से गहन हो सकता है। स्व-मरम्मत डेटा के लाभों और संबंधित प्रदर्शन एवं संसाधन लागतों के बीच ट्रेड-ऑफ का आकलन करना महत्वपूर्ण है।

यह कहा जाए तो, आइए इस शक्तिशाली दृष्टिकोण को कब और कहाँ लागू करना है, इससे जुड़े कारकों में गहराई से जाते हैं।

डेटा महत्वता

स्व-मरम्मत डेटा तकनीकों के अनुप्रयोग पर विचार करते समय, प्रसंस्करण किए जा रहे डेटा की महत्वता का आकलन करना अत्यंत आवश्यक है। महत्वता का स्तर आपके एप्लिकेशन और इसके व्यावसायिक डोमेन के संदर्भ में डेटा के महत्व और संवेदनशीलता को दर्शाता है।

कुछ मामलों में, डेटा त्रुटियों को स्वचालित रूप से सुधारना उचित नहीं हो सकता, विशेष रूप से यदि डेटा अत्यधिक संवेदनशील है या इसके कानूनी निहितार्थ हैं। उदाहरण के लिए, निम्नलिखित परिदृश्यों पर विचार करें:

1. **वित्तीय लेन-देन:** वित्तीय एप्लिकेशन में, जैसे बैंकिंग सिस्टम या ट्रेडिंग प्लेटफॉर्म, डेटा सटीकता सर्वोच्च महत्व की होती है। वित्तीय डेटा में छोटी त्रुटियां भी महत्वपूर्ण परिणाम ला सकती हैं, जैसे गलत खाता शेष, गलत दिशा में भेजी गई धनराशि, या त्रुटिपूर्ण ट्रेडिंग निर्णय। इन मामलों में, विस्तृत सत्यापन और लेखा-परीक्षण के बिना स्वचालित सुधार अस्वीकार्य जोखिम पैदा कर सकते हैं।
2. **चिकित्सा रिकॉर्ड:** स्वास्थ्य सेवा एप्लिकेशन अत्यधिक संवेदनशील और गोपनीय रोगी डेटा से निपटते हैं। चिकित्सा रिकॉर्ड में अशुद्धियों का रोगी सुरक्षा और उपचार निर्णयों पर गंभीर प्रभाव पड़ सकता है। योग्य स्वास्थ्य सेवा पेशेवरों द्वारा उचित निरीक्षण और सत्यापन के बिना चिकित्सा डेटा को स्वचालित रूप से संशोधित करना नियामक आवश्यकताओं का उल्लंघन कर सकता है और रोगी के कल्याण को जोखिम में डाल सकता है।
3. **कानूनी दस्तावेज़:** अनुबंध, समझौते, या न्यायालय फाइलिंग जैसे कानूनी दस्तावेजों को संभालने वाले एप्लिकेशन में कड़ी सटीकता और अखंडता की

> आवश्यकता होती है। कानूनी डेटा में छोटी त्रुटियों के भी महत्वपूर्ण कानूनी परिणाम हो सकते हैं। इस क्षेत्र में स्वचालित सुधार उपयुक्त नहीं हो सकते, क्योंकि डेटा की वैधता और प्रवर्तनीयता सुनिश्चित करने के लिए अक्सर कानूनी विशेषज्ञों द्वारा मैनुअल समीक्षा और सत्यापन की आवश्यकता होती है।

इन महत्वपूर्ण डेटा परिदृश्यों में, स्वचालित सुधारों से जुड़े जोखिम अक्सर संभावित लाभों से अधिक होते हैं। त्रुटियों को शामिल करने या डेटा को गलत तरीके से संशोधित करने के परिणाम गंभीर हो सकते हैं, जिनसे वित्तीय नुकसान, कानूनी देयताएं, या यहां तक कि व्यक्तियों को नुकसान भी हो सकता है।

अत्यंत महत्वपूर्ण डेटा से निपटते समय, मैनुअल सत्यापन और मान्यकरण प्रक्रियाओं को प्राथमिकता देना आवश्यक है। डेटा की सटीकता और अखंडता सुनिश्चित करने में मानवीय निरीक्षण और विशेषज्ञता महत्वपूर्ण है। संभावित त्रुटियों या विसंगतियों को चिह्नित करने के लिए स्व-मरम्मत तकनीकों का उपयोग किया जा सकता है, लेकिन सुधारों पर अंतिम निर्णय में मानवीय निर्णय और अनुमोदन शामिल होना चाहिए।

हालांकि, यह ध्यान रखना महत्वपूर्ण है कि किसी एप्लिकेशन में सभी डेटा की महत्वपूर्णता का स्तर समान नहीं हो सकता। एक ही एप्लिकेशन में, डेटा के कुछ हिस्से कम संवेदनशील हो सकते हैं या त्रुटियों के होने पर उनका प्रभाव कम हो सकता है। ऐसे मामलों में, स्व-मरम्मत डेटा तकनीकों को चुनिंदा रूप से उन विशिष्ट डेटा सबसेट पर लागू किया जा सकता है, जबकि महत्वपूर्ण डेटा मैनुअल सत्यापन के अधीन रहता है।

मुख्य बात यह है कि आपके एप्लिकेशन में प्रत्येक डेटा श्रेणी की महत्वपूर्णता का सावधानीपूर्वक आकलन किया जाए और संबंधित जोखिमों और निहितार्थों के आधार पर सुधारों को संभालने के लिए स्पष्ट दिशानिर्देश और प्रक्रियाएं परिभाषित की जाएं। महत्वपूर्ण (जैसे लेजर्स, चिकित्सा रिकॉर्ड) और गैर-महत्वपूर्ण डेटा (जैसे मेलिंग पते, संसाधन चेतावनियां) के बीच अंतर करके, आप उपयुक्त स्थानों पर स्व-मरम्मत डेटा तकनीकों के लाभों का लाभ उठाने और जहां आवश्यक हो कड़े नियंत्रण और निरीक्षण को बनाए रखने के बीच संतुलन बना सकते हैं।

अंततः, महत्वपूर्ण डेटा पर स्व-मरम्मत डेटा तकनीकों को लागू करने का निर्णय डोमेन विशेषज्ञों, कानूनी सलाहकारों और अन्य संबंधित हितधारकों के परामर्श से लिया

जाना चाहिए। आपके एप्लिकेशन के डेटा से जुड़ी विशिष्ट आवश्यकताओं, नियमों और जोखिमों पर विचार करना और डेटा सुधार रणनीतियों को तदनुसार संरेखित करना आवश्यक है।

त्रुटि की गंभीरता

स्व-मरम्मत डेटा तकनीकों को लागू करते समय, डेटा त्रुटियों की गंभीरता और प्रभाव का आकलन करना महत्वपूर्ण है। सभी त्रुटियां समान नहीं होतीं, और उचित कार्रवाई समस्या की गंभीरता के आधार पर भिन्न हो सकती है।

मामूली विसंगतियां या फॉर्मेटिंग समस्याएं स्वचालित सुधार के लिए उपयुक्त हो सकती हैं। उदाहरण के लिए, टूटे हुए JSON को ठीक करने का काम करने वाला स्व-मरम्मत डेटा वर्कर गायब कॉमा या अनएस्केप्ड डबल कोट्स को डेटा के अर्थ या संरचना को महत्वपूर्ण रूप से बदले बिना संभाल सकता है। इस प्रकार की त्रुटियों को सुधारना अक्सर सीधा होता है और समग्र डेटा अखंडता पर न्यूनतम प्रभाव पड़ता है।

हालांकि, अधिक गंभीर त्रुटियां जो मूल रूप से डेटा के अर्थ या अखंडता को बदलती हैं, उनके लिए एक अलग दृष्टिकोण की आवश्यकता हो सकती है। ऐसे मामलों में, स्वचालित सुधार पर्याप्त नहीं हो सकते हैं, और डेटा की सटीकता और वैधता सुनिश्चित करने के लिए मानवीय हस्तक्षेप आवश्यक हो सकता है।

यहीं पर एआई का उपयोग स्वयं त्रुटि की गंभीरता निर्धारित करने में मदद के लिए करने की अवधारणा सामने आती है। एआई मॉडल की क्षमताओं का लाभ उठाते हुए, हम ऐसे स्व-उपचार डेटा वर्कर डिज़ाइन कर सकते हैं जो न केवल त्रुटियों को सुधारते हैं बल्कि उन त्रुटियों की गंभीरता का आकलन भी करते हैं और उनसे निपटने के लिए सूचित निर्णय लेते हैं।

उदाहरण के लिए, एक स्व-उपचार डेटा वर्कर पर विचार करें जो ग्राहक डेटाबेस में प्रवाहित होने वाले डेटा में विसंगतियों को सुधारने के लिए जिम्मेदार है। वर्कर को डेटा का विश्लेषण करने और संभावित त्रुटियों की पहचान करने के लिए डिज़ाइन किया जा सकता है, जैसे कि गायब या परस्पर विरोधी जानकारी। हालांकि, सभी त्रुटियों को

स्वचालित रूप से सुधारने के बजाय, वर्कर को अतिरिक्त टूल कॉल्स से लैस किया जा सकता है जो गंभीर त्रुटियों को मानवीय समीक्षा के लिए चिह्नित कर सकते हैं।

यहाँ एक उदाहरण है कि इसे कैसे कार्यान्वित किया जा सकता है:

```
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
```

इस उदाहरण में, CustomerDataHealer वर्कर ग्राहक डेटा में असंगतियों की पहचान और सुधार के लिए डिज़ाइन किया गया है। एक बार फिर, हम संरचित आउटपुट प्राप्त करने के लिए रेस्पॉन्स फेन्सिंग और वेंट्रिलोक्विस्ट का उपयोग करते हैं। महत्वपूर्ण रूप से, वर्कर के सिस्टम निर्देश में गंभीर त्रुटियों का सामना होने पर flag_for_review फ़ंक्शन का उपयोग करने के निर्देश शामिल हैं।

जब वर्कर ग्राहक डेटा को प्रोसेस करता है, तो यह डेटा का विश्लेषण करता है और किसी भी असंगति को सुधारने का प्रयास करता है। यदि वर्कर यह निर्धारित करता है कि त्रुटियाँ गंभीर हैं और मानवीय हस्तक्षेप की आवश्यकता है, तो यह डेटा को फ्लैग करने और फ्लैगिंग का कारण प्रदान करने के लिए flag_for_review टूल का उपयोग कर सकता है।

chat_completion मेथड को json: true के साथ कॉल किया जाता है ताकि सुधारित ग्राहक डेटा को JSON के रूप में पार्स किया जा सके। फ़ंक्शन कॉल के बाद लूपिंग का कोई प्रावधान नहीं है, इसलिए यदि flag_for_review को इनवोक किया गया था तो परिणाम खाली होगा। अन्यथा, ग्राहक को समीक्षित और संभावित रूप से सुधारित डेटा के साथ अपडेट किया जाता है।

त्रुटि की गंभीरता के आकलन और मानवीय समीक्षा के लिए डेटा को फ्लैग करने के विकल्प को शामिल करके, स्व-उपचारी डेटा वर्कर अधिक बुद्धिमान और अनुकूलनीय बन जाता है। यह मामूली त्रुटियों को स्वचालित रूप से संभाल सकता है जबकि गंभीर त्रुटियों को मैनुअल हस्तक्षेप के लिए मानव विशेषज्ञों तक पहुंचा सकता है।

त्रुटि की गंभीरता निर्धारित करने के विशिष्ट मानदंड डोमेन ज्ञान और व्यावसायिक आवश्यकताओं के आधार पर वर्कर के निर्देश में परिभाषित किए जा सकते हैं। डेटा अखंडता पर प्रभाव, डेटा हानि या भ्रष्टता की संभावना, और गलत डेटा के परिणामों जैसे कारकों को गंभीरता का आकलन करते समय विचार किया जा सकता है।

त्रुटि की गंभीरता का आकलन करने के लिए AI का लाभ उठाकर और मानवीय हस्तक्षेप के विकल्प प्रदान करके, स्व-उपचारी डेटा तकनीकें स्वचालन और डेटा सटीकता बनाए रखने के बीच संतुलन बना सकती हैं। यह दृष्टिकोण सुनिश्चित करता है कि मामूली त्रुटियों को कुशलतापूर्वक सुधारा जाए जबकि गंभीर त्रुटियों को मानव समीक्षकों से आवश्यक ध्यान और विशेषज्ञता प्राप्त हो।

डोमेन जटिलता

स्व-उपचारी डेटा तकनीकों के अनुप्रयोग पर विचार करते समय, डेटा डोमेन की जटिलता और इसकी संरचना और संबंधों को नियंत्रित करने वाले नियमों का मूल्यांकन करना महत्वपूर्ण है। डोमेन की जटिलता स्वचालित डेटा सुधार दृष्टिकोणों की प्रभावशीलता और व्यवहार्यता को महत्वपूर्ण रूप से प्रभावित कर सकती है।

स्व-उपचारी डेटा तकनीकें तब अच्छी तरह से काम करती हैं जब डेटा सुपरिभाषित पैटर्न और प्रतिबंधों का पालन करता है। उन डोमेन में जहाँ डेटा संरचना अपेक्षाकृत सरल है और डेटा तत्वों के बीच संबंध सीधे हैं, स्वचालित सुधारों को उच्च विश्वास के साथ लागू किया जा सकता है। उदाहरण के लिए, फॉर्मेटिंग समस्याओं को सुधारना या बुनियादी डेटा प्रकार प्रतिबंधों को लागू करना अक्सर स्व-उपचारी डेटा वर्कर्स द्वारा प्रभावी ढंग से संभाला जा सकता है।

हालांकि, जैसे-जैसे डेटा डोमेन की जटिलता बढ़ती है, स्वचालित डेटा सुधार से जुड़ी चुनौतियां भी बढ़ती जाती हैं। जटिल व्यावसायिक तर्क, डेटा इकाइयों के बीच जटिल संबंधों, या डोमेन-विशिष्ट नियमों और अपवादों वाले डोमेन में, स्व-मरम्मत डेटा तकनीकें हमेशा सूक्ष्म बातों को नहीं समझ पाती हैं और अनपेक्षित परिणाम उत्पन्न कर सकती हैं।

आइए एक जटिल डोमेन का उदाहरण लेते हैं: एक वित्तीय ट्रेडिंग सिस्टम। इस डोमेन में, डेटा में विभिन्न वित्तीय साधन, बाजार डेटा, ट्रेडिंग नियम, और नियामक आवश्यकताएं शामिल होती हैं। विभिन्न डेटा तत्वों के बीच संबंध जटिल हो सकते हैं, और डेटा की वैधता और संगति को नियंत्रित करने वाले नियम डोमेन के लिए अत्यधिक विशिष्ट हो सकते हैं।

इस तरह के जटिल डोमेन में, ट्रेड डेटा में असंगतियों को सुधारने के लिए नियुक्त स्व-मरम्मत डेटा वर्कर को डोमेन-विशिष्ट नियमों और प्रतिबंधों की गहरी समझ होनी चाहिए। इसे बाजार नियमों, ट्रेडिंग सीमाओं, जोखिम गणनाओं, और निपटान प्रक्रियाओं जैसे कारकों पर विचार करना होगा। इस संदर्भ में स्वचालित सुधार हमेशा डोमेन की पूर्ण जटिलता को नहीं समझ पाते हैं और अनजाने में त्रुटियां या डोमेन-विशिष्ट नियमों का उल्लंघन कर सकते हैं।

डोमेन जटिलता की चुनौतियों से निपटने के लिए, स्व-मरम्मत डेटा तकनीकों को एआई मॉडल और वर्कर्स में डोमेन-विशिष्ट ज्ञान और नियमों को शामिल करके बेहतर बनाया जा सकता है। यह निम्नलिखित तकनीकों के माध्यम से प्राप्त किया जा सकता है:

1. **डोमेन-विशिष्ट प्रशिक्षण:** स्व-मरम्मत डेटा के लिए उपयोग किए जाने वाले एआई मॉडल को विशेष डोमेन के डेटासेट पर निर्देशित या फाइन-ट्यून किया जा सकता है जो उस विशेष डोमेन की जटिलताओं और नियमों को समझते हैं। मॉडल को प्रतिनिधि डेटा और परिदृश्यों से अवगत कराकर, वे डोमेन-विशिष्ट पैटर्न, प्रतिबंध और अपवादों को सीख सकते हैं।
2. **नियम-आधारित प्रतिबंध:** स्व-मरम्मत डेटा वर्कर्स को स्पष्ट नियम-आधारित प्रतिबंधों से संवर्धित किया जा सकता है जो डोमेन-विशिष्ट ज्ञान को एनकोड करते हैं। ये नियम डोमेन विशेषज्ञों द्वारा परिभाषित किए जा सकते हैं और डेटा सुधार प्रक्रिया में एकीकृत किए जा सकते हैं। एआई मॉडल तब इन नियमों का उपयोग अपने निर्णयों को मार्गदर्शित करने और डोमेन-विशिष्ट आवश्यकताओं के अनुपालन को सुनिश्चित करने के लिए कर सकते हैं।
3. **डोमेन विशेषज्ञों के साथ सहयोग:** जटिल डोमेन में, स्व-मरम्मत डेटा तकनीकों के डिजाइन और विकास में डोमेन विशेषज्ञों को शामिल करना महत्वपूर्ण है। डोमेन विशेषज्ञ डेटा की जटिलताओं, व्यावसायिक नियमों, और संभावित एज केस के बारे में मूल्यवान जानकारी प्रदान कर सकते हैं। उनके ज्ञान को मानव-सहायक प्रणाली पैटर्न का उपयोग करके एआई मॉडल और वर्कर्स में शामिल किया जा सकता है।
4. **क्रमिक और पुनरावर्ती दृष्टिकोण:** जटिल डोमेन से निपटते समय, स्व-मरम्मत डेटा के लिए एक क्रमिक और पुनरावर्ती दृष्टिकोण अपनाना अक्सर लाभदायक होता है। एक बार में पूरे डोमेन के लिए सुधारों को स्वचालित करने का प्रयास करने के बजाय, विशिष्ट उप-डोमेन या डेटा श्रेणियों पर ध्यान केंद्रित करें जहां नियम और प्रतिबंध अच्छी तरह से समझे जाते हैं। जैसे-जैसे डोमेन की समझ बढ़ती है और तकनीकें प्रभावी साबित होती हैं, धीरे-धीरे स्व-मरम्मत तकनीकों के दायरे का विस्तार करें।

डेटा डोमेन की जटिलता पर विचार करते हुए और स्व-उपचार डेटा तकनीकों में

डोमेन-विशिष्ट ज्ञान को शामिल करके, आप स्वचालन और सटीकता के बीच संतुलन बना सकते हैं। यह समझना महत्वपूर्ण है कि स्व-उपचार डेटा एक सर्वमान्य समाधान नहीं है और इस दृष्टिकोण को प्रत्येक डोमेन की विशिष्ट आवश्यकताओं और चुनौतियों के अनुरूप बनाया जाना चाहिए।

जटिल डोमेन में, स्व-उपचार डेटा तकनीकों को मानवीय विशेषज्ञता और निरीक्षण के साथ जोड़ने वाला एक मिश्रित दृष्टिकोण सबसे प्रभावी हो सकता है। स्वचालित सुधार नियमित और सुपरिभाषित मामलों को संभाल सकते हैं, जबकि जटिल परिदृश्यों या अपवादों को मानवीय समीक्षा और हस्तक्षेप के लिए चिह्नित किया जा सकता है। यह सहयोगात्मक दृष्टिकोण सुनिश्चित करता है कि जटिल डेटा डोमेन में आवश्यक नियंत्रण और सटीकता बनाए रखते हुए स्वचालन के लाभों को प्राप्त किया जाए।

व्याख्येयता और पारदर्शिता

व्याख्येयता एआई मॉडल द्वारा लिए गए निर्णयों के पीछे के तर्क को समझने और व्याख्या करने की क्षमता को संदर्भित करती है, जबकि पारदर्शिता डेटा सुधार प्रक्रिया में स्पष्ट दृश्यता प्रदान करने से संबंधित है।

कई संदर्भों में, डेटा संशोधन लेखा-परीक्षण योग्य और न्यायसंगत होने चाहिए। व्यावसायिक उपयोगकर्ताओं, लेखा परीक्षकों और नियामक निकायों सहित हितधारकों को यह स्पष्टीकरण की आवश्यकता हो सकती है कि कुछ विशेष डेटा सुधार क्यों किए गए और एआई मॉडल उन निर्णयों पर कैसे पहुंचे। यह विशेष रूप से उन क्षेत्रों में महत्वपूर्ण है जहां डेटा सटीकता और अखंडता का महत्वपूर्ण प्रभाव पड़ता है, जैसे वित्त, स्वास्थ्य सेवा और कानूनी मामले।

व्याख्येयता और पारदर्शिता की आवश्यकता को पूरा करने के लिए, स्व-उपचार डेटा तकनीकों में ऐसे तंत्र शामिल होने चाहिए जो एआई मॉडल की निर्णय-प्रक्रिया में अंतर्दृष्टि प्रदान करें। यह विभिन्न दृष्टिकोणों के माध्यम से प्राप्त किया जा सकता है:

1. **विचार श्रृंखला:** मॉडल से डेटा में परिवर्तन करने से पहले अपनी सोच को "जोर से" समझाने के लिए कहने से निर्णय-प्रक्रिया को समझना आसान हो

सकता है और किए गए सुधारों के लिए मानव-पठनीय स्पष्टीकरण तैयार किए जा सकते हैं। इसका व्यापार-प्रतिफल स्पष्टीकरण को संरचित डेटा आउटपुट से अलग करने में थोड़ी अधिक जटिलता है, जिसे संबोधित किया जा सकता है...

2. **स्पष्टीकरण उत्पादन:** स्व-उपचार डेटा कार्यकर्ताओं को उनके द्वारा किए गए सुधारों के लिए मानव-पठनीय स्पष्टीकरण उत्पन्न करने की क्षमता से लैस किया जा सकता है। यह मॉडल से डेटा में ही एकीकृत अपनी निर्णय-प्रक्रिया को आसानी से समझने योग्य स्पष्टीकरणों के रूप में आउटपुट करने के लिए कहकर प्राप्त किया जा सकता है। उदाहरण के लिए, एक स्व-उपचार डेटा कार्यकर्ता एक रिपोर्ट तैयार कर सकता है जो उसके द्वारा पहचानी गई विशिष्ट डेटा विसंगतियों, लागू किए गए सुधारों और उन सुधारों के पीछे के तर्क को हाइलाइट करता है।
3. **विशेषता महत्व:** एआई मॉडल को उनके निर्देशों के हिस्से के रूप में डेटा सुधार प्रक्रिया में विभिन्न विशेषताओं या गुणों के महत्व के बारे में जानकारी के साथ निर्देशित किया जा सकता है। बदले में, उन निर्देशों को मानव हितधारकों के लिए प्रकट किया जा सकता है। मॉडल के निर्णयों को प्रभावित करने वाले प्रमुख कारकों की पहचान करके, हितधारक सुधारों के पीछे के तर्क में अंतर्दृष्टि प्राप्त कर सकते हैं और उनकी वैधता का आकलन कर सकते हैं।
4. **लॉगिंग और ऑडिटिंग:** स्व-मरम्मत डेटा प्रक्रिया में पारदर्शिता बनाए रखने के लिए व्यापक लॉगिंग और ऑडिटिंग तंत्र का कार्यान्वयन महत्वपूर्ण है। एआई मॉडल द्वारा किए गए प्रत्येक डेटा सुधार को लॉग किया जाना चाहिए, जिसमें मूल डेटा, सुधारित डेटा और की गई विशिष्ट कार्रवाइयां शामिल हैं। यह ऑडिट ट्रेल पूर्वव्यापी विश्लेषण की अनुमति देता है और डेटा में किए गए संशोधनों का स्पष्ट रिकॉर्ड प्रदान करता है।
5. **मानव-सहभागी दृष्टिकोण:** मानव-सहभागी दृष्टिकोण को शामिल करने से स्व-मरम्मत डेटा तकनीकों की व्याख्येयता और पारदर्शिता बढ़ सकती है। एआई-जनित सुधारों की समीक्षा और सत्यापन में मानव विशेषज्ञों को शामिल करके, संगठन यह सुनिश्चित कर सकते हैं कि सुधार डोमेन ज्ञान और व्यावसायिक आवश्यकताओं के अनुरूप हैं। मानवीय निरीक्षण जवाबदेही की एक अतिरिक्त परत जोड़ता है और एआई मॉडल में किसी भी संभावित पूर्वाग्रह या त्रुटियों की पहचान करने की अनुमति देता है।

6. **निरंतर निगरानी और मूल्यांकन:** पारदर्शिता और विश्वास बनाए रखने के लिए स्व-मरम्मत डेटा तकनीकों के प्रदर्शन की नियमित निगरानी और मूल्यांकन आवश्यक है। समय के साथ एआई मॉडल की सटीकता और प्रभावशीलता का आकलन करके, संगठन किसी भी विचलन या असामान्यताओं की पहचान कर सकते हैं और सुधारात्मक कार्रवाई कर सकते हैं। निरंतर निगरानी यह सुनिश्चित करने में मदद करती है कि स्व-मरम्मत डेटा प्रक्रिया विश्वसनीय बनी रहे और वांछित परिणामों के अनुरूप हो।

स्व-मरम्मत डेटा तकनीकों को लागू करते समय व्याख्येयता और पारदर्शिता महत्वपूर्ण विचार हैं। डेटा सुधारों के लिए स्पष्ट स्पष्टीकरण प्रदान करके, व्यापक ऑडिट ट्रेल बनाए रखकर, और मानवीय निरीक्षण को शामिल करके, संगठन स्व-मरम्मत डेटा प्रक्रिया में विश्वास बना सकते हैं और यह सुनिश्चित कर सकते हैं कि डेटा में किए गए संशोधन न्यायसंगत हैं और व्यावसायिक उद्देश्यों के अनुरूप हैं।

स्वचालन के लाभों और पारदर्शिता की आवश्यकता के बीच संतुलन बनाना महत्वपूर्ण है। जबकि स्व-मरम्मत डेटा तकनीकें डेटा गुणवत्ता और दक्षता में महत्वपूर्ण सुधार कर सकती हैं, यह डेटा सुधार प्रक्रिया पर दृश्यता और नियंत्रण खोने की कीमत पर नहीं होना चाहिए। व्याख्येयता और पारदर्शिता को ध्यान में रखते हुए स्व-मरम्मत डेटा कार्यकर्ताओं को डिज़ाइन करके, संगठन एआई की शक्ति का उपयोग कर सकते हैं, जबकि डेटा में आवश्यक स्तर की जवाबदेही और विश्वास बनाए रख सकते हैं।

अनपेक्षित परिणाम

जबकि स्व-मरम्मत डेटा तकनीकें डेटा गुणवत्ता और संगति में सुधार करने का लक्ष्य रखती हैं, अनपेक्षित परिणामों की संभावना के प्रति सचेत रहना महत्वपूर्ण है। स्वचालित सुधार, यदि सावधानीपूर्वक डिज़ाइन और निगरानी नहीं की जाती है, तो अनजाने में डेटा के अर्थ या संदर्भ को बदल सकते हैं, जिससे अनुप्रवाही समस्याएं उत्पन्न हो सकती हैं।

स्व-मरम्मत डेटा के प्राथमिक जोखिमों में से एक डेटा सुधार प्रक्रिया में पूर्वाग्रह या त्रुटियों का प्रवेश है। एआई मॉडल, किसी अन्य सॉफ्टवेयर सिस्टम की तरह, प्रशिक्षण

डेटा में मौजूद पूर्वाग्रहों या एल्गोरिदम के डिजाइन के माध्यम से प्रस्तुत पूर्वाग्रहों के अधीन हो सकते हैं। यदि इन पूर्वाग्रहों की पहचान और कम नहीं किया जाता है, तो वे स्व-मरम्मत डेटा प्रक्रिया के माध्यम से फैल सकते हैं और विषम या गलत डेटा संशोधनों का परिणाम हो सकते हैं।

उदाहरण के लिए, ग्राहक जनसांख्यिकीय डेटा में विसंगतियों को सुधारने के लिए नियुक्त एक स्व-सुधार डेटा कार्यकर्ता पर विचार करें। यदि एआई मॉडल ने ऐतिहासिक डेटा से पूर्वाग्रह सीखा है, जैसे कि विशिष्ट व्यवसायों या आय स्तरों को विशेष लिंग या जातियों से जोड़ना, तो यह गलत धारणाएं बना सकता है और डेटा को ऐसे तरीके से संशोधित कर सकता है जो उन पूर्वाग्रहों को मजबूत करता है। यह अशुद्ध ग्राहक प्रोफाइल, भ्रामक व्यावसायिक निर्णय और संभावित भेदभावपूर्ण परिणामों की ओर ले जा सकता है।

एक अन्य संभावित अनपेक्षित परिणाम डेटा सुधार प्रक्रिया के दौरान मूल्यवान जानकारी या संदर्भ का नुकसान है। स्व-सुधार डेटा तकनीकें अक्सर एकरूपता सुनिश्चित करने के लिए डेटा को मानकीकृत और सामान्यीकृत करने पर ध्यान केंद्रित करती हैं। हालांकि, कुछ मामलों में, मूल डेटा में बारीकियां, अपवाद, या संदर्भगत जानकारी हो सकती है जो पूरी तस्वीर को समझने के लिए महत्वपूर्ण है। अंधाधुंध मानकीकरण लागू करने वाले स्वचालित सुधार अनजाने में इस मूल्यवान जानकारी को हटा या धुंधला सकते हैं।

उदाहरण के लिए, चिकित्सा रिकॉर्ड में विसंगतियों को सुधारने के लिए जिम्मेदार एक स्व-सुधार डेटा कार्यकर्ता की कल्पना करें। यदि कार्यकर्ता को किसी दुर्लभ स्थिति या असामान्य उपचार योजना वाले रोगी के चिकित्सा इतिहास का सामना करना पड़ता है, तो यह डेटा को एक अधिक सामान्य पैटर्न में फिट करने के लिए सामान्यीकृत करने का प्रयास कर सकता है। हालांकि, ऐसा करने में, यह उन विशिष्ट विवरणों और संदर्भ को खो सकता है जो रोगी की अनूठी स्थिति को सटीक रूप से दर्शाने के लिए महत्वपूर्ण हैं। इस जानकारी की हानि रोगी देखभाल और चिकित्सकीय निर्णय लेने के लिए गंभीर प्रभाव डाल सकती है।

अनपेक्षित परिणामों के जोखिमों को कम करने के लिए, स्व-सुधार डेटा तकनीकों को डिज़ाइन और कार्यान्वित करते समय एक सक्रिय दृष्टिकोण अपनाना आवश्यक है:

1. **व्यापक परीक्षण और सत्यापन:** उत्पादन में स्व-सुधार डेटा कार्यकर्ताओं को तैनात करने से पहले, विभिन्न परिदृश्यों के विरुद्ध उनके व्यवहार का व्यापक परीक्षण और सत्यापन करना महत्वपूर्ण है। इसमें विभिन्न सीमांत मामलों, अपवादों और संभावित पूर्वाग्रहों को कवर करने वाले प्रतिनिधि डेटासेट के साथ परीक्षण शामिल है। कठोर परीक्षण वास्तविक दुनिया के डेटा को प्रभावित करने से पहले किसी भी अनपेक्षित परिणाम की पहचान करने और उन्हें संबोधित करने में मदद करता है।
2. **निरंतर निगरानी और मूल्यांकन:** समय के साथ अनपेक्षित परिणामों का पता लगाने और उन्हें कम करने के लिए निरंतर निगरानी और मूल्यांकन तंत्र को लागू करना आवश्यक है। स्व-सुधार डेटा प्रक्रियाओं के परिणामों की नियमित समीक्षा, अनुप्रवाही प्रणालियों और निर्णय लेने पर प्रभाव का विश्लेषण, और हितधारकों से प्रतिक्रिया एकत्र करना किसी भी प्रतिकूल प्रभाव की पहचान करने और समय पर सुधारात्मक कार्रवाई को प्रेरित करने में मदद कर सकता है। यदि आपके संगठन में परिचालन डैशबोर्ड हैं, तो स्वचालित डेटा परिवर्तनों से संबंधित स्पष्ट रूप से दिखाई देने वाले मेट्रिक्स जोड़ना शायद एक अच्छा विचार है। सामान्य डेटा परिवर्तन गतिविधि से बड़े विचलन से जुड़े अलार्म जोड़ना शायद और भी बेहतर विचार है!
3. **मानवीय निरीक्षण और हस्तक्षेप:** स्व-सुधार डेटा प्रक्रिया में मानवीय निरीक्षण और हस्तक्षेप की क्षमता बनाए रखना महत्वपूर्ण है। जबकि स्वचालन दक्षता को बहुत बढ़ा सकता है, विशेष रूप से महत्वपूर्ण या संवेदनशील डोमेन में, एआई मॉडल द्वारा किए गए सुधारों की समीक्षा और सत्यापन के लिए मानव विशेषज्ञों का होना महत्वपूर्ण है। मानवीय निर्णय और डोमेन विशेषज्ञता किसी भी उत्पन्न होने वाले अनपेक्षित परिणामों की पहचान करने और उन्हें संबोधित करने में मदद कर सकती है।

4. **व्याख्या योग्य एआई (XAI) और पारदर्शिता:** जैसा कि पिछले उपखंड में चर्चा की गई है, स्व-उपचारात्मक डेटा प्रक्रिया में व्याख्या योग्य एआई तकनीकों को शामिल करना और पारदर्शिता सुनिश्चित करना अनपेक्षित परिणामों को कम करने में मदद कर सकता है। डेटा सुधारों के लिए स्पष्ट स्पष्टीकरण प्रदान करके

और व्यापक ऑडिट ट्रेल बनाए रखकर, संगठन एआई मॉडल द्वारा किए गए संशोधनों के पीछे के तर्क को बेहतर ढंग से समझ और ट्रैक कर सकते हैं।

5. **क्रमिक और पुनरावर्ती दृष्टिकोण:** स्व-उपचारात्मक डेटा के लिए एक क्रमिक और पुनरावर्ती दृष्टिकोण अपनाने से अनपेक्षित परिणामों के जोखिम को कम करने में मदद मिल सकती है। एक बार में पूरे डेटासेट पर स्वचालित सुधार लागू करने के बजाय, डेटा के एक उपसमुच्चय से शुरू करें और जैसे-जैसे तकनीकें प्रभावी और विश्वसनीय साबित होती हैं, धीरे-धीरे इसके दायरे का विस्तार करें। यह रास्ते में सावधानीपूर्वक निगरानी और समायोजन की अनुमति देता है, जिससे किसी भी अनपेक्षित परिणाम का प्रभाव कम होता है।
6. **सहयोग और प्रतिक्रिया:** विभिन्न डोमेन के हितधारकों को शामिल करना और स्व-उपचारात्मक डेटा प्रक्रिया के दौरान सहयोग और प्रतिक्रिया को प्रोत्साहित करना अनपेक्षित परिणामों की पहचान और समाधान में मदद कर सकता है। डोमेन विशेषज्ञों, डेटा उपभोक्ताओं और अंतिम उपयोगकर्ताओं से नियमित रूप से इनपुट लेना डेटा सुधारों के वास्तविक प्रभाव के बारे में मूल्यवान अंतर्दृष्टि प्रदान कर सकता है और किसी भी ऐसी समस्या को उजागर कर सकता है जो अनदेखी रह गई हो।

अनपेक्षित परिणामों के जोखिम को सक्रिय रूप से संबोधित करके और उचित सुरक्षा उपायों को लागू करके, संगठन संभावित प्रतिकूल प्रभावों को कम करते हुए स्व-उपचारात्मक डेटा तकनीकों के लाभों का उपयोग कर सकते हैं। स्व-उपचारात्मक डेटा को एक पुनरावर्ती और सहयोगात्मक प्रक्रिया के रूप में देखना महत्वपूर्ण है, जिसमें वांछित परिणामों के साथ संरेखित करने और डेटा की अखंडता और विश्वसनीयता बनाए रखने के लिए तकनीकों की लगातार निगरानी, मूल्यांकन और परिष्करण किया जाता है।

स्व-उपचारात्मक डेटा पैटर्न के उपयोग पर विचार करते समय, इन कारकों का सावधानीपूर्वक मूल्यांकन करना और संभावित जोखिमों और सीमाओं के विरुद्ध लाभों का मूल्यांकन करना आवश्यक है। कुछ मामलों में, स्वचालित सुधारों को

मानवीय निरीक्षण और हस्तक्षेप के साथ जोड़ने वाला एक हाइब्रिड दृष्टिकोण सबसे उपयुक्त समाधान हो सकता है।

यह भी ध्यान रखना महत्वपूर्ण है कि स्व-उपचारात्मक डेटा तकनीकों को मजबूत डेटा सत्यापन, इनपुट सैनिटाइज़ेशन और त्रुटि प्रबंधन तंत्र के विकल्प के रूप में नहीं देखा जाना चाहिए। डेटा की अखंडता और सुरक्षा सुनिश्चित करने के लिए ये मूलभूत प्रथाएं महत्वपूर्ण बनी रहती हैं। स्व-उपचारात्मक डेटा को एक पूरक दृष्टिकोण के रूप में देखा जाना चाहिए जो इन मौजूदा उपायों को बढ़ा और बेहतर बना सकता है।

अंततः, स्व-उपचारात्मक डेटा पैटर्न का उपयोग करने का निर्णय आपके एप्लिकेशन की विशिष्ट आवश्यकताओं, बाधाओं और प्राथमिकताओं पर निर्भर करता है। ऊपर बताए गए विचारों पर सावधानीपूर्वक विचार करके और उन्हें अपने एप्लिकेशन के लक्ष्यों और आर्किटेक्चर के साथ संरेखित करके, आप स्व-उपचारात्मक डेटा तकनीकों का प्रभावी ढंग से उपयोग करने के लिए कब और कैसे लाभ उठाना है, इस पर सूचित निर्णय ले सकते हैं।

संदर्भ-आधारित सामग्री निर्माण

संदर्भ-आधारित सामग्री निर्माण पैटर्न बृहत भाषा मॉडल (LLMs) की शक्ति का उपयोग एप्लिकेशन के भीतर गतिशील और संदर्भ-विशिष्ट सामग्री उत्पन्न करने के लिए करते हैं। पैटर्न की यह श्रेणी उपयोगकर्ताओं की विशिष्ट आवश्यकताओं, प्राथमिकताओं, और यहां तक कि एप्लिकेशन के साथ उनकी पिछली और वर्तमान अंतःक्रियाओं के आधार पर वैयक्तिकृत और प्रासंगिक सामग्री प्रदान करने के महत्व को पहचानती है।

> इस दृष्टिकोण के संदर्भ में, “सामग्री” का अर्थ प्राथमिक सामग्री (जैसे ब्लॉग पोस्ट, लेख, आदि) और मेटा-सामग्री, जैसे प्राथमिक सामग्री के लिए सिफारिशें, दोनों से है।

संदर्भ-आधारित सामग्री निर्माण पैटर्न आपके उपयोगकर्ता जुड़ाव स्तरों को बढ़ाने,

अनुकूलित अनुभव प्रदान करने, और आपके और आपके उपयोगकर्ताओं के लिए सामग्री निर्माण कार्यों को स्वचालित करने में महत्वपूर्ण भूमिका निभा सकते हैं। इस अध्याय में वर्णित पैटर्न का उपयोग करके, आप ऐसे एप्लिकेशन बना सकते हैं जो गतिशील रूप से सामग्री उत्पन्न करते हैं, वास्तविक समय में संदर्भ और इनपुट के अनुकूल होते हैं।

ये पैटर्न LLMs को एप्लिकेशन के आउटपुट में एकीकृत करके काम करते हैं, जो उपयोगकर्ता इंटरफ़ेस (जिसे कभी-कभी "क्रोम" के रूप में संदर्भित किया जाता है) से लेकर ईमेल और अन्य प्रकार की सूचनाओं, साथ ही किसी भी सामग्री निर्माण पाइपलाइन तक फैले होते हैं।

जब कोई उपयोगकर्ता एप्लिकेशन के साथ बातचीत करता है या किसी विशिष्ट सामग्री अनुरोध को ट्रिगर करता है, तो एप्लिकेशन प्रासंगिक संदर्भ को कैप्चर करता है, जैसे उपयोगकर्ता प्राथमिकताएं, पिछली बातचीत, या विशिष्ट प्रॉम्प्ट। यह संदर्भगत जानकारी फिर LLM में किसी भी आवश्यक टेम्पलेट या दिशानिर्देशों के साथ फीड की जाती है और टेक्स्ट आउटपुट उत्पन्न करने के लिए उपयोग की जाती है जो अन्यथा या तो हार्डकोडेड होना पड़ता, डेटाबेस में संग्रहीत होना पड़ता, या एल्गोरिथ्मिक रूप से उत्पन्न किया जाना पड़ता।

LLM द्वारा उत्पन्न सामग्री विभिन्न रूप ले सकती है, जैसे वैयक्तिकृत सिफारिशें, गतिशील उत्पाद विवरण, अनुकूलित ईमेल प्रतिक्रियाएं, या यहां तक कि पूरे लेख या ब्लॉग पोस्ट। इस सामग्री के सबसे क्रांतिकारी उपयोगों में से एक, जिसे मैंने एक वर्ष से अधिक पहले शुरू किया था, फॉर्म लेबल, टूलटिप्स और अन्य प्रकार के व्याख्यात्मक टेक्स्ट जैसे UI तत्वों को गतिशील रूप से उत्पन्न करना है।

वैयक्तिकरण

संदर्भ-आधारित सामग्री निर्माण पैटर्न के प्रमुख लाभों में से एक उपयोगकर्ताओं को अत्यधिक वैयक्तिकृत अनुभव प्रदान करने की क्षमता है। उपयोगकर्ता-विशिष्ट संदर्भ के आधार पर सामग्री उत्पन्न करके, ये पैटर्न एप्लिकेशन को व्यक्तिगत उपयोगकर्ताओं

की रुचियों, प्राथमिकताओं और बातचीत के अनुरूप सामग्री को अनुकूलित करने में सक्षम बनाते हैं।

वैयक्तिकरण सिर्फ सामान्य सामग्री में उपयोगकर्ता का नाम डालने से कहीं आगे जाता है। इसमें प्रत्येक उपयोगकर्ता के बारे में उपलब्ध समृद्ध संदर्भ का उपयोग करके ऐसी सामग्री तैयार करना शामिल है जो उनकी विशिष्ट आवश्यकताओं और इच्छाओं के अनुरूप हो। इस संदर्भ में कई कारक शामिल हो सकते हैं, जैसे:

1. **उपयोगकर्ता प्रोफ़ाइल जानकारी:** इस तकनीक को लागू करने के सबसे सामान्य स्तर पर, जनसांख्यिकीय डेटा, रुचियां, प्राथमिकताएं और अन्य प्रोफ़ाइल विशेषताओं का उपयोग उपयोगकर्ता की पृष्ठभूमि और विशेषताओं के अनुरूप सामग्री तैयार करने के लिए किया जा सकता है।
2. **व्यवहार संबंधी डेटा:** एप्लिकेशन के साथ उपयोगकर्ता की पिछली बातचीत, जैसे देखे गए पृष्ठ, क्लिक किए गए लिंक, या खरीदे गए उत्पाद, उनके व्यवहार और रुचियों के बारे में मूल्यवान जानकारी प्रदान कर सकते हैं। इस डेटा का उपयोग ऐसी सामग्री सुझाव तैयार करने के लिए किया जा सकता है जो उनके संलग्नता पैटर्न को दर्शाती है और उनकी भविष्य की जरूरतों की भविष्यवाणी करती है।
3. **संदर्भात्मक कारक:** उपयोगकर्ता का वर्तमान संदर्भ, जैसे उनका स्थान, डिवाइस, दिन का समय, या यहां तक कि मौसम, सामग्री निर्माण प्रक्रिया को प्रभावित कर सकता है। उदाहरण के लिए, एक यात्रा एप्लिकेशन में एक ऐसा AI कार्यकर्ता हो सकता है जो उपयोगकर्ता के वर्तमान स्थान और मौजूदा मौसम की स्थितियों के आधार पर वैयक्तिकृत सिफारिशें तैयार कर सकता है।

इन संदर्भात्मक कारकों का लाभ उठाकर, संदर्भात्मक सामग्री निर्माण पैटर्न एप्लिकेशन को प्रत्येक व्यक्तिगत उपयोगकर्ता के लिए विशेष रूप से बनाई गई सामग्री प्रदान करने में सक्षम बनाते हैं। वैयक्तिकरण के इस स्तर के कई महत्वपूर्ण लाभ हैं:

1. **बढ़ी हुई सहभागिता:** वैयक्तिकृत सामग्री उपयोगकर्ताओं का ध्यान आकर्षित करती है और उन्हें एप्लिकेशन के साथ जुड़े रहने में मदद करती है। जब उपयोगकर्ताओं को लगता है कि सामग्री प्रासंगिक है और सीधे उनकी जरूरतों

की बात करती है, तो वे एप्लिकेशन के साथ अधिक समय बिताने और इसकी विशेषताओं का पता लगाने की अधिक संभावना रखते हैं।

2. **बेहतर उपयोगकर्ता संतुष्टि:** वैयक्तिकृत सामग्री दर्शाती है कि एप्लिकेशन उपयोगकर्ता की विशिष्ट आवश्यकताओं को समझता है और उनकी परवाह करता है। उपयोगकर्ता की रुचियों के अनुरूप सहायक और जानकारीपूर्ण सामग्री प्रदान करके, एप्लिकेशन उपयोगकर्ता संतुष्टि को बढ़ा सकता है और अपने उपयोगकर्ताओं के साथ मजबूत संबंध बना सकता है।
3. **उच्च रूपांतरण दर:** ई-कॉमर्स या मार्केटिंग एप्लिकेशन के संदर्भ में, वैयक्तिकृत सामग्री रूपांतरण दर को महत्वपूर्ण रूप से प्रभावित कर सकती है। उपयोगकर्ताओं को उनकी प्राथमिकताओं और व्यवहार के अनुरूप उत्पाद, ऑफ़र या सिफारिशें प्रस्तुत करके, एप्लिकेशन उपयोगकर्ताओं द्वारा वांछित कार्रवाई करने की संभावना को बढ़ा सकता है, जैसे खरीदारी करना या किसी सेवा के लिए साइन अप करना।

उत्पादकता

संदर्भात्मक सामग्री निर्माण पैटर्न रचनात्मक प्रक्रियाओं में मैनुअल सामग्री निर्माण और संपादन की आवश्यकता को कम करके कुछ प्रकार की उत्पादकता को काफी बढ़ा सकते हैं। बृहत भाषा मॉडल की शक्ति का लाभ उठाकर, आप बड़े पैमाने पर उच्च-गुणवत्ता वाली सामग्री तैयार कर सकते हैं, जिससे आपके सामग्री निर्माताओं और डेवलपर्स को उबाऊ मैनुअल काम में खर्च करना पड़ने वाला समय और प्रयास बच जाता है।

परंपरागत रूप से, सामग्री निर्माताओं को एप्लिकेशन की आवश्यकताओं और उपयोगकर्ता की अपेक्षाओं को पूरा करने के लिए अनुसंधान, लेखन, संपादन और प्रारूपण करना पड़ता है। यह प्रक्रिया समय लेने वाली और संसाधन-गहन हो सकती है, विशेष रूप से जब सामग्री की मात्रा बढ़ती है।

हालांकि, संदर्भ-आधारित सामग्री निर्माण पैटर्न के साथ, सामग्री निर्माण प्रक्रिया को काफी हद तक स्वचालित किया जा सकता है। एलएलएम दिए गए प्रॉम्प्ट्स और

दिशानिर्देशों के आधार पर सुसंगत, व्याकरण की दृष्टि से सही, और संदर्भगत रूप से प्रासंगिक सामग्री उत्पन्न कर सकते हैं। यह स्वचालन कई उत्पादकता लाभ प्रदान करता है:

1. **कम मैनुअल प्रयास:** सामग्री निर्माण कार्यों को एलएलएम को सौंपकर, सामग्री निर्माता सामग्री रणनीति, विचार-निर्माण और गुणवत्ता आश्वासन जैसे उच्च-स्तरीय कार्यों पर ध्यान केंद्रित कर सकते हैं। वे एलएलएम को आवश्यक संदर्भ, टेम्पलेट्स और दिशानिर्देश प्रदान कर सकते हैं और वास्तविक सामग्री निर्माण का काम उस पर छोड़ सकते हैं। यह लेखन और संपादन के लिए आवश्यक मैनुअल प्रयास को कम करता है, जिससे सामग्री निर्माता अधिक उत्पादक और कुशल हो सकते हैं।
2. **तेज सामग्री निर्माण:** एलएलएम मानव लेखकों की तुलना में बहुत तेजी से सामग्री उत्पन्न कर सकते हैं। सही प्रॉम्प्ट्स और दिशानिर्देशों के साथ, एक एलएलएम कुछ सेकंड या मिनटों में कई सामग्री टुकड़े उत्पन्न कर सकता है। यह गति एप्लिकेशन को बहुत तेजी से सामग्री उत्पन्न करने में सक्षम बनाती है, जो उपयोगकर्ताओं की मांगों और लगातार बदलते डिजिटल परिदृश्य के साथ तालमेल बनाए रखती है।

> क्या तेज सामग्री निर्माण "कॉमन्स की त्रासदी" जैसी स्थिति की ओर ले जा रहा है जहां इंटरनेट ऐसी सामग्री से भर गया है जिसे कोई नहीं पढ़ता। दुर्भाग्य से, मुझे लगता है कि इसका जवाब हां है।

3. **निरंतरता और गुणवत्ता:** एलएलएम आसानी से सामग्री को संशोधित कर सकते हैं ताकि वह शैली, स्वर और गुणवत्ता में एकरूप हो। स्पष्ट दिशानिर्देश और उदाहरण प्रदान किए जाने पर, कुछ प्रकार के एप्लिकेशन (जैसे न्यूजरूम, पीआर, आदि) यह सुनिश्चित कर सकते हैं कि उनकी मानव-निर्मित सामग्री उनकी ब्रांड वॉइस के अनुरूप हो और वांछित गुणवत्ता मानकों को पूरा करे। यह निरंतरता

व्यापक संपादन और संशोधनों की आवश्यकता को कम करती है, जिससे सामग्री निर्माण प्रक्रिया में समय और प्रयास की बचत होती है।

4. **पुनरावृत्ति और अनुकूलन:** संदर्भ-आधारित सामग्री निर्माण पैटर्न सामग्री की त्वरित पुनरावृत्ति और अनुकूलन को सक्षम बनाते हैं। एलएलएम को प्रदान किए गए प्रॉम्प्ट्स, टेम्पलेट्स या दिशानिर्देशों को समायोजित करके, आपके एप्लिकेशन त्वरित रूप से सामग्री के विविध रूप उत्पन्न कर सकते हैं और विभिन्न दृष्टिकोणों का स्वचालित तरीके से परीक्षण कर सकते हैं, जो पहले कभी संभव नहीं था। यह पुनरावर्ती प्रक्रिया सामग्री रणनीतियों के तेज प्रयोग और परिष्करण की अनुमति देती है, जिससे समय के साथ अधिक प्रभावी और आकर्षक सामग्री बनती है। यह विशेष तकनीक ई-कॉमर्स जैसे एप्लिकेशन के लिए गेम-चेंजर हो सकती है जो बाउंस रेट और एंगेजमेंट के आधार पर जीते और मरते हैं।

यह ध्यान रखना महत्वपूर्ण है कि जबकि प्रासंगिक सामग्री निर्माण पैटर्न उत्पादकता को बहुत बढ़ा सकते हैं, वे मानवीय भागीदारी की आवश्यकता को पूरी तरह से समाप्त नहीं करते। सामग्री निर्माता और संपादक अभी भी समग्र सामग्री रणनीति को परिभाषित करने, LLM को मार्गदर्शन प्रदान करने, और उत्पन्न सामग्री की गुणवत्ता और उपयुक्तता सुनिश्चित करने में महत्वपूर्ण भूमिका निभाते हैं।

सामग्री निर्माण के अधिक दोहराव वाले और समय लेने वाले पहलुओं को स्वचालित करके, प्रासंगिक सामग्री निर्माण पैटर्न मूल्यवान मानवीय समय और संसाधनों को मुक्त करते हैं जिन्हें उच्च-मूल्य कार्यों की ओर पुनर्निर्देशित किया जा सकता है। यह बढ़ी हुई उत्पादकता आपको सामग्री निर्माण कार्यप्रवाह को अनुकूलित करते हुए उपयोगकर्ताओं को अधिक वैयक्तिकृत और आकर्षक सामग्री प्रदान करने में सक्षम बनाती है।

त्वरित पुनरावृत्ति और प्रयोग

प्रासंगिक सामग्री निर्माण पैटर्न आपको विभिन्न सामग्री विविधताओं के साथ तेजी से पुनरावृत्ति और प्रयोग करने में सक्षम बनाते हैं, जिससे आपकी सामग्री रणनीति का

तेज अनुकूलन और परिष्करण संभव होता है। आप केवल मॉडल को प्रदान किए गए संदर्भ, टेम्पलेट्स, या दिशानिर्देशों को समायोजित करके, कुछ ही सेकंड में सामग्री के कई संस्करण उत्पन्न कर सकते हैं।

यह त्वरित पुनरावृत्ति क्षमता कई प्रमुख लाभ प्रदान करती है:

1. **परीक्षण और अनुकूलन:** सामग्री विविधताओं को तेजी से उत्पन्न करने की क्षमता के साथ, आप आसानी से विभिन्न दृष्टिकोणों का परीक्षण कर सकते हैं और उनकी प्रभावशीलता को माप सकते हैं। उदाहरण के लिए, आप एक उत्पाद विवरण या मार्केटिंग संदेश के कई संस्करण उत्पन्न कर सकते हैं, जो प्रत्येक विशिष्ट उपयोगकर्ता खंड या संदर्भ के लिए अनुकूलित हो। उपयोगकर्ता संलग्नता मैट्रिक्स का विश्लेषण करके, जैसे क्लिक-थ्रू दर या रूपांतरण दर, आप सबसे प्रभावी सामग्री विविधताओं की पहचान कर सकते हैं और तदनुसार अपनी सामग्री रणनीति को अनुकूलित कर सकते हैं।

2. **A/B टेस्टिंग:** प्रासंगिक सामग्री निर्माण पैटर्न सामग्री का निर्बाध A/B परीक्षण सक्षम करते हैं। आप सामग्री के दो या अधिक विविधताएं उत्पन्न कर सकते हैं और उन्हें विभिन्न उपयोगकर्ता समूहों को यादृच्छिक रूप से प्रदान कर सकते हैं। प्रत्येक विविधता के प्रदर्शन की तुलना करके, आप निर्धारित कर सकते हैं कि कौन सी सामग्री आपके लक्षित दर्शकों के साथ सबसे अच्छा प्रतिध्वनित होती है। यह डेटा-संचालित दृष्टिकोण आपको सूचित निर्णय लेने और उपयोगकर्ता संलग्नता को अधिकतम करने और अपने वांछित परिणामों को प्राप्त करने के लिए अपनी सामग्री को निरंतर परिष्कृत करने की अनुमति देता है।
3. **वैयक्तिकरण प्रयोग:** त्वरित पुनरावृत्ति और प्रयोग वैयक्तिकरण के मामले में विशेष रूप से मूल्यवान हैं। प्रासंगिक सामग्री निर्माण पैटर्न के साथ, आप विभिन्न उपयोगकर्ता खंडों, प्राथमिकताओं, या व्यवहारों के आधार पर तेजी से वैयक्तिकृत सामग्री विविधताएं उत्पन्न कर सकते हैं। विभिन्न वैयक्तिकरण रणनीतियों के साथ प्रयोग करके, आप व्यक्तिगत उपयोगकर्ताओं को आकर्षित करने और अनुकूलित अनुभव प्रदान करने के लिए सबसे प्रभावी दृष्टिकोणों की पहचान कर सकते हैं।
4. **बदलते रुझानों के अनुकूल होना:** तेजी से पुनरावृत्ति और प्रयोग करने की क्षमता आपको चपल रहने और बदलते रुझानों और उपयोगकर्ता प्राथमिकताओं के

अनुकूल होने में सक्षम बनाती है। जैसे-जैसे नए विषय, कीवर्ड, या उपयोगकर्ता व्यवहार उभरते हैं, आप तेजी से इन रुझानों के अनुरूप सामग्री उत्पन्न कर सकते हैं। अपनी सामग्री के साथ निरंतर प्रयोग और परिष्करण करके, आप लगातार विकसित हो रहे डिजिटल परिदृश्य में प्रासंगिक बने रह सकते हैं और प्रतिस्पर्धात्मक बढ़त बनाए रख सकते हैं।

5. **किफायती प्रयोग:** पारंपरिक सामग्री प्रयोग में अक्सर काफी समय और संसाधन लगते हैं, क्योंकि सामग्री निर्माताओं को विभिन्न प्रकार के विकल्पों को मैन्युअल रूप से विकसित और परीक्षण करने की आवश्यकता होती है। हालांकि, संदर्भात्मक सामग्री निर्माण पैटर्न के साथ, प्रयोग की लागत काफी कम हो जाती है। LLMs तेजी से और बड़े पैमाने पर सामग्री विविधताएं उत्पन्न कर सकते हैं, जिससे आप बिना अधिक लागत के विचारों और दृष्टिकोणों की एक विस्तृत श्रृंखला का पता लगा सकते हैं।

त्वरित पुनरावृत्ति और प्रयोग का सर्वोत्तम लाभ उठाने के लिए, एक सुपरिभाषित प्रयोग ढांचा होना महत्वपूर्ण है। इस ढांचे में शामिल होना चाहिए:

- प्रत्येक प्रयोग के लिए स्पष्ट उद्देश्य और परिकल्पनाएं
- सामग्री प्रदर्शन को मापने के लिए उपयुक्त मैट्रिक्स और ट्रैकिंग तंत्र
- सही उपयोगकर्ताओं को प्रासंगिक सामग्री विविधताएं प्रदान करने के लिए विभाजन और लक्ष्यीकरण रणनीतियां
- प्रयोगात्मक डेटा से अंतर्दृष्टि प्राप्त करने के लिए विश्लेषण और रिपोर्टिंग उपकरण
- आपकी सामग्री रणनीति में सीख और अनुकूलन को शामिल करने की प्रक्रिया

त्वरित पुनरावृत्ति और प्रयोग को अपनाकर, आप अपनी सामग्री को लगातार परिष्कृत और अनुकूलित कर सकते हैं, यह सुनिश्चित करते हुए कि यह आपके एप्लिकेशन के लक्ष्यों को प्राप्त करने में आकर्षक, प्रासंगिक और प्रभावी बनी रहे। सामग्री निर्माण का यह चुस्त दृष्टिकोण आपको अग्रणी बने रहने और असाधारण उपयोगकर्ता अनुभव प्रदान करने में मदद करता है।

मापनीयता और दक्षता

जैसे-जैसे एप्लिकेशन बढ़ते हैं और व्यक्तिगत सामग्री की मांग बढ़ती है, संदर्भात्मक सामग्री निर्माण पैटर्न सामग्री निर्माण के कुशल स्केलिंग को सक्षम बनाते हैं। LLMs एक साथ बड़ी संख्या में उपयोगकर्ताओं और संदर्भों के लिए सामग्री उत्पन्न कर सकते हैं, बिना मानव संसाधनों में आनुपातिक वृद्धि की आवश्यकता के। यह मापनीयता एप्लिकेशन को बढ़ते उपयोगकर्ता आधार के लिए व्यक्तिगत अनुभव प्रदान करने की अनुमति देती है, बिना उनकी सामग्री निर्माण क्षमताओं पर दबाव डाले।

ध्यान दें कि संदर्भात्मक सामग्री निर्माण का उपयोग आपके एप्लिकेशन को "ऑन द फ्लाई" अंतर्राष्ट्रीयकृत करने के लिए प्रभावी ढंग से किया जा सकता है। वास्तव में, यही मैंने अपने Instant18n Gem का उपयोग करके Olympia को आधा दर्जन से अधिक भाषाओं में प्रदान करने के लिए किया, भले ही हम एक साल से भी कम पुराने हैं।

एआई संचालित स्थानीयकरण

यदि आप मुझे एक क्षण के लिए गर्व करने की अनुमति दें, तो मुझे लगता है कि रेल्स ऐप्स के लिए मेरी Instant18n लाइब्रेरी "संदर्भात्मक सामग्री निर्माण" पैटर्न का एक क्रांतिकारी उदाहरण है, जो एप्लिकेशन विकास में एआई की परिवर्तनकारी क्षमता को प्रदर्शित करता है। यह जेम OpenAI के GPT बृहत-भाषा मॉडल की शक्ति का लाभ उठाकर रेल्स एप्लिकेशन में अंतर्राष्ट्रीयकरण और स्थानीयकरण को संभालने के तरीके में क्रांति लाता है।

पारंपरिक रूप से, एक रेल्स एप्लिकेशन का अंतर्राष्ट्रीयकरण करने में अनुवाद कुंजियों को मैन्युअल रूप से परिभाषित करना और प्रत्येक समर्थित भाषा के लिए संबंधित अनुवाद प्रदान करना शामिल है। यह प्रक्रिया समय लेने वाली, संसाधन-गहन और असंगतियों के प्रति संवेदनशील हो सकती है। हालांकि, Instant18n जेम के साथ, स्थानीयकरण का प्रतिमान पूरी तरह से पुनर्परिभाषित किया गया है।

एक बृहत भाषा मॉडल को एकीकृत करके, Instant18n जेम आपको संदर्भ और पाठ के अर्थ के आधार पर तत्काल अनुवाद उत्पन्न करने में सक्षम बनाता है। पूर्व-परिभाषित अनुवाद कुंजियों और स्थिर अनुवादों पर निर्भर रहने के बजाय, जेम एआई की शक्ति का उपयोग करके पाठ का गतिशील रूप से अनुवाद करता है। इस दृष्टिकोण के कई प्रमुख लाभ हैं:

1. **निर्बाध स्थानीयकरण:** Instant18n जेम के साथ, डेवलपर्स को प्रत्येक समर्थित भाषा के लिए अनुवाद फ़ाइलों को मैन्युअल रूप से परिभाषित और बनाए रखने की आवश्यकता नहीं है। जेम स्वचालित रूप से प्रदान किए गए पाठ और वांछित लक्ष्य भाषा के आधार पर अनुवाद उत्पन्न करता है, जिससे स्थानीयकरण प्रक्रिया सरल और निर्बाध हो जाती है।
2. **संदर्भगत सटीकता:** एआई को अनुवाद किए जा रहे पाठ की बारीकियों को समझने के लिए पर्याप्त संदर्भ दिया जा सकता है। यह आस-पास के संदर्भ, मुहावरों और सांस्कृतिक संदर्भों को ध्यान में रखकर ऐसे अनुवाद उत्पन्न कर सकता है जो सटीक, स्वाभाविक रूप से लगने वाले और संदर्भ के अनुरूप हों।
3. **व्यापक भाषा समर्थन:** Instant18n जेम GPT के विशाल ज्ञान और भाषाई क्षमताओं का लाभ उठाता है, जो भाषाओं की एक विस्तृत श्रृंखला में अनुवाद को सक्षम बनाता है। स्पेनिश और फ्रेंच जैसी सामान्य भाषाओं से लेकर क्लिंगन और एल्विश जैसी अधिक दुर्लभ या काल्पनिक भाषाओं तक, जेम विभिन्न प्रकार की अनुवाद आवश्यकताओं को संभाल सकता है।
4. **लचीलापन और रचनात्मकता:** जेम पारंपरिक भाषा अनुवादों से आगे जाता है और रचनात्मक और अपरंपरागत स्थानीयकरण विकल्पों की अनुमति देता है। डेवलपर्स विभिन्न शैलियों, बोलियों, या यहां तक कि काल्पनिक भाषाओं में पाठ का अनुवाद कर सकते हैं, जो अनूठे उपयोगकर्ता अनुभवों और आकर्षक सामग्री के लिए नई संभावनाएं खोलता है।
5. **प्रदर्शन अनुकूलन:** Instant18n जेम प्रदर्शन में सुधार और दोहराए गए अनुवादों के ओवरहेड को कम करने के लिए कैशिंग तंत्र को शामिल करता है। अनुवादित पाठ को कैश किया जाता है, जिससे एक ही अनुवाद के लिए बाद के अनुरोधों को अनावश्यक एपीआई कॉल की आवश्यकता के बिना तेजी से सेवा प्रदान की जा सकती है।

Instant18n जेम एआई का उपयोग करके गतिशील रूप से स्थानीयकृत सामग्री उत्पन्न करने के लिए "संदर्भगत सामग्री निर्माण" पैटर्न की शक्ति को प्रदर्शित करता है। यह दिखाता है कि कैसे एआई को एक रेल्स एप्लिकेशन की मूल कार्यक्षमता में एकीकृत किया जा सकता है, जो डेवलपर्स के अंतर्राष्ट्रीयकरण और स्थानीयकरण के दृष्टिकोण को बदल देता है।

मैनुअल अनुवाद प्रबंधन की आवश्यकता को समाप्त करके और संदर्भ के आधार पर तत्काल अनुवाद को सक्षम करके, Instant18n जेम डेवलपर्स का महत्वपूर्ण समय और प्रयास बचाता है। यह उन्हें अपने एप्लिकेशन की मुख्य विशेषताओं के निर्माण पर ध्यान केंद्रित करने की अनुमति देता है, जबकि यह सुनिश्चित करता है कि स्थानीयकरण पहलू निर्बाध और सटीक रूप से संभाला जाए।

उपयोगकर्ता परीक्षण और प्रतिक्रिया का महत्व

अंत में, उपयोगकर्ता परीक्षण और प्रतिक्रिया के महत्व को हमेशा ध्यान में रखें। यह सत्यापित करना महत्वपूर्ण है कि संदर्भगत सामग्री निर्माण उपयोगकर्ता की अपेक्षाओं को पूरा करता है और एप्लिकेशन के लक्ष्यों के अनुरूप है। उपयोगकर्ता अंतर्दृष्टि और विश्लेषण के आधार पर उत्पन्न सामग्री को लगातार सुधारें और परिष्कृत करें। यदि आप बड़े पैमाने पर गतिशील सामग्री उत्पन्न कर रहे हैं जिसे आप और आपकी टीम द्वारा मैन्युअल रूप से मान्य करना असंभव होगा, तो ऐसे प्रतिक्रिया तंत्र जोड़ने पर विचार करें जो उपयोगकर्ताओं को विचित्र या गलत सामग्री की रिपोर्ट करने की अनुमति देते हैं, साथ ही कारण का स्पष्टीकरण भी। वह बहुमूल्य प्रतिक्रिया एक एआई कार्यकर्ता को भी दी जा सकती है जो सामग्री उत्पन्न करने वाले घटक में समायोजन करने का काम करता है!

जेनरेटिव यूआई

आजकल ध्यान इतना मूल्यवान है कि प्रभावी उपयोगकर्ता संलग्नता के लिए अब न केवल सहज और सरल सॉफ्टवेयर अनुभवों की आवश्यकता है, बल्कि व्यक्तिगत जरूरतों, प्राथमिकताओं और संदर्भों के अनुसार अत्यधिक वैयक्तिकृत होने की भी आवश्यकता है। परिणामस्वरूप, डिजाइनर और डेवलपर्स को ऐसे यूजर इंटरफेस बनाने की चुनौती का सामना करना पड़ रहा है जो प्रत्येक उपयोगकर्ता की विशिष्ट आवश्यकताओं के अनुरूप बड़े पैमाने पर अनुकूल हो सकें।

जेनरेटिव यूआई (GenUI) यूजर इंटरफेस डिजाइन का एक वास्तव में क्रांतिकारी दृष्टिकोण है जो अत्यधिक वैयक्तिकृत और गतिशील उपयोगकर्ता अनुभवों को तत्काल बनाने के लिए लार्ज लैंग्वेज मॉडल्स (LLMs) की शक्ति का लाभ उठाता है। मैं इस पुस्तक में आपको GenUI का कम से कम एक प्राथमिक परिचय देना चाहता था, क्योंकि मेरा मानना है कि यह एप्लिकेशन डिजाइन और फ्रेमवर्क के क्षेत्र में वर्तमान में मौजूद सबसे नई और अछूती संभावनाओं में से एक है। मुझे विश्वास है कि इस विशेष

क्षेत्र में दर्जनों या उससे अधिक नई सफल व्यावसायिक और ओपन-सोर्स परियोजनाएं सामने आएंगी।

मूल रूप से, GenUI संदर्भात्मक सामग्री निर्माण के सिद्धांतों को उन्नत AI तकनीकों के साथ जोड़कर उपयोगकर्ता के संदर्भ, प्राथमिकताओं और लक्ष्यों की गहरी समझ के आधार पर गतिशील रूप से यूजर इंटरफेस तत्वों, जैसे टेक्स्ट, छवियों और लेआउट को उत्पन्न करता है। GenUI डिजाइनरों और डेवलपर्स को ऐसे इंटरफेस बनाने में सक्षम बनाता है जो उपयोगकर्ता के इंटरैक्शन के जवाब में अनुकूल और विकसित होते हैं, जो पहले प्राप्त नहीं किया जा सकता था, ऐसा वैयक्तिकरण प्रदान करते हैं।

GenUI यूजर इंटरफेस डिजाइन के हमारे दृष्टिकोण में एक मौलिक परिवर्तन का प्रतिनिधित्व करता है। जनसमूह के लिए डिजाइन करने के बजाय, GenUI हमें व्यक्ति के लिए डिजाइन करने की अनुमति देता है। वैयक्तिकृत सामग्री और इंटरफेस ऐसे उपयोगकर्ता अनुभव बनाने की क्षमता रखते हैं जो प्रत्येक उपयोगकर्ता के साथ गहरे स्तर पर जुड़ते हैं, जिससे संलग्नता, संतुष्टि और निष्ठा बढ़ती है।

एक अत्याधुनिक तकनीक के रूप में, GenUI में संक्रमण वैचारिक और व्यावहारिक चुनौतियों से भरा हुआ है। डिजाइन प्रक्रिया में AI को एकीकृत करना, यह सुनिश्चित करना कि उत्पन्न इंटरफेस न केवल वैयक्तिकृत हैं बल्कि उपयोग योग्य, सुलभ और समग्र ब्रांड और उपयोगकर्ता अनुभव के अनुरूप भी हैं, ये सभी ऐसी चुनौतियां हैं जो GenUI को कुछ लोगों के लिए बनाती हैं, न कि बहुत के लिए। इसके अतिरिक्त, AI की भागीदारी डेटा गोपनीयता, पारदर्शिता और शायद नैतिक प्रभावों के बारे में भी प्रश्न उठाती है।

चुनौतियों के बावजूद, बड़े पैमाने पर व्यक्तिगत अनुभव डिजिटल उत्पादों और सेवाओं के साथ हमारी अंतःक्रिया को पूरी तरह से बदल सकते हैं। यह समावेशी और सुलभ इंटरफ़ेस बनाने की संभावनाएं खोलता है जो उपयोगकर्ताओं की विविध आवश्यकताओं को पूरा करते हैं, चाहे उनकी क्षमताएं, पृष्ठभूमि या प्राथमिकताएं कुछ भी हों।

इस अध्याय में, हम GenUI की अवधारणा का पता लगाएंगे, कुछ परिभाषित विशेषताओं, प्रमुख लाभों और संभावित चुनौतियों की जांच करेंगे। हम GenUI के सबसे बुनियादी और सुलभ रूप पर विचार करके शुरू करते हैं: पारंपरिक रूप से डिज़ाइन और कार्यान्वित उपयोगकर्ता इंटरफ़ेस के लिए टेक्स्ट कॉपी जनरेट करना।

उपयोगकर्ता इंटरफ़ेस के लिए कॉपी जनरेट करना

आपके एप्लिकेशन के क्रोम में मौजूद टेक्स्ट तत्व, जैसे फ़ॉर्म लेबल, टूलटिप्स और व्याख्यात्मक टेक्स्ट, आमतौर पर टेम्पलेट्स या UI कंपोनेंट्स में हार्डकोड किए जाते हैं, जो सभी उपयोगकर्ताओं के लिए एक समान लेकिन सामान्य अनुभव प्रदान करते हैं। संदर्भ-आधारित सामग्री निर्माण पैटर्न का उपयोग करके, आप इन स्थिर तत्वों को गतिशील, संदर्भ-जागरूक और व्यक्तिगत कंपोनेंट्स में बदल सकते हैं।

व्यक्तिगत फ़ॉर्म

फ़ॉर्म वेब और मोबाइल एप्लिकेशन का एक सर्वव्यापी हिस्सा हैं, जो उपयोगकर्ता इनपुट एकत्र करने का प्राथमिक साधन हैं। हालांकि, पारंपरिक फ़ॉर्म अक्सर एक सामान्य और अवैयक्तिक अनुभव प्रस्तुत करते हैं, जिनमें मानक लेबल और फ़ील्ड होते हैं जो हमेशा उपयोगकर्ता के विशिष्ट संदर्भ या आवश्यकताओं के अनुरूप नहीं हो सकते। उपयोगकर्ता उन फ़ॉर्म को भरने की अधिक संभावना रखते हैं जो उनकी आवश्यकताओं और प्राथमिकताओं के अनुरूप महसूस होते हैं, जिससे उच्च रूपांतरण दर और उपयोगकर्ता संतुष्टि प्राप्त होती है।

हालांकि, व्यक्तिगतकरण और निरंतरता के बीच संतुलन बनाना महत्वपूर्ण है। जबकि व्यक्तिगत उपयोगकर्ताओं के अनुरूप फ़ॉर्म को अनुकूलित करना लाभदायक हो सकता है, परिचितता और अनुमानयोग्यता का स्तर बनाए रखना महत्वपूर्ण है। उपयोगकर्ताओं को अभी भी व्यक्तिगत तत्वों के साथ भी फ़ॉर्म को आसानी से पहचानने और नेविगेट करने में सक्षम होना चाहिए।

यहाँ प्रेरणा के लिए कुछ व्यक्तिगत फ़ॉर्म विचार दिए गए हैं:

संदर्भ-आधारित फ़ील्ड सुझाव

GenUI उपयोगकर्ता की पिछली अंतःक्रियाओं, प्राथमिकताओं और डेटा का विश्लेषण करके बुद्धिमान फ़ील्ड सुझाव प्रदान कर सकता है। उदाहरण के लिए, यदि उपयोगकर्ता

ने पहले अपना शिपिंग पता दर्ज किया है, तो फ़ॉर्म संबंधित फ़ील्ड को उनकी सहेजी गई जानकारी से स्वचालित रूप से भर सकता है। यह न केवल समय बचाता है बल्कि यह भी दर्शाता है कि एप्लिकेशन उपयोगकर्ता की प्राथमिकताओं को समझता और याद रखता है।

एक मिनट, क्या यह तकनीक AI को शामिल किए बिना नहीं की जा सकती? बेशक, लेकिन AI के साथ इस तरह की कार्यक्षमता को संचालित करने की खूबसूरती दो तरह से है: 1) इसे लागू करना कितना आसान हो सकता है और 2) जैसे-जैसे आपका UI बदलता और विकसित होता है, यह कितना लचीला हो सकता है।

आइए हमारे सैद्धांतिक ऑर्डर हैंडलिंग सिस्टम के लिए एक सेवा बनाएं, जो उपयोगकर्ता के लिए सही शिपिंग पता सक्रिय रूप से भरने का प्रयास करती है।

```
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
```

```ruby

  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
```

यह उदाहरण बहुत सरलीकृत है, लेकिन अधिकांश मामलों में काम करेगा। विचार यह है कि AI को उसी तरह अनुमान लगाने दिया जाए जैसे एक इंसान लगाएगा। यह स्पष्ट करने के लिए कि मैं किस बारे में बात कर रहा हूं, आइए कुछ नमूना डेटा पर विचार करें:

```
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"}
]
```

क्या आपने डेटा में पैटर्न देखा? मैं आपको गारंटी देता/देती हूं कि यह एलएलएम के लिए बहुत आसान काम है। इसे प्रदर्शित करने के लिए, चलिए GPT-4 से पूछते हैं कि “थर्मामीटर” के लिए सबसे संभावित शिपिंग पता क्या होगा।

```
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.
```

यदि आप सोच रहे हैं कि इस कार्य के लिए GPT-4 जैसे महंगे मॉडल का उपयोग करना जरूरत से ज्यादा है, तो आप सही हैं! मैंने यही प्रॉम्प्ट Mistral 7B Instruct पर आज़माया और इसने 75 टोकन प्रति सेकंड की गति से, और मात्र $0.000218 USD की मामूली लागत पर निम्नलिखित प्रतिक्रिया दी।

```
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.
```

क्या इस तकनीक का ओवरहेड और लागत चेकआउट अनुभव को अधिक जादुई बनाने के लिए उचित है? कई ऑनलाइन रिटेलर्स के लिए, बिल्कुल। और जैसा कि दिख रहा है, एआई कंप्यूटिंग की लागत केवल कम होती जाएगी, विशेष रूप से कमोडिटी ओपन सोर्स मॉडल होस्टिंग प्रोवाइडर्स के लिए जो सबसे कम कीमत की दौड़ में हैं।

इस तरह के चैट कम्प्लीशन को अनुकूलित करने के लिए प्रॉम्प्ट टेम्पलेट और स्ट्रक्चर्ड आईओ के साथ रेस्पॉन्स फेन्सिंग का उपयोग करें।

अनुकूली फ़ील्ड क्रम

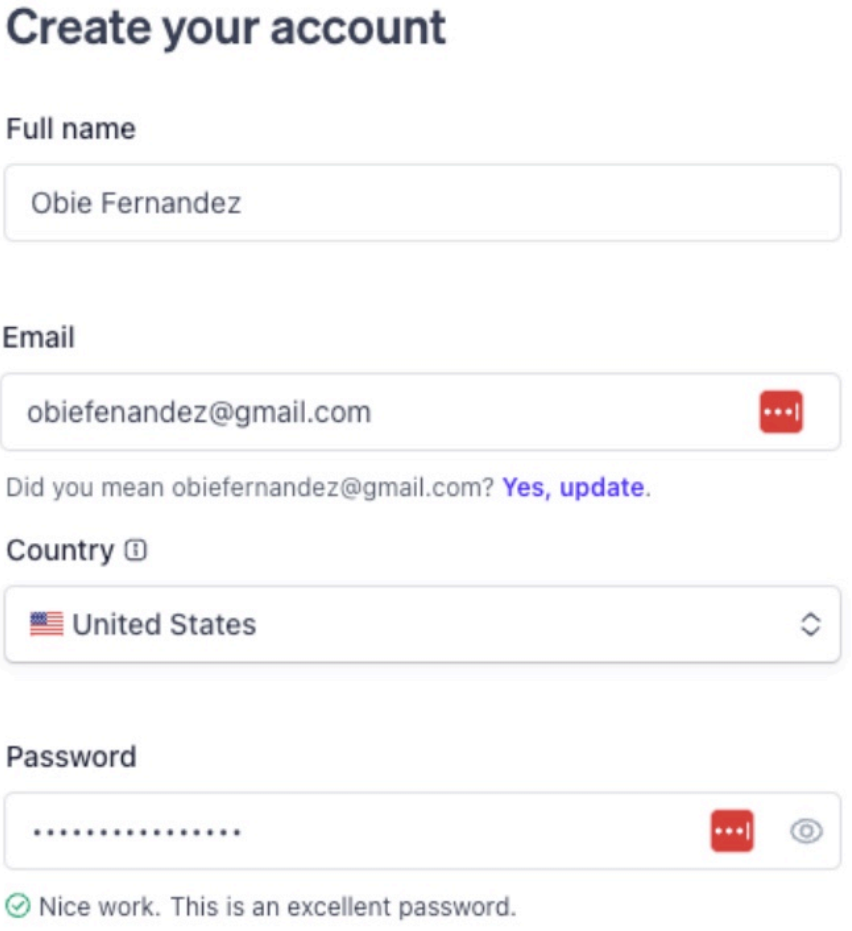
जिस क्रम में फॉर्म फ़ील्ड्स प्रस्तुत की जाती हैं, वह उपयोगकर्ता के अनुभव और पूर्णता दर को महत्वपूर्ण रूप से प्रभावित कर सकती हैं। GenUI के साथ, आप उपयोगकर्ता के संदर्भ और प्रत्येक फ़ील्ड के महत्व के आधार पर फ़ील्ड क्रम को गतिशील रूप से समायोजित कर सकते हैं। उदाहरण के लिए, यदि उपयोगकर्ता एक फिटनेस ऐप के लिए पंजीकरण फॉर्म भर रहा है, तो फॉर्म उनके फिटनेस लक्ष्यों और प्राथमिकताओं से संबंधित फ़ील्ड्स को प्राथमिकता दे सकता है, जिससे प्रक्रिया अधिक प्रासंगिक और आकर्षक बन जाती है।

व्यक्तिगत माइक्रोकॉपी

फॉर्म से जुड़े निर्देशात्मक टेक्स्ट, त्रुटि संदेश और अन्य माइक्रोकॉपी को भी GenUI का उपयोग करके व्यक्तिगत बनाया जा सकता है। “अमान्य ईमेल पता” जैसे सामान्य त्रुटि संदेशों को प्रदर्शित करने के बजाय, आप अधिक सहायक और संदर्भपूर्ण संदेश जनरेट कर सकते हैं जैसे “कृपया अपना ऑर्डर पुष्टिकरण प्राप्त करने के लिए एक वैध ईमेल पता दर्ज करें।” ये व्यक्तिगत छूट फॉर्म अनुभव को अधिक उपयोगकर्ता-मैत्रीपूर्ण और कम निराशाजनक बना सकती हैं।

व्यक्तिगत वैलिडेशन

व्यक्तिगत माइक्रोकॉपी की तरह ही, आप फॉर्म को जादुई तरीके से मान्य करने के लिए एआई का उपयोग कर सकते हैं। कल्पना कीजिए कि एक एआई को उपयोगकर्ता प्रोफ़ाइल फॉर्म को मान्य करने की अनुमति दी जाए, जो सिमैंटिक स्तर पर संभावित गलतियों की जांच करती है।



आकृति 8. क्या आप सिमैंटिक वैलिडेशन को देख सकते हैं?

क्रमिक प्रकटीकरण

GenUI उपयोगकर्ता के संदर्भ के आधार पर यह समझदारी से निर्धारित कर सकता है कि कौन से फॉर्म फ़ील्ड आवश्यक हैं और आवश्यकतानुसार अतिरिक्त फ़ील्ड को धीरे-धीरे प्रकट कर सकता है। यह क्रमिक प्रकटीकरण तकनीक संज्ञानात्मक भार को कम करने में मदद करती है और फॉर्म भरने की प्रक्रिया को अधिक प्रबंधनीय

बनाती है। उदाहरण के लिए, यदि कोई उपयोगकर्ता एक बेसिक सब्सक्रिप्शन के लिए साइन अप कर रहा है, तो फॉर्म शुरू में केवल आवश्यक फ़ील्ड दिखा सकता है, और जैसे-जैसे उपयोगकर्ता आगे बढ़ता है या विशिष्ट विकल्पों का चयन करता है, अतिरिक्त प्रासंगिक फ़ील्ड गतिशील रूप से प्रस्तुत किए जा सकते हैं।

संदर्भ-जागरूक व्याख्यात्मक टेक्स्ट

टूलटिप्स का उपयोग अक्सर उपयोगकर्ताओं को अतिरिक्त जानकारी या मार्गदर्शन प्रदान करने के लिए किया जाता है जब वे विशिष्ट तत्वों पर होवर करते हैं या उनके साथ इंटरैक्ट करते हैं। “संदर्भ-आधारित सामग्री निर्माण” दृष्टिकोण के साथ, आप ऐसे टूलटिप्स जनरेट कर सकते हैं जो उपयोगकर्ता के संदर्भ के अनुकूल होते हैं और प्रासंगिक जानकारी प्रदान करते हैं। उदाहरण के लिए, यदि कोई उपयोगकर्ता एक जटिल सुविधा का पता लगा रहा है, तो टूलटिप उनकी पिछली इंटरैक्शन या कौशल स्तर के आधार पर व्यक्तिगत सुझाव या उदाहरण प्रदान कर सकता है।

व्याख्यात्मक टेक्स्ट, जैसे निर्देश, विवरण, या सहायता संदेश, उपयोगकर्ता के संदर्भ के आधार पर गतिशील रूप से जनरेट किए जा सकते हैं। सामान्य स्पष्टीकरण प्रस्तुत करने के बजाय, आप उपयोगकर्ता की विशिष्ट आवश्यकताओं या प्रश्नों के अनुरूप टेक्स्ट जनरेट करने के लिए LLMs का उपयोग कर सकते हैं। उदाहरण के लिए, यदि कोई उपयोगकर्ता किसी प्रक्रिया में एक विशेष चरण के साथ संघर्ष कर रहा है, तो व्याख्यात्मक टेक्स्ट व्यक्तिगत मार्गदर्शन या समस्या निवारण के सुझाव प्रदान कर सकता है।

माइक्रोकॉपी छोटे टेक्स्ट खंडों को संदर्भित करता है जो उपयोगकर्ताओं को आपके एप्लिकेशन के माध्यम से मार्गदर्शन करते हैं, जैसे बटन लेबल, त्रुटि संदेश, या पुष्टिकरण प्रॉम्प्ट। माइक्रोकॉपी पर संदर्भ-आधारित सामग्री निर्माण दृष्टिकोण को लागू करके, आप एक अनुकूली यूआई बना सकते हैं जो उपयोगकर्ता की क्रियाओं के प्रति प्रतिक्रिया करता है और प्रासंगिक और सहायक टेक्स्ट प्रदान करता है। उदाहरण के लिए, यदि कोई उपयोगकर्ता कोई महत्वपूर्ण कार्रवाई करने वाला है, तो पुष्टिकरण प्रॉम्प्ट को गतिशील रूप से जनरेट किया जा सकता है ताकि एक स्पष्ट और व्यक्तिगत संदेश प्रदान किया जा सके।

व्यक्तिगत व्याख्यात्मक टेक्स्ट और टूलटिप्स नए उपयोगकर्ताओं के लिए ऑनबोर्डिंग प्रक्रिया को काफी बढ़ा सकते हैं। संदर्भ-विशिष्ट मार्गदर्शन और उदाहरण प्रदान करके, आप उपयोगकर्ताओं को एप्लिकेशन को जल्दी से समझने और नेविगेट करने में मदद कर सकते हैं, जिससे सीखने की प्रक्रिया कम होती है और अपनाने की दर बढ़ती है।

गतिशील और संदर्भ-जागरूक क्रोम एलिमेंट्स एप्लिकेशन को अधिक सहज और आकर्षक भी बना सकते हैं। जब साथ में दिया गया टेक्स्ट उनकी विशिष्ट जरूरतों और रुचियों के अनुरूप होता है, तो उपयोगकर्ताओं के सुविधाओं के साथ इंटरैक्ट करने और उनका पता लगाने की संभावना अधिक होती है।

अब तक हमने AI के साथ मौजूदा UI पैराडाइम को बेहतर बनाने के विचारों को कवर किया है, लेकिन उपयोगकर्ता इंटरफ़ेस को एक अधिक मौलिक तरीके से कैसे डिज़ाइन और कार्यान्वित किया जाए, इस बारे में क्या?

जनरेटिव यूआई को परिभाषित करना

पारंपरिक यूआई डिज़ाइन के विपरीत, जहां डिज़ाइनर निश्चित, स्थिर इंटरफ़ेस बनाते हैं, जेनयूआई एक ऐसे भविष्य की ओर इशारा करता है जिसमें हमारा सॉफ्टवेयर लचीले, वैयक्तिकृत अनुभव प्रदान करता है जो वास्तविक समय में विकसित और अनुकूलित हो सकते हैं। जब भी हम एआई-संचालित संवादात्मक इंटरफ़ेस का उपयोग करते हैं, हम एआई को उपयोगकर्ता की विशिष्ट आवश्यकताओं के अनुरूप ढलने देते हैं। जेनयूआई एक कदम आगे बढ़कर सॉफ्टवेयर के दृश्य इंटरफ़ेस में उस स्तर की अनुकूलनशीलता को लागू करता है।

आज जेनयूआई विचारों के साथ प्रयोग करना संभव है क्योंकि बृहत भाषा मॉडल पहले से ही प्रोग्रामिंग को समझते हैं और उनके आधार ज्ञान में यूआई प्रौद्योगिकियां और फ्रेमवर्क शामिल हैं। सवाल यह है कि क्या बृहत भाषा मॉडल का उपयोग यूआई तत्वों को जनरेट करने के लिए किया जा सकता है, जैसे टेक्स्ट, छवियां, लेआउट, और यहां तक कि पूरे इंटरफ़ेस, जो प्रत्येक व्यक्तिगत उपयोगकर्ता के लिए अनुकूलित

हों। मॉडल को विभिन्न कारकों को ध्यान में रखने के लिए निर्देशित किया जा सकता है, जैसे उपयोगकर्ता की पिछली इंटरैक्शन, व्यक्त की गई प्राथमिकताएं, जनसांख्यिकीय जानकारी, और उपयोग का वर्तमान संदर्भ, ताकि अत्यधिक वैयक्तिकृत और प्रासंगिक इंटरफ़ेस बनाए जा सकें।

जेनयूआई कई प्रमुख तरीकों से पारंपरिक उपयोगकर्ता इंटरफ़ेस डिज़ाइन से भिन्न है:

1. **गतिशील और अनुकूलनीय:** पारंपरिक यूआई डिज़ाइन में निश्चित, स्थिर इंटरफ़ेस बनाना शामिल है जो सभी उपयोगकर्ताओं के लिए समान रहता है। इसके विपरीत, जेनयूआई ऐसे इंटरफ़ेस को सक्षम बनाता है जो उपयोगकर्ता की आवश्यकताओं और संदर्भ के आधार पर गतिशील रूप से अनुकूलित और परिवर्तित हो सकते हैं। इसका मतलब है कि एक ही एप्लिकेशन विभिन्न उपयोगकर्ताओं को या यहां तक कि एक ही उपयोगकर्ता को अलग-अलग परिस्थितियों में अलग-अलग इंटरफ़ेस प्रस्तुत कर सकता है।
2. **बड़े पैमाने पर वैयक्तिकरण:** पारंपरिक डिज़ाइन के साथ, प्रत्येक उपयोगकर्ता के लिए वैयक्तिकृत अनुभव बनाना अक्सर आवश्यक समय और संसाधनों के कारण अव्यावहारिक होता है। दूसरी ओर, जेनयूआई बड़े पैमाने पर वैयक्तिकरण की अनुमति देता है। एआई का लाभ उठाकर, डिज़ाइनर ऐसे इंटरफ़ेस बना सकते हैं जो प्रत्येक उपयोगकर्ता की विशिष्ट आवश्यकताओं और प्राथमिकताओं के अनुरूप स्वचालित रूप से ढल जाते हैं, बिना प्रत्येक उपयोगकर्ता खंड के लिए अलग से इंटरफ़ेस डिज़ाइन और विकसित करने की आवश्यकता के।
3. **परिणामों पर ध्यान:** पारंपरिक यूआई डिज़ाइन अक्सर दृश्य रूप से आकर्षक और कार्यात्मक इंटरफ़ेस बनाने पर केंद्रित होता है। हालांकि ये पहलू जेनयूआई में भी महत्वपूर्ण हैं, प्राथमिक ध्यान वांछित उपयोगकर्ता परिणामों को प्राप्त करने की ओर स्थानांतरित हो जाता है। जेनयूआई प्रत्येक उपयोगकर्ता के विशिष्ट लक्ष्यों और कार्यों के लिए अनुकूलित इंटरफ़ेस बनाने का लक्ष्य रखता है, केवल सौंदर्य संबंधी विचारों की तुलना में उपयोगिता और प्रभावशीलता को प्राथमिकता देता है।
4. **निरंतर सीखना और सुधार:** जेनयूआई सिस्टम उपयोगकर्ता इंटरैक्शन और प्रतिक्रिया के आधार पर निरंतर सीख और सुधर सकते हैं। जैसे-जैसे उपयोगकर्ता

> जनरेट किए गए इंटरफ़ेस के साथ जुड़ते हैं, एआई मॉडल उपयोगकर्ता व्यवहार, प्राथमिकताओं और परिणामों पर डेटा एकत्र कर सकते हैं, और इस जानकारी का उपयोग भविष्य के इंटरफ़ेस जनरेशन को परिष्कृत और अनुकूलित करने के लिए कर सकते हैं। यह पुनरावर्ती सीखने की प्रक्रिया जेनयूआई सिस्टम को समय के साथ उपयोगकर्ता की आवश्यकताओं को पूरा करने में अधिक प्रभावी बनने की अनुमति देती है।

यह ध्यान रखना महत्वपूर्ण है कि GenUI, एआई-सहायक डिज़ाइन टूल्स से अलग है, जैसे वे जो सुझाव प्रदान करते हैं या कुछ डिज़ाइन कार्यों को स्वचालित करते हैं। हालांकि ये टूल्स डिज़ाइन प्रक्रिया को सुव्यवस्थित करने में मददगार हो सकते हैं, वे फिर भी अंतिम निर्णय लेने और स्थिर इंटरफ़ेस बनाने के लिए डिज़ाइनरों पर निर्भर करते हैं। दूसरी ओर, GenUI में एआई सिस्टम उपयोगकर्ता डेटा और संदर्भ के आधार पर इंटरफ़ेस के वास्तविक निर्माण और अनुकूलन में एक अधिक सक्रिय भूमिका निभाता है।

GenUI यूज़र इंटरफ़ेस डिज़ाइन के हमारे दृष्टिकोण में एक महत्वपूर्ण बदलाव का प्रतिनिधित्व करता है, जो एक-आकार-सभी-के-लिए समाधानों से हटकर अत्यधिक व्यक्तिगत, अनुकूली अनुभवों की ओर बढ़ रहा है। एआई की शक्ति का लाभ उठाते हुए, GenUI में डिजिटल उत्पादों और सेवाओं के साथ हमारी अंतःक्रिया के तरीके को क्रांतिकारी रूप से बदलने की क्षमता है, जो प्रत्येक व्यक्तिगत उपयोगकर्ता के लिए अधिक सहज, आकर्षक और प्रभावी इंटरफ़ेस बनाता है।

उदाहरण

GenUI की अवधारणा को समझाने के लिए, आइए “FitAI” नामक एक काल्पनिक फिटनेस एप्लिकेशन पर विचार करें। यह ऐप उपयोगकर्ताओं के व्यक्तिगत लक्ष्यों, फिटनेस स्तर और पसंद के आधार पर व्यक्तिगत व्यायाम योजना और पोषण सलाह प्रदान करने का लक्ष्य रखता है।

पारंपरिक यूआई डिज़ाइन दृष्टिकोण में, FitAI में स्क्रीन और एलिमेंट्स का एक निश्चित सेट हो सकता है जो सभी उपयोगकर्ताओं के लिए समान होता है। हालांकि, GenUI

के साथ, ऐप का इंटरफ़ेस प्रत्येक उपयोगकर्ता की विशिष्ट आवश्यकताओं और संदर्भ के अनुसार गतिशील रूप से अनुकूलित हो सकता है।

2024 में इस दृष्टिकोण को लागू करने की कल्पना करना थोड़ा कठिन है और हो सकता है कि इसका पर्याप्त ROI भी न हो, लेकिन यह संभव है।

यह कैसे काम कर सकता है:

1. **ऑनबोर्डिंग:**

    - मानक प्रश्नावली के बजाय, FitAI उपयोगकर्ता के लक्ष्यों, वर्तमान फिटनेस स्तर और पसंद के बारे में जानकारी एकत्र करने के लिए संवादात्मक एआई का उपयोग करता है।
    - इस प्रारंभिक बातचीत के आधार पर, एआई एक व्यक्तिगत डैशबोर्ड लेआउट तैयार करता है, जो उपयोगकर्ता के लक्ष्यों के लिए सबसे प्रासंगिक सुविधाओं और जानकारी को हाइलाइट करता है।
    - वर्तमान एआई तकनीक के पास व्यक्तिगत डैशबोर्ड बनाने के लिए स्क्रीन कंपोनेंट्स का एक चयन हो सकता है।
    - भविष्य की एआई तकनीक एक अनुभवी यूआई डिज़ाइनर की भूमिका निभा सकती है और वास्तव में डैशबोर्ड को शून्य से बना सकती है।

2. **व्यायाम योजनाकार:**

    - व्यायाम योजनाकार इंटरफ़ेस को एआई द्वारा उपयोगकर्ता के अनुभव स्तर और उपलब्ध उपकरणों के अनुरूप अनुकूलित किया जाता है।
    - बिना किसी उपकरण के एक शुरुआती उपयोगकर्ता के लिए, यह विस्तृत निर्देशों और वीडियो के साथ सरल शारीरिक व्यायाम दिखा सकता है।
    - जिम तक पहुंच वाले एक उन्नत उपयोगकर्ता के लिए, यह कम व्याख्यात्मक सामग्री के साथ अधिक जटिल दिनचर्या प्रदर्शित कर सकता है।
    - व्यायाम योजनाकार की सामग्री केवल एक बड़े सुपरसेट से फ़िल्टर नहीं की जाती है। यह एक ज्ञान-आधार से तत्काल उत्पन्न की जा सकती है जिसे उपयोगकर्ता के बारे में ज्ञात सभी जानकारी सहित संदर्भ के साथ क्वेरी किया जाता है।

3. **प्रगति की निगरानी:**

- प्रगति की निगरानी का इंटरफ़ेस उपयोगकर्ता के लक्ष्यों और संलग्नता पैटर्न के आधार पर विकसित होता है।
- यदि कोई उपयोगकर्ता मुख्य रूप से वजन घटाने पर ध्यान केंद्रित कर रहा है, तो इंटरफ़ेस प्रमुखता से वजन के रुझान का ग्राफ और कैलोरी बर्न के आंकड़े प्रदर्शित कर सकता है।
- मांसपेशियां बनाने वाले उपयोगकर्ता के लिए, यह ताकत में वृद्धि और शरीर की संरचना में परिवर्तनों को प्रमुखता से दिखा सकता है।
- एआई एप्लिकेशन के इस हिस्से को उपयोगकर्ता की वास्तविक प्रगति के अनुसार अनुकूलित कर सकता है। यदि प्रगति कुछ समय के लिए रुक जाती है, तो ऐप एक ऐसी स्थिति में स्थानांतरित हो सकता है जहां यह उपयोगकर्ता को बाधाओं के कारणों का खुलासा करने के लिए प्रेरित करने का प्रयास करता है, ताकि उन्हें कम किया जा सके।

4. **पोषण सलाह:**

- पोषण खंड उपयोगकर्ता की आहार संबंधी पसंद और प्रतिबंधों के अनुसार अनुकूलित होता है।
- एक शाकाहारी उपयोगकर्ता के लिए, यह पौधे-आधारित भोजन सुझाव और प्रोटीन स्रोत दिखा सकता है।
- ग्लूटेन असहिष्णुता वाले उपयोगकर्ता के लिए, यह सिफारिशों से ग्लूटेन युक्त खाद्य पदार्थों को स्वचालित रूप से फ़िल्टर कर देगा।
- पुनः, सामग्री सभी उपयोगकर्ताओं पर लागू होने वाले भोजन डेटा के विशाल सुपरसेट से नहीं ली जाती है, बल्कि एक ज्ञान आधार से संश्लेषित की जाती है जो उपयोगकर्ता की विशिष्ट स्थिति और बाधाओं के आधार पर अनुकूलन योग्य जानकारी रखता है।
- उदाहरण के लिए, व्यंजनों को सामग्री विनिर्देशों के साथ तैयार किया जाता है जो उपयोगकर्ता की लगातार बदलती कैलोरी आवश्यकताओं से मेल खाते हैं क्योंकि उनका फिटनेस स्तर और शारीरिक आंकड़े विकसित होते हैं।

5. **प्रेरक तत्व:**

    - ऐप की प्रेरक सामग्री और सूचनाएं उपयोगकर्ता के व्यक्तित्व प्रकार और विभिन्न प्रेरक रणनीतियों पर प्रतिक्रिया के आधार पर व्यक्तिगत बनाई जाती हैं।
    - कुछ उपयोगकर्ताओं को प्रोत्साहक संदेश मिल सकते हैं, जबकि अन्य को अधिक डेटा-संचालित प्रतिक्रिया मिलती है।

इस उदाहरण में, GenUI FitAI को प्रत्येक उपयोगकर्ता के लिए एक अत्यधिक अनुकूलित अनुभव बनाने में सक्षम बनाता है, जो संभवतः संलग्नता, संतुष्टि और फिटनेस लक्ष्यों को प्राप्त करने की संभावना को बढ़ाता है। इंटरफ़ेस तत्व, सामग्री, और यहां तक कि ऐप का "व्यक्तित्व" प्रत्येक व्यक्तिगत उपयोगकर्ता की जरूरतों और पसंद को सर्वोत्तम रूप से पूरा करने के लिए अनुकूलित होता है।

परिणाम-उन्मुख डिज़ाइन की ओर बदलाव

GenUI उपयोगकर्ता इंटरफ़ेस डिज़ाइन! के दृष्टिकोण में एक मौलिक बदलाव का प्रतिनिधित्व करता है, जो विशिष्ट इंटरफ़ेस तत्वों के निर्माण से एक अधिक समग्र, परिणाम-उन्मुख दृष्टिकोण की ओर बढ़ रहा है। इस बदलाव के कई महत्वपूर्ण निहितार्थ हैं:

1. **उपयोगकर्ता लक्ष्यों पर ध्यान:**

    - डिज़ाइनरों को विशिष्ट इंटरफ़ेस घटकों के बजाय उपयोगकर्ता लक्ष्यों और वांछित परिणामों के बारे में अधिक गहराई से सोचने की आवश्यकता होगी।
    - जोर ऐसी प्रणालियों के निर्माण पर होगा जो उपयोगकर्ताओं को उनके उद्देश्यों को कुशलतापूर्वक और प्रभावी ढंग से प्राप्त करने में मदद करने वाले इंटरफ़ेस उत्पन्न कर सकें।
    - नए यूआई फ्रेमवर्क सामने आएंगे जो एआई-आधारित डिज़ाइनरों को पूर्व-निर्धारित स्क्रीन विनिर्देशों के बजाय तत्काल और शुरू से उपयोगकर्ता अनुभवों को उत्पन्न करने के लिए आवश्यक उपकरण प्रदान करेंगे।

2. **डिज़ाइनर्स की बदलती भूमिका:**

- डिज़ाइनर्स निश्चित लेआउट बनाने से बदलकर नियम, सीमाएं और दिशानिर्देश परिभाषित करने की ओर बढ़ेंगे, जिनका पालन एआई सिस्टम इंटरफ़ेस जनरेट करते समय करेंगे।
- उन्हें जेनयूआई सिस्टम को प्रभावी ढंग से मार्गदर्शित करने के लिए डेटा विश्लेषण, एआई प्रॉम्प्ट इंजीनियरिंग, और सिस्टम थिंकिंग जैसे क्षेत्रों में कौशल विकसित करने की आवश्यकता होगी।

3. **उपयोगकर्ता शोध का महत्व:**

- जेनयूआई के संदर्भ में उपयोगकर्ता शोध और भी महत्वपूर्ण हो जाता है, क्योंकि डिज़ाइनर्स को न केवल उपयोगकर्ता प्राथमिकताओं को समझने की आवश्यकता होती है, बल्कि यह भी समझना होता है कि विभिन्न संदर्भों में ये प्राथमिकताएं और आवश्यकताएं कैसे बदलती हैं।
- एआई की प्रभावी इंटरफ़ेस जनरेट करने की क्षमता को परिष्कृत और सुधारने के लिए निरंतर उपयोगकर्ता परीक्षण और फीडबैक लूप आवश्यक होंगे।

4. **विविधता के लिए डिज़ाइन:**

- एक "परफेक्ट" इंटरफ़ेस बनाने के बजाय, डिज़ाइनर्स को कई संभावित विविधताओं पर विचार करना होगा और यह सुनिश्चित करना होगा कि सिस्टम विभिन्न उपयोगकर्ता आवश्यकताओं के लिए उपयुक्त इंटरफ़ेस जनरेट कर सके।
- इसमें सीमांत मामलों के लिए डिज़ाइन करना और यह सुनिश्चित करना शामिल है कि जनरेट किए गए इंटरफ़ेस विभिन्न कॉन्फ़िगरेशन में प्रयोज्यता और पहुंच को बनाए रखें।
- उत्पाद विभेदीकरण उपयोगकर्ता मनोविज्ञान पर विभिन्न दृष्टिकोणों और प्रतिस्पर्धियों को अनुपलब्ध अनूठे डेटा सेट और ज्ञान आधार के उपयोग के नए आयाम लेता है।

चुनौतियां और विचारणीय बिंदु

जहां जेनयूआई रोमांचक संभावनाएं प्रस्तुत करता है, वहीं यह कई चुनौतियां और विचारणीय बिंदु भी प्रस्तुत करता है:

1. **तकनीकी सीमाएं:**

    - वर्तमान एआई तकनीक, हालांकि उन्नत है, फिर भी जटिल उपयोगकर्ता इरादों को समझने और वास्तव में संदर्भ-जागरूक इंटरफ़ेस जनरेट करने में सीमाएं हैं।
    - विशेष रूप से कम शक्तिशाली उपकरणों पर इंटरफ़ेस तत्वों की रीयल-टाइम जनरेशन से संबंधित प्रदर्शन मुद्दे।

2. **डेटा आवश्यकताएं:**

    - उपयोग के मामले के आधार पर, प्रभावी जेनयूआई सिस्टम को वैयक्तिकृत इंटरफ़ेस जनरेट करने के लिए महत्वपूर्ण मात्रा में उपयोगकर्ता डेटा की आवश्यकता हो सकती है।
    - नैतिक रूप से प्रामाणिक उपयोगकर्ता डेटा प्राप्त करने की चुनौतियां डेटा गोपनीयता और सुरक्षा के बारे में चिंताएं उठाती हैं, साथ ही जेनयूआई मॉडल को प्रशिक्षित करने के लिए उपयोग किए जाने वाले डेटा में संभावित पूर्वाग्रह भी।

3. **प्रयोज्यता और स्थिरता:**

    - कम से कम जब तक यह प्रथा व्यापक नहीं हो जाती, लगातार बदलते इंटरफ़ेस वाला एप्लिकेशन प्रयोज्यता समस्याएं पैदा कर सकता है, क्योंकि उपयोगकर्ताओं को परिचित तत्वों को खोजने या कुशलतापूर्वक नेविगेट करने में कठिनाई हो सकती है।
    - वैयक्तिकरण और एक स्थिर, सीखने योग्य इंटरफ़ेस बनाए रखने के बीच संतुलन बनाना महत्वपूर्ण होगा।

4. **एआई पर अत्यधिक निर्भरता:**

- एआई सिस्टम को डिज़ाइन निर्णय अत्यधिक सौंपने का जोखिम है, जो संभवतः अप्रेरित, समस्याग्रस्त, या केवल खराब इंटरफ़ेस विकल्पों की ओर ले जा सकता है।
- निकट भविष्य में मानवीय निरीक्षण और एआई-जनित डिज़ाइन को ओवरराइड करने की क्षमता महत्वपूर्ण बनी रहेगी।

5. **पहुंच-योग्यता संबंधी चिंताएं:**

- गतिशील रूप से उत्पन्न इंटरफ़ेस को विकलांग उपयोगकर्ताओं के लिए सुलभ बनाए रखना पूरी तरह से नई चुनौतियां प्रस्तुत करता है, जो चिंताजनक है क्योंकि सामान्य सिस्टम द्वारा प्रदर्शित पहुंच-योग्यता अनुपालन का स्तर कमजोर है।
- दूसरी ओर, एआई डिज़ाइनर को पहुंच-योग्यता के लिए अंतर्निहित चिंता के साथ कार्यान्वित किया जा सकता है, और सामान्य उपयोगकर्ताओं के लिए यूआई बनाने की तरह ही सुलभ इंटरफ़ेस बनाने की क्षमताएं होंगी।
- किसी भी स्थिति में, जेनयूआई सिस्टम को मजबूत पहुंच-योग्यता दिशानिर्देशों और परीक्षण प्रक्रियाओं के साथ डिज़ाइन किया जाना चाहिए।

6. **उपयोगकर्ता विश्वास और पारदर्शिता:**

- उपयोगकर्ता ऐसे इंटरफ़ेस से असहज महसूस कर सकते हैं जो उनके बारे में “बहुत अधिक जानते” हैं या ऐसे तरीकों से बदलते हैं जिन्हें वे नहीं समझते।
- इंटरफ़ेस को कैसे और क्यों व्यक्तिगत बनाया जाता है, इसके बारे में पारदर्शिता प्रदान करना उपयोगकर्ता विश्वास बनाने के लिए महत्वपूर्ण होगा।

भविष्य का दृष्टिकोण और अवसर

जनरेटिव यूआई (जेनयूआई) का भविष्य डिजिटल उत्पादों और सेवाओं के साथ हमारी अंतःक्रिया के तरीके में क्रांति लाने का अपार वादा रखता है। जैसे-जैसे यह तकनीक विकसित होती जाएगी, हम उपयोगकर्ता इंटरफ़ेस के डिज़ाइन, कार्यान्वयन और अनुभव में एक भूकंपीय बदलाव की उम्मीद कर सकते हैं। मेरा मानना है कि जेनयूआई वह घटना है जो अंततः हमारे सॉफ्टवेयर को विज्ञान कथा माने जाने वाले क्षेत्र में धकेल देगी।

जेनयूआई की सबसे रोमांचक संभावनाओं में से एक इसकी पहुंच-योग्यता को बड़े पैमाने पर बढ़ाने की क्षमता है, जो केवल गंभीर विकलांगता वाले लोगों को आपके सॉफ्टवेयर के उपयोग से पूरी तरह से बाहर न रखने से कहीं आगे जाती है। व्यक्तिगत उपयोगकर्ता की जरूरतों के अनुसार स्वचालित रूप से इंटरफ़ेस को अनुकूलित करके, जेनयूआई डिजिटल अनुभवों को पहले से कहीं अधिक समावेशी बना सकता है। ऐसे इंटरफ़ेस की कल्पना करें जो युवा या दृष्टि बाधित उपयोगकर्ताओं के लिए बड़े टेक्स्ट प्रदान करने या संज्ञानात्मक विकलांगता वाले लोगों के लिए सरलीकृत लेआउट प्रदान करने के लिए निर्बाध रूप से समायोजित हो जाते हैं, वह भी मैनुअल कॉन्फ़िगरेशन या एप्लिकेशन के अलग "सुलभ" संस्करणों की आवश्यकता के बिना।

जेनयूआई की व्यक्तिगतकरण क्षमताएं विभिन्न डिजिटल उत्पादों में उपयोगकर्ता जुड़ाव, संतुष्टि और निष्ठा को बढ़ाने की संभावना रखती हैं। जैसे-जैसे इंटरफ़ेस व्यक्तिगत पसंद और व्यवहार के प्रति अधिक सामंजस्यपूर्ण होते जाएंगे, उपयोगकर्ताओं को डिजिटल अनुभव अधिक सहज और आनंददायक लगेंगे, जिससे संभवतः प्रौद्योगिकी के साथ गहरी और अधिक सार्थक अंतःक्रिया हो सकेगी।

जेनयूआई में नए उपयोगकर्ताओं के लिए ऑनबोर्डिंग प्रक्रिया को बदलने की क्षमता भी है। प्रत्येक उपयोगकर्ता की विशेषज्ञता के स्तर के अनुसार तेजी से अनुकूल होने वाले सहज, व्यक्तिगत पहली बार के उपयोगकर्ता अनुभव बनाकर, जेनयूआई नए एप्लिकेशन से जुड़ी सीखने की प्रक्रिया को काफी कम कर सकता है। इससे तेज अपनाने की दर और नई सुविधाओं और कार्यक्षमताओं की खोज में उपयोगकर्ता का आत्मविश्वास बढ़ सकता है।

एक और रोमांचक संभावना है रचनात्मक यूआई (GenUI) की क्षमता विभिन्न उपकरणों और प्लेटफ़ॉर्म पर एक सुसंगत प्रयोक्ता अनुभव को बनाए रखने की, जबकि प्रत्येक विशिष्ट उपयोग के संदर्भ के लिए अनुकूलन करते हुए। यह स्मार्टफोन और टैबलेट से लेकर डेस्कटॉप कंप्यूटर और संवर्धित वास्तविकता चश्मे जैसी उभरती तकनीकों तक, एक बढ़ते हुए खंडित उपकरण परिदृश्य में सुसंगत अनुभव प्रदान करने की लंबे समय से चली आ रही चुनौती को हल कर सकता है।

डेटा-संचालित प्रकृति के कारण रचनात्मक यूआई यूआई डिज़ाइन में तेज़ी से सुधार और परिवर्तन के अवसर खोलता है। उत्पन्न इंटरफ़ेस के साथ प्रयोक्ताओं की अंतःक्रिया पर वास्तविक समय का डेटा एकत्र करके, डिज़ाइनर और डेवलपर प्रयोक्ता व्यवहार और प्राथमिकताओं में अभूतपूर्व अंतर्दृष्टि प्राप्त कर सकते हैं। यह प्रतिक्रिया चक्र यूआई डिज़ाइन में निरंतर सुधार की ओर ले जा सकता है, जो मान्यताओं या सीमित प्रयोक्ता परीक्षण के बजाय वास्तविक उपयोग पैटर्न से संचालित होगा।

इस बदलाव के लिए तैयार होने के लिए, डिज़ाइनरों को अपने कौशल समूह और मानसिकता को विकसित करने की आवश्यकता होगी। ध्यान निश्चित लेआउट बनाने से हटकर व्यापक डिज़ाइन प्रणालियों और दिशानिर्देशों को विकसित करने पर केंद्रित होगा जो एआई-संचालित इंटरफ़ेस निर्माण को सूचित कर सकते हैं। डिज़ाइनरों को रचनात्मक यूआई प्रणालियों को प्रभावी ढंग से मार्गदर्शित करने के लिए डेटा विश्लेषण, एआई प्रौद्योगिकियों और प्रणाली सोच की गहरी समझ विकसित करने की आवश्यकता होगी।

इसके अलावा, जैसे-जैसे रचनात्मक यूआई डिज़ाइन और प्रौद्योगिकी के बीच की रेखाएं धुंधली होती जाएंगी, डिज़ाइनरों को डेवलपर्स और डेटा वैज्ञानिकों के साथ और अधिक निकटता से सहयोग करने की आवश्यकता होगी। यह अंतर्विषयक दृष्टिकोण ऐसी रचनात्मक यूआई प्रणालियाँ बनाने में महत्वपूर्ण होगा जो न केवल दृश्य रूप से आकर्षक और प्रयोक्ता-अनुकूल हों, बल्कि तकनीकी रूप से मजबूत और नैतिक रूप से उचित भी हों।

जैसे-जैसे प्रौद्योगिकी परिपक्व होगी, रचनात्मक यूआई के नैतिक निहितार्थ भी सामने आएंगे। डिज़ाइनर इंटरफ़ेस डिज़ाइन में जिम्मेदार एआई उपयोग के लिए ढांचे विकसित करने में महत्वपूर्ण भूमिका निभाएंगे, यह सुनिश्चित करते हुए कि व्यक्तिगतकरण प्रयोक्ता

अनुभवों को बढ़ाता है, गोपनीयता से समझौता किए बिना या प्रयोक्ता व्यवहार को अनैतिक तरीकों से प्रभावित किए बिना।

जैसे-जैसे हम भविष्य की ओर देखते हैं, रचनात्मक यूआई रोमांचक अवसर और महत्वपूर्ण चुनौतियां दोनों प्रस्तुत करता है। इसमें दुनिया भर के प्रयोक्ताओं के लिए अधिक सहज, कुशल और संतोषजनक डिजिटल अनुभव बनाने की क्षमता है। हालांकि इसके लिए डिज़ाइनरों को अनुकूलित होने और नए कौशल प्राप्त करने की आवश्यकता होगी, यह मानव-कंप्यूटर अंतःक्रिया के भविष्य को गहन और सार्थक तरीकों से आकार देने का एक अभूतपूर्व अवसर भी प्रदान करता है। पूर्ण रूप से विकसित रचनात्मक यूआई प्रणालियों की ओर यात्रा निश्चित रूप से जटिल होगी, लेकिन बेहतर प्रयोक्ता अनुभवों और डिजिटल पहुंच के संदर्भ में संभावित पुरस्कार इसे एक ऐसा भविष्य बनाते हैं जिसके लिए प्रयास करने योग्य है।

बुद्धिमान कार्यप्रवाह समन्वय

“बुद्धिमान कार्यप्रवाह समन्वय” दृष्टिकोण एप्लिकेशन के भीतर जटिल कार्यप्रवाह को गतिशील रूप से समन्वयित और अनुकूलित करने के लिए AI घटकों का लाभ उठाने पर केंद्रित है। लक्ष्य ऐसे एप्लिकेशन बनाना है जो अधिक कुशल, प्रतिक्रियाशील और वास्तविक समय के डेटा और संदर्भ के प्रति अनुकूलनीय हों।

इस अध्याय में, हम बुद्धिमान कार्यप्रवाह समन्वय दृष्टिकोण के मूल सिद्धांतों और पैटर्न की खोज करेंगे। हम विचार करेंगे कि कैसे AI का उपयोग कार्यों को बुद्धिमानी से रूट करने, निर्णय लेने को स्वचालित करने और उपयोगकर्ता व्यवहार, सिस्टम प्रदर्शन और व्यावसायिक नियमों जैसे विभिन्न कारकों के आधार पर कार्यप्रवाह को गतिशील रूप से अनुकूलित करने के लिए किया जा सकता है। व्यावहारिक उदाहरणों और वास्तविक परिदृश्यों के माध्यम से, हम एप्लिकेशन कार्यप्रवाह को सुव्यवस्थित और अनुकूलित करने में AI की परिवर्तनकारी क्षमता का प्रदर्शन करेंगे।

चाहे आप जटिल व्यावसायिक प्रक्रियाओं वाले एंटरप्राइज एप्लिकेशन बना रहे हों या

गतिशील उपयोगकर्ता यात्राओं वाले उपभोक्ता-केंद्रित एप्लिकेशन, इस अध्याय में चर्चा किए गए पैटर्न और तकनीकें आपको बुद्धिमान और कुशल कार्यप्रवाह बनाने के लिए ज्ञान और उपकरणों से लैस करेंगी जो समग्र उपयोगकर्ता अनुभव को बढ़ाते हैं और व्यावसायिक मूल्य को बढ़ावा देते हैं।

व्यावसायिक आवश्यकता

कार्यप्रवाह प्रबंधन के पारंपरिक दृष्टिकोण अक्सर पूर्व-निर्धारित नियमों और स्थिर निर्णय वृक्षों पर निर्भर करते हैं, जो कठोर, अनम्य हो सकते हैं और आधुनिक एप्लिकेशन की गतिशील प्रकृति से निपटने में असमर्थ हो सकते हैं।

एक ऐसी स्थिति पर विचार करें जहां एक ई-कॉमर्स एप्लिकेशन को एक जटिल आदेश पूर्ति प्रक्रिया को संभालने की आवश्यकता है। कार्यप्रवाह में कई चरण शामिल हो सकते हैं जैसे आदेश सत्यापन, इन्वेंट्री जांच, भुगतान प्रसंस्करण, शिपिंग और ग्राहक सूचनाएं। प्रत्येक चरण के अपने नियम, निर्भरताएं, बाहरी एकीकरण और अपवाद प्रबंधन तंत्र हो सकते हैं। ऐसे कार्यप्रवाह को मैन्युअल रूप से या हार्डकोडेड लॉजिक के माध्यम से प्रबंधित करना जल्दी ही बोझिल, त्रुटि-प्रवण और बनाए रखने में कठिन हो सकता है।

इसके अलावा, जैसे-जैसे एप्लिकेशन का विस्तार होता है और एक साथ उपयोगकर्ताओं की संख्या बढ़ती है, कार्यप्रवाह को वास्तविक समय के डेटा और सिस्टम प्रदर्शन के आधार पर खुद को अनुकूलित और अनुकूलन करने की आवश्यकता हो सकती है। उदाहरण के लिए, अधिक ट्रैफिक की अवधि के दौरान, एप्लिकेशन को कुछ कार्यों को प्राथमिकता देने, संसाधनों का कुशलतापूर्वक आवंटन करने और सुचारू उपयोगकर्ता अनुभव सुनिश्चित करने के लिए कार्यप्रवाह को गतिशील रूप से समायोजित करने की आवश्यकता हो सकती है।

यहीं पर "बुद्धिमान कार्यप्रवाह समन्वय" दृष्टिकोण महत्वपूर्ण हो जाता है। AI घटकों का लाभ उठाकर, डेवलपर्स ऐसे कार्यप्रवाह बना सकते हैं जो बुद्धिमान, अनुकूलनशील और स्व-अनुकूलन करने वाले हों। AI बड़ी मात्रा में डेटा का विश्लेषण कर सकता है,

पिछले अनुभवों से सीख सकता है, और कार्यप्रवाह को प्रभावी ढंग से संचालित करने के लिए वास्तविक समय में सूचित निर्णय ले सकता है।

प्रमुख लाभ

1. **बढ़ी हुई दक्षता:** AI कार्य आवंटन, संसाधन उपयोग और कार्यप्रवाह निष्पादन को अनुकूलित कर सकता है, जिससे तेज प्रसंस्करण समय और समग्र दक्षता में सुधार होता है।
2. **अनुकूलन क्षमता:** AI-संचालित कार्यप्रवाह बदलती परिस्थितियों के अनुसार गतिशील रूप से अनुकूलन कर सकते हैं, जैसे उपयोगकर्ता मांग में उतार-चढ़ाव, सिस्टम प्रदर्शन, या व्यावसायिक आवश्यकताएं, जो सुनिश्चित करता है कि एप्लिकेशन प्रतिक्रियाशील और लचीला बना रहे।
3. **स्वचालित निर्णय-निर्माण:** AI कार्यप्रवाह के भीतर जटिल निर्णय-निर्माण प्रक्रियाओं को स्वचालित कर सकता है, जिससे मैनुअल हस्तक्षेप कम होता है और मानवीय त्रुटियों का जोखिम कम होता है।
4. **वैयक्तिकरण:** AI उपयोगकर्ता व्यवहार, प्राथमिकताओं और संदर्भ का विश्लेषण करके कार्यप्रवाह को वैयक्तिकृत कर सकता है और व्यक्तिगत उपयोगकर्ताओं को अनुकूलित अनुभव प्रदान कर सकता है।
5. **मापनीयता:** AI-संचालित कार्यप्रवाह प्रदर्शन या विश्वसनीयता से समझौता किए बिना, बढ़ती मात्रा में डेटा और उपयोगकर्ता इंटरैक्शन को संभालने के लिए निर्बाध रूप से स्केल कर सकते हैं।

आगामी खंडों में, हम उन प्रमुख पैटर्न और तकनीकों की खोज करेंगे जो बुद्धिमान कार्यप्रवाह के कार्यान्वयन को सक्षम बनाते हैं और दिखाएंगे कि कैसे AI आधुनिक एप्लिकेशन में कार्यप्रवाह प्रबंधन को बदल रहा है।

प्रमुख पैटर्न

एप्लिकेशन में बुद्धिमान कार्यप्रवाह समन्वय को लागू करने के लिए, डेवलपर्स कई प्रमुख पैटर्न का लाभ उठा सकते हैं जो AI की शक्ति का उपयोग करते हैं। ये पैटर्न कार्यप्रवाह को डिज़ाइन और प्रबंधित करने के लिए एक संरचित दृष्टिकोण प्रदान करते हैं, जो एप्लिकेशन को वास्तविक समय के डेटा और संदर्भ के आधार पर प्रक्रियाओं को अनुकूलित, अनुकूलन और स्वचालित करने में सक्षम बनाते हैं। आइए बुद्धिमान कार्यप्रवाह समन्वय में कुछ मौलिक पैटर्न की खोज करें।

गतिशील कार्य मार्गण

इस पैटर्न में कार्य प्राथमिकता, संसाधन उपलब्धता और सिस्टम प्रदर्शन जैसे विभिन्न कारकों के आधार पर कार्यप्रवाह के भीतर कार्यों को बुद्धिमानी से रूट करने के लिए AI का उपयोग शामिल है। AI एल्गोरिदम प्रत्येक कार्य की विशेषताओं का विश्लेषण कर सकते हैं, सिस्टम की वर्तमान स्थिति पर विचार कर सकते हैं, और कार्यों को सबसे उपयुक्त संसाधनों या प्रसंस्करण पथों को असाइन करने के लिए सूचित निर्णय ले सकते हैं। गतिशील कार्य मार्गण सुनिश्चित करता है कि कार्य कुशलतापूर्वक वितरित और निष्पादित किए जाते हैं, जो समग्र कार्यप्रवाह प्रदर्शन को अनुकूलित करता है।

```
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, # ...
```

```
15    function :assess_system_performance, # ...
16    function :assign_task_to_resource, # ...
17
18    DIRECTIVE = "You are a task router, responsible for intelligently
19     assigning tasks to available resources based on priority, resource
20     availability, and system performance..."
21
22    def initialize(task)
23      self.task = task
24      transcript << { system: DIRECTIVE }
25      transcript << { user: task.to_json }
26    end
27
28    def perform
29      while task.unassigned?
30        chat_completion
31
32        # todo: add max loop counter and break
33      end
34
35      # capture the transcript for later analysis
36      task.update(routing_transcript: transcript)
37    end
38  end
```

ध्यान दें कि लाइन 29 पर while एक्सप्रेशन द्वारा बनाया गया लूप, जो AI को तब तक प्रॉम्प्ट करता रहता है जब तक कि कार्य असाइन नहीं हो जाता। लाइन 35 पर, हम बाद में विश्लेषण और डीबगिंग के लिए कार्य का ट्रांसक्रिप्ट सहेज लेते हैं, यदि यह आवश्यक हो।

संदर्भपरक निर्णय लेना

आप वर्कफ्लो के भीतर संदर्भ-जागरूक निर्णय लेने के लिए बहुत समान कोड का उपयोग कर सकते हैं। उपयोगकर्ता प्राथमिकताओं, ऐतिहासिक पैटर्न, और रीयल-टाइम इनपुट जैसे प्रासंगिक डेटा पॉइंट्स का विश्लेषण करके, AI कंपोनेंट्स वर्कफ्लो में प्रत्येक निर्णय बिंदु पर सबसे उपयुक्त कार्रवाई का निर्धारण कर सकते हैं। प्रत्येक उपयोगकर्ता

या परिदृश्य के विशिष्ट संदर्भ के आधार पर अपने वर्कफ्लो के व्यवहार को अनुकूलित करें, जो व्यक्तिगत और अनुकूलित अनुभव प्रदान करता है।

अनुकूली वर्कफ्लो संरचना

यह पैटर्न बदलती आवश्यकताओं या परिस्थितियों के आधार पर वर्कफ्लो को गतिशील रूप से तैयार करने और समायोजित करने पर केंद्रित है। AI वर्कफ्लो की वर्तमान स्थिति का विश्लेषण कर सकता है, बाधाओं या अक्षमताओं की पहचान कर सकता है, और प्रदर्शन को अनुकूलित करने के लिए वर्कफ्लो संरचना को स्वचालित रूप से संशोधित कर सकता है। अनुकूली वर्कफ्लो संरचना एप्लिकेशन को मैनुअल हस्तक्षेप की आवश्यकता के बिना निरंतर विकसित और सुधार करने की अनुमति देती है।

अपवाद प्रबंधन और पुनर्प्राप्ति

अपवाद प्रबंधन और पुनर्प्राप्ति बुद्धिमान वर्कफ्लो ऑर्केस्ट्रेशन के महत्वपूर्ण पहलू हैं। AI कंपोनेंट्स और जटिल वर्कफ्लो के साथ काम करते समय, सिस्टम की स्थिरता और विश्वसनीयता सुनिश्चित करने के लिए अपवादों का सुचारू रूप से पूर्वानुमान और प्रबंधन करना आवश्यक है।

बुद्धिमान वर्कफ्लो में अपवाद प्रबंधन और पुनर्प्राप्ति के लिए कुछ प्रमुख विचार और तकनीकें यहाँ दी गई हैं:

1. **अपवाद प्रसार:** वर्कफ्लो कंपोनेंट्स में अपवादों के प्रसार के लिए एक सुसंगत दृष्टिकोण लागू करें। जब किसी कंपोनेंट के भीतर कोई अपवाद होता है, तो इसे पकड़ा जाना चाहिए, लॉग किया जाना चाहिए, और ऑर्केस्ट्रेटर या अपवादों को संभालने के लिए जिम्मेदार एक अलग कंपोनेंट तक पहुंचाया जाना चाहिए। विचार यह है कि अपवाद प्रबंधन को केंद्रीकृत किया जाए और अपवादों को चुपचाप निगल जाने से रोका जाए, साथ ही बुद्धिमान त्रुटि प्रबंधन के लिए संभावनाएं खोली जाएं।

2. **पुनः प्रयास तंत्र:** पुनः प्रयास तंत्र वर्कफ्लो की लचीलापन में सुधार करने और क्षणिक विफलताओं को सुचारू रूप से संभालने में मदद करते हैं। क्षणिक या पुनर्प्राप्त करने योग्य अपवादों के लिए पुनः प्रयास तंत्र को लागू करने का निश्चित प्रयास करें, जैसे नेटवर्क कनेक्टिविटी या संसाधन की अनुपलब्धता जिसे निर्दिष्ट देरी के बाद स्वचालित रूप से पुनः प्रयास किया जा सकता है। एक AI-संचालित ऑर्केस्ट्रेटर या अपवाद हैंडलर का मतलब है कि आपकी पुनः प्रयास रणनीतियों को यांत्रिक प्रकृति की नहीं होना पड़ता, जो एक्सपोनेंशियल फॉलबैक जैसे निश्चित एल्गोरिथम पर निर्भर करती हैं। आप अपवाद को संभालने के तरीके का निर्णय करने के लिए जिम्मेदार AI कंपोनेंट के "विवेक" पर पुनः प्रयास का प्रबंधन छोड़ सकते हैं।
3. **फॉलबैक रणनीतियाँ:** यदि कोई AI घटक वैध प्रतिक्रिया प्रदान करने में विफल रहता है या किसी त्रुटि का सामना करता है—जो इसकी नवीनतम प्रकृति को देखते हुए एक सामान्य घटना है—तो कार्यप्रवाह को जारी रखने के लिए एक फॉलबैक तंत्र होना चाहिए। इसमें डिफ़ॉल्ट मान का उपयोग, वैकल्पिक एल्गोरिथ्म, या निर्णय लेने और कार्यप्रवाह को आगे बढ़ाने के लिए मानव-इन-द-लूप का उपयोग शामिल हो सकता है।
4. **प्रतिकारक कार्रवाइयाँ:** ऑर्केस्ट्रेटर के निर्देशों में उन अपवादों को संभालने के लिए प्रतिकारक कार्रवाइयों के बारे में निर्देश शामिल होने चाहिए जिन्हें स्वचालित रूप से हल नहीं किया जा सकता। प्रतिकारक कार्रवाइयाँ विफल हुए ऑपरेशन के प्रभावों को पूर्ववत करने या कम करने के लिए उठाए गए कदम हैं। उदाहरण के लिए, यदि भुगतान प्रसंस्करण चरण विफल हो जाता है, तो प्रतिकारक कार्रवाई लेनदेन को वापस रोल करना और उपयोगकर्ता को सूचित करना हो सकती है। प्रतिकारक कार्रवाइयाँ अपवादों के सामने डेटा संगति और अखंडता बनाए रखने में मदद करती हैं।
5. **अपवाद निगरानी और सूचना:** महत्वपूर्ण अपवादों का पता लगाने और संबंधित हितधारकों को सूचित करने के लिए निगरानी और सूचना तंत्र स्थापित करें। ऑर्केस्ट्रेटर को सीमाओं और नियमों से अवगत कराया जा सकता है जो तब चेतावनियाँ ट्रिगर करते हैं जब अपवाद कुछ सीमाओं से अधिक हो जाते हैं या जब विशिष्ट प्रकार के अपवाद होते हैं। यह समस्याओं की सक्रिय पहचान और

समाधान को संभव बनाता है, इससे पहले कि वे समग्र प्रणाली को प्रभावित करें।

यहाँ Ruby कार्यप्रवाह घटक में अपवाद प्रबंधन और पुनर्प्राप्ति का एक उदाहरण दिया गया है:

```
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
```

इस उदाहरण में, InventoryManager कंपोनेंट किसी दिए गए ऑर्डर के लिए प्रोडक्ट की उपलब्धता की जांच करता है। यदि उपलब्ध मात्रा अपर्याप्त है, तो यह एक InsufficientInventoryError उठाता है। इस अपवाद को कैच किया जाता है, लॉग

किया जाता है, और एक पुनः प्रयास तंत्र लागू किया जाता है। यदि पुनः प्रयास की सीमा पार हो जाती है, तो कंपोनेंट एडमिन को सूचित करके मैनुअल हस्तक्षेप पर वापस आ जाता है।

मजबूत अपवाद प्रबंधन और रिकवरी तंत्र को लागू करके, आप यह सुनिश्चित कर सकते हैं कि आपके बुद्धिमान कार्यप्रवाह लचीले, रखरखाव योग्य और अप्रत्याशित स्थितियों को सहजता से संभालने में सक्षम हैं।

ये पैटर्न बुद्धिमान कार्यप्रवाह संयोजन की नींव बनाते हैं और विभिन्न एप्लिकेशन की विशिष्ट आवश्यकताओं के अनुरूप संयोजित और अनुकूलित किए जा सकते हैं। इन पैटर्न का लाभ उठाकर, डेवलपर्स ऐसे कार्यप्रवाह बना सकते हैं जो लचीले, मजबूत और प्रदर्शन एवं उपयोगकर्ता अनुभव के लिए अनुकूलित हैं।

अगले खंड में, हम देखेंगे कि इन पैटर्न को व्यवहार में कैसे लागू किया जा सकता है, वास्तविक-दुनिया के उदाहरणों और कोड स्निपेट का उपयोग करके कार्यप्रवाह प्रबंधन में AI कंपोनेंट के एकीकरण को समझाया जाएगा।

बुद्धिमान कार्यप्रवाह संयोजन को व्यवहार में लागू करना

अब जब हमने बुद्धिमान कार्यप्रवाह संयोजन में प्रमुख पैटर्न की खोज कर ली है, आइए देखें कि इन पैटर्न को वास्तविक-दुनिया के एप्लिकेशन में कैसे लागू किया जा सकता है। हम कार्यप्रवाह प्रबंधन में AI कंपोनेंट के एकीकरण को दर्शाने के लिए व्यावहारिक उदाहरण और कोड स्निपेट प्रदान करेंगे।

बुद्धिमान ऑर्डर प्रोसेसर

आइए Ruby on Rails ई-कॉमर्स एप्लिकेशन में AI-संचालित OrderProcessor कंपोनेंट का उपयोग करके बुद्धिमान कार्यप्रवाह संयोजन को लागू करने का एक व्यावहारिक उदाहरण देखें। OrderProcessor प्रोसेस मैनेजर एंटरप्राइज एकीकरण अवधारणा को

साकार करता है जिसका हमने पहली बार अध्याय 3 में कार्यकर्ताओं की बहुलता पर चर्चा करते समय सामना किया था। यह कंपोनेंट ऑर्डर फुलफिलमेंट कार्यप्रवाह के प्रबंधन, मध्यवर्ती परिणामों के आधार पर रूटिंग निर्णय लेने और विभिन्न प्रोसेसिंग चरणों के निष्पादन के संयोजन के लिए जिम्मेदार होगा।

ऑर्डर फुलफिलमेंट प्रक्रिया में कई चरण शामिल हैं जैसे ऑर्डर वैधीकरण, इन्वेंटरी जांच, भुगतान प्रोसेसिंग और शिपिंग। प्रत्येक चरण एक अलग वर्कर प्रोसेस के रूप में लागू किया जाता है जो एक विशिष्ट कार्य करता है और परिणाम OrderProcessor को वापस करता है। ये चरण अनिवार्य नहीं हैं, और इन्हें सटीक क्रम में किया जाना भी आवश्यक नहीं है।

यहाँ OrderProcessor का एक उदाहरण कार्यान्वयन है। इसमें Raix से दो मिक्सिन हैं। पहला (ChatCompletion) इसे चैट कंप्लीशन की क्षमता प्रदान करता है, जो इसे एक AI कंपोनेंट बनाता है। दूसरा (FunctionDispatch) AI द्वारा फंक्शन कॉलिंग को सक्षम करता है, जिससे यह एक टेक्स्ट मैसेज के बजाय फंक्शन इनवोकेशन के साथ प्रॉम्प्ट का जवाब दे सकता है।

कार्यकर्ता फ़ंक्शन्स (validate_order, check_inventory, इत्यादि) अपने संबंधित कार्यकर्ता क्लासेज को कार्य सौंपते हैं, जो एआई या गैर-एआई कंपोनेंट्स हो सकते हैं, जिनकी एकमात्र आवश्यकता यह है कि वे अपने कार्य के परिणामों को एक ऐसे प्रारूप में वापस करें जिसे स्ट्रिंग के रूप में प्रस्तुत किया जा सके।

जैसा कि इस पुस्तक के इस भाग में दिए गए अन्य सभी उदाहरणों में है, यह कोड व्यावहारिक रूप से स्यूडो-कोड है और केवल पैटर्न का अर्थ समझाने और आपकी खुद की रचनाओं को प्रेरित करने के लिए है। पैटर्न का पूर्ण विवरण और पूर्ण कोड उदाहरण भाग 2 में शामिल किए गए हैं।

```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
```

इस उदाहरण में, OrderProcessor को एक ऑर्डर ऑब्जेक्ट के साथ आरंभ किया जाता है और कार्यप्रवाह निष्पादन का एक प्रतिलेख बनाए रखता है, जो बृहत भाषा मॉडल के लिए प्राकृतिक वार्तालाप प्रतिलेख प्रारूप में होता है। एआई को विभिन्न प्रसंस्करण चरणों के निष्पादन को संचालित करने के लिए पूर्ण नियंत्रण दिया जाता है, जैसे ऑर्डर सत्यापन, इन्वेंट्री जांच, भुगतान प्रसंस्करण और शिपिंग।

हर बार जब chat_completion मेथड को कॉल किया जाता है, प्रतिलेख को एआई को एक फ़ंक्शन कॉल के रूप में पूर्णता प्रदान करने के लिए भेजा जाता है। पिछले चरण के परिणाम का विश्लेषण करने और उचित कार्रवाई करने का निर्धारण पूरी तरह से एआई पर निर्भर करता है। उदाहरण के लिए, यदि इन्वेंट्री जांच में कम स्टॉक स्तर का पता चलता है, तो OrderProcessor एक पुनःपूर्ति कार्य निर्धारित कर सकता है। यदि भुगतान प्रसंस्करण विफल होता है, तो यह पुनः प्रयास कर सकता है या ग्राहक सहायता को सूचित कर सकता है।

> उपरोक्त उदाहरण में पुनःपूर्ति या ग्राहक सहायता को सूचित करने के लिए फ़ंक्शन्स परिभाषित नहीं हैं, लेकिन बिल्कुल हो सकते हैं।

प्रतिलेख हर बार एक फ़ंक्शन कॉल होने पर बढ़ता जाता है और कार्यप्रवाह निष्पादन का एक रिकॉर्ड के रूप में काम करता है, जिसमें प्रत्येक चरण के परिणाम और अगले चरणों के लिए एआई-जनित निर्देश शामिल होते हैं। इस प्रतिलेख का उपयोग डीबगिंग, ऑडिटिंग और ऑर्डर फुलफिलमेंट प्रक्रिया में दृश्यता प्रदान करने के लिए किया जा सकता है।

OrderProcessor में एआई का लाभ उठाकर, ई-कॉमर्स एप्लिकेशन वास्तविक समय के डेटा के आधार पर कार्यप्रवाह को गतिशील रूप से अनुकूलित कर सकता है और बुद्धिमानी से अपवादों को संभाल सकता है। एआई घटक सूचित निर्णय ले सकता है, कार्यप्रवाह को अनुकूलित कर सकता है, और जटिल परिदृश्यों में भी सुचारू ऑर्डर प्रसंस्करण सुनिश्चित कर सकता है।

यह तथ्य कि कार्यकर्ता प्रक्रियाओं पर एकमात्र आवश्यकता एआई के लिए अगला क्या करना है यह तय करने के लिए कुछ समझने योग्य आउटपुट लौटाना है, आपको यह एहसास होने लगेगा कि यह दृष्टिकोण एक-दूसरे के साथ विभिन्न सिस्टम्स को एकीकृत करते समय आमतौर पर शामिल इनपुट/आउटपुट मैपिंग कार्य को कैसे कम कर सकता है।

बुद्धिमान कंटेंट मॉडरेटर

सोशल मीडिया एप्लिकेशन्स को आमतौर पर एक सुरक्षित और स्वस्थ समुदाय सुनिश्चित करने के लिए कम से कम न्यूनतम कंटेंट मॉडरेशन की आवश्यकता होती है। यह उदाहरण ContentModerator घटक बुद्धिमानी से मॉडरेशन कार्यप्रवाह को संचालित करने के लिए एआई का लाभ उठाता है, जो कंटेंट की विशेषताओं और विभिन्न मॉडरेशन चरणों के परिणामों के आधार पर निर्णय लेता है।

मॉडरेशन प्रक्रिया में कई चरण शामिल हैं जैसे टेक्स्ट विश्लेषण, छवि पहचान, उपयोगकर्ता प्रतिष्ठा मूल्यांकन और मैनुअल समीक्षा। प्रत्येक चरण एक अलग कार्यकर्ता प्रक्रिया के रूप में लागू किया जाता है जो एक विशिष्ट कार्य करता है और परिणाम ContentModerator को वापस करता है।

यहाँ ContentModerator का एक उदाहरण कार्यान्वयन दिया गया है:

```
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
```

इस उदाहरण में, ContentModerator को एक कंटेंट ऑब्जेक्ट के साथ आरंभ किया जाता है और वार्तालाप प्रारूप में एक मॉडरेशन प्रतिलेख बनाए रखता है। एआई घटक को मॉडरेशन कार्यप्रवाह पर पूर्ण नियंत्रण होता है, जो कंटेंट की विशेषताओं और प्रत्येक चरण के परिणामों के आधार पर यह तय करता है कि किन चरणों को निष्पादित करना है।

एआई द्वारा उपयोग किए जाने वाले उपलब्ध कार्यकर्ता फ़ंक्शन में analyze_text, recognize_image, assess_user_reputation, और escalate_to_manual_review शामिल हैं। प्रत्येक फ़ंक्शन कार्य को संबंधित कार्यकर्ता प्रक्रिया (TextAnalysisWorker, ImageRecognitionWorker, आदि) को सौंपता है और परिणाम को मॉडरेशन प्रतिलेख में जोड़ता है, एस्केलेशन फ़ंक्शन को छोड़कर, जो एक अंतिम स्थिति के रूप में कार्य करता है। अंत में, approve_content और reject_content फ़ंक्शन भी अंतिम स्थितियों के रूप में कार्य करते हैं।

एआई घटक सामग्री का विश्लेषण करता है और उचित कार्रवाई करने का निर्धारण करता है। यदि सामग्री में छवि संदर्भ शामिल हैं, तो यह दृश्य समीक्षा में सहायता के लिए recognize_image कार्यकर्ता को कॉल कर सकता है। यदि कोई कार्यकर्ता

संभावित हानिकारक सामग्री के बारे में चेतावनी देता है, तो एआई सामग्री को मैनुअल समीक्षा के लिए एस्केलेट करने या सीधे अस्वीकार करने का निर्णय ले सकता है। लेकिन चेतावनी की गंभीरता के आधार पर, एआई उस सामग्री को संभालने के तरीके के बारे में निर्णय लेने में उपयोगकर्ता प्रतिष्ठा मूल्यांकन के परिणामों का उपयोग करने का विकल्प चुन सकता है जिसके बारे में वह अन्यथा निश्चित नहीं है। उपयोग के मामले के आधार पर, शायद विश्वसनीय उपयोगकर्ताओं को पोस्ट करने में अधिक छूट मिल सकती है। और इसी तरह आगे भी...

पिछले प्रक्रिया प्रबंधक उदाहरण की तरह, मॉडरेशन प्रतिलेख कार्यप्रवाह निष्पादन का एक रिकॉर्ड है, जिसमें प्रत्येक चरण के परिणाम और एआई-जनित निर्णय शामिल हैं। इस प्रतिलेख का उपयोग ऑडिटिंग, पारदर्शिता और समय के साथ मॉडरेशन प्रक्रिया को बेहतर बनाने के लिए किया जा सकता है।

ContentModerator में एआई का लाभ उठाकर, सोशल मीडिया एप्लिकेशन सामग्री की विशेषताओं के आधार पर मॉडरेशन कार्यप्रवाह को गतिशील रूप से अनुकूलित कर सकता है और जटिल मॉडरेशन परिदृश्यों को बुद्धिमानी से संभाल सकता है। एआई घटक सूचित निर्णय ले सकता है, कार्यप्रवाह को अनुकूलित कर सकता है, और एक सुरक्षित और स्वस्थ सामुदायिक अनुभव सुनिश्चित कर सकता है।

आइए बुद्धिमान कार्यप्रवाह ऑर्केस्ट्रेशन के संदर्भ में पूर्वानुमानित कार्य निर्धारण और अपवाद प्रबंधन और पुनर्प्राप्ति को प्रदर्शित करने वाले दो और उदाहरणों की खोज करें।

ग्राहक सहायता प्रणाली में पूर्वानुमानित कार्य निर्धारण

Ruby on Rails के साथ निर्मित एक ग्राहक सहायता एप्लिकेशन में, ग्राहकों को समय पर सहायता प्रदान करने के लिए सहायता टिकटों का कुशलतापूर्वक प्रबंधन और प्राथमिकता निर्धारण महत्वपूर्ण है। SupportTicketScheduler घटक टिकट की तात्कालिकता, एजेंट विशेषज्ञता और कार्यभार जैसे विभिन्न कारकों के आधार पर उपलब्ध एजेंटों को सहायता टिकट पूर्वानुमानित रूप से शेड्यूल और असाइन करने के लिए एआई का लाभ उठाता है।

```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
```

इस उदाहरण में, SupportTicketScheduler को एक सपोर्ट टिकट ऑब्जेक्ट के साथ आरंभ किया जाता है और एक शेड्यूलिंग ट्रांसक्रिप्ट को बनाए रखता है। AI घटक टिकट विवरणों का विश्लेषण करता है और टिकट की तात्कालिकता, एजेंट की विशेषज्ञता, और अनुमानित एजेंट कार्यभार जैसे कारकों के आधार पर पूर्वानुमानित रूप से टिकट असाइनमेंट को शेड्यूल करता है।

AI द्वारा उपयोग किए जाने वाले उपलब्ध फंक्शन्स में analyze_ticket_urgency, list_available_agents, predict_agent_workload, और assign_ticket_to_agent शामिल हैं। प्रत्येक फंक्शन कार्य को संबंधित विश्लेषक या प्रेडिक्टर घटक को सौंपता है और परिणाम को शेड्यूलिंग ट्रांसक्रिप्ट में जोड़ता है। AI के पास delay_assignment फंक्शन का उपयोग करके असाइनमेंट में देरी करने का विकल्प भी है।

AI घटक शेड्यूलिंग ट्रांसक्रिप्ट की जांच करता है और टिकट असाइनमेंट पर सूचित निर्णय लेता है। यह टिकट की तात्कालिकता, उपलब्ध एजेंट्स की विशेषज्ञता, और प्रत्येक एजेंट के अनुमानित कार्यभार पर विचार करता है ताकि टिकट को संभालने के लिए सबसे उपयुक्त एजेंट का निर्धारण किया जा सके।

पूर्वानुमानित कार्य शेड्यूलिंग का लाभ उठाकर, ग्राहक सहायता एप्लिकेशन टिकट असाइनमेंट को अनुकूलित कर सकता है, प्रतिक्रिया समय को कम कर सकता है, और समग्र ग्राहक संतुष्टि में सुधार कर सकता है। सपोर्ट टिकट्स का सक्रिय और कुशल प्रबंधन यह सुनिश्चित करता है कि सही टिकट सही एजेंट्स को सही समय पर असाइन किए जाएं।

डेटा प्रोसेसिंग पाइपलाइन में अपवाद प्रबंधन और रिकवरी

अपवादों को संभालना और विफलताओं से रिकवर करना डेटा अखंडता सुनिश्चित करने और डेटा हानि को रोकने के लिए आवश्यक है। DataProcessingOrchestrator घटक एक डेटा प्रोसेसिंग पाइपलाइन में बुद्धिमत्तापूर्ण ढंग से अपवादों को संभालने और रिकवरी प्रक्रिया के संचालन के लिए AI का उपयोग करता है।

```ruby
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",
```

```ruby
        # ...
      },
      {
        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
```

इस उदाहरण में, DataProcessingOrchestrator को एक डेटा बैच ऑब्जेक्ट के साथ आरंभ किया जाता है और एक प्रोसेसिंग ट्रांसक्रिप्ट को बनाए रखता है। एआई कंपोनेंट डेटा प्रोसेसिंग पाइपलाइन का संचालन करता है, अपवादों को संभालता है और आवश्यकतानुसार विफलताओं से पुनर्प्राप्ति करता है।

एआई द्वारा उपयोग किए जाने वाले उपलब्ध फंक्शंस में validate_data, process_data, request_fix, retry_processing, और mark_data_as_failed शामिल हैं। प्रत्येक फंक्शन कार्य को संबंधित डेटा प्रोसेसिंग कंपोनेंट को सौंपता है और परिणाम या अपवाद विवरण को प्रोसेसिंग ट्रांसक्रिप्ट में जोड़ता है।

यदि validate_data चरण के दौरान वैलिडेशन अपवाद होता है, तो handle_validation_exception फंक्शन अपवाद डेटा को ट्रांसक्रिप्ट में जोड़ता है और नियंत्रण एआई को वापस सौंप देता है। इसी तरह, यदि process_data चरण के दौरान प्रोसेसिंग अपवाद होता है, तो एआई रिकवरी रणनीति पर निर्णय ले सकता है।

सामने आए अपवाद की प्रकृति के आधार पर, एआई अपने विवेक से request_fix को कॉल करने का निर्णय ले सकता है, जो एआई-संचालित SmartDataFixer कंपोनेंट

को कार्य सौंपता है (सेल्फ हीलिंग डेटा अध्याय देखें)। डेटा फिक्सर को सरल अंग्रेजी में यह विवरण मिलता है कि उसे @data_batch को कैसे संशोधित करना चाहिए ताकि प्रोसेसिंग को पुनः प्रयास किया जा सके। शायद एक सफल पुनर्प्रयास में उन रिकॉर्ड्स को डेटा बैच से हटाना शामिल होगा जो वैलिधेशन में विफल हो गए हैं और/या उन्हें मानवीय समीक्षा के लिए एक अलग प्रोसेसिंग पाइपलाइन में कॉपी करना होगा? संभावनाएं लगभग अनंत हैं।

एआई-संचालित अपवाद प्रबंधन और रिकवरी को शामिल करके, डेटा प्रोसेसिंग एप्लिकेशन अधिक लचीला और त्रुटि-सहिष्णु बन जाता है। DataProcessingOrchestrator बुद्धिमानी से अपवादों का प्रबंधन करता है, डेटा हानि को कम करता है, और डेटा प्रोसेसिंग कार्यप्रवाह के सुचारु निष्पादन को सुनिश्चित करता है।

निगरानी और लॉगिंग

निगरानी और लॉगिंग एआई-संचालित कार्यप्रवाह कंपोनेंट्स की प्रगति, प्रदर्शन और स्वास्थ्य की दृश्यता प्रदान करते हैं, जो डेवलपर्स को सिस्टम के व्यवहार को ट्रैक और विश्लेषण करने में सक्षम बनाते हैं। बुद्धिमान कार्यप्रवाह के डीबगिंग, ऑडिटिंग और निरंतर सुधार के लिए प्रभावी निगरानी और लॉगिंग तंत्र का कार्यान्वयन आवश्यक है।

कार्यप्रवाह प्रगति और प्रदर्शन की निगरानी

बुद्धिमान कार्यप्रवाह के सुचारु निष्पादन को सुनिश्चित करने के लिए, प्रत्येक कार्यप्रवाह कंपोनेंट की प्रगति और प्रदर्शन की निगरानी करना महत्वपूर्ण है। इसमें कार्यप्रवाह जीवनचक्र के दौरान प्रमुख मैट्रिक्स और घटनाओं को ट्रैक करना शामिल है।

निगरानी के कुछ महत्वपूर्ण पहलू हैं:

1. कार्यप्रवाह निष्पादन समय: प्रत्येक कार्यप्रवाह कंपोनेंट द्वारा अपना कार्य पूरा करने में लिए गए समय को मापें। यह प्रदर्शन बाधाओं की पहचान करने और समग्र कार्यप्रवाह दक्षता को अनुकूलित करने में मदद करता है।

2. संसाधन उपयोग: प्रत्येक कार्यप्रवाह कंपोनेंट द्वारा CPU, मेमोरी और स्टोरेज जैसे सिस्टम संसाधनों के उपयोग की निगरानी करें। यह सुनिश्चित करने में मदद करता है कि सिस्टम अपनी क्षमता के भीतर काम कर रहा है और कार्यभार को प्रभावी ढंग से संभाल सकता है।

3. त्रुटि दर और अपवाद: कार्यप्रवाह घटकों के भीतर त्रुटियों और अपवादों की घटनाओं को ट्रैक करें। यह संभावित समस्याओं की पहचान करने में मदद करता है और सक्रिय त्रुटि प्रबंधन और पुनर्प्राप्ति को सक्षम बनाता है।

4. निर्णय बिंदु और परिणाम: कार्यप्रवाह के भीतर निर्णय बिंदुओं और एआई-संचालित निर्णयों के परिणामों की निगरानी करें। यह एआई घटकों के व्यवहार और प्रभावशीलता में अंतर्दृष्टि प्रदान करता है।

निगरानी प्रक्रियाओं द्वारा एकत्रित डेटा को डैशबोर्ड में प्रदर्शित किया जा सकता है या निर्धारित रिपोर्ट के इनपुट के रूप में उपयोग किया जा सकता है जो सिस्टम प्रशासकों को सिस्टम के स्वास्थ्य के बारे में सूचित करते हैं।

निगरानी डेटा को समीक्षा और संभावित कार्रवाई के लिए एआई-संचालित सिस्टम प्रशासक प्रक्रिया में फीड किया जा सकता है!

महत्वपूर्ण घटनाओं और निर्णयों की लॉगिंग

लॉगिंग एक आवश्यक अभ्यास है जिसमें कार्यप्रवाह निष्पादन के दौरान होने वाली प्रमुख घटनाओं, निर्णयों और अपवादों के बारे में प्रासंगिक जानकारी को कैप्चर और स्टोर करना शामिल है।

लॉग करने के लिए कुछ महत्वपूर्ण पहलू हैं:

1. कार्यप्रवाह प्रारंभ और समापन: प्रत्येक कार्यप्रवाह इंस्टेंस के प्रारंभ और समाप्ति समय को लॉग करें, साथ ही इनपुट डेटा और उपयोगकर्ता संदर्भ जैसे किसी भी प्रासंगिक मेटाडेटा को भी।

2. घटक निष्पादन: प्रत्येक कार्यप्रवाह घटक के निष्पादन विवरण को लॉग करें, जिसमें इनपुट पैरामीटर, आउटपुट परिणाम और कोई भी मध्यवर्ती डेटा जो उत्पन्न हुआ है, शामिल हैं।

3. एआई निर्णय और तर्क: एआई घटकों द्वारा लिए गए निर्णयों को अंतर्निहित तर्क या विश्वास स्कोर के साथ लॉग करें। यह पारदर्शिता प्रदान करता है और एआई-संचालित निर्णयों की ऑडिटिंग को सक्षम बनाता है।

4. अपवाद और त्रुटि संदेश: कार्यप्रवाह निष्पादन के दौरान आने वाले किसी भी अपवाद या त्रुटि संदेश को लॉग करें, जिसमें स्टैक ट्रेस और प्रासंगिक संदर्भ जानकारी शामिल है।

लॉगिंग को विभिन्न तकनीकों का उपयोग करके लागू किया जा सकता है, जैसे लॉग फ़ाइलों में लिखना, डेटाबेस में लॉग स्टोर करना, या लॉग को एक केंद्रीकृत लॉगिंग सेवा में भेजना। यह महत्वपूर्ण है कि एक ऐसा लॉगिंग फ्रेमवर्क चुना जाए जो लचीलापन, स्केलेबिलिटी और एप्लिकेशन की आर्किटेक्चर के साथ आसान एकीकरण प्रदान करता हो।

यहाँ एक उदाहरण है कि ActiveSupport::Logger क्लास का उपयोग करके Ruby on Rails एप्लिकेशन में लॉगिंग को कैसे लागू किया जा सकता है:

```
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)
```

कार्यप्रवाह घटकों और AI निर्णय बिंदुओं में रणनीतिक रूप से लॉगिंग स्टेटमेंट्स को

स्थापित करके, डेवलपर्स डीबगिंग, लेखा-परीक्षण और विश्लेषण के लिए महत्वपूर्ण जानकारी प्राप्त कर सकते हैं।

निगरानी और लॉगिंग के लाभ

बुद्धिमान कार्यप्रवाह संचालन में निगरानी और लॉगिंग को लागू करने से कई लाभ मिलते हैं:

1. डीबगिंग और समस्या निवारण: विस्तृत लॉग्स और निगरानी डेटा डेवलपर्स को समस्याओं की पहचान और निदान जल्दी करने में मदद करते हैं। वे कार्यप्रवाह निष्पादन प्रवाह, घटक अंतःक्रियाओं और किसी भी त्रुटि या अपवाद की जानकारी प्रदान करते हैं।

2. प्रदर्शन अनुकूलन: प्रदर्शन मैट्रिक्स की निगरानी से डेवलपर्स बाधाओं की पहचान कर सकते हैं और बेहतर दक्षता के लिए कार्यप्रवाह घटकों को अनुकूलित कर सकते हैं। निष्पादन समय, संसाधन उपयोग और अन्य मैट्रिक्स का विश्लेषण करके, डेवलपर्स सिस्टम के समग्र प्रदर्शन को सुधारने के लिए सूचित निर्णय ले सकते हैं।

3. लेखा-परीक्षण और अनुपालन: प्रमुख घटनाओं और निर्णयों की लॉगिंग नियामक अनुपालन और जवाबदेही के लिए एक लेखा-परीक्षण निशान प्रदान करती है। यह संगठनों को AI घटकों द्वारा की गई कार्रवाइयों को ट्रैक और सत्यापित करने तथा व्यावसायिक नियमों और कानूनी आवश्यकताओं का पालन सुनिश्चित करने में सक्षम बनाती है।

4. निरंतर सुधार: निगरानी और लॉगिंग डेटा बुद्धिमान कार्यप्रवाह के निरंतर सुधार के लिए महत्वपूर्ण इनपुट के रूप में काम करते हैं। ऐतिहासिक डेटा का विश्लेषण करके, पैटर्न की पहचान करके और AI निर्णयों की प्रभावशीलता को मापकर, डेवलपर्स कार्यप्रवाह संचालन तर्क को क्रमिक रूप से परिष्कृत और बेहतर बना सकते हैं।

विचारणीय बिंदु और सर्वोत्तम प्रथाएं

बुद्धिमान कार्यप्रवाह संचालन में निगरानी और लॉगिंग को लागू करते समय, निम्नलिखित सर्वोत्तम प्रथाओं पर विचार करें:

1. स्पष्ट निगरानी मैट्रिक्स परिभाषित करें: कार्यप्रवाह की विशिष्ट आवश्यकताओं के आधार पर निगरानी किए जाने वाले प्रमुख मैट्रिक्स और घटनाओं की पहचान करें। सिस्टम के प्रदर्शन, स्वास्थ्य और व्यवहार में सार्थक अंतर्दृष्टि प्रदान करने वाले मैट्रिक्स पर ध्यान केंद्रित करें।

2. सूक्ष्म लॉगिंग लागू करें: सुनिश्चित करें कि लॉगिंग स्टेटमेंट्स कार्यप्रवाह घटकों और AI निर्णय बिंदुओं में उचित स्थानों पर रखे गए हैं। प्रासंगिक संदर्भ जानकारी, जैसे इनपुट पैरामीटर्स, आउटपुट परिणाम और कोई भी मध्यवर्ती डेटा कैप्चर करें।

3. संरचित लॉगिंग का उपयोग करें: लॉग डेटा के आसान पार्सिंग और विश्लेषण की सुविधा के लिए एक संरचित लॉगिंग प्रारूप अपनाएं। संरचित लॉगिंग लॉग प्रविष्टियों की बेहतर खोज, फ़िल्टरिंग और एकत्रीकरण की अनुमति देती है।

4. लॉग प्रतिधारण और रोटेशन का प्रबंधन करें: लॉग फ़ाइलों के भंडारण और जीवन चक्र को प्रबंधित करने के लिए लॉग प्रतिधारण और रोटेशन नीतियां लागू करें। कानूनी आवश्यकताओं, भंडारण बाधाओं और विश्लेषण आवश्यकताओं के आधार पर उचित प्रतिधारण अवधि निर्धारित करें। यदि संभव हो, तो लॉगिंग को Papertrail जैसी तृतीय-पक्ष सेवा में स्थानांतरित करें।

5. संवेदनशील जानकारी सुरक्षित करें: व्यक्तिगत पहचान योग्य जानकारी (PII) या गोपनीय व्यावसायिक डेटा जैसी संवेदनशील जानकारी को लॉग करते समय सावधान रहें। लॉग फ़ाइलों में संवेदनशील जानकारी की सुरक्षा के लिए डेटा मास्किंग या एन्क्रिप्शन जैसे उचित सुरक्षा उपाय लागू करें।

6. निगरानी और चेतावनी टूल्स के साथ एकीकरण करें: निगरानी और लॉगिंग डेटा के संग्रह, विश्लेषण और विज़ुअलाइज़ेशन को केंद्रीकृत करने के लिए मॉनिटरिंग और अलर्टिंग टूल्स का लाभ उठाएं। ये टूल्स रीयल-टाइम अंतर्दृष्टि प्रदान कर सकते हैं, पूर्वनिर्धारित सीमाओं के आधार पर अलर्ट जनरेट कर सकते हैं, और सक्रिय समस्या पहचान और समाधान की सुविधा प्रदान कर सकते हैं। इन टूल्स में मेरा पसंदीदा Datadog है।

व्यापक निगरानी और लॉगिंग तंत्र को लागू करके, डेवलपर्स बुद्धिमान कार्यप्रवाह के व्यवहार और प्रदर्शन में मूल्यवान अंतर्दृष्टि प्राप्त कर सकते हैं। ये अंतर्दृष्टियां AI-

संचालित कार्यप्रवाह ऑर्केस्ट्रेशन सिस्टम के प्रभावी डीबगिंग, अनुकूलन और निरंतर सुधार को सक्षम बनाती हैं।

मापनीयता और प्रदर्शन विचार

मापनीयता और प्रदर्शन बुद्धिमान कार्यप्रवाह ऑर्केस्ट्रेशन सिस्टम को डिज़ाइन और कार्यान्वित करते समय विचार करने योग्य महत्वपूर्ण पहलू हैं। जैसे-जैसे समवर्ती कार्यप्रवाह की मात्रा और AI-संचालित घटकों की जटिलता बढ़ती है, यह सुनिश्चित करना आवश्यक हो जाता है कि सिस्टम कार्यभार को कुशलतापूर्वक संभाल सके और बढ़ती मांगों को पूरा करने के लिए निर्बाध रूप से स्केल कर सके।

समवर्ती कार्यप्रवाह की उच्च मात्रा को संभालना

बुद्धिमान कार्यप्रवाह ऑर्केस्ट्रेशन सिस्टम को अक्सर बड़ी संख्या में समवर्ती कार्यप्रवाह को संभालने की आवश्यकता होती है। मापनीयता सुनिश्चित करने के लिए, निम्नलिखित रणनीतियों पर विचार करें:

1. अतुल्यकालिक प्रसंस्करण: कार्यप्रवाह घटकों के निष्पादन को अलग करने के लिए अतुल्यकालिक प्रसंस्करण तंत्र लागू करें। यह सिस्टम को प्रत्येक घटक के पूरा होने की प्रतीक्षा किए बिना या उसे ब्लॉक किए बिना कई कार्यप्रवाह को समवर्ती रूप से संभालने की अनुमति देता है। अतुल्यकालिक प्रसंस्करण को संदेश कतारों, इवेंट-संचालित आर्किटेक्चर, या Sidekiq जैसे बैकग्राउंड जॉब प्रोसेसिंग फ्रेमवर्क का उपयोग करके प्राप्त किया जा सकता है।

2. वितरित आर्किटेक्चर: सर्वरलेस घटकों (जैसे AWS Lambda) का उपयोग करने या आपके मुख्य एप्लिकेशन सर्वर के साथ कई नोड्स या सर्वर में कार्यभार को वितरित करने के लिए सिस्टम आर्किटेक्चर डिज़ाइन करें। यह क्षैतिज मापनीयता को सक्षम करता है, जहां बढ़े हुए कार्यप्रवाह वॉल्यूम को संभालने के लिए अतिरिक्त नोड्स जोड़े जा सकते हैं।

3. समानांतर निष्पादन: कार्यप्रवाह के भीतर समानांतर निष्पादन के अवसरों की पहचान करें। कुछ कार्यप्रवाह घटक एक-दूसरे से स्वतंत्र हो सकते हैं और समवर्ती रूप से निष्पादित किए जा सकते हैं। मल्टी-थ्रेडिंग या वितरित टास्क कतारों जैसी समानांतर प्रोसेसिंग तकनीकों का लाभ उठाकर, सिस्टम संसाधन उपयोग को अनुकूलित कर सकता है और समग्र कार्यप्रवाह निष्पादन समय को कम कर सकता है।

AI-संचालित घटकों के प्रदर्शन का अनुकूलन

एआई-संचालित घटक, जैसे मशीन लर्निंग मॉडल या नेचुरल लैंग्वेज प्रोसेसिंग इंजन, कम्प्यूटेशनल रूप से गहन हो सकते हैं और कार्यप्रवाह नियोजन प्रणाली के समग्र प्रदर्शन को प्रभावित कर सकते हैं। एआई घटकों के प्रदर्शन को अनुकूलित करने के लिए, निम्नलिखित तकनीकों पर विचार करें:

1. कैशिंग: यदि आपकी एआई प्रोसेसिंग पूरी तरह से जेनरेटिव है और चैट पूर्णता उत्पन्न करने के लिए रीयलटाइम सूचना लुकअप या बाहरी एकीकरण शामिल नहीं है, तो आप बार-बार एक्सेस किए जाने वाले या कम्प्यूटेशनल रूप से महंगे ऑपरेशन के परिणामों को स्टोर करने और पुन: उपयोग करने के लिए कैशिंग तंत्र का उपयोग कर सकते हैं।

2. मॉडल अनुकूलन: कार्यप्रवाह घटकों में एआई मॉडल के उपयोग को निरंतर अनुकूलित करें। इसमें प्रॉम्प्ट डिस्टिलेशन जैसी तकनीकें शामिल हो सकती हैं या यह केवल नए मॉडल के उपलब्ध होने पर उनका परीक्षण करने का मामला हो सकता है।

3. बैच प्रोसेसिंग: यदि आप GPT-4 क्लास मॉडल के साथ काम कर रहे हैं, तो आप कई डेटा पॉइंट्स या अनुरोधों को व्यक्तिगत रूप से प्रोसेस करने के बजाय एक बैच में प्रोसेस करने के लिए बैच प्रोसेसिंग तकनीकों का लाभ उठा सकते हैं। डेटा को बैच में प्रोसेस करके, सिस्टम संसाधन उपयोग को अनुकूलित कर सकता है और बार-बार मॉडल अनुरोधों के ओवरहेड को कम कर सकता है।

प्रदर्शन की निगरानी और प्रोफाइलिंग

बुद्धिमान कार्यप्रवाह नियोजन प्रणाली की स्केलेबिलिटी को अनुकूलित करने और प्रदर्शन बाधाओं की पहचान करने के लिए, निगरानी और प्रोफाइलिंग तंत्र को लागू करना महत्वपूर्ण है। निम्नलिखित दृष्टिकोणों पर विचार करें:

1. प्रदर्शन मैट्रिक्स: प्रमुख प्रदर्शन मैट्रिक्स को परिभाषित और ट्रैक करें, जैसे प्रतिक्रिया समय, थ्रूपुट, संसाधन उपयोग और विलंबता। ये मैट्रिक्स सिस्टम के प्रदर्शन में अंतर्दृष्टि प्रदान करते हैं और अनुकूलन के क्षेत्रों की पहचान में मदद करते हैं। लोकप्रिय एआई मॉडल एग्रीगेटर OpenRouter प्रत्येक API प्रतिक्रिया में Host[1] और Speed[2] मैट्रिक्स शामिल करता है, जो इन प्रमुख मैट्रिक्स को ट्रैक करना सरल बनाता है।

2. प्रोफाइलिंग टूल्स: व्यक्तिगत कार्यप्रवाह घटकों और एआई ऑपरेशंस के प्रदर्शन का विश्लेषण करने के लिए प्रोफाइलिंग टूल्स का उपयोग करें। प्रोफाइलिंग टूल्स प्रदर्शन हॉटस्पॉट, अक्षम कोड पाथ, या संसाधन-गहन ऑपरेशंस की पहचान में मदद कर सकते हैं। लोकप्रिय प्रोफाइलिंग टूल्स में New Relic, Scout, या प्रोग्रामिंग भाषा या फ्रेमवर्क द्वारा प्रदान किए गए बिल्ट-इन प्रोफाइलर शामिल हैं।

3. लोड टेस्टिंग: विभिन्न स्तरों के समवर्ती कार्यभार के तहत सिस्टम के प्रदर्शन का मूल्यांकन करने के लिए लोड टेस्टिंग करें। लोड टेस्टिंग सिस्टम की मापनीयता सीमाओं की पहचान करने, प्रदर्शन में गिरावट का पता लगाने और यह सुनिश्चित करने में मदद करती है कि सिस्टम प्रदर्शन से समझौता किए बिना अपेक्षित ट्रैफ़िक को संभाल सकता है।

4. निरंतर निगरानी: प्रदर्शन संबंधी समस्याओं और बाधाओं का सक्रिय रूप से पता लगाने के लिए निरंतर निगरानी और चेतावनी तंत्र लागू करें। प्रमुख प्रदर्शन संकेतकों (KPI) को ट्रैक करने और पूर्वनिर्धारित सीमाओं के उल्लंघन होने पर सूचनाएं प्राप्त करने के लिए निगरानी डैशबोर्ड और अलर्ट सेट करें। यह प्रदर्शन समस्याओं की त्वरित

[1]Host मॉडल होस्ट से स्ट्रीम्ड जेनरेशन का पहला बाइट प्राप्त करने में लगा समय है, जिसे "टाइम टू फर्स्ट बाइट" भी कहा जाता है।

[2]Speed की गणना पूर्णता टोकन की संख्या को कुल जेनरेशन समय से विभाजित करके की जाती है। नॉन-स्ट्रीम्ड अनुरोधों के लिए विलंबता को जेनरेशन समय का हिस्सा माना जाता है।

पहचान और समाधान को सक्षम बनाता है।

स्केलिंग रणनीतियां

बढ़ते कार्यभार को संभालने और बुद्धिमान वर्कफ़्लो ऑर्केस्ट्रेशन सिस्टम की मापनीयता सुनिश्चित करने के लिए, निम्नलिखित स्केलिंग रणनीतियों पर विचार करें:

1. वर्टिकल स्केलिंग: वर्टिकल स्केलिंग में उच्च कार्यभार को संभालने के लिए व्यक्तिगत नोड्स या सर्वर के संसाधनों (जैसे, CPU, मेमोरी) को बढ़ाना शामिल है। यह दृष्टिकोण तब उपयुक्त होता है जब सिस्टम को जटिल वर्कफ़्लो या AI संचालन को संभालने के लिए अधिक प्रोसेसिंग पावर या मेमोरी की आवश्यकता होती है।

2. होरिज़ॉन्टल स्केलिंग: होरिज़ॉन्टल स्केलिंग में कार्यभार को वितरित करने के लिए सिस्टम में अधिक नोड्स या सर्वर जोड़ना शामिल है। यह दृष्टिकोण तब प्रभावी होता है जब सिस्टम को बड़ी संख्या में समवर्ती वर्कफ़्लो को संभालने की आवश्यकता होती है या जब कार्यभार को आसानी से कई नोड्स में वितरित किया जा सकता है। होरिज़ॉन्टल स्केलिंग के लिए ट्रैफ़िक के समान वितरण को सुनिश्चित करने के लिए एक वितरित आर्किटेक्चर और लोड बैलेंसिंग तंत्र की आवश्यकता होती है।

3. ऑटो-स्केलिंग: कार्यभार मांग के आधार पर नोड्स या संसाधनों की संख्या को स्वचालित रूप से समायोजित करने के लिए ऑटो-स्केलिंग तंत्र लागू करें। ऑटो-स्केलिंग सिस्टम को आने वाले ट्रैफ़िक के आधार पर गतिशील रूप से ऊपर या नीचे स्केल करने की अनुमति देता है, जो इष्टतम संसाधन उपयोग और लागत-दक्षता सुनिश्चित करता है। Amazon Web Services (AWS) या Google Cloud Platform (GCP) जैसे क्लाउड प्लेटफ़ॉर्म ऑटो-स्केलिंग क्षमताएं प्रदान करते हैं जिनका उपयोग बुद्धिमान वर्कफ़्लो ऑर्केस्ट्रेशन सिस्टम के लिए किया जा सकता है।

प्रदर्शन अनुकूलन तकनीकें

स्केलिंग रणनीतियों के अलावा, बुद्धिमान वर्कफ़्लो ऑर्केस्ट्रेशन सिस्टम की दक्षता बढ़ाने के लिए निम्नलिखित प्रदर्शन अनुकूलन तकनीकों पर विचार करें:

1. कुशल डेटा भंडारण और पुनर्प्राप्ति: वर्कफ़्लो घटकों द्वारा उपयोग किए जाने वाले डेटा भंडारण और पुनर्प्राप्ति तंत्रों को अनुकूलित करें। डेटा-गहन संचालन की विलंबता को कम करने और प्रदर्शन में सुधार करने के लिए कुशल डेटाबेस इंडेक्सिंग, क्वेरी अनुकूलन तकनीकों और डेटा कैशिंग का उपयोग करें।

2. असिंक्रोनस I/O: ब्लॉकिंग को रोकने और सिस्टम की प्रतिक्रिया को बेहतर बनाने के लिए असिंक्रोनस I/O संचालन का उपयोग करें। असिंक्रोनस I/O सिस्टम को I/O संचालन के पूरा होने की प्रतीक्षा किए बिना कई अनुरोधों को समवर्ती रूप से संभालने की अनुमति देता है, जिससे संसाधन उपयोग अधिकतम होता है।

3. कुशल सीरियलाइज़ेशन और डीसीरियलाइज़ेशन: वर्कफ़्लो कंपोनेंट्स के बीच डेटा एक्सचेंज के लिए उपयोग किए जाने वाली सीरियलाइज़ेशन और डीसीरियलाइज़ेशन प्रक्रियाओं को अनुकूलित करें। कुशल सीरियलाइज़ेशन फॉर्मेट्स का उपयोग करें, जैसे Protocol Buffers या MessagePack, जो डेटा सीरियलाइज़ेशन के ओवरहेड को कम करते हैं और इंटर-कंपोनेंट कम्युनिकेशन का प्रदर्शन बेहतर बनाते हैं।

Ruby-आधारित एप्लिकेशन्स के लिए, Universal ID पर विचार करें।। Universal ID, MessagePack और Brotli दोनों का लाभ उठाता है (एक ऐसा कॉम्बो जो स्पीड और सर्वश्रेष्ठ डेटा कम्प्रेशन के लिए बनाया गया है)। जब इन लाइब्रेरीज को संयुक्त किया जाता है, तो ये Protocol Buffers की तुलना में 30% तक तेज होती हैं और कम्प्रेशन दर 2-5% के भीतर रहती है।

4. कम्प्रेशन और एनकोडिंग: वर्कफ़्लो कंपोनेंट्स के बीच स्थानांतरित डेटा का आकार कम करने के लिए कम्प्रेशन और एनकोडिंग तकनीकों का प्रयोग करें। कम्प्रेशन एल्गोरिथम, जैसे gzip या Brotli, नेटवर्क बैंडविड्थ उपयोग को काफी कम कर सकते हैं और सिस्टम के समग्र प्रदर्शन को बेहतर बना सकते हैं।

बुद्धिमान वर्कफ़्लो ऑर्केस्ट्रेशन सिस्टम के डिज़ाइन और कार्यान्वयन के दौरान स्केलेबिलिटी और प्रदर्शन पहलुओं पर विचार करके, आप सुनिश्चित कर सकते हैं कि आपका सिस्टम समवर्ती वर्कफ़्लोज़ की उच्च मात्रा को संभाल सकता है, AI-संचालित कंपोनेंट्स के प्रदर्शन को अनुकूलित कर सकता है, और बढ़ती मांगों को पूरा

करने के लिए निर्बाध रूप से स्केल कर सकता है। वर्कलोड और जटिलता में समय के साथ वृद्धि होने पर सिस्टम के प्रदर्शन और प्रतिक्रिया को बनाए रखने के लिए निरंतर निगरानी, प्रोफाइलिंग और अनुकूलन प्रयास आवश्यक हैं।

वर्कफ़्लोज़ का परीक्षण और सत्यापन

परीक्षण और सत्यापन बुद्धिमान वर्कफ़्लो ऑर्केस्ट्रेशन सिस्टम के विकास और रखरखाव के महत्वपूर्ण पहलू हैं। AI-संचालित वर्कफ़्लोज़ की जटिल प्रकृति को देखते हुए, यह सुनिश्चित करना आवश्यक है कि प्रत्येक कंपोनेंट अपेक्षित रूप से कार्य करे, समग्र वर्कफ़्लो सही ढंग से व्यवहार करे, और AI निर्णय सटीक और विश्वसनीय हों। इस खंड में, हम बुद्धिमान वर्कफ़्लोज़ के परीक्षण और सत्यापन के लिए विभिन्न तकनीकों और विचारों की खोज करेंगे।

वर्कफ़्लो कंपोनेंट्स का यूनिट टेस्टिंग

यूनिट टेस्टिंग में व्यक्तिगत वर्कफ़्लो कंपोनेंट्स का अलग-अलग परीक्षण शामिल होता है ताकि उनकी सटीकता और मजबूती की जांच की जा सके। AI-संचालित वर्कफ़्लो कंपोनेंट्स का यूनिट टेस्टिंग करते समय निम्नलिखित बातों पर विचार करें:

1. इनपुट वैलिडेशन: विभिन्न प्रकार के इनपुट को संभालने की कंपोनेंट की क्षमता का परीक्षण करें, जिसमें वैध और अवैध डेटा शामिल हैं। सत्यापित करें कि कंपोनेंट एज केस को सहजता से संभालता है और उपयुक्त त्रुटि संदेश या अपवाद प्रदान करता है।

2. आउटपुट वेरिफिकेशन: सत्यापित करें कि कंपोनेंट दिए गए इनपुट सेट के लिए अपेक्षित आउटपुट उत्पन्न करता है। सटीकता सुनिश्चित करने के लिए वास्तविक आउटपुट की अपेक्षित परिणामों से तुलना करें।

3. त्रुटि प्रबंधन: विभिन्न त्रुटि परिदृश्यों का अनुकरण करके घटक के त्रुटि प्रबंधन तंत्रों का परीक्षण करें, जैसे अमान्य इनपुट, संसाधन अनुपलब्धता, या अप्रत्याशित

अपवाद। सत्यापित करें कि घटक त्रुटियों को उचित रूप से पकड़ता और संभालता है।

4. सीमा स्थितियाँ: सीमा स्थितियों के तहत घटक के व्यवहार का परीक्षण करें, जैसे खाली इनपुट, अधिकतम इनपुट आकार, या चरम मान। सुनिश्चित करें कि घटक इन स्थितियों को क्रैश हुए बिना या गलत परिणाम उत्पन्न किए बिना सुचारू रूप से संभालता है।

यहाँ RSpec परीक्षण ढाँचे का उपयोग करते हुए Ruby में एक वर्कफ़्लो घटक के लिए एकक परीक्षण का एक उदाहरण है:

```
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
```

इस उदाहरण में, OrderValidator घटक का परीक्षण दो परीक्षण मामलों का उपयोग करके किया जाता है: एक वैध ऑर्डर के लिए और दूसरा अवैध ऑर्डर के लिए। परीक्षण मामले यह सत्यापित करते हैं कि validate विधि ऑर्डर की वैधता के आधार पर अपेक्षित बूलियन मान लौटाती है।

कार्यप्रवाह अंतःक्रियाओं का एकीकरण परीक्षण

एकीकरण परीक्षण विभिन्न कार्यप्रवाह घटकों के बीच अंतःक्रियाओं और डेटा प्रवाह के सत्यापन पर केंद्रित होता है। यह सुनिश्चित करता है कि घटक निर्बाध रूप से एक साथ काम करें और अपेक्षित परिणाम उत्पन्न करें। बुद्धिमान कार्यप्रवाह का एकीकरण परीक्षण करते समय, निम्नलिखित बातों पर विचार करें:

1. घटक अंतःक्रिया: कार्यप्रवाह घटकों के बीच संचार और डेटा विनिमय का परीक्षण करें। सत्यापित करें कि एक घटक का आउटपुट कार्यप्रवाह में अगले घटक के इनपुट के रूप में सही ढंग से पारित किया जाता है।

2. डेटा संगतता: सुनिश्चित करें कि कार्यप्रवाह में डेटा सुसंगत और सटीक रहता है। सत्यापित करें कि डेटा रूपांतरण, गणनाएं और एकत्रीकरण सही ढंग से किए जाते हैं।

3. अपवाद प्रसार: कार्यप्रवाह घटकों में अपवादों और त्रुटियों के प्रसार और प्रबंधन का परीक्षण करें। सत्यापित करें कि कार्यप्रवाह व्यवधान को रोकने के लिए अपवादों को उचित रूप से पकड़ा, लॉग किया और प्रबंधित किया जाता है।

4. अतुल्यकालिक व्यवहार: यदि कार्यप्रवाह में अतुल्यकालिक घटक या समानांतर निष्पादन शामिल है, तो समन्वय और सिंक्रनाइज़ेशन तंत्र का परीक्षण करें। सुनिश्चित करें कि कार्यप्रवाह समवर्ती और अतुल्यकालिक परिदृश्यों में सही ढंग से व्यवहार करता है।

यहाँ RSpec परीक्षण फ्रेमवर्क का उपयोग करके Ruby में एक कार्यप्रवाह के लिए एकीकरण परीक्षण का एक उदाहरण दिया गया है:

```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
```

इस उदाहरण में, OrderProcessingWorkflow का परीक्षण विभिन्न कार्यप्रवाह घटकों के बीच अंतःक्रिया को सत्यापित करके किया जाता है। परीक्षण केस प्रत्येक घटक के व्यवहार के लिए अपेक्षाएं स्थापित करता है और यह सुनिश्चित करता है कि कार्यप्रवाह ऑर्डर को सफलतापूर्वक संसाधित करता है, तदनुसार ऑर्डर की स्थिति को अपडेट करता है।

AI निर्णय बिंदुओं का परीक्षण

AI निर्णय बिंदुओं का परीक्षण AI-संचालित कार्यप्रवाह की सटीकता और विश्वसनीयता सुनिश्चित करने के लिए महत्वपूर्ण है। AI निर्णय बिंदुओं का परीक्षण करते समय, निम्नलिखित बातों पर विचार करें:

1. निर्णय सटीकता: सुनिश्चित करें कि AI घटक इनपुट डेटा और प्रशिक्षित मॉडल के आधार पर सटीक निर्णय लेता है। AI निर्णयों की तुलना अपेक्षित परिणामों या आधार सत्य डेटा से करें।

2. एज केस: एज केस और असामान्य परिस्थितियों में AI घटक के व्यवहार का

परीक्षण करें। सत्यापित करें कि AI घटक इन मामलों को सहजता से संभालता है और उचित निर्णय लेता है।

3. पक्षपात और निष्पक्षता: संभावित पक्षपात के लिए AI घटक का मूल्यांकन करें और सुनिश्चित करें कि यह निष्पक्ष और बिना पक्षपात के निर्णय लेता है। विविध इनपुट डेटा के साथ घटक का परीक्षण करें और किसी भी भेदभावपूर्ण पैटर्न के लिए परिणामों का विश्लेषण करें।

4. व्याख्येयता: यदि AI घटक अपने निर्णयों के लिए स्पष्टीकरण या तर्क प्रदान करता है, तो स्पष्टीकरण की सटीकता और स्पष्टता की जांच करें। सुनिश्चित करें कि स्पष्टीकरण अंतर्निहित निर्णय लेने की प्रक्रिया के अनुरूप हैं।

यहाँ RSpec परीक्षण ढांचे का उपयोग करके Ruby में AI निर्णय बिंदु के परीक्षण का एक उदाहरण दिया गया है:

```ruby
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
```

इस उदाहरण में, FraudDetector AI कंपोनेंट को दो टेस्ट केस के साथ परीक्षण किया गया है: एक धोखाधड़ी वाले लेनदेन के लिए और दूसरा वैध लेनदेन के लिए। टेस्ट केस यह सत्यापित करते हैं कि detect_fraud मेथड लेनदेन की विशेषताओं के आधार पर अपेक्षित बूलियन वैल्यू लौटाता है।

एंड-टू-एंड टेस्टिंग

एंड-टू-एंड टेस्टिंग में शुरू से अंत तक पूरे कार्यप्रवाह का परीक्षण शामिल होता है, जो वास्तविक दुनिया के परिदृश्यों और उपयोगकर्ता इंटरैक्शन का अनुकरण करता है। यह सुनिश्चित करता है कि कार्यप्रवाह सही ढंग से व्यवहार करता है और वांछित परिणाम देता है। बुद्धिमान कार्यप्रवाह के लिए एंड-टू-एंड टेस्टिंग करते समय निम्नलिखित बातों पर विचार करें:

1. उपयोगकर्ता परिदृश्य: सामान्य उपयोगकर्ता परिदृश्यों की पहचान करें और इन परिदृश्यों के तहत कार्यप्रवाह के व्यवहार का परीक्षण करें। सत्यापित करें कि कार्यप्रवाह उपयोगकर्ता इनपुट को सही ढंग से संभालता है, उचित निर्णय लेता है, और अपेक्षित आउटपुट उत्पन्न करता है।

2. डेटा सत्यापन: सुनिश्चित करें कि कार्यप्रवाह डेटा असंगतियों या सुरक्षा कमजोरियों को रोकने के लिए उपयोगकर्ता इनपुट का सत्यापन और शुद्धिकरण करता है। विभिन्न प्रकार के इनपुट डेटा के साथ कार्यप्रवाह का परीक्षण करें, जिसमें वैध और अवैध डेटा शामिल हैं।

3. त्रुटि पुनर्प्राप्ति: कार्यप्रवाह की त्रुटियों और अपवादों से उबरने की क्षमता का परीक्षण करें। त्रुटि परिदृश्यों का अनुकरण करें और सत्यापित करें कि कार्यप्रवाह उन्हें सुचारू रूप से संभालता है, त्रुटियों को लॉग करता है, और उचित पुनर्प्राप्ति कार्रवाई करता है।

4. प्रदर्शन और स्केलेबिलिटी: विभिन्न लोड परिस्थितियों में कार्यप्रवाह के प्रदर्शन और स्केलेबिलिटी का मूल्यांकन करें। बड़ी मात्रा में समवर्ती अनुरोधों के साथ कार्यप्रवाह का परीक्षण करें और प्रतिक्रिया समय, संसाधन उपयोग और समग्र सिस्टम स्थिरता को मापें।

यहाँ Ruby में RSpec परीक्षण फ्रेमवर्क और उपयोगकर्ता इंटरैक्शन का अनुकरण करने के लिए Capybara लाइब्रेरी का उपयोग करके एक कार्यप्रवाह के लिए एंड-टू-एंड टेस्ट का एक उदाहरण दिया गया है:

```
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
```

इस उदाहरण में, एंड-टू-एंड टेस्ट वेब इंटरफ़ेस के माध्यम से उपयोगकर्ता द्वारा ऑर्डर प्लेस करने की प्रक्रिया का अनुकरण करता है। यह आवश्यक फॉर्म फ़ील्ड भरता है, ऑर्डर सबमिट करता है, और यह सत्यापित करता है कि ऑर्डर सफलतापूर्वक प्रोसेस किया गया है, उचित पुष्टिकरण संदेश प्रदर्शित करता है और डेटाबेस में ऑर्डर की स्थिति को अपडेट करता है।

कंटीन्युअस इंटीग्रेशन और डिप्लॉयमेंट

बुद्धिमान वर्कफ़्लो की विश्वसनीयता और रखरखाव सुनिश्चित करने के लिए, कंटीन्युअस इंटीग्रेशन और डिप्लॉयमेंट (CI/CD) पाइपलाइन में परीक्षण और सत्यापन को एकीकृत करने की सिफारिश की जाती है। यह प्रोडक्शन में डिप्लॉय करने से पहले वर्कफ़्लो परिवर्तनों के स्वचालित परीक्षण और सत्यापन की अनुमति देता है। निम्नलिखित प्रथाओं पर विचार करें:

1. स्वचालित परीक्षण निष्पादन: वर्कफ़्लो कोडबेस में परिवर्तन होने पर स्वचालित रूप से टेस्ट सूट चलाने के लिए CI/CD पाइपलाइन को कॉन्फ़िगर करें। यह सुनिश्चित

करता है कि विकास प्रक्रिया के प्रारंभिक चरण में किसी भी रिग्रेशन या विफलता का पता चल जाए।

2. टेस्ट कवरेज निगरानी: वर्कफ़्लो घटकों और AI निर्णय बिंदुओं के टेस्ट कवरेज को मापें और मॉनिटर करें। महत्वपूर्ण पथों और परिदृश्यों का पूरी तरह से परीक्षण सुनिश्चित करने के लिए उच्च टेस्ट कवरेज का लक्ष्य रखें।

3. निरंतर प्रतिक्रिया: विकास वर्कफ़्लो में टेस्ट परिणामों और कोड गुणवत्ता मेट्रिक्स को एकीकृत करें। CI/CD प्रक्रिया के दौरान परीक्षणों की स्थिति, कोड गुणवत्ता और पता चली किसी भी समस्या के बारे में डेवलपर्स को निरंतर प्रतिक्रिया प्रदान करें।

4. स्टेजिंग एनवायरनमेंट: वर्कफ़्लो को स्टेजिंग एनवायरनमेंट में डिप्लॉय करें जो प्रोडक्शन एनवायरनमेंट के बहुत करीब हो। इंफ्रास्ट्रक्चर, कॉन्फ़िगरेशन, या डेटा एकीकरण से संबंधित किसी भी समस्या को पकड़ने के लिए स्टेजिंग एनवायरनमेंट में अतिरिक्त परीक्षण और सत्यापन करें।

5. रोलबैक मैकेनिज्म: डिप्लॉयमेंट विफलताओं या प्रोडक्शन में पता चली गंभीर समस्याओं के लिए रोलबैक मैकेनिज्म लागू करें। सुनिश्चित करें कि डाउनटाइम और उपयोगकर्ताओं पर प्रभाव को कम करने के लिए वर्कफ़्लो को जल्दी से पिछले स्थिर वर्जन पर वापस लाया जा सकता है।

बुद्धिमान वर्कफ़्लो के विकास जीवनचक्र में परीक्षण और सत्यापन को शामिल करके, संगठन अपने AI-संचालित सिस्टम की विश्वसनीयता, सटीकता और रखरखाव क्षमता सुनिश्चित कर सकते हैं। नियमित परीक्षण और सत्यापन बग्स को पकड़ने, रिग्रेशन को रोकने और वर्कफ़्लो के व्यवहार और परिणामों में विश्वास बनाने में मदद करते हैं।

भाग 2: पैटर्न्स

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रॉम्प्ट इंजीनियरिंग

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

विचार श्रृंखला

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

सामग्री निर्माण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

संरचित इकाई निर्माण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

एलएलएम एजेंट मार्गदर्शन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

लाभ और विचारणीय बिंदु

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

मोड स्विच

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कब उपयोग करें

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

भूमिका निर्धारण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कब इसका उपयोग करें

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Prompt Object

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रॉम्प्ट टेम्पलेट

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

लाभ और विचारणीय बिंदु

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कब उपयोग करें:

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

संरचित इनपुट-आउटपुट

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

संरचित इनपुट-आउटपुट को स्केल करना

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

लाभ और विचारणीय बिंदु

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रॉम्प्ट चेनिंग

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

इसका उपयोग कब करें

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण: Olympia की ऑनबोर्डिंग

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रॉम्प्ट रीराइटर

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

रेस्पॉन्स फेन्सिंग

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

लाभ और विचारणीय बिंदु

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

त्रुटि प्रबंधन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

क्वेरी एनालाइज़र

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कार्यान्वयन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

वाक् भाग टैगिंग (POS) और नामित इकाई पहचान (NER)

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

इरादा वर्गीकरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कीवर्ड निष्कर्षण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

लाभ

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Query Rewriter

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

लाभ

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Ventriloquist

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कब इस्तेमाल करें

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

विभिन्न घटक

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रेडिकेट

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कब इस्तेमाल करें

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

एपीआई फसाड

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रमुख लाभ

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कब इसका उपयोग करें

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रमाणीकरण और अधिकृतिकरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

अनुरोध प्रबंधन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रतिक्रिया स्वरूपण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

त्रुटि प्रबंधन और सीमांत मामले

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

स्केलेबिलिटी और प्रदर्शन संबंधी विचार

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

अन्य डिज़ाइन पैटर्न से तुलना

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Result Interpreter

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

इसका उपयोग कब करें

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

वर्चुअल मशीन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कब इस्तेमाल करें

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

जादू के पीछे

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

विनिर्देशन और परीक्षण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

व्यवहार का विनिर्देशन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

परीक्षण मामले लिखना

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण: ट्रांसलेटर कंपोनेंट का परीक्षण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

HTTP इंटरैक्शन्स का पुनः प्रदर्शन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

मानव-इन-द-लूप (HITL)

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उच्च-स्तरीय पैटर्न

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

संकर बुद्धिमत्ता

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

अनुकूली प्रतिक्रिया

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

मानव-एआई भूमिका परिवर्तन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

एस्केलेशन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रमुख लाभ

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

वास्तविक-दुनिया का अनुप्रयोग: स्वास्थ्य सेवा

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रतिक्रिया चक्र

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

अनुप्रयोग और उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

मानवीय प्रतिक्रिया एकीकरण में उन्नत तकनीकें

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

निष्क्रिय सूचना प्रसारण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

संदर्भगत सूचना प्रदर्शन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

सक्रिय सूचनाएं

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

व्याख्यात्मक अंतर्दृष्टि

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

इंटरैक्टिव एक्सप्लोरेशन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

प्रमुख लाभ

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

अनुप्रयोग और उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

सहयोगात्मक निर्णय लेना (CDM)

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

सतत सीखने की प्रक्रिया

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

अनुप्रयोग और उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

नैतिक विचार

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

AI जोखिमों को कम करने में मानव-निर्देशित प्रणाली की भूमिका

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

तकनीकी प्रगति और भविष्य का दृष्टिकोण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

एचआईटीएल प्रणालियों की चुनौतियां और सीमाएं

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

बुद्धिमान त्रुटि प्रबंधन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

पारंपरिक त्रुटि प्रबंधन दृष्टिकोण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

संदर्भगत त्रुटि निदान

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

संदर्भगत त्रुटि निदान के लिए प्रॉम्प्ट इंजीनियरिंग

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

संदर्भात्मक त्रुटि निदान के लिए पुनर्प्राप्ति-संवर्धित उत्पादन

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

बुद्धिमान त्रुटि रिपोर्टिंग

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

पूर्वानुमानित त्रुटि निवारण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

स्मार्ट त्रुटि पुनर्प्राप्ति

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

व्यक्तिगत त्रुटि संचार

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

अनुकूली त्रुटि प्रबंधन कार्यप्रवाह

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

गुणवत्ता नियंत्रण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Eval

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

समस्या

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

समाधान

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

विचारणीय बिंदु

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

स्वर्ण मानकों को समझना

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

संदर्भ-मुक्त मूल्यांकन कैसे काम करते हैं

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

सुरक्षा सीमा

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

समस्या

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

समाधान

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

यह कैसे काम करता है

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

उदाहरण

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

विचारणीय बिंदु

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

सुरक्षा नियंत्रण और मूल्यांकन: एक ही सिक्के के दो पहलू

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

सुरक्षा नियंत्रण और संदर्भ-मुक्त मूल्यांकन की परस्पर विनिमेयता

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

द्विउद्देशीय सुरक्षा सीमाओं और मूल्यांकनों को लागू करना

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

शब्दकोश

शब्दकोश

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

A

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

B

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

C

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

D

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

E

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

F

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

G

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

H

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

I

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

J

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

K

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

L

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

M

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

N

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

O

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

P

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Q

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

R

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

स

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

T

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

U

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

V

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

W

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Z

यह सामग्री नमूना पुस्तक में उपलब्ध नहीं है। इस पुस्तक को Leanpub पर खरीदा जा सकता है http://leanpub.com/patterns-of-application-development-using-ai-hi पर।

Index