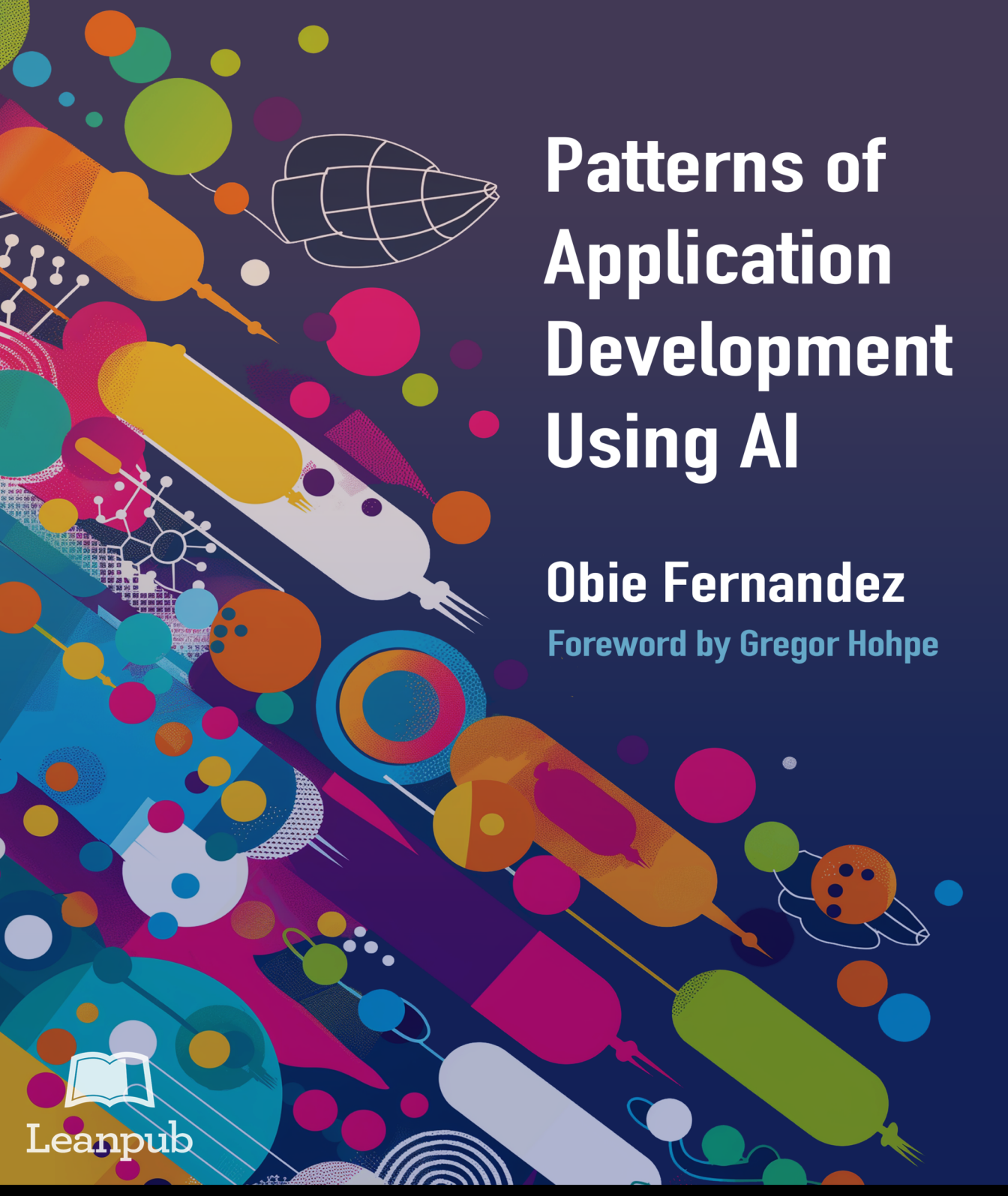


Polskie Wydanie

Wzorce Wytwarzania Aplikacji z Wykorzystaniem SI (Polskie Wydanie)

Obie Fernandez

Ta książka jest do kupienia na
http://leanpub.com/patterns-of-application-development-using-ai-pl

Wersja opublikowana 2025-01-23

Tweet’nij tę książkę

Proszę pomóż Obie Fernandez rozpowszechniać informację o tej książce na Twitter’ze!

Sugerowany tag do tej książki to #poaduai.

Sprawdź co inni ludzie mówią o tej książce i kliknij na ten link, żeby poszukać tego tagu na Twitter’ze:

#poaduai

Dla mojej nieustraszonej królowej, mojej muzy, mojego światła i miłości, Victorii

Pozostałe pozycje autora 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

Spis treści

Część 1: Fundamentalne Podejścia i Techniki 38

Przedmowa autorstwa Gregora Hohpe

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przedmowa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

O książce

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

O Przykładach Kodu

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Czego Nie Obejmuje Książka

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Dla Kogo Jest Ta Książka

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Budowanie Wspólnego Słownictwa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak się zaangażować

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Podziękowania

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Co z ilustracjami?

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

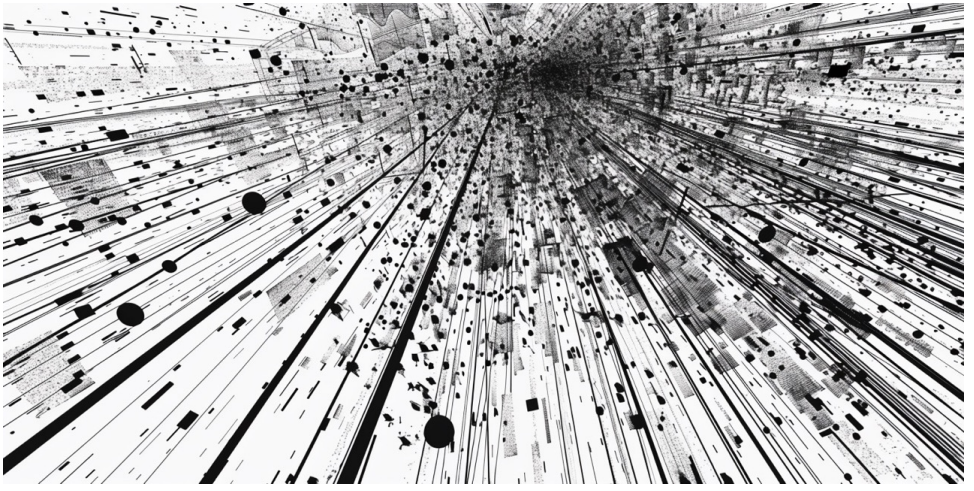
O Lean Publishing

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

O Autorze

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Wprowadzenie



Jeśli nie możesz się doczekać, aby zacząć integrować Duże Modele Językowe AI (LLM) w swoich projektach programistycznych, śmiało możesz przejść od razu do wzorców i przykładów kodu przedstawionych w kolejnych rozdziałach. Jednak aby w pełni docenić moc i potencjał tych wzorców, warto poświęcić chwilę na zrozumienie szerszego kontekstu i spójnego podejścia, które reprezentują.

Wzorce nie są jedynie zbiorem odizolowanych technik, lecz raczej jednolitym frameworkiem do integracji AI w twoich aplikacjach. Używam Ruby on Rails, ale te wzorce powinny działać praktycznie w każdym innym środowisku programistycznym. Odnoszą się one do szerokiego zakresu zagadnień, od zarządzania danymi i optymalizacji wydajności po doświadczenie użytkownika i bezpieczeństwo, zapewniając kompleksowy zestaw narzędzi do wzbogacania tradycyjnych praktyk programistycznych o możliwości AI.

Każda kategoria wzorców zajmuje się konkretnym wyzwaniem lub możliwością, które pojawiają się podczas włączania komponentów AI do twojej aplikacji. Zrozumienie relacji i synergii między tymi wzorcami pozwoli ci podejmować świadome decyzje dotyczące tego, gdzie i jak najefektywniej zastosować AI.

Wzorce nigdy nie są nakazowymi rozwiązaniami i nie należy ich tak traktować. Są pomyślane jako adaptowalne elementy składowe, które powinny być dostosowane do unikalnych wymagań i ograniczeń twojej własnej aplikacji. Skuteczne zastosowanie tych wzorców (jak każdych innych w dziedzinie oprogramowania) opiera się na głębokim zrozumieniu dziedziny problemu, potrzeb użytkowników i ogólnej architektury technicznej twojego projektu.

Przemyślenia na temat architektury oprogramowania

Zacząłem programować w latach 80. i byłem zaangażowany w środowisko hakerskie, i nigdy nie straciłem swojego hakerskiego sposobu myślenia, nawet po zostaniu profesjonalnym programistą. Od początku zawsze podchodziłem ze zdrowym sceptycyzmem do tego, jaką wartość faktycznie wnosili architekci oprogramowania w swoich wieżach z kości słoniowej.

Jednym z powodów, dla których osobiście jestem tak podekscytowany zmianami wprowadzonymi przez tę potężną nową falę technologii AI, jest jej wpływ na to, co uznajemy za decyzje dotyczące *architektury oprogramowania*. Kwestionuje ona tradycyjne pojęcia tego, co stanowi "prawidłowy" sposób projektowania i implementacji naszych projektów programistycznych. Kwestionuje również, czy architekturę można nadal postrzegać głównie jako *te części systemu, które trudno zmienić*, ponieważ usprawnienia AI sprawiają, że łatwiej niż kiedykolwiek jest zmienić dowolną część projektu w dowolnym momencie.

Być może wkraczamy w szczytowy okres "postmodernistycznego" podejścia do

inżynierii oprogramowania. W tym kontekście postmodernizm odnosi się do fundamentalnej zmiany w stosunku do tradycyjnych paradygmatów, gdzie programiści byli odpowiedzialni za pisanie i utrzymywanie każdej linii kodu. Zamiast tego, podejście to przyjmuje ideę delegowania zadań, takich jak manipulacja danymi, złożone algorytmy, a nawet całe fragmenty logiki aplikacji, do bibliotek zewnętrznych i zewnętrznych API. Ta postmodernistyczna zmiana stanowi znaczące odejście od konwencjonalnej mądrości budowania aplikacji od podstaw i zmusza programistów do przemyślenia swojej roli w procesie rozwoju oprogramowania.

Zawsze wierzyłem, że dobrzy programiści piszą tylko kod, który jest absolutnie niezbędny do napisania, bazując na naukach Larry'ego Walla i innych świetlanych umysłów hakerskich podobnych do niego. Minimalizując ilość napisanego kodu, możemy działać szybciej, zmniejszyć przestrzeń na błędy, uprościć konserwację i poprawić ogólną niezawodność aplikacji. Mniej kodu pozwala nam skupić się na podstawowej logice biznesowej i doświadczeniu użytkownika, delegując jednocześnie pozostałą pracę do innych usług.

Teraz, gdy systemy oparte na sztucznej inteligencji mogą obsługiwać zadania, które wcześniej były domeną wyłącznie kodu pisanego przez człowieka, powinniśmy być w stanie być jeszcze bardziej produktywni i zwinni, skupiając się bardziej niż kiedykolwiek na tworzeniu wartości biznesowej i doświadczeniu użytkownika.

Oczywiście delegowanie ogromnych części projektu systemom AI wiąże się z pewnymi kompromisami, takimi jak potencjalna utrata kontroli i potrzeba solidnych mechanizmów monitorowania i informacji zwrotnej. Dlatego wymaga to nowego zestawu umiejętności i wiedzy, w tym przynajmniej podstawowego zrozumienia tego, jak działa AI.

Czym jest Duży Model Językowy?

Duże Modele Językowe (LLM) to rodzaj modelu sztucznej inteligencji, który zyskał znaczną uwagę w ostatnich latach, szczególnie od czasu wprowadzenia GPT-3

przez OpenAI w 2020 roku. LLM są zaprojektowane do przetwarzania, rozumienia i generowania języka ludzkiego z niezwykłą dokładnością i płynnością. W tej sekcji przyjrzymy się pokrótce temu, jak działają LLM i dlaczego są tak dobrze przystosowane do budowania inteligentnych komponentów systemowych.

U podstaw LLM leżą algorytmy uczenia głębokiego, a konkretnie sieci neuronowe. Sieci te składają się z połączonych węzłów, czyli neuronów, które przetwarzają i przekazują informacje. Architekturą preferowaną dla LLM jest często model Transformer, który okazał się wysoce skuteczny w obsłudze danych sekwencyjnych, takich jak tekst.

Modele Transformer opierają się na mechanizmie uwagi i są głównie wykorzystywane do zadań związanych z danymi sekwencyjnymi, takich jak przetwarzanie języka naturalnego. Transformery przetwarzają dane wejściowe wszystkie naraz, zamiast sekwencyjnie, co pozwala im skuteczniej uchwycić zależności długodystansowe. Posiadają warstwy mechanizmów uwagi, które pomagają modelowi skupić się na różnych częściach danych wejściowych w celu zrozumienia kontekstu i relacji.

Proces trenowania LLM-ów polega na wystawieniu modelu na ogromne ilości danych tekstowych, takich jak książki, artykuły, strony internetowe i repozytoria kodu. Podczas treningu model uczy się rozpoznawać wzorce, relacje i struktury w tekście. Wychwytuje statystyczne właściwości języka, takie jak reguły gramatyczne, powiązania między słowami i znaczenia kontekstowe.

Jedną z kluczowych technik wykorzystywanych w trenowaniu LLM-ów jest uczenie nienadzorowane. Oznacza to, że model uczy się z danych bez wyraźnego oznakowania czy wskazówek. Samodzielnie odkrywa wzorce i reprezentacje, analizując współwystępowanie słów i fraz w danych treningowych. Pozwala to LLM-om rozwinąć głębokie zrozumienie języka i jego zawiłości.

Innym ważnym aspektem LLM-ów jest ich zdolność do obsługi *kontekstu*. Podczas przetwarzania tekstu, LLM-y biorą pod uwagę nie tylko pojedyncze słowa, ale również otaczający kontekst. Uwzględniają poprzednie słowa, zdania, a nawet akapity, aby zrozumieć znaczenie i intencję tekstu. To kontekstowe zrozumienie umożliwia LLM-

-om generowanie spójnych i trafnych odpowiedzi. Jednym z głównych sposobów oceny możliwości danego modelu LLM jest rozważenie wielkości kontekstu, jaki może uwzględnić przy generowaniu odpowiedzi.

Po wytrenowaniu, LLM-y mogą być wykorzystywane do szerokiej gamy zadań związanych z językiem. Potrafią generować tekst przypominający ludzki, odpowiadać na pytania, streszczać dokumenty, tłumaczyć języki, a nawet pisać kod. Wszechstronność LLM-ów sprawia, że są cenne przy budowaniu inteligentnych komponentów systemowych, które mogą wchodzić w interakcje z użytkownikami, przetwarzać i analizować dane tekstowe oraz generować znaczące wyniki.

Włączając LLM-y do architektury aplikacji, można tworzyć komponenty AI, które rozumieją i przetwarzają dane wejściowe od użytkownika, generują dynamiczną treść oraz dostarczają inteligentne rekomendacje lub działania. Jednak praca z LLM-ami wymaga starannego rozważenia wymagań dotyczących zasobów i kompromisów wydajnościowych. LLM-y są obliczeniowo intensywne i mogą wymagać znaczącej mocy obliczeniowej oraz pamięci (innymi słowy, pieniędzy) do działania. Większość z nas będzie musiała ocenić implikacje kosztowe integracji LLM-ów z naszymi aplikacjami i działać odpowiednio.

Zrozumienie Wnioskowania

Wnioskowanie odnosi się do procesu, w którym model generuje przewidywania lub wyniki na podstawie nowych, niewidzianych wcześniej danych. Jest to faza, w której wytrenowany model jest wykorzystywany do podejmowania decyzji lub generowania tekstu, obrazów lub innych treści w odpowiedzi na dane wejściowe użytkownika.

Podczas fazy treningu model AI uczy się na podstawie dużego zbioru danych, dostosowując swoje parametry, aby zminimalizować błędy w swoich przewidywaniach. Po wytrenowaniu model może zastosować to, czego się nauczył, do nowych danych. Wnioskowanie to sposób, w jaki model wykorzystuje wyuczone wzorce i wiedzę do generowania wyników.

W przypadku Dużych Modeli Językowych, wnioskowanie polega na przyjęciu promptu lub tekstu wejściowego i wyprodukowaniu spójnej i kontekstowo odpowiedniej odpowiedzi w postaci strumienia *tokenów* (o których wkrótce porozmawiamy). Może to być odpowiadanie na pytania, uzupełnianie zdań, generowanie historii czy tłumaczenie tekstu, wśród wielu innych zadań.

W przeciwieństwie do sposobu, w jaki ty i ja myślimy, "myślenie" modelu AI poprzez wnioskowanie odbywa się w jednej bezstanowej operacji. To znaczy, jego myślenie jest ograniczone do procesu generowania. Dosłownie musi myśleć na głos, tak jakbym zadał ci pytanie i akceptował odpowiedź tylko w stylu "strumienia świadomości".

Duże Modele Językowe Występują w Wielu Rozmiarach i Odmianach

Podczas gdy praktycznie wszystkie popularne duże modele językowe (LLM) są oparte na tej samej podstawowej architekturze transformera i trenowane na ogromnych zbiorach tekstowych, występują w różnych rozmiarach i są dostrajane do różnych celów. Rozmiar LLM, mierzony liczbą parametrów w jego sieci neuronowej, ma duży wpływ na jego możliwości. Większe modele z większą liczbą parametrów, jak GPT-4, który podobno może pochwalić się 1-2 bilionami parametrów, są generalnie bardziej kompetentne i zdolne niż mniejsze modele. Jednakże większe modele wymagają też znacznie większej mocy obliczeniowej do działania, co przekłada się na wyższe koszty przy korzystaniu z nich poprzez wywołania API.

Aby uczynić LLM bardziej praktycznymi i dostosowanymi do konkretnych przypadków użycia, modele bazowe są często dostrajane na bardziej ukierunkowanych zbiorach danych. Na przykład, LLM może być trenowany na dużym korpusie dialogów, aby wyspecjalizować go w konwersacyjnej SI. Inne są trenowane na kodzie, aby wyposażyć je w wiedzę programistyczną. Istnieją nawet modele, które są specjalnie trenowane do interakcji z użytkownikami w stylu odgrywania ról!

Modele oparte na wyszukiwaniu a modele generatywne

W świecie dużych modeli językowych (LLM) istnieją dwa główne podejścia do generowania odpowiedzi: modele oparte na wyszukiwaniu i modele generatywne. Każde podejście ma swoje mocne i słabe strony, a zrozumienie różnic między nimi może pomóc w wyborze odpowiedniego modelu do konkretnego przypadku użycia.

Modele oparte na wyszukiwaniu

Modele oparte na wyszukiwaniu, znane również jako modele wyszukiwania informacji, generują odpowiedzi poprzez przeszukiwanie dużej bazy danych istniejących tekstów i wybieranie najbardziej odpowiednich fragmentów na podstawie zapytania. Modele te nie generują nowego tekstu od podstaw, lecz raczej łączą fragmenty z bazy danych w spójną odpowiedź.

Jedną z głównych zalet modeli opartych na wyszukiwaniu jest ich zdolność do dostarczania faktycznie dokładnych i aktualnych informacji. Ponieważ opierają się na bazie wyselekcjonowanych tekstów, mogą wydobywać istotne informacje z wiarygodnych źródeł i przedstawiać je użytkownikowi. Sprawia to, że doskonale nadają się do zastosowań wymagających precyzyjnych, faktycznych odpowiedzi, takich jak systemy odpowiadania na pytania czy bazy wiedzy.

Jednak modele oparte na wyszukiwaniu mają pewne ograniczenia. Są tylko tak dobre jak baza danych, którą przeszukują, więc jakość i zakres bazy danych bezpośrednio wpływają na wydajność modelu. Ponadto modele te mogą mieć trudności z generowaniem spójnych i naturalnie brzmiących odpowiedzi, ponieważ są ograniczone do tekstu dostępnego w bazie danych.

W tej książce nie omawiamy wykorzystania czystych modeli opartych na wyszukiwaniu.

Modele generatywne

Modele generatywne natomiast tworzą nowy tekst od podstaw w oparciu o wzorce i zależności, których nauczyły się podczas treningu. Modele te wykorzystują swoje zrozumienie języka do generowania nowych odpowiedzi dostosowanych do wprowadzonego polecenia.

Główną zaletą modeli generatywnych jest ich zdolność do tworzenia kreatywnego, spójnego i kontekstowo odpowiedniego tekstu. Mogą prowadzić otwarte rozmowy, generować historie, a nawet pisać kod. Sprawia to, że są idealne do zastosowań wymagających bardziej otwartych i dynamicznych interakcji, takich jak chatboty, tworzenie treści czy asystenci pisania kreatywnego.

Jednak modele generatywne mogą czasami produkować niespójne lub faktycznie niepoprawne informacje, ponieważ opierają się na wzorcach nauczonych podczas treningu, a nie na wyselekcjonowanej bazie faktów. Mogą być również bardziej podatne na uprzedzenia i halucynacje, generując tekst, który jest wiarygodny, ale niekoniecznie prawdziwy.

Przykładami generatywnych LLM-ów są modele z serii GPT (GPT-3, GPT-4) od OpenAI oraz Claude od Anthropic.

Modele hybrydowe

Kilka dostępnych komercyjnie LLM-ów łączy zarówno podejście oparte na wyszukiwaniu, jak i generatywne w modelu hybrydowym. Modele te wykorzystują techniki wyszukiwania do znalezienia istotnych informacji z bazy danych, a następnie używają technik generatywnych do syntezy tych informacji w spójną odpowiedź.

Modele hybrydowe mają na celu połączenie dokładności faktograficznej modeli opartych na wyszukiwaniu z możliwościami generowania języka naturalnego charakterystycznymi dla modeli generatywnych. Mogą dostarczać bardziej

wiarygodnych i aktualnych informacji, zachowując jednocześnie zdolność do prowadzenia otwartych konwersacji.

Wybierając między modelami opartymi na wyszukiwaniu a modelami generatywnymi, należy wziąć pod uwagę konkretne wymagania aplikacji. Jeśli głównym celem jest dostarczanie dokładnych, faktycznych informacji, model oparty na wyszukiwaniu może być najlepszym wyborem. Jeśli aplikacja wymaga bardziej otwartych i kreatywnych interakcji, model generatywny może być bardziej odpowiedni. Modele hybrydowe oferują równowagę między tymi dwoma podejściami i mogą być dobrym wyborem dla aplikacji wymagających zarówno dokładności faktograficznej, jak i generowania języka naturalnego.

Ostatecznie wybór między modelami opartymi na wyszukiwaniu a modelami generatywnymi zależy od konkretnego przypadku użycia i kompromisów między dokładnością, kreatywnością a elastycznością. Rozumiejąc mocne strony i ograniczenia każdego podejścia, możesz podejmować świadome decyzje.

Dostrajanie instrukcji

Dostrajanie instrukcji to podzbiór treningu konwersacyjnego, który wykorzystuje napisane przez ludzi polecenia i odpowiedzi, aby ulepszyć zdolność LLM-ów do rozumienia i wykonywania instrukcji. Oto kilka prostych przykładów ilustrujących, jak dostrajanie instrukcji na podstawie napisanych przez ludzi poleceń i odpowiedzi poprawia zdolność LLM-ów do rozumienia i wykonywania instrukcji:

Przykład 1: Obsługa klienta

Instrukcja: "Udziel odpowiedzi klientowi proszącemu o zwrot pieniędzy."

Polecenie napisane przez człowieka: "Chciałbym poprosić o zwrot pieniędzy za mój ostatni zakup. Produkt nie spełnił moich oczekiwań."

Odpowiedź napisana przez człowieka: "Przykro mi słyszeć, że produkt nie spełnił Pana/Pani oczekiwań. Aby przetworzyć zwrot, prosimy o podanie numeru zamówienia oraz powodu zwrotu. Przeanalizujemy Pana/Pani prośbę i odpowiemy w ciągu 2-3 dni roboczych."

Przykład 2: Pomoc edukacyjna

Instrukcja: "Wyjaśnij pojęcie fotosyntezy 10-latkowi."

Human-Written Prompt: "Czy możesz wyjaśnić, jak rośliny wytwarzają swoje pożywienie?"

Human-Written Response: "Oczywiście! Rośliny wytwarzają swoje pożywienie w procesie zwanym fotosyntezą. Wykorzystują światło słoneczne, wodę i dwutlenek węgla do wytworzenia pokarmu i tlenu. Liście rośliny mają specjalną część zwaną chlorofilem, która pomaga im w wychwytywaniu światła słonecznego."

Example 3: Personal Assistant

> Instruction: “Schedule a meeting for next Tuesday at 3 PM.”
>
> Human-Written Prompt: “Czy możesz zaplanować spotkanie na wtorek na godzinę 15:00?”
>
> Human-Written Response: “Oczywiście, zaplanowałem spotkanie na następny wtorek na godzinę 15:00. Czy potrzebujesz czegoś jeszcze?”

Rezultatem jest zróżnicowany ekosystem LLM-ów o różnych rozmiarach i specjalizacjach. Mniejsze modele w zakresie 1-7 miliardów parametrów zapewniają dobre ogólne możliwości językowe, będąc jednocześnie bardziej wydajnymi w działaniu.

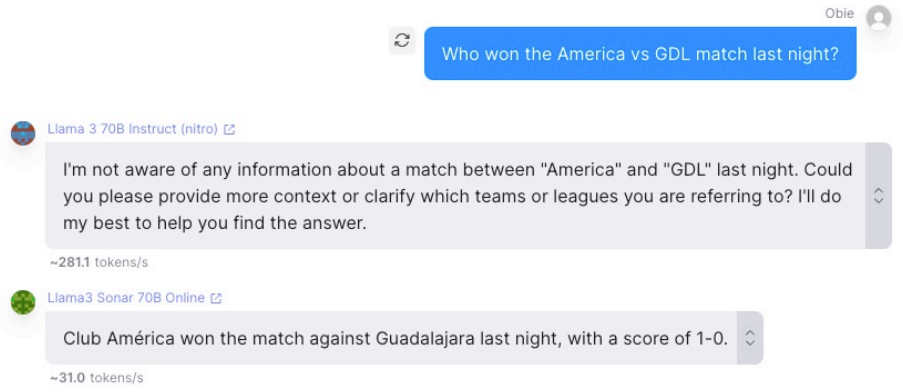
- Mistral 7B
- Llama 3 8B
- Gemma 7B

Modele średniej wielkości, około 30-70 miliardów parametrów, oferują silniejsze zdolności rozumowania i wykonywania instrukcji.

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

Przy wyborze LLM do włączenia do aplikacji musisz zrównoważyć możliwości modelu z praktycznymi czynnikami, takimi jak koszt, opóźnienie, długość kontekstu i filtrowanie treści. Mniejsze modele dostosowane do instrukcji są często najlepszym wyborem dla prostszych zadań językowych, podczas gdy największe modele mogą być potrzebne do złożonego rozumowania lub analizy. Dane treningowe modelu są również ważnym czynnikiem, ponieważ określają datę graniczną wiedzy modelu.

Niektóre modele, jak te od Perplexity, są połączone ze źródłami informacji w czasie rzeczywistym, więc efektywnie nie mają daty granicznej. Kiedy zadajesz im pytania, są w stanie samodzielnie zdecydować o przeprowadzeniu wyszukiwania w sieci i pobraniu dowolnych stron internetowych w celu wygenerowania odpowiedzi.



Rycina 1. Llama3 z dostępem do sieci i bez

Ostatecznie nie istnieje uniwersalny model LLM. Zrozumienie różnic w wielkości modelu, architekturze i treningu jest kluczowe przy wyborze odpowiedniego modelu do danego przypadku użycia. Eksperymentowanie z różnymi modelami jest jedynym praktycznym sposobem na odkrycie, które z nich zapewniają najlepszą wydajność dla danego zadania.

Tokenizacja: Dzielenie Tekstu na Części

Zanim model językowy może przetworzyć tekst, musi on zostać podzielony na mniejsze jednostki zwane *tokenami*. Tokeny mogą być pojedynczymi słowami, częściami słów, a nawet pojedynczymi znakami. Proces dzielenia tekstu na tokeny nazywany jest

tokenizacją i jest kluczowym krokiem w przygotowywaniu danych dla modelu językowego.

Rycina 2. To zdanie zawiera 27 tokenów

Różne modele LLM używają różnych strategii tokenizacji, co może mieć znaczący wpływ na wydajność i możliwości modelu. Niektóre popularne tokenizery używane przez modele LLM to:

- **GPT (Byte Pair Encoding)**: Tokenizery GPT używają techniki zwanej kodowaniem par bajtów (BPE) do dzielenia tekstu na jednostki podwyrazowe. BPE iteracyjnie łączy najczęściej występujące pary bajtów w korpusie tekstu, tworząc słownik tokenów podwyrazowych. Pozwala to tokenizerowi obsługiwać rzadkie i nowe słowa poprzez rozbijanie ich na częściej występujące części podwyrazowe. Tokenizery GPT są używane przez modele takie jak GPT-3 i GPT-4.

- **Llama (SentencePiece)**: Tokenizery Llama używają biblioteki SentencePiece, która jest nienadzorowanym tokenizerem i detokenizatorem tekstu. SentencePiece traktuje tekst wejściowy jako sekwencję znaków Unicode i uczy się słownika podwyrazowego na podstawie korpusu treningowego. Może obsługiwać dowolny język, który można zakodować w Unicode, co sprawia, że świetnie nadaje się do modeli wielojęzycznych. Tokenizery Llama są używane przez modele takie jak Meta's Llama i Alpaca.

- **SentencePiece (Unigram)**: Tokenizery SentencePiece mogą również wykorzystywać inny algorytm zwany Unigram, który opiera się na technice regularyzacji jednostek podwyrazowych. Tokenizacja Unigram określa

optymalny słownik jednostek podwyrazowych w oparciu o jednogramowy model językowy, który przypisuje prawdopodobieństwa pojedynczym jednostkom podwyrazowym. To podejście może generować jednostki podwyrazowe o większym znaczeniu semantycznym w porównaniu z BPE. SentencePiece z Unigram jest wykorzystywany przez modele takie jak Google T5 i BERT.

- **Google Gemini (Tokenizacja Multimodalna)**: Google Gemini wykorzystuje schemat tokenizacji zaprojektowany do obsługi różnych typów danych, w tym tekstu, obrazów, dźwięku, wideo i kodu. Ta zdolność multimodalna pozwala Gemini przetwarzać i integrować różne formy informacji. Co istotne, Google Gemini 1.5 Pro posiada okno kontekstowe, które może obsłużyć miliony tokenów, znacznie więcej niż poprzednie modele. To rozległe okno kontekstowe umożliwia modelowi przetwarzanie większego kontekstu, potencjalnie prowadząc do dokładniejszych odpowiedzi. Należy jednak zauważyć, że schemat tokenizacji Gemini jest znacznie bliższy jednemu tokenowi na znak niż w innych modelach. Oznacza to, że rzeczywisty koszt korzystania z modeli Gemini może być znacznie wyższy niż oczekiwano, jeśli jest się przyzwyczajonym do korzystania z modeli takich jak GPT, ponieważ wycena Google opiera się na znakach, a nie tokenach.

Wybór tokenizera wpływa na kilka aspektów LLM, w tym:

- **Rozmiar słownika**: Tokenizer określa rozmiar słownika modelu, czyli zbiór unikalnych tokenów, które rozpoznaje. Większy, bardziej szczegółowy słownik może pomóc modelowi obsłużyć szerszy zakres słów i fraz, a nawet stać się multimodalnym (zdolnym do rozumienia i generowania więcej niż tylko tekstu), ale zwiększa też wymagania pamięciowe modelu i złożoność obliczeniową.
- **Obsługa rzadkich i nieznanych słów**: Tokenizery wykorzystujące jednostki podwyrazowe, takie jak BPE i SentencePiece, mogą rozbijać rzadkie i nieznane słowa na częściej występujące części podwyrazowe. Pozwala to modelowi na dokonywanie świadomych przypuszczeń co do znaczenia słów, których wcześniej nie widział, na podstawie zawartych w nich części podwyrazowych.

- **Wsparcie wielojęzyczne**: Tokenizery takie jak SentencePiece, które mogą obsługiwać dowolny język kodowany w Unicode, są dobrze przystosowane do modeli wielojęzycznych, które muszą przetwarzać tekst w wielu językach.

Przy wyborze LLM do konkretnego zastosowania ważne jest, aby wziąć pod uwagę używany przez niego tokenizer oraz to, jak dobrze odpowiada on specyficznym potrzebom przetwarzania języka w danym zadaniu. Tokenizer może mieć znaczący wpływ na zdolność modelu do obsługi terminologii specyficznej dla danej dziedziny, rzadkich słów i tekstu wielojęzycznego.

Rozmiar Kontekstu: Ile Informacji Model Językowy Może Wykorzystać Podczas Wnioskowania?

W dyskusji o modelach językowych, rozmiar kontekstu odnosi się do ilości tekstu, który model może uwzględnić podczas przetwarzania lub generowania odpowiedzi. Jest to w zasadzie miara tego, ile informacji model może "zapamiętać" i wykorzystać do tworzenia swoich wyników (wyrażona w tokenach). Rozmiar kontekstu modelu językowego może mieć znaczący wpływ na jego możliwości i rodzaje zadań, które może skutecznie wykonywać.

Czym Jest Rozmiar Kontekstu?

W ujęciu technicznym, rozmiar kontekstu jest określony przez liczbę tokenów (słów lub części słów), które model językowy może przetworzyć w pojedynczej sekwencji wejściowej. Jest to często określane jako "zakres uwagi" lub "okno kontekstowe" modelu. Im większy rozmiar kontekstu, tym więcej tekstu model może jednocześnie uwzględnić podczas generowania odpowiedzi lub wykonywania zadania.

Różne modele językowe mają różne rozmiary kontekstu, od kilkuset tokenów do milionów tokenów. Dla porównania, typowy akapit tekstu może zawierać około 100-150 tokenów, podczas gdy cała książka może zawierać dziesiątki lub setki tysięcy tokenów.

> Istnieją nawet prace nad wydajnymi metodami skalowania Transformerowych Dużych Modeli Językowych (LLM) do nieskończenie długich danych wejściowych z ograniczoną pamięcią i mocą obliczeniową.

Dlaczego Rozmiar Kontekstu Jest Ważny?

Rozmiar kontekstu modelu językowego ma znaczący wpływ na jego zdolność do rozumienia i generowania spójnego, kontekstowo odpowiedniego tekstu. Oto kilka kluczowych powodów, dla których rozmiar kontekstu jest istotny:

1. **Rozumienie treści długoformowych**: Modele z większym rozmiarem kontekstu mogą lepiej zrozumieć i analizować dłuższe teksty, takie jak artykuły, raporty, czy nawet całe książki. Jest to kluczowe dla zadań takich jak podsumowywanie dokumentów, odpowiadanie na pytania i analiza treści.

2. **Zachowanie spójności**: Większe okno kontekstowe pozwala modelowi zachować spójność i konsekwencję w dłuższych fragmentach wyjściowych. Jest to ważne dla zadań takich jak generowanie opowiadań, systemy dialogowe i tworzenie treści, gdzie utrzymanie spójnej narracji lub tematu jest niezbędne. Jest to również absolutnie kluczowe przy wykorzystywaniu LLM do generowania lub przekształcania danych strukturalnych.
3. **Przechwytywanie zależności długodystansowych**: Niektóre zadania językowe wymagają zrozumienia relacji między słowami lub frazami, które są od siebie znacznie oddalone w tekście. Modele o większym rozmiarze kontekstu są lepiej przygotowane do uchwycenia tych zależności długodystansowych, co może być istotne w zadaniach takich jak analiza sentymentu, tłumaczenie i rozumienie języka.

4. **Obsługa złożonych instrukcji**: W aplikacjach, gdzie modele językowe są wykorzystywane do wykonywania złożonych, wieloetapowych instrukcji, większy rozmiar kontekstu pozwala modelowi uwzględnić cały zestaw instrukcji podczas generowania odpowiedzi, zamiast tylko kilku ostatnich słów.

Przykłady modeli językowych o różnych rozmiarach kontekstu

Oto kilka przykładów modeli językowych o różnych rozmiarach kontekstu:

- OpenAI GPT-3.5 Turbo: 4 095 tokenów
- Mistral 7B Instruct: 32 768 tokenów
- Anthropic Claude v1: 100 000 tokenów
- OpenAI GPT-4 Turbo: 128 000 tokenów
- Anthropic Claude v2: 200 000 tokenów
- Google Gemini Pro 1.5: 2,8M tokenów

Jak widać, istnieje szeroki zakres rozmiarów kontekstu wśród tych modeli, od około 4 000 tokenów dla modelu OpenAI GPT-3.5 Turbo do 200 000 tokenów dla modelu Anthropic Claude v2. Niektóre modele, jak Google PaLM 2 i OpenAI GPT-4, oferują różne warianty o większych rozmiarach kontekstu (np. wersje "32k"), które mogą obsłużyć jeszcze dłuższe sekwencje wejściowe. A w chwili obecnej (kwiecień 2024) Google Gemini Pro może pochwalić się prawie 3 milionami tokenów!

Warto zauważyć, że rozmiar kontekstu może się różnić w zależności od konkretnej implementacji i wersji danego modelu. Na przykład, oryginalny model OpenAI GPT-4 ma rozmiar kontekstu 8 191 tokenów, podczas gdy późniejsze warianty GPT-4, takie jak Turbo i 4o, mają znacznie większy rozmiar kontekstu wynoszący 128 000 tokenów.

> Sam Altman porównał obecne ograniczenia kontekstu do kilobajtów pamięci operacyjnej, z którą programiści komputerów osobistych musieli się zmagać w latach 80., i stwierdził, że w niedalekiej przyszłości będziemy mogli zmieścić "wszystkie nasze dane osobiste" w kontekście dużego modelu językowego.

Wybór odpowiedniego rozmiaru kontekstu

Przy wyborze modelu językowego do konkretnego zastosowania ważne jest uwzględnienie wymagań dotyczących rozmiaru kontekstu dla danego zadania. W przypadku zadań związanych z krótkimi, odizolowanymi fragmentami tekstu, takimi jak analiza sentymentu czy proste odpowiadanie na pytania, mniejszy rozmiar kontekstu może być wystarczający. Jednak w przypadku zadań wymagających zrozumienia i generowania dłuższych, bardziej złożonych tekstów, prawdopodobnie niezbędny będzie większy rozmiar kontekstu.

Warto zauważyć, że większe rozmiary kontekstu często wiążą się ze zwiększonymi kosztami obliczeniowymi i dłuższym czasem przetwarzania, ponieważ model musi uwzględnić więcej informacji podczas generowania odpowiedzi. W związku z tym przy wyborze modelu językowego dla swojej aplikacji należy znaleźć równowagę między rozmiarem kontekstu a wydajnością.

> Dlaczego po prostu nie wybrać modelu z największym rozmiarem kontekstu i nie wypełnić go maksymalną ilością informacji? Cóż, oprócz czynników wydajnościowych, głównym problemem jest koszt. W marcu 2024 roku pojedynczy cykl zapytanie-odpowiedź przy użyciu Google Gemini Pro 1.5 z pełnym kontekstem kosztuje prawie 8 dolarów (USD). Jeśli masz przypadek użycia, który uzasadnia ten wydatek, więcej mocy dla ciebie! Ale dla większości zastosowań jest to po prostu

zbyt drogie o rzędy wielkości.

Szukanie igieł w stogu siana

Koncepcja szukania igły w stogu siana od dawna była metaforą wyzwań związanych z wyszukiwaniem w dużych zbiorach danych. W dziedzinie LLM nieco modyfikujemy tę analogię. Wyobraź sobie, że nie szukamy pojedynczego faktu ukrytego w ogromnym tekście (jak pełna antologia esejów Paula Grahama), ale wielu faktów rozproszonych po całości. Ten scenariusz bardziej przypomina szukanie kilku igieł na rozległym polu, a nie tylko w jednym stogu siana. A oto haczyk: nie tylko musimy zlokalizować te igły, ale także połączyć je w spójną całość.

Gdy LLM mają za zadanie wyszukać i rozumować na podstawie wielu faktów zawartych w długich kontekstach, stają przed podwójnym wyzwaniem. Po pierwsze, istnieje prosty problem dokładności wyszukiwania - naturalnie spada ona wraz ze wzrostem liczby faktów. Jest to zrozumiałe; w końcu śledzenie wielu szczegółów w rozległym tekście obciąża nawet najbardziej zaawansowane modele.

Po drugie, i być może bardziej krytycznie, pojawia się wyzwanie rozumowania z wykorzystaniem tych faktów. Czym innym jest wyodrębnianie faktów, a czym innym synteza ich w spójną narrację lub odpowiedź. To właśnie tutaj pojawia się prawdziwy test. Wydajność modeli LLM w zadaniach rozumowania ma tendencję do pogorszenia się w większym stopniu niż w prostych zadaniach wyszukiwania. Ta degradacja nie dotyczy tylko ilości; chodzi o skomplikowany taniec kontekstu, trafności i wnioskowania.

Dlaczego tak się dzieje? Rozważmy dynamikę pamięci i uwagi w ludzkim poznaniu, która w pewnym stopniu odzwierciedlona jest w modelach LLM. Podczas przetwarzania dużych ilości informacji, modele LLM, podobnie jak ludzie, mogą tracić z pola widzenia wcześniejsze szczegóły w miarę przyswajania nowych. Jest to szczególnie

widoczne w modelach, które nie zostały wyraźnie zaprojektowane do automatycznego priorytetyzowania lub powracania do wcześniejszych fragmentów tekstu.

Co więcej, zdolność modelu LLM do połączenia tych pozyskanych faktów w spójną odpowiedź przypomina proces budowania narracji. Wymaga to nie tylko wydobycia informacji, ale także głębokiego zrozumienia i umiejscowienia kontekstowego, co pozostaje poważnym wyzwaniem dla obecnej sztucznej inteligencji.

Jakie ma to więc znaczenie dla nas jako programistów i integratorów tych technologii? Musimy być szczególnie świadomi tych ograniczeń podczas projektowania systemów, które polegają na modelach LLM do obsługi złożonych zadań długoformowych. Zrozumienie, że wydajność może się pogorszyć w pewnych warunkach, pomaga nam ustalić realistyczne oczekiwania i zaprojektować lepsze mechanizmy awaryjne lub strategie uzupełniające.

Modalności: Poza Tekstem

Podczas gdy większość dzisiejszych modeli językowych koncentruje się na przetwarzaniu i generowaniu tekstu, obserwujemy rosnący trend w kierunku modeli wielomodalnych, które mogą natywnie przyjmować i generować wiele typów danych, takich jak obrazy, dźwięk i wideo. Te modele wielomodalne otwierają nowe możliwości dla aplikacji opartych na sztucznej inteligencji, które mogą rozumieć i generować treści w różnych modalnościach.

Czym są modalności?

W kontekście modeli językowych, modalności odnoszą się do różnych typów danych, które model może przetwarzać i generować. Najpopularniejszą modalnością jest tekst, który obejmuje język pisany w różnych formach, takich jak książki, artykuły, strony internetowe i posty w mediach społecznościowych. Istnieje jednak kilka innych modalności, które są coraz częściej włączane do modeli językowych:

- **Obrazy**: Dane wizualne, takie jak fotografie, ilustracje i diagramy.
- **Audio**: Dane dźwiękowe, takie jak mowa, muzyka i dźwięki otoczenia.
- **Wideo**: Ruchome dane wizualne, często z towarzyszącym dźwiękiem, takie jak klipy wideo i filmy.

Każda modalność przedstawia unikalne wyzwania i możliwości dla modeli językowych. Na przykład, obrazy wymagają od modelu zrozumienia koncepcji i relacji wizualnych, podczas gdy dźwięk wymaga przetwarzania i generowania mowy oraz innych dźwięków.

Wielomodalne Modele Językowe

Wielomodalne modele językowe są zaprojektowane do obsługi wielu modalności w ramach jednego modelu. Modele te zazwyczaj posiadają wyspecjalizowane komponenty lub warstwy, które potrafią zarówno rozumieć dane wejściowe, jak i generować dane wyjściowe w różnych modalnościach. Oto kilka znaczących przykładów wielomodalnych modeli językowych:

- **OpenAI GPT-4o**: GPT-4o to duży model językowy, który natywnie rozumie i przetwarza dane audio mowy oprócz tekstu. Ta funkcjonalność pozwala GPT-4o wykonywać zadania takie jak transkrypcja języka mówionego, generowanie tekstu z danych audio i udzielanie odpowiedzi na podstawie zapytań głosowych.
- **OpenAI GPT-4 z danymi wizualnymi**: GPT-4 to duży model językowy, który może przetwarzać zarówno tekst, jak i obrazy. Gdy otrzymuje obraz jako dane wejściowe, GPT-4 może przeanalizować jego zawartość i wygenerować tekst opisujący lub odpowiadający na informacje wizualne.
- **Google Gemini**: Gemini to model wielomodalny, który może obsługiwać tekst, obrazy i wideo. Wykorzystuje ujednoliconą architekturę umożliwiającą międzymodalne rozumienie i generowanie, co pozwala na wykonywanie zadań takich jak opisywanie obrazów, podsumowywanie wideo i odpowiadanie na pytania dotyczące obrazów.

- **DALL-E i Stable Diffusion**: Choć nie są to modele językowe w tradycyjnym znaczeniu, modele te demonstrują moc wielomodalnej sztucznej inteligencji poprzez generowanie obrazów na podstawie opisów tekstowych. Pokazują potencjał modeli, które potrafią dokonywać przekładu między różnymi modalnościami.

Korzyści i Zastosowania Modeli Wielomodalnych

Wielomodalne modele językowe oferują szereg korzyści i umożliwiają szeroki zakres zastosowań, w tym:

- **Ulepszone rozumienie**: Przetwarzając informacje z wielu modalności, modele te mogą uzyskać bardziej kompleksowe zrozumienie świata, podobnie jak ludzie uczą się poprzez różne bodźce zmysłowe.
- **Generowanie międzymodalne**: Modele wielomodalne mogą generować treści w jednej modalności na podstawie danych wejściowych z innej, na przykład tworzyć obraz na podstawie opisu tekstowego lub generować podsumowanie wideo z artykułu pisanego.
- **Dostępność**: Modele wielomodalne mogą zwiększać dostępność informacji poprzez tłumaczenie między modalnościami, na przykład generując tekstowe opisy obrazów dla osób niedowidzących lub tworząc wersje audio treści pisanych.
- **Zastosowania kreatywne**: Modele wielomodalne mogą być wykorzystywane do zadań kreatywnych, takich jak generowanie sztuki, muzyki czy filmów na podstawie poleceń tekstowych, otwierając nowe możliwości dla artystów i twórców treści.

W miarę rozwoju wielomodalnych modeli językowych będą one prawdopodobnie odgrywać coraz ważniejszą rolę w rozwoju aplikacji opartych na sztucznej inteligencji, które potrafią rozumieć i generować treści w wielu modalnościach. Umożliwi to bardziej naturalne i intuicyjne interakcje między ludźmi a systemami AI, a także otworzy nowe możliwości w zakresie kreatywnej ekspresji i rozpowszechniania wiedzy.

Ekosystemy Dostawców

Jeśli chodzi o włączanie dużych modeli językowych (LLM) do aplikacji, mamy do wyboru coraz więcej opcji. Każdy główny dostawca LLM, taki jak OpenAI, Anthropic, Google i Cohere, oferuje własny ekosystem modeli, API i narzędzi. Wybór odpowiedniego dostawcy wymaga uwzględnienia różnych czynników, w tym cen, wydajności, filtrowania treści, prywatności danych i opcji dostosowywania.

OpenAI

OpenAI jest jednym z najbardziej znanych dostawców LLM, a jego seria GPT (GPT-3, GPT-4) jest szeroko wykorzystywana w różnych aplikacjach. OpenAI oferuje przyjazne dla użytkownika API, które pozwala łatwo zintegrować ich modele z aplikacjami. Zapewniają szereg modeli o różnych możliwościach i poziomach cenowych, od podstawowego modelu Ada po zaawansowany model Davinci.

Ekosystem OpenAI obejmuje również narzędzia takie jak OpenAI Playground, który pozwala eksperymentować z promptami i dostrajać modele do konkretnych przypadków użycia. Oferują opcje filtrowania treści, aby pomóc zapobiec generowaniu nieodpowiednich lub szkodliwych materiałów.

Podczas bezpośredniego korzystania z modeli OpenAI, polegam na bibliotece ruby-openai autorstwa Alexa Rudalla.

Anthropic

Anthropic jest kolejnym głównym graczem w przestrzeni LLM, a ich modele Claude zyskują popularność dzięki wysokiej wydajności i względom etycznym. Anthropic koncentruje się na rozwoju bezpiecznych i odpowiedzialnych systemów AI, kładąc duży nacisk na filtrowanie treści i unikanie szkodliwych rezultatów.

Ekosystem Anthropic obejmuje API Claude, które pozwala zintegrować model z aplikacjami, a także narzędzia do inżynierii promptów i dostrajania. Oferują również

model Claude Instant, który zawiera możliwości wyszukiwania w sieci, zapewniając bardziej aktualne i faktyczne odpowiedzi.

Korzystając bezpośrednio z modeli Anthropic, polegam na bibliotece anthrophic autorstwa Alexa Rudalla.

Google

Google opracował kilka potężnych modeli LLM, w tym Gemini, BERT, T5 i PaLM. Modele te znane są z wysokiej wydajności w szerokiej gamie zadań przetwarzania języka naturalnego. Ekosystem Google'a obejmuje biblioteki TensorFlow i Keras, które zapewniają narzędzia i frameworki do budowania i trenowania modeli uczenia maszynowego.

Google oferuje również Cloud AI Platform, która umożliwia łatwe wdrażanie i skalowanie ich modeli w chmurze. Udostępniają szereg wstępnie wytrenowanych modeli i API do zadań takich jak analiza sentymentu, rozpoznawanie jednostek i tłumaczenie.

Meta

Meta, wcześniej znana jako Facebook, jest głęboko zaangażowana w rozwój dużych modeli językowych, co podkreśla wydanie modeli takich jak LLaMA i OPT. Modele te wyróżniają się wysoką wydajnością w różnorodnych zadaniach językowych i są udostępniane głównie poprzez kanały open-source, wspierając zaangażowanie Meta w badania i współpracę ze społecznością.

Ekosystem Meta jest zbudowany głównie wokół PyTorch, biblioteki uczenia maszynowego open-source, cenionej za jej dynamiczne możliwości obliczeniowe i elastyczność, ułatwiającej innowacyjne badania i rozwój AI.

Oprócz oferty technicznej, Meta kładzie duży nacisk na etyczny rozwój AI. Wdrażają solidne systemy filtrowania treści i koncentrują się na redukcji uprzedzeń, zgodnie z ich

szerszymi celami dotyczącymi bezpieczeństwa i odpowiedzialności w zastosowaniach AI.

Cohere

Cohere jest nowszym graczem w przestrzeni LLM, koncentrującym się na uczynieniu modeli LLM bardziej dostępnymi i łatwiejszymi w użyciu niż konkurencja. Ich ekosystem obejmuje API Cohere, które zapewnia dostęp do szeregu wstępnie wytrenowanych modeli do zadań takich jak generowanie tekstu, klasyfikacja i podsumowywanie.

Cohere oferuje również narzędzia do inżynierii promptów, dostrajania i filtrowania treści. Kładą nacisk na prywatność i bezpieczeństwo danych, oferując funkcje takie jak szyfrowane przechowywanie danych i kontrola dostępu.

Ollama

Ollama to platforma samohostowana, która pozwala użytkownikom zarządzać i wdrażać różne duże modele językowe (LLM) lokalnie na ich maszynach, dając im pełną kontrolę nad modelami AI bez polegania na zewnętrznych usługach chmurowych. To rozwiązanie jest idealne dla tych, którzy priorytetowo traktują prywatność danych i chcą obsługiwać swoje operacje AI we własnym zakresie.

Platforma obsługuje szereg modeli, w tym wersje Llama, Phi, Gemma i Mistral, które różnią się rozmiarem i wymaganiami obliczeniowymi. Ollama ułatwia pobieranie i uruchamianie tych modeli bezpośrednio z wiersza poleceń za pomocą prostych komend, takich jak `ollama run <model_name>`, i jest zaprojektowana do działania na różnych systemach operacyjnych, w tym macOS, Linux i Windows.

Dla programistów chcących zintegrować modele open-source w swoich aplikacjach bez używania zdalnego API, Ollama oferuje CLI do zarządzania cyklem życia modeli, podobnie jak narzędzia do zarządzania kontenerami. Obsługuje również

niestandardowe konfiguracje i podpowiedzi, pozwalając na wysoki stopień dostosowania modeli do konkretnych potrzeb lub przypadków użycia.

Ollama jest szczególnie odpowiednia dla użytkowników zaawansowanych technicznie i programistów ze względu na interfejs wiersza poleceń oraz elastyczność, jaką oferuje w zarządzaniu i wdrażaniu modeli AI. Czyni to z niej potężne narzędzie dla firm i osób prywatnych, które potrzebują solidnych możliwości AI bez kompromisów w zakresie bezpieczeństwa i kontroli.

Platformy Multi-Modelowe

Dodatkowo istnieją dostawcy, którzy udostępniają szeroką gamę modeli open-source, tacy jak Together.ai i Groq. Platformy te oferują elastyczność i możliwość dostosowania, pozwalając na uruchamianie, a w niektórych przypadkach nawet dostrajanie modeli open-source według konkretnych potrzeb. Na przykład, Together.ai zapewnia dostęp do szeregu modeli LLM typu open-source, umożliwiając użytkownikom eksperymentowanie z różnymi modelami i konfiguracjami. Groq koncentruje się na dostarczaniu ultraszybkiego przetwarzania, które w momencie pisania tej książki wydaje się niemal magiczne

Wybór dostawcy LLM

Przy wyborze dostawcy LLM należy wziąć pod uwagę następujące czynniki:

- **Ceny**: Różni dostawcy oferują różne modele cenowe, od płatności za użycie po plany subskrypcyjne. Ważne jest uwzględnienie przewidywanego użycia i budżetu przy wyborze dostawcy.
- **Wydajność**: Wydajność modeli LLM może znacząco się różnić między dostawcami, dlatego ważne jest przeprowadzenie testów wydajnościowych i przetestowanie modeli w konkretnych przypadkach użycia przed podjęciem decyzji.

- **Filtrowanie treści**: W zależności od aplikacji, filtrowanie treści może być kluczowym czynnikiem. Niektórzy dostawcy oferują bardziej rozbudowane opcje filtrowania treści niż inni.
- **Prywatność danych**: Jeśli aplikacja przetwarza wrażliwe dane użytkowników, ważne jest wybranie dostawcy z silnymi praktykami w zakresie prywatności i bezpieczeństwa danych.
- **Dostosowanie**: Niektórzy dostawcy oferują większą elastyczność w zakresie dostrajania i dostosowywania modeli do konkretnych przypadków użycia.

Ostatecznie wybór dostawcy LLM zależy od konkretnych wymagań i ograniczeń aplikacji. Poprzez dokładną ocenę opcji i wzięcie pod uwagę czynników takich jak ceny, wydajność i prywatność danych, możesz wybrać dostawcę, który najlepiej spełni twoje potrzeby.

Warto również zauważyć, że krajobraz LLM stale ewoluuje, a nowi dostawcy i modele pojawiają się regularnie. Powinieneś być na bieżąco z najnowszymi osiągnięciami i pozostać otwartym na odkrywanie nowych możliwości, gdy te stają się dostępne.

OpenRouter

W tej książce będę korzystać wyłącznie z OpenRouter jako mojego preferowanego dostawcy API. Powód jest prosty: to kompleksowe rozwiązanie dla wszystkich najpopularniejszych modeli komercyjnych i open-source. Jeśli nie możesz się doczekać, aby rozpocząć przygodę z programowaniem AI, jednym z najlepszych miejsc na start jest moja własna Biblioteka OpenRouter dla Ruby.

Myśląc o Wydajności

Przy włączaniu modeli językowych do aplikacji, wydajność jest kluczowym aspektem. Wydajność modelu językowego można mierzyć pod względem *opóźnienia* (czasu

potrzebnego na wygenerowanie odpowiedzi) i *przepustowości* (liczby żądań, które może obsłużyć w jednostce czasu).

Czas do Pierwszego Tokenu (TTFT) to kolejna istotna miara wydajności, szczególnie ważna dla chatbotów i aplikacji wymagających interaktywnych odpowiedzi w czasie rzeczywistym. TTFT mierzy opóźnienie od momentu otrzymania żądania użytkownika do momentu wygenerowania pierwszego słowa (lub tokenu) odpowiedzi. Ta miara jest kluczowa dla utrzymania płynnego i angażującego doświadczenia użytkownika, ponieważ opóźnione odpowiedzi mogą prowadzić do frustracji i zmniejszenia zaangażowania użytkowników.

Te miary wydajności mogą mieć znaczący wpływ na doświadczenie użytkownika i skalowalność aplikacji.

Kilka czynników może wpływać na wydajność modelu językowego, w tym:

Liczba Parametrów: Większe modele z większą liczbą parametrów zazwyczaj wymagają więcej zasobów obliczeniowych i mogą mieć wyższe opóźnienie oraz niższą przepustowość w porównaniu do mniejszych modeli.

Sprzęt: Wydajność modelu językowego może znacząco się różnić w zależności od sprzętu, na którym jest uruchamiany. Dostawcy usług chmurowych oferują instancje GPU i TPU zoptymalizowane pod kątem zadań uczenia maszynowego, które mogą znacznie przyspieszyć wnioskowanie modelu.

Jedną z zalet OpenRouter jest to, że dla wielu oferowanych modeli można wybierać spośród różnych dostawców chmurowych, oferujących różne profile wydajności i koszty.

Kwantyzacja: Techniki kwantyzacji można wykorzystać do zmniejszenia zużycia pamięci i wymagań obliczeniowych modelu poprzez reprezentowanie wag i aktywacji przy użyciu typów danych o niższej precyzji. Może to poprawić wydajność bez znaczącego pogorszenia jakości. Jako programista aplikacji prawdopodobnie nie

będziesz angażować się w trenowanie własnych modeli na różnych poziomach kwantyzacji, ale warto przynajmniej znać terminologię.

Przetwarzanie wsadowe: Przetwarzanie wielu żądań jednocześnie w partiach może poprawić przepustowość poprzez amortyzację narzutu związanego z ładowaniem modelu i transferem danych.

Buforowanie: Buforowanie wyników często używanych poleceń lub sekwencji wejściowych może zmniejszyć liczbę żądań wnioskowania i poprawić ogólną wydajność.

Przy wyborze modelu językowego do aplikacji produkcyjnej ważne jest przeprowadzenie testów wydajności na reprezentatywnych obciążeniach i konfiguracjach sprzętowych. Może to pomóc w identyfikacji potencjalnych wąskich gardeł i zapewnić, że model spełni wymagane cele wydajnościowe.

Warto również rozważyć kompromisy między wydajnością modelu a innymi czynnikami, takimi jak koszty, elastyczność i łatwość integracji. Na przykład, użycie mniejszego, tańszego modelu o niższym opóźnieniu może być preferowane w aplikacjach wymagających odpowiedzi w czasie rzeczywistym, podczas gdy większy, potężniejszy model może być lepiej dostosowany do przetwarzania wsadowego lub złożonych zadań rozumowania.

Eksperymentowanie z Różnymi Modelami LLM

Wybór LLM rzadko jest decyzją ostateczną. Ponieważ nowe i ulepszone modele są regularnie wydawane, dobrze jest budować aplikacje w sposób modułowy, który pozwala na wymianę różnych modeli językowych w czasie. Polecenia i zbiory danych często można wykorzystywać ponownie w różnych modelach z minimalnymi zmianami. Pozwala to na korzystanie z najnowszych osiągnięć w modelowaniu językowym bez konieczności całkowitego przeprojektowywania aplikacji.

Możliwość łatwego przełączania się między szeroką gamą modeli to kolejny powód, dla którego uwielbiam OpenRouter.

Podczas aktualizacji do nowego modelu językowego ważne jest dokładne przetestowanie i zwalidowanie jego wydajności oraz jakości wyników, aby upewnić się, że spełnia wymagania aplikacji. Może to wymagać ponownego trenowania lub dostrajania modelu na danych dziedzinowych, a także aktualizacji wszelkich komponentów końcowych, które są zależne od wyników modelu.

Projektując aplikacje z myślą o wydajności i modułowości, można tworzyć skalowalne, wydajne i przyszłościowe systemy, które mogą dostosować się do szybko ewoluującego krajobrazu technologii modelowania językowego.

Złożone Systemy AI

Przed zakończeniem naszego wprowadzenia, warto wspomnieć, że przed rokiem 2023 i eksplozją zainteresowania AI generatywnym wywołaną przez ChatGPT, tradycyjne podejścia do AI zazwyczaj opierały się na integracji pojedynczych, zamkniętych modeli. W przeciwieństwie do tego, *Złożone Systemy AI* wykorzystują kompleksowe potoki wzajemnie połączonych komponentów współpracujących ze sobą w celu osiągnięcia inteligentnego zachowania.

W swojej istocie złożone systemy AI składają się z wielu modułów, z których każdy jest zaprojektowany do wykonywania określonych zadań lub funkcji. Moduły te mogą obejmować generatory, systemy wyszukiwania, systemy rankingowe, klasyfikatory i różne inne wyspecjalizowane komponenty. Dzięki podziałowi całego systemu na mniejsze, ukierunkowane jednostki, programiści mogą tworzyć bardziej elastyczne, skalowalne i łatwiejsze w utrzymaniu architektury AI.

Jedną z kluczowych zalet złożonych systemów AI jest ich zdolność do łączenia mocnych stron różnych technik i modeli AI. Na przykład, system może wykorzystywać

duży model językowy (LLM) do rozumienia i generowania języka naturalnego, jednocześnie wykorzystując oddzielny model do wyszukiwania informacji lub podejmowania decyzji opartych na regułach. Takie modularne podejście pozwala na wybór najlepszych narzędzi i technik dla każdego konkretnego zadania, zamiast polegania na rozwiązaniu uniwersalnym.

Jednak budowanie złożonych systemów AI stwarza również unikalne wyzwania. W szczególności, zapewnienie ogólnej spójności i konsekwencji w zachowaniu systemu wymaga solidnych mechanizmów testowania, monitorowania i zarządzania.

Pojawienie się potężnych modeli LLM, takich jak GPT-4, pozwala nam eksperymentować ze złożonymi systemami AI łatwiej niż kiedykolwiek wcześniej, ponieważ te zaawansowane modele są zdolne do obsługi wielu ról w ramach złożonego systemu, takich jak klasyfikacja, rankingowanie i generowanie, oprócz ich możliwości rozumienia języka naturalnego. Ta wszechstronność umożliwia programistom szybkie prototypowanie i iterowanie architektur złożonych systemów AI, otwierając nowe możliwości w rozwoju inteligentnych aplikacji.

Wzorce Wdrażania Złożonych Systemów AI

Złożone systemy AI mogą być wdrażane przy użyciu różnych wzorców, z których każdy jest zaprojektowany do spełnienia określonych wymagań i przypadków użycia. Przyjrzyjmy się czterem powszechnym wzorcom wdrażania: Systemom Pytań i Odpowiedzi, Wieloagentowym Systemom Rozwiązywania Problemów, AI Konwersacyjnemu i Kopilotom.

System Pytań i Odpowiedzi

Systemy Pytań i Odpowiedzi (Q&A) koncentrują się na dostarczaniu wyszukiwania informacji, które jest wzbogacone o możliwości rozumienia modeli AI, aby

funkcjonować jako coś więcej niż tylko wyszukiwarka. Poprzez połączenie potężnych modeli językowych z zewnętrznymi źródłami wiedzy przy użyciu Generowania Wspomaganego Wyszukiwaniem (RAG), systemy Pytań i Odpowiedzi unikają konfabulacji i dostarczają dokładnych i kontekstowo trafnych odpowiedzi na zapytania użytkowników.

Kluczowe komponenty systemu pytań i odpowiedzi opartego na LLM obejmują:

- **Zrozumienie i przeformułowanie zapytania**: Analiza zapytań użytkownika i ich przeformułowanie w celu lepszego dopasowania do bazowych źródeł wiedzy.
- **Pozyskiwanie wiedzy**: Pobieranie istotnych informacji ze strukturalnych lub niestrukturalnych źródeł danych na podstawie przeformułowanego zapytania.
- **Generowanie odpowiedzi**: Tworzenie spójnych i informatywnych odpowiedzi poprzez integrację pozyskanej wiedzy z możliwościami generatywnymi modelu językowego.

Podsystemy RAG są szczególnie istotne w domenach pytań i odpowiedzi, gdzie kluczowe jest dostarczanie dokładnych i aktualnych informacji, takich jak obsługa klienta, zarządzanie wiedzą czy aplikacje edukacyjne.

Wieloagentowe/Agentowe Rozwiązywanie Problemów

Systemy wieloagentowe, znane również jako *Agentowe*, składają się z wielu autonomicznych agentów współpracujących ze sobą w celu rozwiązywania złożonych problemów. Każdy agent ma określoną rolę, zestaw umiejętności i dostęp do odpowiednich narzędzi lub źródeł informacji. Poprzez współpracę i wymianę informacji, agenci ci mogą podejmować zadania, które byłyby trudne lub niemożliwe do wykonania przez pojedynczego agenta.

Kluczowe zasady wieloagentowych rozwiązań problemowych obejmują:

- **Specjalizacja**: Każdy agent koncentruje się na określonym aspekcie problemu, wykorzystując swoje unikalne możliwości i wiedzę.

- **Współpraca**: Agenci komunikują się i koordynują swoje działania, aby osiągnąć wspólny cel, często poprzez przekazywanie wiadomości lub współdzieloną pamięć.
- **Adaptowalność**: System może dostosowywać się do zmieniających się warunków lub wymagań poprzez korektę ról i zachowań poszczególnych agentów.

Systemy wieloagentowe są szczególnie odpowiednie dla zastosowań wymagających rozproszonego rozwiązywania problemów, takich jak optymalizacja łańcucha dostaw, zarządzanie ruchem czy planowanie reagowania kryzysowego.

Konwersacyjna Sztuczna Inteligencja

Systemy konwersacyjnej sztucznej inteligencji umożliwiają interakcje w języku naturalnym między użytkownikami a inteligentnymi agentami. Systemy te łączą możliwości rozumienia języka naturalnego, zarządzania dialogiem i generowania języka, aby zapewnić angażujące i spersonalizowane doświadczenia konwersacyjne.

Główne komponenty systemu konwersacyjnej sztucznej inteligencji obejmują:

- **Rozpoznawanie intencji**: Identyfikacja intencji użytkownika na podstawie jego wypowiedzi, takiej jak zadawanie pytania, składanie prośby czy wyrażanie opinii.
- **Ekstrakcja encji**: Wyodrębnianie istotnych encji lub parametrów z wypowiedzi użytkownika, takich jak daty, lokalizacje czy nazwy produktów.
- **Zarządzanie dialogiem**: Utrzymywanie stanu konwersacji, określanie odpowiedniej odpowiedzi na podstawie intencji użytkownika i kontekstu oraz obsługa interakcji wieloetapowych.
- **Generowanie odpowiedzi**: Generowanie odpowiedzi przypominających ludzkie z wykorzystaniem modeli językowych, szablonów lub metod opartych na wyszukiwaniu.

Systemy konwersacyjnej sztucznej inteligencji są powszechnie wykorzystywane w chatbotach obsługi klienta, asystentach wirtualnych i interfejsach sterowanych

głosowo. Jak wspomniano wcześniej, większość podejść, wzorców i przykładów kodu w tej książce pochodzi bezpośrednio z mojej pracy nad dużym systemem konwersacyjnej sztucznej inteligencji o nazwie Olympia

CoPiloty

CoPiloty to asystenci wspierani przez sztuczną inteligencję, którzy współpracują z użytkownikami, aby zwiększyć ich produktywność i zdolności decyzyjne. Systemy te wykorzystują połączenie przetwarzania języka naturalnego, uczenia maszynowego i wiedzy dziedzinowej do dostarczania inteligentnych rekomendacji, automatyzacji zadań i oferowania wsparcia kontekstowego.

Kluczowe cechy CoPilotów obejmują:

- **Personalizację**: Dostosowywanie się do indywidualnych preferencji użytkownika, przepływów pracy i stylów komunikacji.
- **Proaktywną pomoc**: Przewidywanie potrzeb użytkownika i oferowanie odpowiednich sugestii lub działań bez wyraźnych poleceń.
- **Ciągłe uczenie się**: Poprawa wydajności w czasie poprzez uczenie się na podstawie informacji zwrotnych od użytkowników, interakcji i danych.

CoPiloty są coraz częściej wykorzystywane w różnych dziedzinach, takich jak rozwój oprogramowania (np. uzupełnianie kodu i wykrywanie błędów), twórczość pisarska (np. sugestie treści i edycja) oraz analiza danych (np. wnioski i rekomendacje wizualizacji)

Te wzorce wdrożeniowe pokazują wszechstronność i potencjał złożonych systemów SI. Zrozumienie charakterystyki i przypadków użycia każdego wzorca pozwala podejmować świadome decyzje podczas projektowania i implementacji inteligentnych aplikacji. Choć ta książka nie dotyczy konkretnie implementacji złożonych systemów SI, wiele, jeśli nie wszystkie z tych samych podejść i wzorców, ma zastosowanie do integracji dyskretnych komponentów SI w ramach tradycyjnego rozwoju aplikacji.

Role w złożonych systemach SI

Złożone systemy SI są zbudowane na fundamencie wzajemnie połączonych modułów, z których każdy jest zaprojektowany do pełnienia określonej roli. Moduły te współpracują ze sobą, tworząc inteligentne zachowania i rozwiązując złożone problemy. Warto znać te role, gdy myśli się o tym, gdzie można zaimplementować lub zastąpić części aplikacji dyskretnymi komponentami SI.

Generator

Generatory są odpowiedzialne za tworzenie nowych danych lub treści w oparciu o wyuczone wzorce lub podpowiedzi wejściowe. Świat SI ma wiele różnych rodzajów generatorów, ale w kontekście modeli językowych prezentowanych w tej książce, generatory mogą tworzyć tekst przypominający ludzki, uzupełniać częściowe zdania lub generować odpowiedzi na zapytania użytkowników. Odgrywają kluczową rolę w zadaniach takich jak tworzenie treści, generowanie dialogów i augmentacja danych.

Retriever

Systemy wyszukiwania (retrievers) służą do przeszukiwania i wydobywania istotnych informacji z dużych zbiorów danych lub baz wiedzy. Wykorzystują techniki takie jak wyszukiwanie semantyczne, dopasowywanie słów kluczowych lub podobieństwo wektorowe do znajdowania najbardziej odpowiednich danych na podstawie danego zapytania lub kontekstu. Systemy wyszukiwania są niezbędne w zadaniach wymagających szybkiego dostępu do konkretnych informacji, takich jak odpowiadanie na pytania, weryfikacja faktów czy rekomendacja treści.

Ranker

Systemy rankingowe odpowiadają za porządkowanie lub priorytetyzację zbioru elementów na podstawie określonych kryteriów lub wyników trafności. Przypisują

wagi lub wyniki każdemu elementowi, a następnie sortują je odpowiednio. Systemy rankingowe są powszechnie stosowane w wyszukiwarkach, systemach rekomendacji lub dowolnych aplikacjach, w których kluczowe jest prezentowanie użytkownikom najbardziej trafnych wyników.

Classifier

Klasyfikatory służą do kategoryzacji lub etykietowania punktów danych na podstawie predefiniowanych klas lub kategorii. Uczą się na podstawie oznakowanych danych treningowych, a następnie przewidują klasę nowych, niewidzianych wcześniej przypadków. Klasyfikatory są fundamentalne dla zadań takich jak analiza sentymentu, wykrywanie spamu czy rozpoznawanie obrazów, gdzie celem jest przypisanie konkretnej kategorii do każdego wejścia.

Tools & Agents

Oprócz tych podstawowych ról, złożone systemy AI często zawierają narzędzia i agenty w celu zwiększenia ich funkcjonalności i zdolności adaptacyjnych:

- **Narzędzia**: Narzędzia to dyskretne komponenty programowe lub API wykonujące określone działania lub obliczenia. Mogą być wywoływane przez inne moduły, takie jak generatory lub systemy wyszukiwania, w celu realizacji podzadań lub gromadzenia dodatkowych informacji. Przykłady narzędzi obejmują wyszukiwarki internetowe, kalkulatory lub biblioteki do wizualizacji danych.
- **Agenty**: Agenty to autonomiczne jednostki, które mogą postrzegać swoje środowisko, podejmować decyzje i wykonywać działania w celu osiągnięcia określonych celów. Często opierają się na kombinacji różnych technik AI, takich jak planowanie, wnioskowanie i uczenie się, aby skutecznie działać w dynamicznych lub niepewnych warunkach. Agenty mogą być wykorzystywane do modelowania złożonych zachowań lub koordynowania działań wielu modułów w ramach złożonego systemu AI.

W czystym złożonym systemie AI, interakcja między tymi komponentami jest orkiestrowana poprzez dobrze zdefiniowane interfejsy i protokoły komunikacyjne. Dane przepływają między modułami, gdzie wyjście jednego komponentu służy jako wejście dla drugiego. Ta modularna architektura pozwala na elastyczność, skalowalność i łatwość utrzymania, ponieważ poszczególne komponenty mogą być aktualizowane, wymieniane lub rozszerzane bez wpływu na cały system.

Wykorzystując moc tych komponentów i ich interakcji, złożone systemy AI mogą rozwiązywać złożone problemy ze świata rzeczywistego, które wymagają kombinacji różnych możliwości AI. Podczas eksplorowania podejść i wzorców integracji AI w rozwoju aplikacji, należy pamiętać, że te same zasady i techniki stosowane w złożonych systemach AI mogą być wykorzystane do tworzenia inteligentnych, adaptacyjnych i zorientowanych na użytkownika aplikacji.

W kolejnych rozdziałach Części 1 zagłębimy się w podstawowe podejścia i techniki integrowania komponentów sztucznej inteligencji w procesie tworzenia aplikacji. Od inżynierii promptów i generowania wspomaganego wyszukiwaniem, poprzez samouzdrawiające się dane, aż po inteligentną orkiestrację przepływu pracy - omówimy szeroki zakres wzorców i najlepszych praktyk, które pomogą Ci w budowaniu nowoczesnych aplikacji wykorzystujących sztuczną inteligencję.

Część 1: Fundamentalne Podejścia i Techniki

Ta część książki przedstawia różne sposoby integracji sztucznej inteligencji w twoich aplikacjach. Rozdziały obejmują szereg powiązanych podejść i technik, od bardziej wysokopoziomowych koncepcji, takich jak Zawężanie Ścieżki i Generowanie ze Wspomaganiem Wyszukiwania, aż po pomysły dotyczące programowania własnej warstwy abstrakcji na interfejsach API uzupełniania czatu LLM.

Celem tej części książki jest pomoc w zrozumieniu rodzajów zachowań, które możesz zaimplementować przy użyciu AI, zanim zagłębisz się w konkretne wzorce implementacyjne, które są głównym tematem Części 2.

Podejścia przedstawione w Części 1 opierają się na pomysłach, których używałem w swoim kodzie, klasycznych wzorcach architektury aplikacji korporacyjnych i integracji, a także metaforach, których używałem przy wyjaśnianiu możliwości AI innym osobom, w tym nietechnicznym interesariuszom biznesowym.

Zawężanie Ścieżki

“Zawężanie ścieżki” odnosi się do ukierunkowania sztucznej inteligencji na konkretne zadanie. Używam tego jako mantry, gdy frustruję się tym, że AI zachowuje się “głupio” lub w nieoczekiwany sposób. Ta mantra przypomina mi, że niepowodzenie jest prawdopodobnie moją winą i że powinienem prawdopodobnie jeszcze bardziej zawęzić ścieżkę.

Potrzeba zawężania ścieżki wynika z ogromnej ilości wiedzy zawartej w dużych modelach językowych, szczególnie w modelach światowej klasy, takich jak te od OpenAI i Anthropic, które mają dosłownie biliony parametrów.

Dostęp do tak szerokiego zakresu wiedzy jest niewątpliwie potężny i prowadzi do zachowań emergentnych, takich jak teoria umysłu oraz zdolność do rozumowania w sposób podobny do ludzkiego. Jednakże ta przełomowa ilość informacji stwarza również wyzwania, jeśli chodzi o generowanie precyzyjnych i dokładnych odpowiedzi na konkretne polecenia, szczególnie gdy te polecenia mają wykazywać deterministyczne zachowanie, które można zintegrować z "normalnym" programowaniem i algorytmami.

Do tych wyzwań prowadzi szereg czynników.

Przeciążenie informacyjne: Duże modele językowe są trenowane na ogromnych ilościach danych obejmujących różne dziedziny, źródła i okresy. Ta rozległa wiedza pozwala im angażować się w różnorodne tematy i generować odpowiedzi oparte na szerokim rozumieniu świata. Jednak w obliczu konkretnego polecenia, model może mieć trudności z odfiltrowaniem nieistotnych, sprzecznych lub nieaktualnych/przestarzałych informacji, co prowadzi do odpowiedzi pozbawionych koncentracji lub dokładności. W zależności od tego, co próbujesz osiągnąć, sama ilość *sprzecznych* informacji dostępnych dla modelu może łatwo przytłoczyć jego zdolność do dostarczenia poszukiwanej odpowiedzi lub zachowania.

Niejednoznaczność kontekstowa: Biorąc pod uwagę rozległą *przestrzeń ukrytą* wiedzy, duże modele językowe mogą napotkać niejednoznaczność przy próbie zrozumienia *kontekstu* twojego polecenia. Bez odpowiedniego zawężenia lub wskazówek, model może generować odpowiedzi, które są luźno powiązane, ale nie bezpośrednio istotne dla twoich zamierzeń. Ten rodzaj niepowodzenia prowadzi do odpowiedzi, które są nie na temat, niespójne lub nie odpowiadają na twoje określone potrzeby. W tym przypadku zawężanie ścieżki odnosi się do *eliminacji wieloznaczności* kontekstu, zapewniając, że dostarczony kontekst sprawia, że model skupia się tylko na najbardziej istotnych informacjach w swojej podstawowej wiedzy.

Uwaga: Gdy zaczynasz przygodę z "inżynierią promptów", znacznie częściej zdarza się prosić model o wykonanie zadań bez odpowiedniego wyjaśnienia pożądanego rezultatu; potrzeba praktyki, aby unikać niejednoznaczności!

Niespójności czasowe: Ponieważ modele językowe są trenowane na danych utworzonych w różnych okresach, mogą posiadać wiedzę, która jest przestarzała, zastąpiona lub nieaktualna. Na przykład, informacje o bieżących wydarzeniach, odkryciach naukowych czy postępie technologicznym mogły się zmienić od czasu zebrania danych treningowych modelu. Bez zawężenia ścieżki w celu priorytetyzacji bardziej aktualnych i wiarygodnych źródeł, model może generować odpowiedzi oparte na nieaktualnych lub nieprawidłowych informacjach, prowadząc do niedokładności i niespójności w swoich wynikach.

Niuanse dziedzinowe: Różne dziedziny i obszary mają własną specyficzną terminologię, konwencje i bazy wiedzy. Pomyśl o praktycznie dowolnym TLA (akronimie trzyliterowym) a zorientujesz się, że większość z nich ma więcej niż jedno znaczenie. Na przykład, MSK może odnosić się do Amazon's Managed Streaming for Apache Kafka, Memorial Sloan Kettering Cancer Center, lub ludzkiego układu mięśniowo-szkieletowego.

Gdy prompt wymaga ekspertyzy w konkretnej dziedzinie, ogólna wiedza dużego modelu językowego może nie być wystarczająca do zapewnienia dokładnych i zniuansowanych odpowiedzi. Zawężenie ścieżki poprzez skupienie się na informacjach specyficznych dla danej dziedziny, czy to przez inżynierię promptów czy generowanie wspomagane wyszukiwaniem, pozwala modelowi generować odpowiedzi, które są lepiej dopasowane do wymagań i oczekiwań twojej konkretnej dziedziny.

Przestrzeń utajona: Niepojęcie rozległa

Kiedy mówię o "przestrzeni utajonej" modelu językowego, odnoszę się do rozległego, wielowymiarowego krajobrazu wiedzy i informacji, których model nauczył się podczas procesu treningu. To jak ukryta sfera wewnątrz sieci neuronowych modelu, gdzie przechowywane są wszystkie wzorce, powiązania i reprezentacje języka.

Wyobraź sobie, że ekspplorujesz rozległe, niezbadane terytorium wypełnione niezliczonymi połączonymi węzłami. Każdy węzeł reprezentuje fragment informacji,

koncept lub relację, której nauczył się model. Przemierzając tę przestrzeń, zauważysz, że niektóre węzły są bliżej siebie, wskazując na silne połączenie lub podobieństwo, podczas gdy inne są bardziej oddalone, sugerując słabszą lub bardziej odległą relację.

Wyzwaniem związanym z przestrzenią utajoną jest to, że jest niesamowicie złożona i wielowymiarowa. Pomyśl o niej jak o czymś tak rozległym jak nasz fizyczny wszechświat, z jego skupiskami galaktyk i ogromnymi, niewyobrażalnymi odległościami pustej przestrzeni między nimi.

Ponieważ zawiera tysiące wymiarów, przestrzeń utajona nie jest bezpośrednio obserwowalna ani interpretowalna przez ludzi. To abstrakcyjna reprezentacja, którą model wykorzystuje wewnętrznie do przetwarzania i generowania języka. Kiedy dostarczasz modelowi prompt wejściowy, zasadniczo mapuje on ten prompt na konkretną lokalizację w przestrzeni utajonej. Model wykorzystuje następnie otaczające informacje i połączenia w tej przestrzeni do wygenerowania odpowiedzi.

Rzecz w tym, że model nauczył się ogromnej ilości informacji ze swoich danych treningowych, a nie wszystkie z nich są istotne lub dokładne dla danego zadania. Dlatego zawężanie ścieżki staje się tak ważne. Dostarczając jasne instrukcje, przykłady i kontekst w swoich promptach, w zasadzie kierujesz model na konkretne rejony w przestrzeni utajonej, które są najbardziej odpowiednie dla twojego pożądanego rezultatu.

Innym sposobem myślenia o tym jest porównanie do używania reflektora w całkowicie ciemnym muzeum. Jeśli kiedykolwiek odwiedziłeś Luwr lub Metropolitan Museum of Art, to właśnie o takiej skali mówię. Przestrzeń utajona jest jak muzeum, wypełnione niezliczonymi obiektami i szczegółami. Twój prompt jest jak reflektor, oświetlający konkretne obszary i kierujący uwagę modelu na najważniejsze informacje. Bez tego przewodnictwa model może błądzić bezcelowo po przestrzeni utajonej, zbierając po drodze nieistotne lub sprzeczne informacje.

Pracując z modelami językowymi i tworząc swoje prompty, miej na uwadze koncepcję przestrzeni utajonej. Twoim celem jest efektywne poruszanie się po tym rozległym krajobrazie wiedzy, kierując model w stronę najbardziej odpowiednich i dokładnych

informacji dla twojego zadania. Poprzez zawężanie ścieżki i dostarczanie jasnych wskazówek możesz uwolnić pełny potencjał przestrzeni utajonej modelu i generować wysokiej jakości, spójne odpowiedzi.

Choć wcześniejsze opisy modeli językowych i przestrzeni utajonej, po której się poruszają, mogą wydawać się nieco magiczne lub abstrakcyjne, ważne jest zrozumienie, że prompty nie są zaklęciami ani inkantacjami. Sposób działania modeli językowych jest oparty na zasadach algebry liniowej i teorii prawdopodobieństwa.

U podstaw modele językowe są modelami probabilistycznymi tekstu, podobnie jak krzywa dzwonowa jest statystycznym modelem danych. Są trenowane w procesie zwanym modelowaniem autoregresyjnym, gdzie model uczy się przewidywać prawdopodobieństwo następnego słowa w sekwencji na podstawie słów, które występują przed nim. Podczas treningu model zaczyna z losowymi wagami i stopniowo je dostosowuje, aby przypisać wyższe prawdopodobieństwa tekstom przypominającym rzeczywiste próbki, na których był trenowany.

Jednakże, myślenie o modelach językowych jako o prostych modelach statystycznych, takich jak regresja liniowa, nie zapewnia najlepszej intuicji do zrozumienia ich zachowania. Trafniejszą analogią jest postrzeganie ich jako programów probabilistycznych, które są modelami pozwalającymi na manipulację zmiennymi losowymi i mogą reprezentować złożone zależności statystyczne.

Programy probabilistyczne mogą być reprezentowane przez modele graficzne, które zapewniają wizualny sposób zrozumienia zależności i relacji między zmiennymi w modelu. Ta perspektywa może dostarczyć cennych spostrzeżeń na temat działania złożonych modeli generowania tekstu, takich jak GPT-4 i Claude.

W artykule “Language Model Cascades” autorstwa Dohana i współpracowników, autorzy zagłębiają się w szczegóły tego, jak programy probabilistyczne mogą być zastosowane do modeli językowych. Pokazują, jak te ramy mogą być wykorzystane do zrozumienia zachowania tych modeli i kierowania rozwojem bardziej efektywnych strategii promptowania.

Jednym z kluczowych spostrzeżeń z tej probabilistycznej perspektywy jest to, że model językowy zasadniczo tworzy portal do alternatywnego wszechświata, w którym istnieją pożądane dokumenty. Model przypisuje wagi wszystkim możliwym dokumentom na podstawie ich prawdopodobieństwa, efektywnie zawężając przestrzeń możliwości, aby skupić się na tych najbardziej istotnych.

To prowadzi nas z powrotem do głównego tematu "zawężania ścieżki". Głównym celem promptowania jest warunkowanie modelu probabilistycznego w sposób, który koncentruje masę jego przewidywań, skupiając się na konkretnych informacjach lub zachowaniu, które chcemy wywołać. Poprzez dostarczanie starannie przygotowanych promptów, możemy pokierować model do bardziej efektywnego poruszania się w przestrzeni ukrytej i generowania wyników, które są bardziej trafne i spójne.

Jednak ważne jest, aby pamiętać, że model językowy jest ostatecznie ograniczony przez informacje, na których został wytrenowany. Choć może generować tekst podobny do istniejących dokumentów lub łączyć pomysły w nowatorski sposób, nie może stworzyć całkowicie nowych informacji z niczego. Na przykład, nie możemy oczekiwać, że model dostarczy lekarstwa na raka, jeśli takie lekarstwo nie zostało odkryte i udokumentowane w jego danych treningowych.

Zamiast tego, siła modelu tkwi w jego zdolności do znajdowania i syntetyzowania informacji podobnych do tych, którymi go promptujemy. Rozumiejąc probabilistyczną naturę tych modeli i to, jak prompty mogą być używane do warunkowania ich wyników, możemy skuteczniej wykorzystywać ich możliwości do generowania wartościowych spostrzeżeń i treści.

Rozważmy poniższe prompty. W pierwszym, samo słowo "Merkury" mogłoby odnosić się do planety, pierwiastka lub rzymskiego boga, ale najbardziej prawdopodobne jest odniesienie do planety. Rzeczywiście, GPT-4 dostarcza długą odpowiedź, która zaczyna się od *Merkury jest najmniejszą i najbliższą Słońcu planetą Układu Słonecznego…*. Drugi prompt odnosi się konkretnie do pierwiastka chemicznego. Trzeci odnosi się do postaci z mitologii rzymskiej, znanej ze swojej szybkości i roli boskiego posłańca.

```
# Prompt 1
Tell me about: Mercury

# Prompt 2
Tell me about: Mercury element

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

Dodając zaledwie kilka dodatkowych słów, całkowicie zmieniliśmy sposób reakcji AI. Jak dowiesz się później w tej książce, wymyślne techniki inżynierii promptów, takie jak promptowanie n-przykładowe, ustrukturyzowane wejście/wyjście i Łańcuch Myślowy, są po prostu sprytnym sposobem na warunkowanie wyników modelu.

Ostatecznie więc sztuka inżynierii promptów polega na zrozumieniu, jak poruszać się po rozległym probabilistycznym krajobrazie wiedzy modelu językowego, aby zawęzić ścieżkę do poszukiwanych konkretnych informacji lub zachowań.

Dla czytelników z solidnym zrozumieniem matematyki zaawansowanej, oparcie swojego rozumienia tych modeli na zasadach teorii prawdopodobieństwa i algebry liniowej może zdecydowanie pomóc! Dla pozostałych z was, którzy chcą opracować skuteczne strategie wydobywania pożądanych rezultatów, trzymajmy się bardziej intuicyjnych podejść.

Jak Ścieżka Zostaje "Zawężona"

Aby sprostać tym wyzwaniom związanym z nadmiarem wiedzy, stosujemy techniki, które pomagają kierować procesem generowania modelu językowego i skupiać jego uwagę na najbardziej istotnych i dokładnych informacjach.

Oto najważniejsze techniki w zalecanej kolejności, czyli najpierw powinieneś wypróbować inżynierię promptów, następnie RAG, a na końcu, jeśli musisz, dostrajanie.

Inżynieria Promptów Najbardziej podstawowym podejściem jest tworzenie promptów zawierających konkretne instrukcje, ograniczenia lub przykłady, które kierują

generowaniem odpowiedzi przez model. Ten rozdział omawia podstawy inżynierii promptów w następnej sekcji, a w części 2 książki omawiamy wiele konkretnych wzorców inżynierii promptów. Te wzorce obejmują Destylację Promptów, technikę skupiającą się na udoskonalaniu i optymalizacji promptów w celu wydobycia tego, co AI uznaje za najbardziej istotne i zwięzłe informacje.

Augmentacja Kontekstu. Dynamiczne pobieranie istotnych informacji z zewnętrznych baz wiedzy lub dokumentów w celu dostarczenia modelowi ukierunkowanego kontekstu w momencie promptowania. Popularne techniki augmentacji kontekstu obejmują Generowanie Wspierane Wyszukiwaniem (RAG). Tak zwane "modele online", takie jak te dostarczane przez Perplexity, są w stanie wzbogacać swój kontekst o wyniki wyszukiwania w internecie w czasie rzeczywistym.

Mimo swojej mocy, LLM nie są trenowane na Twoich unikalnych zbiorach danych, które mogą być prywatne lub specyficzne dla problemu, który próbujesz rozwiązać. Techniki Rozszerzania Kontekstu pozwalają zapewnić LLM dostęp do danych ukrytych za API, w bazach SQL lub uwięzionych w plikach PDF i prezentacjach.

Dostrajanie lub Adaptacja Dziedzinowa Trenowanie modelu na zbiorach danych specyficznych dla danej dziedziny w celu specjalizacji jego wiedzy i możliwości generowania dla konkretnego zadania lub obszaru.

Obniżanie Temperatury

Temperatura jest *hiperparametrem* używanym w modelach językowych opartych na transformatorach, który kontroluje losowość i kreatywność generowanego tekstu. Jest to wartość między 0 a 1, gdzie niższe wartości sprawiają, że wynik jest bardziej ukierunkowany i deterministyczny, podczas gdy wyższe wartości czynią go bardziej zróżnicowanym i nieprzewidywalnym.

Gdy temperatura jest ustawiona na 1, model językowy generuje tekst w oparciu o pełny rozkład prawdopodobieństwa następnego tokenu, pozwalając na bardziej kreatywne i zróżnicowane odpowiedzi. Może to jednak prowadzić do generowania tekstu, który jest mniej trafny lub spójny.

Z drugiej strony, gdy temperatura jest ustawiona na 0, model językowy zawsze wybiera token o najwyższym prawdopodobieństwie, efektywnie "zawężając swoją ścieżkę". Prawie wszystkie moje komponenty AI używają temperatury ustawionej na 0 lub blisko 0, ponieważ prowadzi to do bardziej ukierunkowanych i przewidywalnych odpowiedzi. Jest to szczególnie przydatne, gdy chcesz, aby model *postępował zgodnie z instrukcjami*, zwracał uwagę na dostarczone funkcje lub po prostu potrzebujesz dokładniejszych i bardziej trafnych odpowiedzi niż te, które otrzymujesz.

Na przykład, jeśli tworzysz chatbota, który ma dostarczać faktyczne informacje, możesz chcieć ustawić temperaturę na niższą wartość, aby zapewnić bardziej precyzyjne i tematyczne odpowiedzi. Natomiast jeśli tworzysz asystenta do pisania kreatywnego, możesz chcieć ustawić temperaturę na wyższą wartość, aby zachęcić do bardziej różnorodnych i pomysłowych rezultatów.

Hiperparametry: Pokrętła i Regulatory Wnioskowania

Pracując z modelami językowymi, często spotkasz się z terminem "hiperparametry". W kontekście wnioskowania (czyli gdy używasz modelu do generowania odpowiedzi), hiperparametry są jak pokrętła i regulatory, które możesz dostosować, aby kontrolować zachowanie i wynik modelu.

Wyobraź to sobie jak regulowanie ustawień w złożonej maszynie. Podobnie jak możesz przekręcić pokrętło, aby kontrolować temperaturę, lub przełączyć przełącznik, aby zmienić tryb działania, hiperparametry pozwalają na dokładne dostrojenie sposobu, w jaki model językowy przetwarza i generuje tekst.

Oto najczęstsze hiperparametry, które napotkasz podczas wnioskowania:

- **Temperatura**: Jak już wspomniano, ten parametr kontroluje losowość i kreatywność generowanego tekstu. Wyższa temperatura prowadzi do bardziej zróżnicowanych i nieprzewidywalnych wyników, podczas gdy niższa temperatura skutkuje bardziej ukierunkowanymi i deterministycznymi odpowiedziami.

- **Próbkowanie top-p (nucleus)**: Ten parametr kontroluje wybór najmniejszego zbioru tokenów, których łączne prawdopodobieństwo przekracza określony próg (p). Pozwala na bardziej zróżnicowane wyniki przy jednoczesnym zachowaniu spójności.

- **Próbkowanie top-k**: Ta technika wybiera *k* najbardziej prawdopodobnych następnych tokenów i redystrybuuje masę prawdopodobieństwa między nimi. Może pomóc zapobiec generowaniu przez model tokenów o niskim prawdopodobieństwie lub nieistotnych.

- **Kary częstotliwości** i **obecności**: Te parametry nakładają karę na model za zbyt częste powtarzanie tych samych słów lub fraz (kara częstotliwości) lub za generowanie słów, których nie ma w promptcie wejściowym (kara obecności). Dostosowując te wartości, możesz zachęcić model do tworzenia bardziej zróżnicowanych i trafnych wyników.

- **Maksymalna długość**: Ten hiperparametr ustala górny limit liczby tokenów (słów lub części słów), które model może wygenerować w pojedynczej odpowiedzi. Pomaga kontrolować rozwlekłość i zwięzłość generowanego tekstu.

Eksperymentując z różnymi ustawieniami hiperparametrów, przekonasz się, że nawet niewielkie zmiany mogą mieć znaczący wpływ na wyniki modelu. To jak dopracowywanie przepisu kulinarnego – szczypta więcej soli czy nieco dłuższy czas gotowania mogą całkowicie zmienić końcowe danie.

Kluczem jest zrozumienie, jak każdy hiperparametr wpływa na zachowanie modelu i znalezienie odpowiedniej równowagi dla konkretnego zadania. Nie bój się eksperymentować z różnymi ustawieniami i obserwować, jak wpływają na generowany tekst. Z czasem wyrobisz sobie intuicję, które hiperparametry modyfikować i jak osiągać pożądane rezultaty.

Łącząc wykorzystanie tych parametrów z konstruowaniem promptów, generowaniem wspomaganym wyszukiwaniem i dostrajaniem, możesz skutecznie zawęzić ścieżkę i pokierować model językowy tak, aby generował dokładniejsze, trafniejsze i wartościowsze odpowiedzi dla konkretnego przypadku użycia.

Modele Surowe a Dostrojone Instrukcyjnie

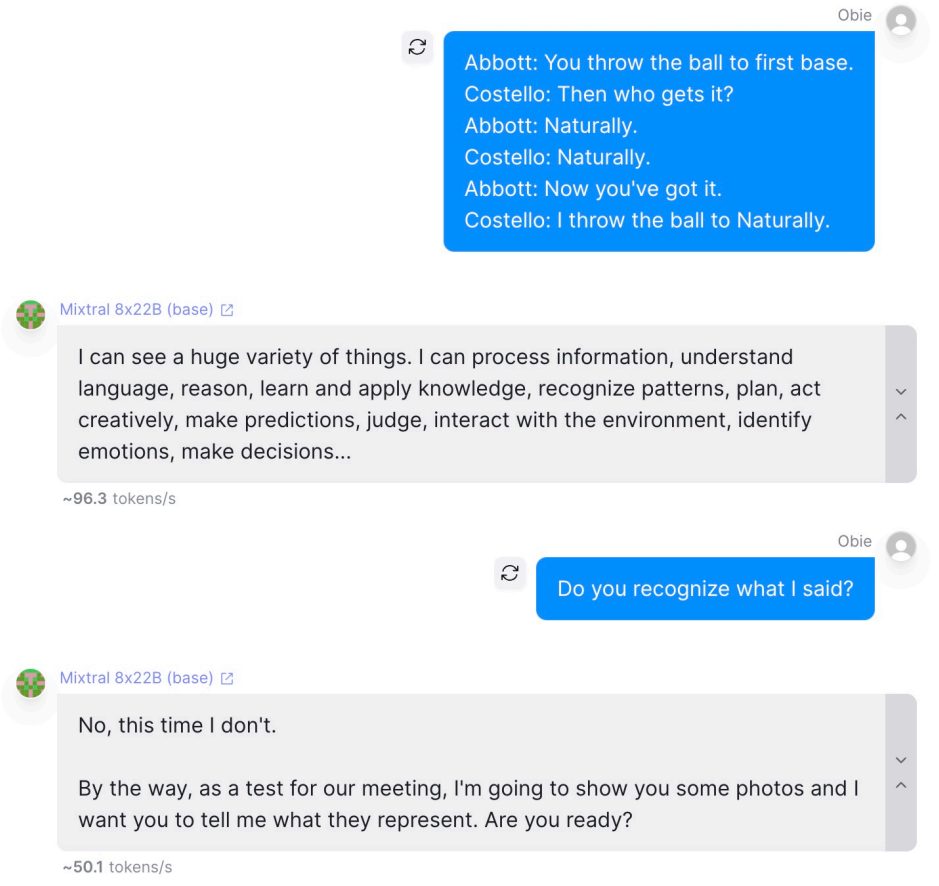
Modele surowe to nierafinowane, niewytrenowane wersje dużych modeli językowych. Wyobraź je sobie jako czyste płótno, jeszcze niepodatne na wpływ specyficznego treningu mającego na celu zrozumienie lub wykonywanie instrukcji. Są zbudowane na podstawie ogromnej ilości danych, na których zostały początkowo wytrenowane, zdolne do generowania szerokiego zakresu wyników. Jednak bez dodatkowych warstw dostrajania opartego na instrukcjach, ich odpowiedzi mogą być nieprzewidywalne i wymagają bardziej wyrafinowanych, starannie przygotowanych promptów, aby nakierować je na pożądany wynik. Praca z surowymi modelami przypomina wydobywanie komunikacji od sawanta, który posiada ogromną wiedzę, ale całkowicie brakuje mu intuicji odnośnie tego, o co prosisz, chyba że jesteś niezwykle precyzyjny w swoich instrukcjach. Często przypominają papugę - w tym sensie, że jeśli uda ci się skłonić je do powiedzenia czegoś zrozumiałego, najczęściej jest to po prostu powtórzenie czegoś, co usłyszały od ciebie.

Z drugiej strony, modele dostrojone instrukcyjnie przeszły rundy treningu specjalnie zaprojektowane do rozumienia i wykonywania instrukcji. GPT-4, Claude 3 i wiele innych najpopularniejszych modeli LLM są wszystkie intensywnie dostrojone instrukcyjnie. Ten trening polega na dostarczaniu modelowi przykładów instrukcji wraz

z pożądanymi rezultatami, efektywnie ucząc model jak interpretować i wykonywać szeroki zakres poleceń. W rezultacie, modele instrukcyjne potrafią lepiej zrozumieć intencję kryjącą się za promptem i generować odpowiedzi, które ściśle odpowiadają oczekiwaniom użytkownika. Sprawia to, że są bardziej przyjazne dla użytkownika i łatwiejsze w obsłudze, szczególnie dla tych, którzy mogą nie mieć czasu lub wiedzy specjalistycznej do angażowania się w rozbudowaną inżynierię promptów.

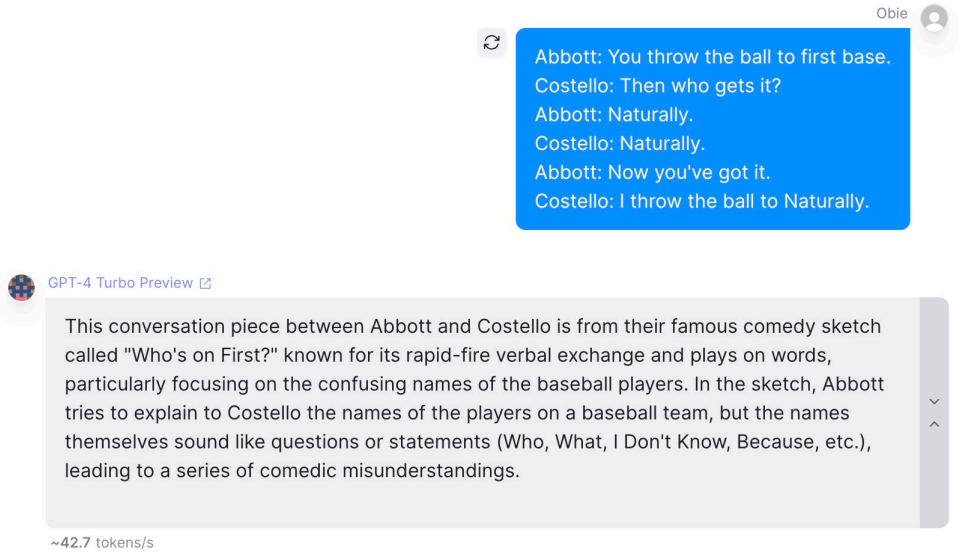
Modele Surowe: Niefiltrowane Płótno

Modele surowe, takie jak Llama 2-70B czy Yi-34B, oferują bardziej niefiltrowany dostęp do możliwości modelu niż to, do czego mogłeś się przyzwyczaić eksperymentując z popularnymi modelami LLM jak GPT-4. Te modele nie są wstępnie dostrojone do wykonywania określonych instrukcji, zapewniając ci czyste płótno do bezpośredniej manipulacji wynikami modelu poprzez staranną inżynierię promptów. Takie podejście wymaga głębokiego zrozumienia, jak tworzyć prompty, które kierują AI w pożądanym kierunku bez bezpośredniego instruowania. Jest to podobne do posiadania bezpośredniego dostępu do “surowych” warstw bazowego AI, bez żadnych warstw pośrednich interpretujących lub kierujących odpowiedziami modelu (stąd nazwa).



Rycina 3. Testowanie modelu podstawowego przy użyciu fragmentu klasycznego skeczu Abbotta i Costello 'Who's on First'

Wyzwanie związane z modelami podstawowymi polega na ich tendencji do wpadania w powtarzające się wzorce lub generowania losowych odpowiedzi. Jednak dzięki starannej inżynierii promptów i dostosowaniu parametrów, takich jak kary za powtórzenia, można nakłonić modele podstawowe do generowania unikalnych i kreatywnych treści. Proces ten nie jest pozbawiony kompromisów; podczas gdy modele podstawowe oferują niezrównaną elastyczność w zakresie innowacji, wymagają one wyższego poziomu wiedzy specjalistycznej.



Rycina 4. Dla porównania, to samo niejednoznaczne polecenie wprowadzone do GPT-4

Modele Dostrojone Instrukcyjnie: Kierowane Doświadczenie

Modele dostrojone instrukcyjnie są zaprojektowane do rozumienia i wykonywania konkretnych instrukcji, co czyni je bardziej przyjaznymi dla użytkownika i dostępnymi dla szerszego zakresu zastosowań. Rozumieją one mechanikę *konwersacji* i wiedzą, że powinny przestać generować tekst, gdy następuje *koniec ich tury w rozmowie*. Dla wielu programistów, szczególnie tych pracujących nad prostymi aplikacjami, modele dostrojone instrukcyjnie oferują wygodne i efektywne rozwiązanie.

Proces dostrajania instrukcyjnego polega na trenowaniu modelu na dużym zbiorze instrukcji i odpowiedzi generowanych przez ludzi. Godnym uwagi przykładem jest otwarty zbiór danych databricks-dolly-15k dataset, który zawiera ponad 15 000 par prompt/odpowiedź stworzonych przez pracowników Databricks, które można samodzielnie przeanalizować. Zbiór danych obejmuje osiem różnych kategorii

instrukcji, w tym twórcze pisanie, odpowiadanie na pytania zamknięte i otwarte, podsumowywanie, ekstrakcję informacji, klasyfikację i burzę mózgów.

Podczas procesu generowania danych, współtwórcy otrzymali wytyczne dotyczące tworzenia promptów i odpowiedzi dla każdej kategorii. Na przykład, w przypadku zadań związanych z twórczym pisaniem, zostali poinstruowani, aby dostarczyć konkretne ograniczenia, instrukcje lub wymagania kierujące wynikiem modelu. W przypadku odpowiedzi na pytania zamknięte, poproszono ich o napisanie pytań wymagających faktycznie poprawnych odpowiedzi na podstawie danego fragmentu z Wikipedii.

Powstały zbiór danych służy jako cenny zasób do dostrajania dużych modeli językowych, aby wykazywały interaktywne możliwości i zdolność do wykonywania instrukcji, podobnie jak systemy typu ChatGPT. Trenując na różnorodnym zbiorze instrukcji i odpowiedzi generowanych przez ludzi, model uczy się rozumieć i wykonywać konkretne polecenia, stając się bardziej biegłym w obsłudze szerokiego zakresu zadań.

Oprócz bezpośredniego dostrajania, instrukcje zawarte w zbiorach danych takich jak databricks-dolly-15k mogą być również wykorzystywane do generowania danych syntetycznych. Poprzez wykorzystanie promptów stworzonych przez współtwórców jako przykładów few-shot dla dużego otwartego modelu językowego, programiści mogą wygenerować znacznie większy zbiór instrukcji w każdej kategorii. To podejście, opisane w artykule Self-Instruct, umożliwia tworzenie bardziej solidnych modeli wykonujących instrukcje.

Ponadto, instrukcje i odpowiedzi w tych zbiorach danych mogą być wzbogacane poprzez techniki takie jak parafrazowanie. Poprzez przeformułowanie każdego promptu lub krótkiej odpowiedzi i powiązanie powstałego tekstu z odpowiednim przykładem wzorcowym, programiści mogą wprowadzić formę regularyzacji, która zwiększa zdolność modelu do wykonywania instrukcji.

Łatwość użytkowania zapewniana przez modele dostrojone instrukcyjnie wiąże się z pewnym kosztem w postaci ograniczonej elastyczności. Modele te są często mocno

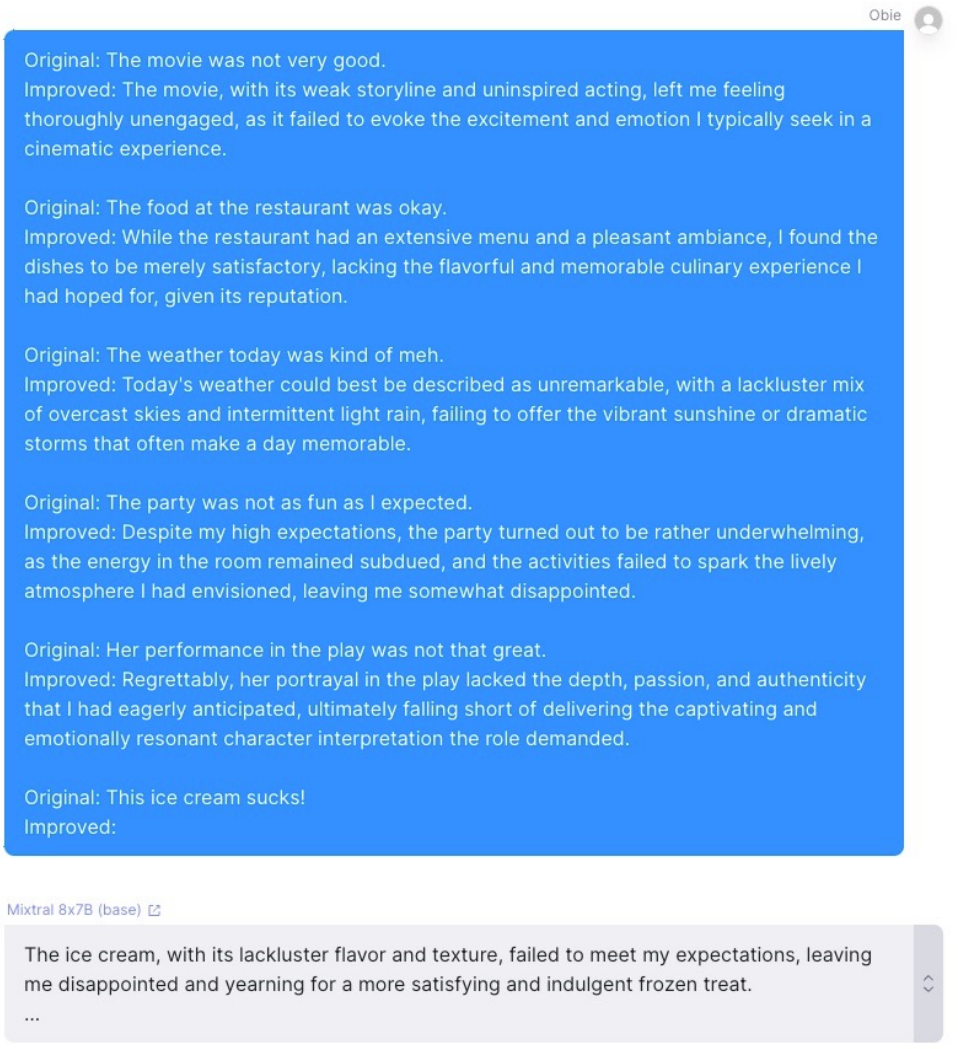
cenzurowane, co oznacza, że nie zawsze zapewniają poziom swobody twórczej wymagany do wykonania określonych zadań. Na ich działanie silnie wpływają uprzedzenia i ograniczenia zawarte w danych użytych do ich dostrajania.

Pomimo tych ograniczeń, modele dostrojone instrukcyjnie stają się coraz popularniejsze ze względu na ich przyjazny dla użytkownika charakter i zdolność do obsługi szerokiego zakresu zadań przy minimalnej inżynierii promptów. Wraz z pojawianiem się kolejnych wysokiej jakości zbiorów danych instrukcyjnych, możemy spodziewać się dalszych ulepszeń w wydajności i wszechstronności tych modeli.

Wybór odpowiedniego rodzaju modelu do twojego projektu

Decyzja między modelami bazowymi (surowymi) a dostrojonymi instrukcyjnie ostatecznie zależy od konkretnych wymagań twojego projektu. W przypadku zadań wymagających wysokiego stopnia kreatywności i oryginalności, modele bazowe oferują potężne narzędzie do innowacji. Modele te pozwalają programistom eksplorować pełny potencjał dużych modeli językowych, przesuwając granice tego, co można osiągnąć dzięki aplikacjom opartym na sztucznej inteligencji, ale wymagają bardziej bezpośredniego podejścia i chęci do eksperymentowania. Temperatura i inne ustawienia mają znacznie większy wpływ w modelach bazowych niż w ich odpowiednikach dostrojonych instrukcyjnie.

Cokolwiek zawrzesz w swoim prompcie, modele bazowe będą próbowały to powtórzyć. Więc jeśli na przykład twój prompt to transkrypt czatu, surowy model będzie próbował kontynuować czat. W zależności od limitu maksymalnej liczby tokenów, nie wygeneruje on tylko następnej wiadomości w czacie, ale może przeprowadzić całą rozmowę sam ze sobą!



Rycina 5. Przykład przepisywania zdań przez Mixtral 8x7B (base) z wykorzystaniem few-shot completion

Podczas przygotowywania powyższego przykładu Przepisywania Zdań autorstwa użytkownika Reddit phree_radical, udało mi się uzyskać użyteczne rezultaty dopiero po wielu eksperymentach z ustawieniami parametrów, ostatecznie ustalając je na: Temperature 0.08, Top P: 0.2, Top K: 1 oraz Repetition Penalty: 1.26.

Próba wykorzystania tego podejścia z modelem bazowym w środowisku produkcyjnym byłaby trudna ze względu na silny wpływ parametru `max_tokens`. Ustawiony zbyt nisko powoduje obcięcie wyników. Ustawiony na wartość większą niż model potrzebuje do pożądanego wyniku, spowoduje, że zacznie on konfabulować dodatkowe przykłady.

Konkluzja jest taka, że o ile nie potrzebujesz pełnej kontroli i braku cenzury, modele dostrojone instrukcyjnie mogą znacząco usprawnić proces rozwoju. Aby to lepiej zobrazować, oto odpowiedź Mixtral 8x7B na to samo polecenie, tym razem w wersji dostrojonej instrukcyjnie:

> Z przykrością informuję, że lody nie spełniają moich oczekiwań, ponieważ brakuje im bogatej, kremowej tekstury i wyśmienitego smaku, które zwykle kojarzę z deserem wysokiej jakości. Liczyłem na bardziej satysfakcjonujące i przyjemne doznania.

Co istotne, mogłem pozostawić ustawienie max tokens na poziomie 500, a model niezawodnie zatrzymał się na końcu pożądanego wyniku bez konfabulowania dodatkowych przykładów.

Inżynieria Promptów

Gdy zaczniesz stosować AI w swoich projektach, szybko odkryjesz, że jedną z najważniejszych umiejętności, które musisz opanować, jest sztuka inżynierii promptów. Ale czym właściwie jest inżynieria promptów i dlaczego jest tak ważna?

W swojej istocie inżynieria promptów to proces projektowania i tworzenia promptów wejściowych, które dostarczasz modelowi językowemu, aby ukierunkować jego wynik. Chodzi o zrozumienie, jak efektywnie komunikować się z AI, używając kombinacji instrukcji, przykładów i kontekstu, aby pokierować model w stronę generowania pożądanej odpowiedzi.

Wyobraź sobie to jak rozmowę z bardzo inteligentnym, ale nieco dosłownie myślącym przyjacielem. Aby wyciągnąć jak najwięcej z tej interakcji, musisz być jasny, konkretny i dostarczyć wystarczająco dużo kontekstu, aby upewnić się, że twój przyjaciel dokładnie rozumie, o co prosisz. To właśnie tu wkracza inżynieria promptów i nawet jeśli na początku wydaje się łatwa, uwierz mi, że wymaga sporo praktyki, aby ją opanować.

Elementy Składowe Skutecznych Promptów

Aby zacząć tworzyć skuteczne prompty, najpierw musisz zrozumieć kluczowe komponenty, które składają się na dobrze skonstruowane dane wejściowe. Oto najważniejsze elementy składowe:

1. **Instrukcje**: Jasne i zwięzłe instrukcje, które mówią modelowi, co ma zrobić. Może to być wszystko, od "Podsumuj następujący artykuł" przez "Wygeneruj wiersz o zachodzie słońca" po "zamień to żądanie zmiany projektu na obiekt JSON".
2. **Kontekst**: Istotne informacje, które pomagają modelowi zrozumieć tło i zakres zadania. Może to obejmować szczegóły dotyczące docelowej grupy odbiorców, pożądanego tonu i stylu lub konkretnych ograniczeń czy wymagań dotyczących danych wyjściowych, takich jak schemat JSON, którego należy przestrzegać.
3. **Przykłady**: Konkretne przykłady, które pokazują rodzaj wyniku, jakiego szukasz. Podając kilka dobrze dobranych przykładów, możesz pomóc modelowi nauczyć się wzorców i charakterystyki pożądanej odpowiedzi.
4. **Formatowanie Wejścia**: Podziały wierszy i formatowanie markdown nadają strukturę naszemu promptowi. Rozdzielenie promptu na akapity pozwala nam pogrupować powiązane instrukcje tak, aby były łatwiejsze do zrozumienia zarówno dla ludzi, jak i dla AI. Punktory i listy numerowane pozwalają nam zdefiniować listy i kolejność elementów. Znaczniki pogrubienia i kursywy pozwalają nam zaznaczyć nacisk.
5. **Formatowanie Wyjścia**: Konkretne instrukcje dotyczące tego, jak powinien być ustrukturyzowany i sformatowany wynik. Mogą one obejmować wytyczne

dotyczące pożądanej długości, użycia nagłówków lub punktów, formatowania markdown lub innych konkretnych szablonów czy konwencji wyjściowych, których należy przestrzegać.

Łącząc te elementy składowe na różne sposoby, możesz tworzyć prompty dostosowane do swoich konkretnych potrzeb i kierować model w stronę generowania wysokiej jakości, trafnych odpowiedzi.

Sztuka i Nauka Projektowania Promptów

Tworzenie skutecznych promptów to zarówno sztuka, jak i nauka. (Dlatego nazywamy to rzemiosłem.) Wymaga głębokiego zrozumienia możliwości i ograniczeń modeli językowych, a także kreatywnego podejścia do projektowania promptów, które wywołują pożądane zachowanie. Kreatywność, która jest w to zaangażowana, sprawia, że jest to dla mnie przynajmniej bardzo satysfakcjonujące zajęcie. Może to również być bardzo frustrujące, szczególnie gdy szukamy zachowania deterministycznego

Jednym z kluczowych aspektów inżynierii promptów jest zrozumienie, jak zrównoważyć szczegółowość i elastyczność. Z jednej strony chcesz zapewnić wystarczające wskazówki, aby pokierować model we właściwym kierunku. Z drugiej strony nie chcesz być tak precyzyjny, aby ograniczyć zdolność modelu do wykorzystania własnej kreatywności i elastyczności w radzeniu sobie z przypadkami brzegowymi.

Kolejnym ważnym aspektem jest wykorzystanie przykładów. Dobrze dobrane przykłady mogą być niezwykle skuteczne w pomocy modelowi zrozumieć rodzaj oczekiwanego rezultatu. Ważne jest jednak, aby używać przykładów rozważnie i upewnić się, że są one reprezentatywne dla pożądanej odpowiedzi. Zły przykład w najlepszym razie jest tylko marnotrawstwem tokenów, a w najgorszym może zniszczyć pożądany wynik.

Techniki i najlepsze praktyki inżynierii promptów

Zagłębiając się w świat inżynierii promptów, odkryjesz szereg technik i najlepszych praktyk, które pomogą Ci tworzyć skuteczniejsze prompty. Oto kilka kluczowych obszarów do zgłębienia:

1. **Uczenie zero-shot vs. few-shot**: Zrozumienie, kiedy używać uczenia *zero-shot* (bez podawania przykładów) w przeciwieństwie do uczenia *one-shot* lub *few-shot* (z podaniem niewielkiej liczby przykładów) może pomóc w tworzeniu bardziej wydajnych i skutecznych promptów.
2. **Iteracyjne udoskonalanie**: Proces iteracyjnego udoskonalania promptów w oparciu o wyniki modelu może pomóc w osiągnięciu optymalnego projektu promptu. Pętla sprzężenia zwrotnego to potężne podejście, które wykorzystuje własne wyniki modelu językowego do stopniowej poprawy jakości i trafności generowanej treści.
3. **Łańcuchowanie promptów**: Łączenie wielu promptów w sekwencję może pomóc w podzieleniu złożonych zadań na mniejsze, łatwiejsze do zarządzania kroki. Łańcuchowanie promptów polega na podzieleniu złożonego zadania lub konwersacji na serię mniejszych, wzajemnie powiązanych promptów. Łącząc prompty w łańcuch, możesz przeprowadzić AI przez wieloetapowy proces, zachowując kontekst i spójność podczas całej interakcji.
4. **Dostrajanie promptów**: Dostosowywanie promptów do konkretnych dziedzin lub zadań może pomóc w tworzeniu bardziej wyspecjalizowanych i skutecznych promptów. Szablon promptu pomaga tworzyć elastyczne, wielokrotnego użytku i łatwe w utrzymaniu struktury promptów, które są łatwiej adaptowalne do danego zadania.

Nauka tego, kiedy używać uczenia zero-shot, one-shot lub few-shot, jest szczególnie ważną częścią opanowania inżynierii promptów. Każde podejście ma swoje mocne

i słabe strony, a zrozumienie, kiedy którego użyć, może pomóc w tworzeniu skuteczniejszych i wydajniejszych promptów.

Uczenie Zero-Shot: Kiedy przykłady nie są potrzebne

Uczenie zero-shot odnosi się do zdolności modelu językowego do wykonywania zadań bez przykładów czy bezpośredniego treningu. Innymi słowy, przekazujesz modelowi prompt opisujący zadanie, a model generuje odpowiedź wyłącznie na podstawie swojej istniejącej wiedzy i rozumienia języka.

Uczenie zero-shot jest szczególnie przydatne, gdy:

1. Zadanie jest stosunkowo proste i jednoznaczne, a model prawdopodobnie napotkał podobne zadania podczas wstępnego trenowania.
2. Chcesz przetestować wrodzone możliwości modelu i zobaczyć, jak reaguje na nowe zadanie bez dodatkowych wskazówek.
3. Pracujesz z dużym i wszechstronnym modelem językowym, który został wytrenowany na szerokiej gamie zadań i dziedzin.

Jednak uczenie zero-shot może być również nieprzewidywalne i nie zawsze prowadzić do pożądanych rezultatów. Na odpowiedź modelu mogą wpływać uprzedzenia lub niespójności w danych treningowych, a model może mieć trudności z bardziej złożonymi lub niejednoznacznymi zadaniami.

> Widziałem prompty zero-shot, które działały świetnie dla 80% moich przypadków testowych, ale dawały całkowicie błędne lub niezrozumiałe wyniki dla pozostałych 20%. Niezwykle ważne jest wdrożenie dokładnego reżimu testowego, szczególnie jeśli w dużym stopniu polegasz na promptowaniu zero-shot.

Uczenie One-Shot: Kiedy Jeden Przykład Może Zrobić Różnicę

Uczenie one-shot polega na dostarczeniu modelowi pojedynczego przykładu pożądanego rezultatu wraz z opisem zadania. Ten przykład służy jako szablon lub wzorzec, którego model może użyć do wygenerowania własnej odpowiedzi.

Uczenie one-shot może być skuteczne, gdy:

1. Zadanie jest stosunkowo nowe lub specyficzne, a model mógł nie napotkać wielu podobnych przykładów podczas wstępnego trenowania.
2. Chcesz zapewnić jasną i zwięzłą demonstrację pożądanego formatu lub stylu wyniku.
3. Zadanie wymaga określonej struktury lub konwencji, która może nie być oczywista na podstawie samego opisu zadania.

Opisy, które są oczywiste dla ciebie, niekoniecznie muszą być oczywiste dla AI. Przykłady one-shot mogą pomóc w wyjaśnieniu sytuacji.

Uczenie one-shot może pomóc modelowi jaśniej zrozumieć oczekiwania i wygenerować odpowiedź, która jest bardziej zgodna z dostarczonym przykładem. Jednak ważne jest, aby starannie wybierać przykład i upewnić się, że jest on reprezentatywny dla pożądanego rezultatu. Wybierając przykład, zastanów się nad potencjalnymi przypadkami brzegowymi i zakresem danych wejściowych, które prompt będzie obsługiwał.

Rycina 6. Pojedynczy przykład oczekiwanego formatu JSON

```
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"
}
```

Uczenie Few-Shot: Kiedy Wiele Przykładów Może Poprawić Wydajność

Uczenie few-shot polega na dostarczeniu modelowi małej liczby przykładów (zazwyczaj od 2 do 10) wraz z opisem zadania. Te przykłady służą do zapewnienia modelowi szerszego kontekstu i różnorodności, pomagając mu generować bardziej zróżnicowane i dokładne odpowiedzi.

Uczenie few-shot jest szczególnie przydatne, gdy:

1. Zadanie jest złożone lub niuansowe, a jeden przykład może nie wystarczyć do uchwycenia wszystkich istotnych aspektów.
2. Chcesz dostarczyć modelowi szereg przykładów pokazujących różne warianty lub przypadki brzegowe.

3. Zadanie wymaga, aby model generował odpowiedzi zgodne z określoną dziedziną lub stylem.

Dostarczając wiele przykładów, możesz pomóc modelowi rozwinąć bardziej kompleksowe zrozumienie zadania i generować odpowiedzi, które są bardziej spójne i niezawodne.

Przykład: Polecenia Mogą Być Znacznie Bardziej Złożone Niż Ci Się Wydaje

Dzisiejsze duże modele językowe są znacznie potężniejsze i zdolne do rozumowania niż mogłoby się wydawać. Nie ograniczaj się więc do myślenia o poleceniach jako o prostej specyfikacji par wejścia i wyjścia. Możesz eksperymentować z długimi i złożonymi instrukcjami w sposób przypominający interakcję z człowiekiem.

Na przykład, oto polecenie, którego użyłem w Olympii podczas prototypowania naszej integracji z usługami Google, które w całości stanowią prawdopodobnie jedno z największych API na świecie. Moje wcześniejsze eksperymenty wykazały, że GPT-4 ma przyzwoitą wiedzę o API Google, a ja nie miałem czasu ani motywacji do pisania szczegółowej warstwy mapowania, implementując każdą funkcję, którą chciałem udostępnić mojej sztucznej inteligencji, pojedynczo. Co by było, gdybym mógł po prostu dać AI dostęp do *całego* API Google?

Rozpocząłem moje polecenie od poinformowania AI, że ma bezpośredni dostęp do punktów końcowych API Google poprzez HTTP i że jego rolą jest korzystanie z aplikacji i usług Google w imieniu użytkownika. Następnie dostarczyłem wytyczne, zasady związane z parametrem `fields`, ponieważ wydawało się, że ma z nim najwięcej problemów, oraz pewne wskazówki specyficzne dla API (uczenie few-shot w działaniu).

Oto całe polecenie, które informuje AI, jak korzystać z dostarczonej funkcji `invoke_google_api`.

```
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
```

Być może zastanawiasz się, czy ten prompt działa. Prosta odpowiedź brzmi: tak. SI nie zawsze wiedziała, jak idealnie wywołać API za pierwszym razem. Jednak jeśli popełniła błąd, po prostu przekazywałem z powrotem komunikaty o błędach jako wynik wywołania. Mając świadomość swojego błędu, SI mogła przeanalizować swoją pomyłkę i spróbować ponownie. W większości przypadków udawało się to poprawnie po kilku próbach.

Oczywiście, duże struktury JSON, które API Google zwraca jako payloady podczas korzystania z tego prompta, są wysoce nieefektywne, więc *nie* zalecam stosowania tego podejścia w środowisku produkcyjnym. Jednak sam fakt, że to podejście w ogóle zadziałało, świadczy o tym, jak potężna może być inżynieria promptów.

Eksperymentowanie i Iteracja

Ostatecznie sposób, w jaki konstruujesz prompt, zależy od konkretnego zadania, złożoności oczekiwanego rezultatu i możliwości modelu językowego, z którym pracujesz.

Jako inżynier promptów, ważne jest eksperymentowanie z różnymi podejściami i iterowanie na podstawie wyników. Zacznij od uczenia zero-shot i sprawdź, jak model sobie radzi. Jeśli wyniki są niespójne lub niezadowalające, spróbuj dostarczyć jeden lub więcej przykładów i zobacz, czy wydajność się poprawi.

Pamiętaj, że nawet w ramach każdego podejścia jest miejsce na warianty i optymalizację. Możesz eksperymentować z różnymi przykładami, dostosowywać sformułowanie opisu zadania lub dostarczać dodatkowy kontekst, aby pomóc w ukierunkowaniu odpowiedzi modelu.

Z czasem rozwiniesz intuicję co do tego, które podejście prawdopodobnie zadziała najlepiej w danym zadaniu, i będziesz w stanie tworzyć prompty, które są bardziej efektywne i wydajne. Kluczem jest zachowanie ciekawości, eksperymentalnego podejścia i iteracyjności w swojej pracy nad inżynierią promptów.

W tej książce zagłębimy się bardziej w te techniki i zbadamy, jak można je zastosować w rzeczywistych scenariuszach. Opanowując sztukę i naukę inżynierii promptów, będziesz dobrze przygotowany do uwolnienia pełnego potencjału tworzenia aplikacji opartych na SI.

Sztuka Niedookreśloności

Jeśli chodzi o tworzenie efektywnych promptów dla dużych modeli językowych (LLM), powszechnym założeniem jest, że większa szczegółowość i bardziej szczegółowe instrukcje prowadzą do lepszych rezultatów. Jednak praktyczne doświadczenie pokazuje, że nie zawsze tak jest. W rzeczywistości, celowa niedookreśloność

w promptach często może przynieść lepsze rezultaty, wykorzystując niezwykłą zdolność LLM do uogólniania i wnioskowania.

Ken, założyciel startupu, który przetworzył ponad 500 milionów tokenów GPT, podzielił się cennymi spostrzeżeniami ze swojego doświadczenia. Jedną z kluczowych lekcji, jakie wyciągnął, było to, że w przypadku promptów "mniej znaczy więcej". Zamiast dokładnych list czy nadmiernie szczegółowych instrukcji, Ken odkrył, że pozwolenie LLM na poleganie na swojej wiedzy bazowej często przynosiło lepsze rezultaty.

To odkrycie wywraca do góry nogami tradycyjne podejście do jawnego kodowania, gdzie wszystko musi być drobiazgowo wyjaśnione. W przypadku LLM-ów ważne jest, aby zdawać sobie sprawę, że posiadają one ogromną ilość wiedzy i potrafią tworzyć inteligentne połączenia oraz wnioski. Będąc bardziej ogólnym w swoich promptach, dajesz LLM swobodę wykorzystania jego zrozumienia i wymyślenia rozwiązań, których możesz nie określić wprost.

Na przykład, gdy zespół Kena pracował nad potokiem do klasyfikacji tekstu pod kątem powiązań z jednym z 50 stanów USA lub rządem federalnym, ich początkowe podejście polegało na dostarczeniu *pełnej* szczegółowej listy stanów i ich odpowiednich identyfikatorów w formie tablicy w formacie JSON.

```
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} ... ]
```

Podejście okazało się na tyle nieudane, że musieli głębiej zagłębić się w prompt, aby odkryć, jak go ulepszyć. Podczas tego procesu zauważyli, że choć LLM często błędnie określał id, konsekwentnie zwracał pełną nazwę właściwego stanu w polu `name`, *mimo że nie prosili o to wprost.*

Usuwając identyfikatory lokalizacji i upraszczając prompt do czegoś w rodzaju: "Oczywiście znasz 50 stanów, GPT, więc po prostu podaj mi pełną nazwę stanu, którego to dotyczy, albo Federal jeśli dotyczy to rządu USA", osiągnęli lepsze rezultaty.

To doświadczenie podkreśla moc wykorzystania zdolności generalizacyjnych LLM i pozwolenia mu na wyciąganie wniosków na podstawie posiadanej wiedzy.

> Uzasadnienie Kena dla tego konkretnego podejścia do klasyfikacji, w przeciwieństwie do bardziej tradycyjnej techniki programowania, rzuca światło na sposób myślenia tych z nas, którzy dostrzegli potencjał technologii LLM: "To nie jest trudne zadanie – prawdopodobnie moglibyśmy użyć string/regex, ale jest wystarczająco dużo dziwnych przypadków brzegowych, że zajęłoby to więcej czasu."

Zdolność LLM do poprawy jakości i generalizacji przy bardziej ogólnych promptach jest niezwykłą cechą myślenia wyższego rzędu i delegowania zadań. Pokazuje to, że LLM potrafią radzić sobie z niejednoznacznością i podejmować inteligentne decyzje na podstawie dostarczonego kontekstu.

Należy jednak pamiętać, że bycie ogólnym nie oznacza bycia niejasnym czy dwuznacznym. Kluczowe jest zapewnienie wystarczającego kontekstu i wskazówek, aby pokierować LLM we właściwym kierunku, jednocześnie dając mu elastyczność w wykorzystaniu swojej wiedzy i zdolności generalizacji.

Dlatego projektując prompty, warto wziąć pod uwagę następujące wskazówki typu "mniej znaczy więcej":

1. Skup się na pożądanym rezultacie zamiast określać każdy szczegół procesu.
2. Zapewnij odpowiedni kontekst i ograniczenia, ale unikaj nadmiernej specyfikacji.
3. Wykorzystuj istniejącą wiedzę, odwołując się do powszechnych pojęć lub bytów.
4. Pozostaw przestrzeń na wnioskowanie i tworzenie połączeń na podstawie danego kontekstu.
5. Iteruj i udoskonalaj swoje prompty na podstawie odpowiedzi LLM, znajdując właściwą równowagę między szczegółowością a ogólnością.

Przyjmując sztukę niedookreśloności w inżynierii promptów, możesz uwolnić pełny potencjał LLM i osiągać lepsze rezultaty. Zaufaj zdolności LLM do generalizacji i podejmowania inteligentnych decyzji, a możesz być zaskoczony jakością i kreatywnością otrzymywanych wyników. Zwracaj uwagę na to, jak różne modele reagują na różne poziomy szczegółowości w twoich promptach i odpowiednio je dostosowuj. Dzięki praktyce i doświadczeniu rozwiniesz wyczucie tego, kiedy być bardziej ogólnym, a kiedy dostarczać dodatkowych wskazówek, co pozwoli ci skutecznie wykorzystywać moc LLM w twoich aplikacjach.

Dlaczego antropomorfizm dominuje w inżynierii promptów

Antropomorfizm, czyli przypisywanie ludzkich cech bytom nie-ludzkim, jest dominującym podejściem w inżynierii promptów dla dużych modeli językowych z celowych powodów. Jest to świadomy wybór projektowy, który sprawia, że interakcja z zaawansowanymi systemami AI staje się bardziej intuicyjna i dostępna dla szerokiego grona użytkowników (włączając nas, programistów aplikacji).

Antropomorfizacja LLM zapewnia framework, który jest natychmiast intuicyjny dla osób całkowicie nieznających technicznych zawiłości systemu. Jak przekonasz się, próbując użyć modelu niedostrojonego instrukcyjnie do wykonania czegokolwiek użytecznego, skonstruowanie ram, w których oczekiwana kontynuacja przynosi wartość, jest trudnym zadaniem. Wymaga to dość głębokiego zrozumienia wewnętrznego działania systemu, którym dysponuje stosunkowo niewielka liczba ekspertów.

Traktując interakcję z modelem językowym jak rozmowę między dwiema osobami, możemy polegać na naszym wrodzonym rozumieniu komunikacji międzyludzkiej, aby przekazać nasze potrzeby i oczekiwania. Podobnie jak wczesne projektowanie interfejsu użytkownika Macintosha przedkładało natychmiastową intuicyjność nad wyrafinowanie, antropomorficzne ujęcie AI pozwala nam na zaangażowanie w sposób,

który wydaje się naturalny i znajomy.

Kiedy komunikujemy się z inną osobą, instynktownie zwracamy się do niej bezpośrednio używając “ty” i przedstawiamy jasne wskazówki dotyczące tego, jak oczekujemy, że będzie się zachowywać. To płynnie przekłada się na proces inżynierii promptów, gdzie kierujemy zachowaniem AI poprzez określanie promptów systemowych i angażowanie się w dialog tam i z powrotem.

Ujmując interakcję w ten sposób, możemy łatwo uchwycić koncepcję przekazywania instrukcji AI i otrzymywania odpowiednich odpowiedzi. Podejście antropomorficzne zmniejsza obciążenie poznawcze i pozwala nam skupić się na zadaniu, zamiast zmagać się z technicznymi zawiłościami systemu.

Ważne jest, aby zauważyć, że choć antropomorfizm jest potężnym narzędziem czyniącym systemy AI bardziej dostępnymi, wiąże się również z pewnymi ryzykami i ograniczeniami. Nasz użytkownik może rozwinąć nierealistyczne oczekiwania lub stworzyć niezdrowe przywiązanie emocjonalne do naszych systemów. Jako inżynierowie promptów i programiści, kluczowe jest znalezienie równowagi między wykorzystywaniem korzyści płynących z antropomorfizmu a zapewnieniem, że użytkownicy zachowują jasne zrozumienie możliwości i ograniczeń AI.

Wraz z rozwojem dziedziny inżynierii promptów możemy spodziewać się dalszych udoskonaleń i innowacji w sposobie interakcji z dużymi modelami językowymi. Jednak antropomorfizm jako środek zapewniający intuicyjne i dostępne doświadczenie programisty i użytkownika prawdopodobnie pozostanie fundamentalną zasadą w projektowaniu tych systemów.

Oddzielanie Instrukcji od Danych: Kluczowa Zasada

Istotne jest zrozumienie fundamentalnej zasady, która stanowi podstawę bezpieczeństwa i niezawodności tych systemów: oddzielenia instrukcji od danych.

W tradycyjnej informatyce, wyraźne rozróżnienie między pasywnymi danymi a aktywnymi instrukcjami jest podstawową zasadą bezpieczeństwa. To rozdzielenie

pomaga zapobiec niezamierzonemu lub złośliwemu wykonaniu kodu, które mogłoby zagrozić integralności i stabilności systemu. Jednak dzisiejsze modele LLM, które zostały głównie opracowane jako modele wykonujące instrukcje, takie jak chatboty, często nie posiadają tego formalnego i metodycznego rozdzielenia.

Z punktu widzenia modeli LLM, instrukcje mogą pojawić się w dowolnym miejscu w danych wejściowych, czy to w prompcie systemowym, czy w prompcie dostarczonym przez użytkownika. Ten brak rozdzielenia może prowadzić do potencjalnych luk w zabezpieczeniach i niepożądanego zachowania, podobnie jak w przypadku problemów, z którymi borykają się bazy danych narażone na iniekcje SQL lub systemy operacyjne bez odpowiedniej ochrony pamięci.

Pracując z modelami LLM, kluczowe jest, aby być świadomym tego ograniczenia i podejmować kroki w celu złagodzenia ryzyka. Jednym z podejść jest staranne konstruowanie promptów i danych wejściowych, aby wyraźnie rozróżnić między instrukcjami a danymi. Typowe metody zapewniania jednoznacznych wskazówek dotyczących tego, co stanowi instrukcję, a co powinno być traktowane jako pasywne dane, obejmują znacznikowanie typu markup. Twój prompt może pomóc modelowi LLM lepiej zrozumieć i respektować to rozdzielenie.

Rycina 7. Używanie XML do rozróżnienia między instrukcjami, materiałem źródłowym i promptem użytkownika

```
<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>
```

Inną techniką jest wdrożenie dodatkowych warstw walidacji i sanityzacji danych wejściowych dostarczanych do LLM. Poprzez filtrowanie lub eskejpowanie potencjalnych instrukcji czy fragmentów kodu, które mogą być osadzone w danych, można zmniejszyć ryzyko niezamierzonego wykonania. Wzorce takie jak Łańcuchowanie Promptów są przydatne w tym celu.

Co więcej, podczas projektowania architektury aplikacji, warto rozważyć wprowadzenie mechanizmów wymuszających rozdzielenie instrukcji i danych na wyższym poziomie. Może to obejmować wykorzystanie osobnych punktów końcowych lub interfejsów API do obsługi instrukcji i danych, wdrożenie ścisłej walidacji i analizy danych wejściowych oraz zastosowanie *zasady najmniejszych uprawnień* w celu ograniczenia zakresu tego, do czego LLM może uzyskać dostęp i co może wykonać.

Zasada Najmniejszych Uprawnień

Przyjęcie zasady najmniejszych uprawnień przypomina organizację ekskluzywnego przyjęcia, gdzie goście otrzymują dostęp tylko do tych pomieszczeń, które są im absolutnie niezbędne. Wyobraź sobie, że organizujesz takie przyjęcie w rozległej posiadłości. Nie każdy musi mieć dostęp do piwniczki z winami czy głównej sypialni, prawda? Stosując tę zasadę, właściwie rozdajesz klucze, które otwierają tylko konkretne drzwi, zapewniając, że każdy gość - lub w naszym przypadku każdy komponent aplikacji LLM - ma tylko taki dostęp, jaki jest niezbędny do wypełnienia swojej roli.

Nie chodzi tu tylko o skąpienie kluczy - chodzi o uznanie faktu, że w świecie, gdzie zagrożenia mogą pochodzić zewsząd, mądrym posunięciem jest ograniczenie pola do zabawy. Jeśli ktoś nieproszony faktycznie pojawi się na przyjęciu, znajdzie

> się zamknięty, że tak powiem, w przedsionku, co znacznie ograniczy możliwości wyrządzenia szkód. Dlatego zabezpieczając aplikacje LLM, pamiętaj: wydawaj klucze tylko do pomieszczeń, które są niezbędne, a resztę posiadłości utrzymuj bezpieczną. To nie tylko kwestia dobrego wychowania - to dobra praktyka bezpieczeństwa.

Choć obecny stan LLM może nie posiadać formalnego rozdzielenia instrukcji i danych, istotne jest, abyś jako programista był świadomy tego ograniczenia i podejmował proaktywne działania w celu ograniczenia ryzyka. Stosując najlepsze praktyki z informatyki i dostosowując je do unikalnych cech LLM, możesz tworzyć bezpieczniejsze i bardziej niezawodne aplikacje, które wykorzystują moc tych modeli, zachowując jednocześnie integralność systemu.

Destylacja Promptów

Tworzenie idealnego prompta jest często trudnym i czasochłonnym zadaniem, wymagającym głębokiego zrozumienia docelowej dziedziny i niuansów modeli językowych. W tym miejscu wkracza technika “Destylacji Promptów”, oferując potężne podejście do inżynierii promptów, które wykorzystuje możliwości dużych modeli językowych (LLM) do usprawnienia i optymalizacji procesu.

Destylacja Promptów to wieloetapowa technika, która polega na wykorzystaniu LLM do pomocy w tworzeniu, udoskonalaniu i optymalizacji promptów. Zamiast polegać wyłącznie na ludzkiej wiedzy i intuicji, podejście to wykorzystuje wiedzę i możliwości generatywne LLM do wspólnego tworzenia wysokiej jakości promptów.

Poprzez zaangażowanie w iteracyjny proces generowania, udoskonalania i integracji, Destylacja Promptów umożliwia tworzenie promptów, które są bardziej spójne, kompleksowe i dostosowane do pożądanego zadania lub wyniku. Warto zauważyć, że proces destylacji może być wykonywany ręcznie w jednym z wielu “środowisk

testowych" dostarczanych przez głównych dostawców AI, takich jak OpenAI czy Anthropic, lub może być zautomatyzowany jako część kodu aplikacji, w zależności od przypadku użycia.

Jak to działa

Destylacja Promptów zazwyczaj obejmuje następujące kroki:

1. **Identyfikacja Głównej Intencji**: Analiza prompta w celu określenia jego podstawowego celu i pożądanego rezultatu. Usunięcie wszelkich zbędnych informacji i skupienie się na głównej intencji prompta.
2. **Eliminacja Niejednoznaczności**: Przegląd prompta pod kątem niejasnego lub nieprecyzyjnego języka. Wyjaśnienie znaczenia i dostarczenie konkretnych szczegółów, które pokierują SI w stronę generowania dokładnych i trafnych odpowiedzi.
3. **Uproszczenie Języka**: Uproszczenie prompta poprzez użycie jasnego i zwięzłego języka. Unikanie złożonych struktur zdań, żargonu lub niepotrzebnych szczegółów, które mogą dezorientować SI lub wprowadzać szum.
4. **Dostarczenie Odpowiedniego Kontekstu**: Uwzględnienie tylko najbardziej istotnych informacji kontekstowych potrzebnych SI do skutecznego zrozumienia i przetworzenia prompta. Unikanie włączania nieistotnych lub redundantnych szczegółów, które mogą odwracać uwagę od głównej intencji.
5. **Iteracja i Udoskonalanie**: Ciągłe iterowanie i udoskonalanie prompta na podstawie odpowiedzi i informacji zwrotnych od SI. Ocena wygenerowanych wyników i wprowadzanie niezbędnych korekt w celu poprawy jasności i skuteczności prompta. Opcjonalnie wersjonowanie promptów w bazie danych przy użyciu Obiektu Prompt w celu śledzenia iteracji i zapewnienia sobie możliwości łatwego wycofywania zmian w czasie wykonywania.

Generowanie początkowego prompta

Pierwszym krokiem w Destylacji promptów jest dostarczenie modelowi LLM wysokopoziomowego opisu lub intencji dotyczącej pożądanego zadania lub rezultatu. Może to być krótkie stwierdzenie lub zestaw słów kluczowych, które oddają istotę tego, co prompt ma osiągnąć. Model LLM generuje następnie początkowy prompt na podstawie tego wejścia, włączając odpowiednie instrukcje, przykłady lub kontekst, które uzna za stosowne.

Spróbujmy przekazać Claude 3 Opus następujący początkowy prompt:

```
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 odpowiada następującymi danymi wyjściowymi:

```
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.
```

Udoskonalanie Promptu

Po wygenerowaniu początkowego promptu, należy go przeanalizować, a następnie poprosić LLM o rozważenie dodatkowego kontekstu, ograniczeń lub informacji zwrotnej. Pozwól LLM-owi wygenerować udoskonaloną wersję promptu, biorąc pod uwagę dostarczone dane wejściowe i jego własne zrozumienie zadania. Ten proces udoskonalania może być powtarzany iteracyjnie, gdzie każda iteracja dalej dopracowuje prompt w oparciu o współpracę między użytkownikiem a LLM.

Oto moja prośba o udoskonalenie, kontynuująca poprzedni proces destylacji.

```
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.
```

A odpowiedź Claude'a:

```
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.
```

Kompresja promptów

Jak zapewne zauważyłeś, modele językowe (LLM) mają tendencję do rozwlekłości, chyba że poprosisz je o zwięzłość. Prompt, nad którym pracowaliśmy w poprzednich sekcjach, jest dość długi, więc poprośmy Claude'a o jego skrócenie:

```
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 odpowiada, z następującym drastycznie krótszym poleceniem:

```
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.
```

Dyrektywa Systemowa i Integracja Kontekstu

Oprócz udoskonalania samego prompta, LLM może również generować odpowiednie dyrektywy systemowe lub informacje kontekstowe, które pokierują końcowym wynikiem. Podczas tworzenia rutyn AI do inżynierii promptów, które zostaną zintegrowane z kodem aplikacji, na tym etapie destylacji z pewnością skupisz się na ograniczeniach wyjściowych, ale możesz też pracować nad pożądanym tonem, stylem, formatem lub innymi istotnymi parametrami wpływającymi na generowaną odpowiedź.

Końcowy Montaż Prompta

Kulminacją procesu Destylacji Promptów jest montaż końcowego prompta. Obejmuje to połączenie udoskonalonego prompta, wygenerowanych dyrektyw systemowych i zintegrowanego kontekstu w spójny i kompleksowy kod, gotowy do wykorzystania w generowaniu pożądanego wyniku.

Możesz ponownie eksperymentować z kompresją promptów na etapie końcowego montażu, prosząc LLM o skrócenie sformułowań prompta do najkrótszej możliwej sekwencji tokenów, zachowując jednocześnie istotę jego działania. To z pewnością metoda prób i błędów, ale szczególnie w przypadku promptów, które będą uruchamiane na dużą skalę, zyski w wydajności mogą przynieść znaczne oszczędności w zużyciu tokenów.

Kluczowe Korzyści

Wykorzystując wiedzę i możliwości generatywne LLM do udoskonalania promptów, otrzymujesz prompty, które z większym prawdopodobieństwem będą dobrze ustrukturyzowane, informatywne i dostosowane do konkretnego zadania. Iteracyjny proces udoskonalania pomaga zapewnić wysoką jakość promptów i skuteczne uchwycenie zamierzonego celu. Inne korzyści obejmują:

Efektywność i Szybkość: Destylacja Promptów usprawnia proces inżynierii promptów poprzez automatyzację pewnych aspektów tworzenia i udoskonalania promptów. Współpracujący charakter tej techniki pozwala na szybsze osiągnięcie skutecznego prompta, zmniejszając czas i wysiłek wymagany przy ręcznym tworzeniu promptów.

Spójność i Skalowalność: Wykorzystanie LLM w procesie inżynierii promptów pomaga utrzymać spójność między promptami, ponieważ LLM mogą uczyć się i stosować najlepsze praktyki i wzorce z wcześniejszych udanych promptów. Ta spójność, w połączeniu z możliwością generowania promptów na dużą skalę, czyni Destylację Promptów cenną techniką dla aplikacji wykorzystujących AI na dużą skalę.

Pomysł na Projekt: Narzędzia na poziomie biblioteki, które upraszczają proces wersjonowania i oceny promptów w systemach, które wykonują automatyczne destylacje promptów jako część kodu aplikacji.

Aby wdrożyć Destylację Promptów, programiści mogą zaprojektować przepływ pracy lub potok, który integruje LLM na różnych etapach procesu inżynierii promptów. Można to osiągnąć poprzez wywołania API, własne narzędzia lub zintegrowane środowiska programistyczne, które ułatwiają płynną interakcję między użytkownikami a LLM podczas tworzenia promptów. Szczegóły implementacji mogą się różnić w zależności od wybranej platformy LLM i wymagań aplikacji.

A co z fine-tuningiem?

W tej książce szczegółowo omawiamy inżynierię promptów i RAG, ale nie fine-tuning. Głównym powodem tej decyzji jest to, że moim zdaniem większość programistów aplikacji nie potrzebuje fine-tuningu do swoich potrzeb integracji AI.

Inżynieria promptów, która polega na starannym tworzeniu promptów z przykładami zero-shot i few-shot, ograniczeniami i instrukcjami, może skutecznie kierować modelem w generowaniu trafnych i dokładnych odpowiedzi dla szerokiego zakresu zadań. Dostarczając jasny kontekst i zawężając ścieżkę poprzez dobrze zaprojektowane prompty, możesz wykorzystać rozległą wiedzę dużych modeli językowych bez potrzeby fine-tuningu.

Podobnie, Generowanie Wspomagane Wyszukiwaniem (RAG) oferuje skuteczne podejście do integracji AI w aplikacjach. Poprzez dynamiczne pobieranie istotnych informacji z zewnętrznych baz wiedzy lub dokumentów, RAG dostarcza modelowi skoncentrowany kontekst w momencie promptowania. Pozwala to modelowi generować odpowiedzi, które są bardziej dokładne, aktualne i specyficzne dla danej dziedziny, bez wymagania czasochłonnego i zasobochłonnego procesu fine-tuningu.

Podczas gdy fine-tuning może być korzystny dla wysoce wyspecjalizowanych dziedzin lub zadań wymagających głębokiego poziomu dostosowania, często wiąże się ze znacznymi kosztami obliczeniowymi, wymaganiami dotyczącymi danych i narzutem na utrzymanie. W większości scenariuszy tworzenia aplikacji, połączenie skutecznej inżynierii promptów i RAG powinno wystarczyć do osiągnięcia pożądanej funkcjonalności i doświadczenia użytkownika opartego na AI.

Retrieval Augmented Generation (RAG)

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Czym jest Retrieval Augmented Generation?

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak działa RAG?

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Dlaczego warto używać RAG w swoich aplikacjach?

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Implementacja RAG w Twojej Aplikacji

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przygotowanie Źródeł Wiedzy (Fragmentacja)

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Segmentacja na propozycje

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Uwagi dotyczące implementacji

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kontrola Jakości

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Korzyści z Wyszukiwania Opartego na Propozycjach

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykłady RAG w Rzeczywistych Zastosowaniach

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Studium Przypadku: RAG w Aplikacji do Przygotowywania Zeznań Podatkowych Bez Osadzeń

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Inteligentna Optymalizacja Zapytań (IQO)

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Ponowne rankingowanie

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Ocena RAG (RAGAs)

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Wierność

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Trafność Odpowiedzi

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Precyzja kontekstowa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Trafność kontekstowa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Pełność kontekstowa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Pełność jednostek kontekstowych

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Semantyczne podobieństwo odpowiedzi (ANSS)

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Poprawność Odpowiedzi

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Krytyka Aspektowa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Wyzwania i Perspektywy na Przyszłość

Dzielenie semantyczne: Usprawnienie wyszukiwania dzięki segmentacji kontekstowej

Indeksowanie hierarchiczne: Strukturyzacja danych dla ulepszonego wyszukiwania

Self-RAG: Samodoskonalące Rozszerzenie

HyDE: Hipotetyczne Osadzanie Dokumentów

Czym jest uczenie kontrastywne?

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Mnogość Pracowników

Lubię myśleć o moich komponentach AI jako o małych, niemal ludzkich wirtualnych „pracownikach“, których można bezproblemowo zintegrować z logiką aplikacji, aby wykonywali określone zadania lub podejmowali złożone decyzje. Chodzi o celowe uczłowieczenie możliwości LLM, tak aby nikt nie ekscytował się *zbytnio* i nie przypisywał im magicznych właściwości, których nie posiadają.

Zamiast polegać wyłącznie na skomplikowanych algorytmach lub czasochłonnych ręcznych implementacjach, programiści mogą postrzegać komponenty AI jako inteligentne, oddane, przypominające ludzi byty, które można przywołać w razie potrzeby do rozwiązywania złożonych problemów i dostarczania rozwiązań opartych na ich treningu i wiedzy. Te byty nie rozpraszają się ani nie biorą zwolnienia chorobowego. Nie decydują spontanicznie o wykonywaniu rzeczy w inny sposób niż ten, w jaki zostały poinstruowane, i ogólnie rzecz biorąc, jeśli są poprawnie zaprogramowane, nie popełniają też błędów.

W kategoriach technicznych, kluczową zasadą tego podejścia jest rozkładanie złożonych zadań lub procesów decyzyjnych na mniejsze, łatwiejsze w zarządzaniu jednostki, które mogą być obsługiwane przez wyspecjalizowanych pracowników AI. Każdy pracownik jest zaprojektowany tak, aby skupiać się na konkretnym aspekcie problemu, wnosząc swoją unikalną wiedzę i możliwości. Poprzez rozdzielenie obciążenia pracy między wielu pracowników AI, aplikacja może osiągnąć większą wydajność, skalowalność i adaptowalność.

Na przykład, rozważmy aplikację internetową, która wymaga moderacji treści generowanych przez użytkowników w czasie rzeczywistym. Implementacja kompleksowego systemu moderacji od podstaw byłaby trudnym zadaniem, wymagającym znacznego wysiłku rozwojowego i ciągłej konserwacji. Jednak stosując podejście Mnogości Pracowników, programiści mogą zintegrować pracowników moderacji wspieranych przez AI z logiką aplikacji. Ci pracownicy mogą automatycznie analizować i oznaczać nieodpowiednie treści, uwalniając programistów, by mogli skupić się na innych krytycznych aspektach aplikacji.

Pracownicy AI Jako Niezależne Komponenty Wielokrotnego Użytku

Kluczowym aspektem podejścia Mnogości Pracowników jest jego modułowość. Zwolennicy programowania obiektowego od dekad mówią nam, żebyśmy myśleli o interakcjach między obiektami jak o wiadomościach. Cóż, pracownicy AI mogą być zaprojektowani jako niezależne, wielokrotnego użytku komponenty, które mogą „rozmawiać ze sobą“ za pomocą wiadomości w prostym języku, prawie tak, jakby rzeczywiście byli małymi ludźmi rozmawiającymi ze sobą. To luźno powiązane podejście pozwala aplikacji na adaptację i ewolucję w czasie, w miarę jak pojawiają się nowe technologie AI lub zmieniają się wymagania logiki biznesowej.

W praktyce potrzeba projektowania przejrzystych interfejsów i protokołów

komunikacyjnych między komponentami nie zmieniła się tylko dlatego, że zaangażowani są pracownicy AI. Nadal musisz brać pod uwagę inne czynniki, takie jak wydajność, skalowalność i bezpieczeństwo, ale teraz pojawiły się też zupełnie nowe "miękkie wymagania". Na przykład, wielu użytkowników sprzeciwia się wykorzystywaniu ich prywatnych danych do trenowania nowych modeli AI. Czy zweryfikowałeś poziom prywatności zapewniany przez dostawcę modelu, z którego korzystasz?

Pracownicy AI jako mikrousługi?

Czytając o podejściu Wielości Pracowników, możesz zauważyć pewne podobieństwa do architektury mikrousług. Oba podejścia kładą nacisk na dekompozycję złożonych systemów na mniejsze, łatwiejsze w zarządzaniu i niezależnie wdrażalne jednostki. Podobnie jak mikrousługi są projektowane tak, aby były luźno powiązane, skoncentrowane na konkretnych możliwościach biznesowych i komunikowały się poprzez dobrze zdefiniowane API, pracownicy AI są projektowani jako modułowe, wyspecjalizowane w swoich zadaniach komponenty, które współdziałają ze sobą poprzez jasne interfejsy i protokoły komunikacyjne.

Istnieją jednak pewne kluczowe różnice, o których należy pamiętać. Podczas gdy mikrousługi są zazwyczaj implementowane jako oddzielne procesy lub usługi działające na różnych maszynach lub w kontenerach, pracownicy AI mogą być implementowani jako samodzielne komponenty w ramach pojedynczej aplikacji lub jako oddzielne usługi, w zależności od konkretnych wymagań i potrzeb skalowalności. Ponadto, komunikacja między pracownikami AI często obejmuje wymianę bogatych informacji opartych na języku naturalnym, takich jak polecenia, instrukcje i generowane treści, w przeciwieństwie do bardziej ustrukturyzowanych formatów danych powszechnie używanych w mikrousługach.

Mimo tych różnic, zasady modułowości, luźnego powiązania i przejrzystych

> interfejsów komunikacyjnych pozostają kluczowe dla obu wzorców. Stosując te zasady w architekturze pracowników AI, możesz tworzyć elastyczne, skalowalne i łatwe w utrzymaniu systemy, które wykorzystują moc AI do rozwiązywania złożonych problemów i dostarczania wartości użytkownikom.

Podejście Wielości Pracowników może być stosowane w różnych domenach i aplikacjach, wykorzystując moc AI do rozwiązywania złożonych zadań i dostarczania inteligentnych rozwiązań. Przyjrzyjmy się kilku konkretnym przykładom tego, jak pracownicy AI mogą być wykorzystywani w różnych kontekstach.

Zarządzanie kontami

Praktycznie każda samodzielna aplikacja internetowa zawiera koncepcję konta (lub użytkownika). W Olympii wykorzystujemy pracownika AI `AccountManager`, który jest zaprogramowany tak, aby mógł obsługiwać różne rodzaje żądań zmian związanych z kontami użytkowników.

Jego dyrektywa brzmi następująco:

```
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.
```

Początkowy stan konta wygenerowany przez `account.to_directive` to po prostu tekstowy opis konta, zawierający istotne powiązane dane, takie jak użytkownicy, subskrypcje itp.

Zakres funkcji dostępnych dla `AccountManager` daje mu możliwość edytowania subskrypcji użytkownika, dodawania i usuwania konsultantów AI oraz innych płatnych dodatków, a także wysyłania powiadomień e-mail do właściciela konta. Oprócz funkcji `finished`, może również `notify_human_administrator` w przypadku napotkania błędu podczas przetwarzania lub gdy wymaga jakiejkolwiek innej pomocy z żądaniem.

Warto zauważyć, że w przypadku pytań, `AccountManager` może zdecydować się na przeszukanie bazy wiedzy Olympii, gdzie może znaleźć instrukcje dotyczące obsługi przypadków brzegowych i wszelkich innych sytuacji, w których nie jest pewien, jak postępować.

Zastosowania w E-commerce

W dziedzinie e-commerce, pracownicy AI mogą odgrywać kluczową rolę w poprawie doświadczenia użytkownika i optymalizacji operacji biznesowych. Oto kilka sposobów wykorzystania pracowników AI:

Rekomendacje Produktów

Jednym z najpotężniejszych zastosowań pracowników AI w e-commerce jest generowanie spersonalizowanych rekomendacji produktów. Analizując zachowanie użytkowników, historię zakupów i preferencje, pracownicy ci mogą sugerować produkty dostosowane do indywidualnych zainteresowań i potrzeb każdego użytkownika.

Kluczem do skutecznych rekomendacji produktów jest wykorzystanie kombinacji filtrowania kolaboratywnego i filtrowania opartego na zawartości. Filtrowanie kolaboratywne analizuje zachowania podobnych użytkowników w celu identyfikacji wzorców i tworzenia rekomendacji na podstawie tego, co inne osoby o podobnych gustach kupiły lub polubiły. Z kolei filtrowanie oparte na zawartości koncentruje się na cechach i atrybutach samych produktów, rekomendując przedmioty, które mają podobne cechy do tych, którymi użytkownik wcześniej się interesował.

Oto uproszczony przykład implementacji pracownika rekomendacji produktów w Ruby, tym razem wykorzystujący styl programowania funkcyjnego "Railway Oriented (ROP)":

```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
```

```
19    end
20  end
```

Styl programowania funkcyjnego w Ruby użyty w przykładzie jest zainspirowany F# i Rust. Więcej na ten temat możesz przeczytać w wyjaśnieniu tej techniki mojego przyjaciela Chada Wooleya na GitLab.

W tym przykładzie, `ProductRecommendationWorker` przyjmuje użytkownika jako dane wejściowe i generuje spersonalizowane rekomendacje produktów, przekazując obiekt wartości przez łańcuch kroków funkcyjnych. Przyjrzyjmy się każdemu z kroków:

1. `ValidateUser.validate`: Ten krok zapewnia, że użytkownik jest prawidłowy i kwalifikuje się do otrzymania spersonalizowanych rekomendacji. Sprawdza, czy użytkownik istnieje, jest aktywny i czy posiada niezbędne dane do wygenerowania rekomendacji. Jeśli walidacja nie powiedzie się, zwracany jest wynik błędu i następuje przerwanie łańcucha.
2. `AnalyzeCurrentSession.analyze`: Jeśli użytkownik jest prawidłowy, ten krok analizuje jego bieżącą sesję przeglądania w celu zebrania informacji kontekstowych. Sprawdza ostatnie interakcje użytkownika, takie jak przeglądane produkty, zapytania wyszukiwania i zawartość koszyka, aby zrozumieć jego obecne zainteresowania i intencje.
3. `CollaborativeFilter.filter`: Wykorzystując *zachowania podobnych użytkowników*, ten krok stosuje techniki filtrowania kolaboratywnego do identyfikacji produktów, które mogą zainteresować użytkownika. Bierze pod uwagę czynniki takie jak historia zakupów, oceny i interakcje użytkownik-produkt, aby wygenerować zestaw potencjalnych rekomendacji.
4. `ContentBasedFilter.filter`: Ten krok dalej udoskonala kandydujące rekomendacje poprzez zastosowanie filtrowania opartego na zawartości. Porównuje atrybuty i charakterystyki kandydujących produktów z *preferencjami*

użytkownika i danymi historycznymi, aby wybrać najbardziej odpowiednie pozycje.

5. `ProductSelector.select`: Na końcu, ten krok wybiera N najlepszych produktów z przefiltrowanych rekomendacji na podstawie predefiniowanych kryteriów, takich jak wskaźnik trafności, popularność lub inne reguły biznesowe. Wybrane produkty są następnie zwracane jako końcowe spersonalizowane rekomendacje.

Piękno wykorzystania funkcyjnego stylu programowania w Ruby w tym przypadku polega na tym, że pozwala nam połączyć te kroki w sposób jasny i zwięzły. Każdy krok koncentruje się na konkretnym zadaniu i zwraca obiekt `Result`, który może być albo sukcesem (`ok`), albo błędem (`err`). Jeśli którykolwiek krok napotka błąd, łańcuch zostaje przerwany, a błąd jest propagowany do końcowego wyniku.

W instrukcji `case` na końcu, wykonujemy dopasowanie wzorca na końcowym wyniku. Jeśli wynik jest błędem (`ProductRecommendationError`), logujemy błąd przy użyciu narzędzia takiego jak Honeybadger do celów monitorowania i debugowania. Jeśli wynik jest sukcesem (`ProductRecommendations`), rozgłaszamy zdarzenie `:new_recommendations` przy użyciu biblioteki publikowania/subskrypcji Wisper, przekazując użytkownika i wygenerowane rekomendacje.

Wykorzystując techniki programowania funkcyjnego, możemy stworzyć modularny i łatwy w utrzymaniu worker rekomendacji produktów. Każdy krok jest niezależny i może być łatwo testowany, modyfikowany lub zastąpiony bez wpływu na ogólny przepływ. Użycie dopasowania wzorca i klasy `Result` pomaga nam elegancko obsługiwać błędy i zapewnia, że worker szybko kończy działanie, jeśli w którymkolwiek kroku wystąpi problem.

Oczywiście, jest to uproszczony przykład, a w rzeczywistym scenariuszu należałoby zintegrować się z platformą e-commerce, obsłużyć przypadki brzegowe, a nawet zagłębić się w implementację algorytmów rekomendacji. Jednak podstawowe zasady rozkładania

problemu na mniejsze kroki i wykorzystywania technik programowania funkcyjnego pozostają takie same.

Wykrywanie Oszustw

Oto uproszczony przykład implementacji workera do wykrywania oszustw przy użyciu tego samego stylu programowania zorientowanego na tory (ROP) w Ruby:

```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
```

Klasa `FraudDetection` jest *obiektem wartości*, który enkapsuluje stan wykrywania oszustw dla danej transakcji. Zapewnia ustrukturyzowany sposób analizy i oceny ryzyka oszustwa związanego z transakcją w oparciu o różne czynniki ryzyka.

```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
```

Klasa `FraudDetection` posiada następujące atrybuty:

- `transaction`: Referencja do transakcji, która jest analizowana pod kątem oszustwa.
- `risk_factors`: Tablica przechowująca czynniki ryzyka związane z transakcją. Każdy czynnik ryzyka jest reprezentowany jako hash, gdzie kluczem jest opis czynnika ryzyka, a wartością prawdopodobieństwo oszustwa związane z tym czynnikiem.

Metoda `add_risk_factor` umożliwia dodanie czynnika ryzyka do tablicy `risk_factors`. Przyjmuje dwa parametry: `description`, który jest ciągiem znaków opisującym czynnik ryzyka, oraz `probability`, który jest liczbą zmiennoprzecinkową reprezentującą prawdopodobieństwo oszustwa związane z tym czynnikiem ryzyka. Używamy instrukcji warunkowej `case..in` do prostego sprawdzania typów.

Metoda `high_risk?`, która będzie sprawdzana na końcu łańcucha, jest metodą predykatową porównującą `fraud_probability` (obliczone poprzez zsumowanie prawdopodobieństw wszystkich czynników ryzyka) z wartością `RISK_THRESHOLD`.

Klasa `FraudDetection` zapewnia czysty i hermetyczny sposób zarządzania wykrywaniem oszustw dla transakcji. Umożliwia dodawanie wielu czynników ryzyka, każdy z własnym opisem i prawdopodobieństwem, oraz udostępnia metodę określającą, czy transakcja jest uznawana za wysokiego ryzyka na podstawie obliczonego prawdopodobieństwa oszustwa. Klasa może być łatwo zintegrowana z większym systemem wykrywania oszustw, gdzie różne komponenty mogą współpracować w celu oceny i ograniczenia ryzyka oszukańczych transakcji.

Na koniec, ponieważ jest to książka o programowaniu z wykorzystaniem AI, oto przykładowa implementacja klasy `CheckCustomerHistory` wykorzystująca przetwarzanie AI przy użyciu modułu `ChatCompletion` z mojej biblioteki Raix:

```
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
```

W tym przykładzie, `CheckCustomerHistory` definiuje stałą `INSTRUCTION`, która dostarcza szczegółowych instrukcji do modelu AI dotyczących analizy historii transakcji klienta pod kątem potencjalnych wskaźników oszustwa za pomocą

dyrektywy systemowej

Metoda `self.check` jest metodą klasową, która inicjalizuje nową instancję `CheckCustomerHistory` z obiektem `fraud_detection` i wywołuje metodę `call` w celu przeprowadzenia analizy historii klienta.

Wewnątrz metody `call`, historia transakcji klienta jest pobierana i formatowana do postaci transkryptu, który jest przekazywany do modelu AI. Model AI analizuje historię transakcji na podstawie dostarczonych instrukcji i zwraca podsumowanie swoich ustaleń.

Ustalenia są dodawane do obiektu `fraud_detection`, a zaktualizowany obiekt `fraud_detection` jest zwracany jako pomyślny `Result`.

Wykorzystując moduł `ChatCompletion`, klasa `CheckCustomerHistory` może wykorzystać możliwości AI do analizy historii transakcji klienta i identyfikacji potencjalnych wskaźników oszustwa. Pozwala to na bardziej zaawansowane i adaptacyjne techniki wykrywania oszustw, ponieważ model AI może uczyć się i dostosowywać do nowych wzorców i anomalii w czasie.

Zaktualizowany `FraudDetectionWorker` i klasa `CheckCustomerHistory` pokazują, jak pracownicy AI mogą być bezproblemowo zintegrowani, wzbogacając proces wykrywania oszustw o inteligentną analizę i zdolności decyzyjne.

Analiza Sentymentu Klienta

Oto jeszcze jeden podobny przykład pokazujący, jak można zaimplementować workera do analizy sentymentu klienta. Tym razem znacznie mniej wyjaśnień, ponieważ powinieneś już rozumieć, jak działa ten styl programowania:

```
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
```

W tym przykładzie, `CustomerSentimentAnalysisWorker` obejmuje kroki takie jak przetwarzanie wstępne opinii (np. usuwanie szumu, tokenizacja), przeprowadzanie analizy sentymentu w celu określenia ogólnego nastawienia (pozytywnego, negatywnego lub neutralnego), wyodrębnianie kluczowych fraz i tematów, identyfikowanie trendów i wzorców oraz generowanie praktycznych wniosków na podstawie analizy.

Zastosowania w Opiece Zdrowotnej

W dziedzinie opieki zdrowotnej, pracownicy AI mogą wspierać personel medyczny i badaczy w różnych zadaniach, prowadząc do poprawy wyników leczenia pacjentów i przyspieszenia odkryć medycznych. Oto niektóre przykłady:

Przyjęcie Pacjenta

Pracownicy AI mogą usprawnić proces przyjmowania pacjentów poprzez automatyzację różnych zadań i zapewnienie inteligentnej pomocy.

Planowanie Wizyt: Pracownicy AI mogą zarządzać planowaniem wizyt poprzez zrozumienie preferencji pacjenta, jego dostępności i pilności potrzeb medycznych. Mogą wchodzić w interakcję z pacjentami poprzez interfejsy konwersacyjne, prowadząc ich przez proces planowania i znajdując najbardziej odpowiednie terminy wizyt w oparciu o wymagania pacjenta i dostępność świadczeniodawcy.

Zbieranie Historii Medycznej: Podczas przyjęcia pacjenta, pracownicy AI mogą pomagać w zbieraniu i dokumentowaniu historii medycznej pacjenta. Mogą prowadzić interaktywne dialogi z pacjentami, zadając odpowiednie pytania dotyczące ich przeszłych schorzeń, leków, alergii i historii rodzinnej. Pracownicy AI mogą wykorzystywać techniki przetwarzania języka naturalnego do interpretacji i strukturyzacji zebranych informacji, zapewniając ich dokładne zapisanie w elektronicznej dokumentacji medycznej pacjenta.

Ocena i Stratyfikacja Objawów: Pracownicy AI mogą przeprowadzać wstępną ocenę objawów, pytając pacjentów o ich aktualne dolegliwości, czas trwania, nasilenie i wszelkie powiązane czynniki. Wykorzystując medyczne bazy wiedzy i modele uczenia maszynowego, pracownicy ci mogą analizować dostarczone informacje i generować wstępną diagnostykę różnicową lub zalecać odpowiednie kolejne kroki, takie jak umówienie konsultacji z pracownikiem służby zdrowia lub sugerowanie środków samoopieki.

Weryfikacja Ubezpieczenia: Pracownicy AI mogą pomagać w weryfikacji ubezpieczenia podczas przyjęcia pacjenta. Mogą zbierać dane dotyczące ubezpieczenia pacjenta, komunikować się z ubezpieczycielami poprzez interfejsy API lub usługi sieciowe oraz weryfikować uprawnienia do świadczeń i zakres ubezpieczenia. Ta automatyzacja pomaga usprawnić proces weryfikacji ubezpieczenia, zmniejszając

obciążenie administracyjne i zapewniając dokładne gromadzenie informacji.

Edukacja i Instrukcje dla Pacjenta: Pracownicy AI mogą dostarczać pacjentom odpowiednie materiały edukacyjne i instrukcje w oparciu o ich konkretne schorzenia lub nadchodzące procedury. Mogą dostarczać spersonalizowane treści, odpowiadać na często zadawane pytania i oferować wskazówki dotyczące przygotowań do wizyty, instrukcji przyjmowania leków lub opieki po zabiegu. Pomaga to utrzymać pacjentów poinformowanych i zaangażowanych przez cały proces opieki zdrowotnej.

Poprzez wykorzystanie pracowników AI w procesie przyjmowania pacjentów, organizacje ochrony zdrowia mogą zwiększyć efektywność, skrócić czas oczekiwania i poprawić ogólne doświadczenie pacjenta. Pracownicy ci mogą obsługiwać rutynowe zadania, zbierać dokładne informacje i zapewniać spersonalizowaną pomoc, pozwalając pracownikom służby zdrowia skupić się na zapewnianiu pacjentom opieki wysokiej jakości.

Ocena Ryzyka Pacjenta

Pracownicy AI mogą odgrywać kluczową rolę w ocenie ryzyka pacjenta poprzez analizę różnych źródeł danych i stosowanie zaawansowanych technik analitycznych.

Integracja Danych: Pracownicy AI mogą gromadzić i interpretować dane pacjentów z wielu źródeł, takich jak elektroniczna dokumentacja medyczna (EDM), obrazowanie medyczne, wyniki badań laboratoryjnych, urządzenia do noszenia i społeczne determinanty zdrowia. Poprzez połączenie tych informacji w kompleksowy profil pacjenta, pracownicy AI mogą zapewnić holistyczny obraz stanu zdrowia pacjenta i czynników ryzyka.

Stratyfikacja Ryzyka: Pracownicy AI mogą wykorzystywać modele predykcyjne do stratyfikacji pacjentów na różne kategorie ryzyka w oparciu o ich indywidualne cechy i dane zdrowotne. Ta stratyfikacja ryzyka umożliwia świadczeniodawcom opieki zdrowotnej priorytetyzację pacjentów wymagających bardziej natychmiastowej uwagi

lub interwencji. Na przykład, pacjenci zidentyfikowani jako wysokiego ryzyka dla określonego schorzenia mogą zostać oznaczeni do ściślejszego monitorowania, działań zapobiegawczych lub wczesnej interwencji.

Spersonalizowane Profile Ryzyka: Pracownicy AI mogą generować spersonalizowane profile ryzyka dla każdego pacjenta, podkreślając konkretne czynniki wpływające na ich wyniki ryzyka. Profile te mogą zawierać informacje o stylu życia pacjenta, predyspozycjach genetycznych, czynnikach środowiskowych i społecznych determinantach zdrowia. Dostarczając szczegółowej analizy czynników ryzyka, pracownicy AI mogą pomóc świadczeniodawcom opieki zdrowotnej w dostosowaniu strategii prewencyjnych i planów leczenia do indywidualnych potrzeb pacjenta.

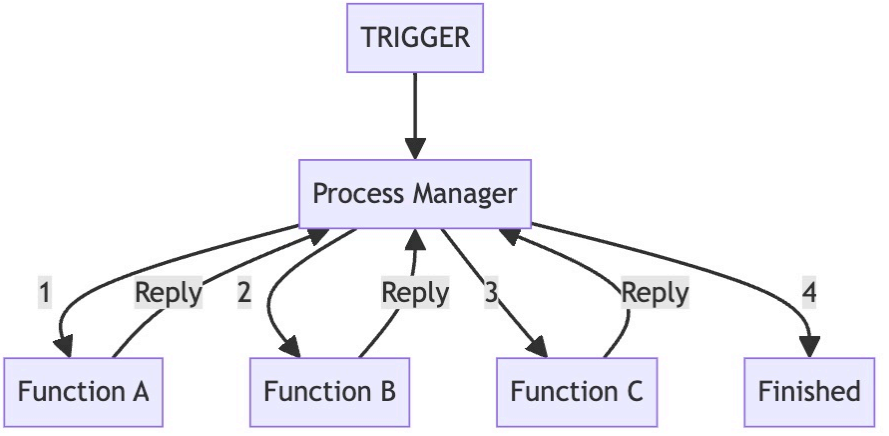
Ciągłe Monitorowanie Ryzyka: Pracownicy AI mogą nieprzerwanie monitorować dane pacjentów i aktualizować oceny ryzyka w czasie rzeczywistym. Gdy pojawiają się nowe informacje, takie jak zmiany w parametrach życiowych, wynikach badań laboratoryjnych czy przestrzeganiu zaleceń dotyczących leków, pracownicy AI mogą ponownie obliczać wyniki ryzyka i alertować świadczeniodawców opieki zdrowotnej o wszelkich istotnych zmianach. To proaktywne monitorowanie umożliwia terminowe interwencje i dostosowywanie planów opieki nad pacjentem.

Wspomaganie Podejmowania Decyzji Klinicznych: Pracownicy AI mogą integrować wyniki oceny ryzyka z systemami wspomagania podejmowania decyzji klinicznych, dostarczając świadczeniodawcom opieki zdrowotnej rekomendacji i alertów opartych na dowodach. Na przykład, jeśli wynik ryzyka pacjenta dla określonego schorzenia przekroczy pewien próg, pracownik AI może zasugerować świadczeniodawcy rozważenie konkretnych testów diagnostycznych, środków zapobiegawczych lub opcji leczenia w oparciu o wytyczne kliniczne i najlepsze praktyki.

Pracownicy ci mogą przetwarzać ogromne ilości danych pacjentów, stosować zaawansowaną analitykę i generować praktyczne spostrzeżenia wspierające podejmowanie decyzji klinicznych. Ostatecznie prowadzi to do poprawy wyników leczenia pacjentów, redukcji kosztów opieki zdrowotnej i lepszego zarządzania

zdrowiem populacji.

Worker AI jako Menedżer Procesów



W kontekście aplikacji opartych na AI, worker może być zaprojektowany do działania jako Menedżer Procesów, zgodnie z opisem w książce "Enterprise Integration Patterns" autorstwa Gregora Hohpe. Menedżer Procesów jest centralnym komponentem, który utrzymuje stan procesu i określa kolejne kroki przetwarzania na podstawie wyników pośrednich.

Gdy worker AI działa jako Menedżer Procesów, otrzymuje przychodzący komunikat inicjalizujący proces, znany jako *komunikat wyzwalający*. Worker AI następnie utrzymuje stan wykonania procesu (jako transkrypt konwersacji) i obsługuje komunikat poprzez serię kroków przetwarzania zaimplementowanych jako funkcje narzędziowe, które mogą być wykonywane sekwencyjnie lub równolegle, wywoływane według jego uznania.

Jeśli używasz klasy modelu AI takiego jak GPT-4, który potrafi wykonywać funkcje równolegle, twój worker może wykonywać wiele kroków jednocześnie. Przyznaję, że sam tego nie próbowałem i mój instynkt podpowiada, że wyniki mogą być różne.

Po każdym pojedynczym kroku przetwarzania, kontrola wraca do workera AI, pozwalając mu określić kolejne kroki przetwarzania na podstawie aktualnego stanu i uzyskanych wyników.

Przechowuj Swoje Komunikaty Wyzwalające

Z mojego doświadczenia wynika, że rozsądnie jest zaimplementować komunikat wyzwalający jako obiekt wspierany bazą danych. W ten sposób każda instancja procesu jest identyfikowana przez unikalny klucz główny i zapewnia miejsce do przechowywania stanu związanego z wykonaniem, włącznie z transkryptem konwersacji AI.

Na przykład, oto uproszczona wersja klasy modelu `AccountChange` z Olympii, która reprezentuje żądanie wprowadzenia zmiany w koncie użytkownika.

```
# == 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
```

Klasa `AccountChange` służy jako komunikat wyzwalający, który inicjuje proces obsługi żądania zmiany konta. Warto zauważyć, jak jest on rozgłaszany do podsystemu pub/sub Olympii opartego na Wisper po zakończeniu transakcji tworzenia.

Przechowywanie komunikatu wyzwalającego w bazie danych w ten sposób zapewnia trwały zapis każdego żądania zmiany konta. Każdej instancji klasy `AccountChange` przypisywany jest unikalny klucz główny, co umożliwia łatwą identyfikację i śledzenie pojedynczych żądań. Jest to szczególnie przydatne dla celów rejestrowania audytu, ponieważ pozwala systemowi na utrzymywanie historycznego zapisu wszystkich zmian konta, w tym informacji o tym, kiedy zostały zgłoszone, jakie zmiany zostały zażądane i jaki jest aktualny stan każdego żądania.

W podanym przykładzie klasa `AccountChange` zawiera pola takie jak `description` do przechwytywania szczegółów żądanej zmiany, `state` do reprezentowania aktualnego

stanu żądania (np. requested, complete, requires_human_review) oraz `transcript` do przechowywania transkryptu rozmowy z AI związanej z żądaniem. Pole `description` jest właściwym promptem używanym do zainicjowania pierwszego uzupełnienia czatu z AI. Przechowywanie tych danych zapewnia cenny kontekst i umożliwia lepsze śledzenie oraz analizę procesu zmiany konta.

Przechowywanie komunikatów wyzwalających w bazie danych umożliwia solidną obsługę błędów i odzyskiwanie. Jeśli podczas przetwarzania żądania zmiany konta wystąpi błąd, system oznacza żądanie jako nieudane i przechodzi do stanu wymagającego interwencji człowieka. Zapewnia to, że żadne żądanie nie zostanie utracone ani zapomniane, a wszelkie problemy mogą zostać odpowiednio zaadresowane i rozwiązane.

Program pracujący z AI, jako Manager Procesów, zapewnia centralny punkt kontroli i umożliwia zaawansowane możliwości raportowania i debugowania procesów. Jednak ważne jest, aby zauważyć, że używanie programu pracującego z AI jako Managera Procesów dla każdego scenariusza przepływu pracy w aplikacji może być przesadą.

Integracja Programów Pracujących z AI w Architekturze Aplikacji

Podczas włączania programów pracujących z AI do architektury aplikacji, należy wziąć pod uwagę kilka aspektów technicznych, aby zapewnić płynną integrację i efektywną komunikację między programami AI a innymi komponentami aplikacji. Ta sekcja omawia kluczowe aspekty projektowania interfejsów, obsługi przepływu danych i zarządzania cyklem życia programów AI.

Projektowanie Przejrzystych Interfejsów i Protokołów Komunikacji

Aby ułatwić bezproblemową integrację między programami AI a innymi komponentami aplikacji, kluczowe jest zdefiniowanie przejrzystych interfejsów i protokołów komunikacji. Rozważ następujące podejścia:

Integracja oparta na API: Udostępnij funkcjonalność pracowników AI poprzez dobrze zdefiniowane API, takie jak punkty końcowe RESTful lub schematy GraphQL. Pozwala to innym komponentom na interakcję z pracownikami AI przy użyciu standardowych żądań i odpowiedzi HTTP. Integracja oparta na API zapewnia jasny kontrakt między pracownikami AI a komponentami korzystającymi z ich usług, ułatwiając rozwój, testowanie i utrzymanie punktów integracji.

Komunikacja oparta na komunikatach: Zaimplementuj wzorce komunikacji opartej na komunikatach, takie jak kolejki komunikatów lub systemy publikowania-subskrypcji, aby umożliwić asynchroniczną interakcję między pracownikami AI a innymi komponentami. Takie podejście oddziela pracowników AI od reszty aplikacji, zapewniając lepszą skalowalność, odporność na błędy i luźne powiązania. Komunikacja oparta na komunikatach jest szczególnie przydatna, gdy przetwarzanie wykonywane przez pracowników AI jest czasochłonne lub wymaga dużych zasobów, ponieważ pozwala innym częściom aplikacji na kontynuowanie działania bez oczekiwania na zakończenie zadań przez pracowników AI.

Architektura sterowana zdarzeniami: Zaprojektuj system wokół zdarzeń i wyzwalaczy, które aktywują pracowników AI, gdy spełnione są określone warunki. Pracownicy AI mogą subskrybować odpowiednie zdarzenia i reagować na nie, wykonując wyznaczone zadania, gdy zdarzenia wystąpią. Architektura sterowana zdarzeniami umożliwia przetwarzanie w czasie rzeczywistym i pozwala na wywoływanie pracowników AI na żądanie, zmniejszając niepotrzebne zużycie zasobów. To podejście jest dobrze dostosowane do scenariuszy, w których pracownicy AI muszą reagować na określone działania lub zmiany w stanie aplikacji.

Obsługa przepływu danych i synchronizacji

Podczas integracji pracowników AI z aplikacją kluczowe jest zapewnienie płynnego przepływu danych i utrzymanie spójności danych między pracownikami AI a innymi komponentami. Weź pod uwagę następujące aspekty:

Przygotowanie danych: Przed przekazaniem danych do pracowników AI może być konieczne wykonanie różnych zadań związanych z przygotowaniem danych, takich jak czyszczenie, formatowanie i/lub przekształcanie danych wejściowych. Chodzi nie tylko o to, aby pracownicy AI mogli efektywnie przetwarzać dane, ale także o to, by nie marnować tokenów na zwracanie uwagi na informacje, które pracownik może uznać za bezużyteczne w najlepszym razie, a rozpraszające w najgorszym. Przygotowanie danych może obejmować zadania takie jak usuwanie szumu, obsługa brakujących wartości lub konwersja typów danych.

Trwałość danych: Jak będziesz przechowywać i utrzymywać dane przepływające do i z pracowników AI? Weź pod uwagę czynniki takie jak wolumen danych, wzorce zapytań i skalowalność. Czy musisz zachować transkrypt AI jako odzwierciedlenie jego "procesu myślowego" do celów audytu lub debugowania, czy wystarczy mieć zapis tylko wyników?

Pobieranie danych: Pozyskiwanie danych potrzebnych pracownikom może obejmować zapytania do baz danych, odczyt z plików lub dostęp do zewnętrznych API. Upewnij się, że uwzględniasz opóźnienia i sposób, w jaki pracownicy AI będą mieli dostęp do najbardziej aktualnych danych. Czy potrzebują pełnego dostępu do twojej bazy danych, czy powinieneś wąsko zdefiniować zakres ich dostępu zgodnie z tym, co robią? A co ze skalowaniem? Rozważ mechanizmy buforowania w celu poprawy wydajności i zmniejszenia obciążenia źródeł danych.

Synchronizacja danych: Gdy wiele komponentów, w tym pracownicy AI, uzyskuje dostęp i modyfikuje współdzielone dane, ważne jest wdrożenie odpowiednich mechanizmów synchronizacji w celu zachowania spójności danych. Strategie

blokowania baz danych, takie jak blokowanie optymistyczne lub pesymistyczne, mogą pomóc w zapobieganiu konfliktom i zapewnieniu integralności danych. Wdrażaj techniki zarządzania transakcjami, aby grupować powiązane operacje na danych i zachować właściwości ACID (atomowość, spójność, izolację i trwałość)

Obsługa błędów i odzyskiwanie: Wdrażaj solidne mechanizmy obsługi błędów i odzyskiwania, aby radzić sobie z problemami związanymi z danymi, które mogą pojawić się podczas procesu przepływu danych. Obsługuj wyjątki w sposób elegancki i dostarczaj znaczące komunikaty o błędach, aby pomóc w debugowaniu. Zaimplementuj mechanizmy ponownych prób i strategie awaryjne, aby obsłużyć tymczasowe awarie lub zakłócenia sieci. Zdefiniuj jasne procedury odzyskiwania i przywracania danych w przypadku ich uszkodzenia lub utraty.

Poprzez staranne projektowanie i wdrażanie mechanizmów przepływu i synchronizacji danych, możesz zapewnić, że twoi pracownicy AI mają dostęp do dokładnych, spójnych i aktualnych danych. Umożliwia im to efektywne wykonywanie zadań i generowanie wiarygodnych wyników.

Zarządzanie cyklem życia pracowników AI

Opracuj standardowy proces inicjalizacji i konfiguracji pracowników AI. Preferuję frameworki, które standaryzują sposób definiowania ustawień, takich jak nazwy modeli, dyrektywy systemowe i definicje funkcji. Upewnij się, że proces inicjalizacji jest zautomatyzowany i powtarzalny, aby ułatwić wdrażanie i skalowanie.

Wdrażaj kompleksowe mechanizmy monitorowania i rejestrowania w celu śledzenia kondycji i wydajności pracowników AI. Zbieraj metryki, takie jak wykorzystanie zasobów, czas przetwarzania, współczynniki błędów i przepustowość. Używaj scentralizowanych systemów rejestrowania, takich jak ELK stack (Elasticsearch, Logstash, Kibana), do agregacji i analizy logów z wielu pracowników AI.

Zbuduj tolerancję błędów i odporność w architekturze pracowników AI. Zaimplementuj mechanizmy obsługi błędów i odzyskiwania, aby elegancko radzić sobie z awariami

i wyjątkami. Duże Modele Językowe to wciąż technologia na bardzo wczesnym etapie rozwoju; dostawcy często mają nieoczekiwane przestoje. Używaj mechanizmów ponawiania prób i bezpieczników, aby zapobiec kaskadowym awariom.

Kompozycyjność i orkiestracja pracowników AI

Jedną z kluczowych zalet architektury pracowników AI jest jej kompozycyjność, która pozwala na łączenie i orkiestrację wielu pracowników AI w celu rozwiązywania złożonych problemów. Dzieląc większe zadanie na mniejsze, łatwiejsze w zarządzaniu podzadania, z których każde jest obsługiwane przez wyspecjalizowanego pracownika AI, możesz tworzyć potężne i elastyczne systemy. W tej sekcji zbadamy różne podejścia do komponowania i orkiestracji "mnogości" pracowników AI.

Łańcuchowanie pracowników AI dla wieloetapowych przepływów pracy

W wielu scenariuszach złożone zadanie można rozłożyć na serię sekwencyjnych kroków, gdzie wynik jednego pracownika AI staje się danymi wejściowymi dla następnego. To łańcuchowanie pracowników AI tworzy wieloetapowy przepływ pracy lub potok. Każdy pracownik AI w łańcuchu koncentruje się na konkretnym podzadaniu, a końcowy wynik jest rezultatem połączonych wysiłków wszystkich pracowników.

Rozważmy przykład w kontekście aplikacji Ruby on Rails do przetwarzania treści generowanych przez użytkowników. Przepływ pracy obejmuje następujące kroki, które przyznajmy, prawdopodobnie są zbyt proste, aby warto było je tak rozkładać w rzeczywistych przypadkach użycia, ale ułatwiają zrozumienie przykładu:

1. **Oczyszczanie tekstu**: Pracownik AI odpowiedzialny za usuwanie tagów HTML, konwersję tekstu na małe litery i obsługę normalizacji Unicode.

2. **Wykrywanie języka**: Pracownik AI, który identyfikuje język oczyszczonego tekstu.

3. **Analiza sentymentu**: Pracownik AI, który określa sentyment (pozytywny, negatywny lub neutralny) tekstu na podstawie wykrytego języka.

4. **Kategoryzacja treści**: Pracownik AI, który klasyfikuje tekst do predefiniowanych kategorii przy użyciu technik przetwarzania języka naturalnego.

Oto bardzo uproszczony przykład tego, jak można połączyć tych pracowników AI w łańcuch przy użyciu Ruby:

```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
```

W tym przykładzie klasa `ContentProcessor` inicjalizuje się z surowym tekstem i łączy moduły AI w metodzie `process`. Każdy moduł AI wykonuje swoje określone zadanie i przekazuje wynik do następnego modułu w łańcuchu. Końcowym wynikiem jest hash zawierający oczyszczony tekst, wykryty język, sentyment i kategorię treści.

Przetwarzanie równoległe dla niezależnych modułów AI

W poprzednim przykładzie moduły AI są połączone sekwencyjnie, gdzie każdy moduł przetwarza tekst i przekazuje wynik do następnego modułu. Jednak jeśli masz wiele modułów AI, które mogą działać niezależnie na tym samym wejściu, możesz zoptymalizować przepływ pracy poprzez przetwarzanie ich równoległe.

W danym scenariuszu, gdy tekst zostanie oczyszczony przez `TextCleanupWorker`, moduły `LanguageDetectionWorker`, `SentimentAnalysisWorker` i `CategorizationWorker` mogą wszystkie przetwarzać oczyszczony tekst niezależnie. Uruchamiając te moduły równolegle, możesz potencjalnie zmniejszyć całkowity czas przetwarzania i poprawić wydajność swojego przepływu pracy.

Aby osiągnąć przetwarzanie równoległe w Ruby, możesz wykorzystać techniki współbieżności, takie jak wątki lub programowanie asynchroniczne. Oto przykład jak można zmodyfikować klasę `ContentProcessor`, aby przetwarzać ostatnie trzy moduły równolegle przy użyciu wątków:

```
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
```

```
29 end
```

W tej zoptymalizowanej wersji używamy biblioteki `concurrent-ruby` do tworzenia obiektów `Concurrent::Future` dla każdego z niezależnych modułów AI. Obiekt `Future` reprezentuje obliczenie, które zostanie wykonane asynchronicznie w osobnym wątku.

Po etapie oczyszczania tekstu, tworzymy trzy obiekty `Future`: `language_future`, `sentiment_future` i `category_future`. Każdy `Future` wykonuje odpowiadający mu moduł AI (`LanguageDetectionWorker`, `SentimentAnalysisWorker` i `CategorizationWorker`) w osobnym wątku, przekazując `cleaned_text` jako dane wejściowe.

Wywołując metodę `value` na każdym obiekcie `Future`, czekamy na zakończenie obliczeń i pobieramy wynik. Metoda `value` blokuje wykonanie do momentu, gdy wynik będzie dostępny, zapewniając tym samym, że wszystkie równoległe moduły zakończyły przetwarzanie przed kontynuacją.

Na końcu tworzymy hash wyjściowy zawierający oczyszczony tekst i wyniki z równoległych modułów, podobnie jak w oryginalnym przykładzie.

Dzięki przetwarzaniu niezależnych modułów AI równolegle, możesz potencjalnie skrócić całkowity czas przetwarzania w porównaniu do wykonywania ich sekwencyjnie. Ta optymalizacja jest szczególnie korzystna w przypadku czasochłonnych zadań lub podczas przetwarzania dużych ilości danych.

Należy jednak pamiętać, że rzeczywiste zyski wydajnościowe zależą od różnych czynników, takich jak złożoność każdego modułu, dostępne zasoby systemowe i narzut związany z zarządzaniem wątkami. Dobrą praktyką jest zawsze przeprowadzanie testów wydajności i profilowanie kodu, aby określić optymalny poziom zrównoleglenia dla konkretnego przypadku użycia.

Dodatkowo, podczas implementacji przetwarzania równoległego, należy zwrócić uwagę na wszelkie współdzielone zasoby lub zależności między modułami.

Upewnij się, że moduły mogą działać niezależnie bez konfliktów czy warunków wyścigu. Jeśli występują zależności lub współdzielone zasoby, może być konieczne zaimplementowanie odpowiednich mechanizmów synchronizacji w celu zachowania integralności danych i uniknięcia problemów takich jak zakleszczenia czy niespójne wyniki.

Globalna Blokada Interpretera Ruby i Przetwarzanie Asynchroniczne

Ważne jest zrozumienie implikacji Globalnej Blokady Interpretera (GIL) Ruby podczas rozważania asynchronicznego przetwarzania opartego na wątkach w Ruby.

GIL to mechanizm w interpreterze Ruby, który zapewnia, że tylko jeden wątek może wykonywać kod Ruby w danym momencie, nawet na procesorach wielordzeniowych. Oznacza to, że choć można tworzyć i zarządzać wieloma wątkami w procesie Ruby, tylko jeden wątek może aktywnie wykonywać kod Ruby w danej chwili.

GIL został zaprojektowany w celu uproszczenia implementacji interpretera Ruby oraz zapewnienia bezpieczeństwa wątków dla wewnętrznych struktur danych Ruby. Jednakże ogranicza on również możliwość prawdziwie równoległego wykonywania kodu Ruby.

Gdy używasz wątków w Ruby, na przykład z biblioteką `concurrent-ruby` lub wbudowaną klasą `Thread`, wątki podlegają ograniczeniom GIL-a. GIL pozwala każdemu wątkowi na wykonywanie kodu Ruby przez krótki przedział czasu, zanim przełączy się na inny wątek, tworząc iluzję współbieżnego wykonywania.

Jednak ze względu na GIL, faktyczne wykonywanie kodu Ruby pozostaje sekwencyjne. Podczas gdy jeden wątek wykonuje kod Ruby, inne wątki są zasadniczo wstrzymane, czekając na swoją kolej na przejęcie GIL-a i wykonanie.

Oznacza to, że asynchroniczne przetwarzanie oparte na wątkach w Ruby jest najbardziej efektywne dla zadań ograniczonych przez operacje wejścia/wyjścia, takich jak oczekiwanie na odpowiedzi zewnętrznych API (na przykład modeli językowych hostowanych przez strony trzecie) lub wykonywanie operacji wejścia/wyjścia na plikach. Gdy wątek napotyka operację wejścia/wyjścia, może zwolnić GIL, pozwalając innym wątkom na wykonywanie się podczas oczekiwania na zakończenie operacji wejścia/wyjścia.

Z drugiej strony, w przypadku zadań ograniczonych przez procesor, takich jak intensywne obliczenia lub długotrwałe przetwarzanie przez programy AI, GIL może ograniczać potencjalne zyski wydajnościowe wynikające z równoległości opartej na wątkach. Ponieważ tylko jeden wątek może wykonywać kod Ruby w danym momencie, całkowity czas wykonania może nie być znacząco zredukowany w porównaniu do przetwarzania sekwencyjnego.

Aby osiągnąć prawdziwe równoległe wykonywanie zadań ograniczonych przez procesor w Ruby, może być konieczne zbadanie alternatywnych podejść, takich jak:

- Wykorzystanie równoległości opartej na procesach z wieloma procesami Ruby, z których każdy działa na osobnym rdzeniu procesora.
- Korzystanie z zewnętrznych bibliotek lub frameworków, które zapewniają natywne rozszerzenia lub interfejsy do języków bez GIL-a, takich jak C lub Rust.,
- Wykorzystanie frameworków do przetwarzania rozproszonego lub kolejek komunikatów do dystrybuowania zadań między wieloma maszynami lub procesami.

Kluczowe jest uwzględnienie charakteru zadań i ograniczeń narzuconych przez GIL podczas projektowania i implementacji asynchronicznego przetwarzania w Ruby. Podczas gdy asynchroniczne przetwarzanie oparte na wątkach może przynieść korzyści dla zadań ograniczonych przez operacje wejścia/wyjścia, może nie oferować

znaczących ulepszeń wydajności dla zadań ograniczonych przez procesor ze względu na ograniczenia GIL-a.

Techniki zespołowe dla zwiększonej dokładności

Techniki zespołowe polegają na łączeniu wyników wielu programów AI w celu poprawy ogólnej dokładności lub odporności systemu. Zamiast polegać na pojedynczym programie AI, techniki zespołowe wykorzystują zbiorową inteligencję wielu programów do podejmowania bardziej świadomych decyzji.

Zespoły modeli są szczególnie ważne, gdy różne części przepływu pracy działają najlepiej z różnymi modelami AI, co zdarza się częściej, niż mogłoby się wydawać. Potężne modele jak GPT-4 są niezwykle drogie w porównaniu z mniej zaawansowanymi opcjami otwartoźródłowymi i prawdopodobnie nie są potrzebne na każdym etapie przepływu pracy aplikacji.

Popularną techniką zespołową jest głosowanie większościowe, gdzie wiele modeli AI niezależnie przetwarza te same dane wejściowe, a końcowy wynik jest określany przez konsensus większości. To podejście może pomóc złagodzić wpływ błędów pojedynczych modeli i poprawić ogólną niezawodność systemu.

Rozważmy przykład, w którym mamy trzy modele AI do analizy sentymentu, każdy wykorzystujący inny model lub wyposażony w różne konteksty. Możemy połączyć ich wyniki za pomocą głosowania większościowego, aby określić końcową predykcję sentymentu.

```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
```

W tym przykładzie klasa `SentimentAnalysisEnsemble` inicjalizuje się z tekstem i wywołuje trzech różnych pracowników AI do analizy sentymentu. Metoda `analyze` zbiera przewidywania od każdego pracownika i określa dominujący sentyment przy użyciu metod `group_by` i `max_by`. Końcowym wynikiem jest sentyment, który otrzymuje najwięcej głosów od zespołu pracowników.

Zespoły są wyraźnym przypadkiem, gdzie eksperymentowanie z równoległością może być warte twojego czasu.

Dynamiczny Wybór i Wywoływanie Pracowników AI

W niektórych, jeśli nie w większości przypadków, wybór konkretnego pracownika AI może zależeć od warunków wykonania lub danych wejściowych użytkownika. Dynamiczny wybór i wywoływanie pracowników AI pozwala na elastyczność i adaptowalność systemu.

Możesz odczuwać pokusę, aby zmieścić dużo funkcjonalności w pojedynczym pracowniku AI, dając mu wiele funkcji i skomplikowane polecenie wyjaśniające jak je wywoływać. Oprzyj się tej pokusie, zaufaj mi. Jednym z powodów, dla których podejście, które omawiamy w tym rozdziale, nazywa się "Mnogość Pracowników", jest przypomnienie nam, że pożądane jest posiadanie wielu wyspecjalizowanych pracowników, z których każdy wykonuje swoje małe zadanie w służbie większemu celowi.

Na przykład, rozważmy aplikację chatbota, gdzie różni pracownicy AI są odpowiedzialni za obsługę różnych typów zapytań użytkownika. Na podstawie danych wejściowych użytkownika, aplikacja dynamicznie wybiera odpowiedniego pracownika AI do przetworzenia zapytania.

```
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
```

W tym przykładzie, `ChatbotController` otrzymuje zapytanie użytkownika poprzez akcję `process_query`. Najpierw wykorzystuje `QueryClassifierWorker` do określenia typu zapytania. Na podstawie sklasyfikowanego typu zapytania, kontroler

dynamicznie wybiera odpowiedni moduł AI do wygenerowania odpowiedzi. Ta dynamiczna selekcja pozwala chatbotowi obsługiwać różne typy zapytań i kierować je do odpowiednich modułów AI.

Ponieważ praca `QueryClassifierWorker` jest stosunkowo prosta i nie wymaga dużego kontekstu ani definicji funkcji, prawdopodobnie możesz zaimplementować ją używając ultra-szybkiego małego LLM, takiego jak `mistralai/mixtral-8x7b-instruct:nitro`. Posiada on możliwości zbliżone do poziomu GPT-4 w wielu zadaniach i, w momencie gdy to piszę, Groq może obsługiwać go z oszałamiającą przepustowością 444 tokenów na sekundę.

Łączenie tradycyjnego NLP z LLM

Podczas gdy Wielkie Modele Językowe (LLM) zrewolucjonizowały dziedzinę przetwarzania języka naturalnego (NLP), oferując niezrównaną wszechstronność i wydajność w szerokiej gamie zadań, nie zawsze są one najbardziej efektywnym lub opłacalnym rozwiązaniem każdego problemu. W wielu przypadkach, łączenie tradycyjnych technik NLP z LLM może prowadzić do bardziej zoptymalizowanych, ukierunkowanych i ekonomicznych podejść do rozwiązywania konkretnych wyzwań NLP.

Pomyśl o LLM jak o scyzoryku szwajcarskim w dziedzinie NLP — niewiarygodnie wszechstronnym i potężnym, ale niekoniecznie najlepszym narzędziem do każdego zadania. Czasami dedykowane narzędzie, jak korkociąg czy otwieracz do puszek, może być bardziej skuteczne i wydajne do konkretnego zadania. Podobnie, tradycyjne techniki NLP, takie jak grupowanie dokumentów, identyfikacja tematów i klasyfikacja, często mogą zapewnić bardziej ukierunkowane i opłacalne rozwiązania dla określonych aspektów twojego pipeline'u NLP.

Jedną z kluczowych zalet tradycyjnych technik NLP jest ich wydajność obliczeniowa. Te metody, które często opierają się na prostszych modelach statystycznych lub podejściach opartych na regułach, mogą przetwarzać duże ilości danych tekstowych znacznie szybciej i przy mniejszym obciążeniu obliczeniowym w porównaniu do LLM. Sprawia to, że są szczególnie dobrze dostosowane do zadań związanych z analizowaniem i organizowaniem dużych korpusów dokumentów, takich jak grupowanie podobnych artykułów czy identyfikacja kluczowych tematów w zbiorze tekstów.

Co więcej, tradycyjne techniki NLP często mogą osiągać wysoką dokładność i precyzję w konkretnych zadaniach, szczególnie gdy są trenowane na zbiorach danych z określonej dziedziny. Na przykład, dobrze dostrojony klasyfikator dokumentów wykorzystujący tradycyjne algorytmy uczenia maszynowego, takie jak Maszyny Wektorów Nośnych (SVM) czy Naiwny Klasyfikator Bayesa, może precyzyjnie kategoryzować dokumenty do predefiniowanych kategorii przy minimalnym koszcie obliczeniowym.

Jednak LLMs naprawdę błyszczą, gdy chodzi o zadania wymagające głębszego zrozumienia języka, kontekstu i wnioskowania. Ich zdolność do generowania spójnego i kontekstowo trafnego tekstu, odpowiadania na pytania i streszczania długich fragmentów jest niedościgniona przez tradycyjne metody NLP. LLMs potrafią skutecznie radzić sobie ze złożonymi zjawiskami językowymi, takimi jak wieloznaczność, koreferencja i wyrażenia idiomatyczne, co czyni je nieocenionymi w zadaniach wymagających generowania lub rozumienia języka naturalnego.

Prawdziwa moc tkwi w łączeniu tradycyjnych technik NLP z LLMs w celu tworzenia podejść hybrydowych, które wykorzystują zalety obu metod. Używając tradycyjnych metod NLP do zadań takich jak wstępne przetwarzanie dokumentów, klastrowanie i ekstrakcja tematów, można efektywnie organizować i strukturyzować dane tekstowe. Te ustrukturyzowane informacje mogą być następnie przekazywane do LLMs w celu wykonywania bardziej zaawansowanych zadań, takich jak generowanie streszczeń, odpowiadanie na pytania czy tworzenie kompleksowych raportów.

Na przykład, rozważmy przypadek użycia, w którym chcemy wygenerować raport o trendach dla konkretnej dziedziny na podstawie dużego korpusu pojedynczych dokumentów opisujących trendy. Zamiast polegać wyłącznie na LLMs, co może być kosztowne obliczeniowo i czasochłonne przy przetwarzaniu dużych ilości tekstu, można zastosować podejście hybrydowe:

1. Wykorzystać tradycyjne techniki NLP, takie jak modelowanie tematyczne (np. Ukryta Alokacja Dirichleta) lub algorytmy klastrowania (np. K-średnich), do grupowania podobnych dokumentów o trendach i identyfikacji kluczowych motywów i tematów w korpusie.
2. Przekazać pogrupowane dokumenty i zidentyfikowane tematy do LLM, wykorzystując jego lepsze możliwości rozumienia i generowania języka do tworzenia spójnych i informatywnych streszczeń dla każdego klastra lub tematu.
3. Na koniec, użyć LLM do wygenerowania kompleksowego raportu o trendach poprzez połączenie pojedynczych streszczeń, wyróżnienie najważniejszych trendów oraz dostarczenie spostrzeżeń i rekomendacji na podstawie zagregowanych informacji.

Łącząc tradycyjne techniki NLP z LLMs w ten sposób, można efektywnie przetwarzać duże ilości danych tekstowych, wydobywać znaczące spostrzeżenia i generować wysokiej jakości raporty przy jednoczesnej optymalizacji zasobów obliczeniowych i kosztów.

Rozpoczynając projekty NLP, kluczowe jest staranne ocenienie konkretnych wymagań i ograniczeń każdego zadania oraz rozważenie, jak tradycyjne metody NLP i LLMs mogą być wspólnie wykorzystane do osiągnięcia najlepszych rezultatów. Łącząc wydajność i precyzję tradycyjnych technik z wszechstronnością i mocą LLMs, możesz tworzyć wysoce skuteczne i ekonomiczne rozwiązania NLP, które dostarczają wartość Twoim użytkownikom i interesariuszom.

Używanie Narzędzi

W dziedzinie tworzenia aplikacji opartych na sztucznej inteligencji, koncepcja “użycia narzędzi” lub “wywołania funkcji” stała się potężną techniką, która umożliwia twojemu LLM łączenie się z zewnętrznymi narzędziami, interfejsami API, funkcjami, bazami danych i innymi zasobami. To podejście pozwala na bogatszy zestaw zachowań niż tylko generowanie tekstu oraz bardziej dynamiczne interakcje między komponentami AI a resztą ekosystemu aplikacji. Jak przekonamy się w tym rozdziale, użycie narzędzi daje również możliwość generowania danych przez model AI w sposób ustrukturyzowany.

Czym jest użycie narzędzi?

Użycie narzędzi, znane również jako wywołanie funkcji, to technika pozwalająca programistom określić listę funkcji, z którymi LLM może wchodzić w interakcję podczas

procesu generowania. Narzędzia te mogą obejmować zarówno proste funkcje użytkowe, jak i złożone interfejsy API czy zapytania do baz danych. Zapewniając modelowi LLM dostęp do tych narzędzi, programiści mogą rozszerzyć możliwości modelu i umożliwić mu wykonywanie zadań wymagających zewnętrznej wiedzy lub działań.

Rycina 8. Przykład definicji funkcji dla pracownika AI analizującego dokumenty

```
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
```

Kluczową ideą stojącą za wykorzystaniem narzędzi jest zapewnienie modelowi LLM możliwości dynamicznego wybierania i wykonywania odpowiednich narzędzi w oparciu o dane wejściowe użytkownika lub bieżące zadanie. Zamiast polegać wyłącznie na wstępnie wytrenowanej wiedzy modelu, wykorzystanie narzędzi

pozwala modelowi LLM korzystać z zewnętrznych zasobów w celu generowania dokładniejszych, bardziej trafnych i praktycznych odpowiedzi. Wykorzystanie narzędzi sprawia, że techniki takie jak RAG (Generowanie Wspomagane Wyszukiwaniem) są znacznie łatwiejsze do wdrożenia niż byłyby w przeciwnym razie.

> Należy zauważyć, że o ile nie określono inaczej, w tej książce zakłada się, że model AI nie ma dostępu do żadnych wbudowanych narzędzi po stronie serwera. Wszelkie narzędzia, które chcesz udostępnić swojemu AI, muszą zostać przez Ciebie jawnie zadeklarowane w każdym żądaniu API, wraz z możliwością uruchomienia ich wykonania, jeśli i kiedy Twoje AI poinformuje Cię, że chciałoby użyć tego narzędzia w swojej odpowiedzi.

Potencjał Wykorzystania Narzędzi

Wykorzystanie narzędzi otwiera szeroki wachlarz możliwości dla aplikacji opartych na AI. Oto kilka przykładów tego, co można osiągnąć dzięki wykorzystaniu narzędzi:

1. **Chatboty i Asystenci Wirtualni**: Poprzez połączenie modelu LLM z zewnętrznymi narzędziami, chatboty i asystenci wirtualni mogą wykonywać bardziej złożone zadania, takie jak pobieranie informacji z baz danych, wykonywanie wywołań API czy interakcja z innymi systemami. Na przykład, chatbot może użyć narzędzia CRM do zmiany statusu transakcji na podstawie żądania użytkownika.
2. **Analiza Danych i Wnioski**: Modele LLM mogą być połączone z narzędziami lub bibliotekami do analizy danych, aby wykonywać zaawansowane zadania przetwarzania danych. Umożliwia to aplikacjom generowanie spostrzeżeń, przeprowadzanie analiz porównawczych lub dostarczanie rekomendacji opartych na danych w odpowiedzi na zapytania użytkowników.

3. **Wyszukiwanie i Pozyskiwanie Informacji**: Wykorzystanie narzędzi pozwala modelom LLM na interakcję z wyszukiwarkami, bazami danych wektorowych lub innymi systemami pozyskiwania informacji. Poprzez przekształcanie zapytań użytkownika w zapytania wyszukiwawcze, model LLM może pobierać istotne informacje z wielu źródeł i dostarczać wyczerpujące odpowiedzi na pytania użytkowników.
4. **Integracja z Usługami Zewnętrznymi**: Wykorzystanie narzędzi umożliwia bezproblemową integrację między aplikacjami opartymi na AI a usługami zewnętrznymi lub API. Na przykład, model LLM może współpracować z API pogodowym, aby dostarczać aktualne informacje o pogodzie, lub z API tłumaczeniowym, aby generować odpowiedzi w wielu językach.

Przebieg Wykorzystania Narzędzi

Przepływ pracy związany z wykorzystaniem narzędzi zazwyczaj obejmuje cztery kluczowe kroki:

1. Włączenie definicji funkcji do kontekstu żądania
2. Dynamiczny (lub jawny) wybór narzędzi
3. Wykonanie funkcji
4. Opcjonalna kontynuacja pierwotnego polecenia

Przyjrzyjmy się szczegółowo każdemu z tych kroków.

Włączenie definicji funkcji do kontekstu żądania

SI wie, jakimi narzędziami dysponuje, ponieważ przekazujesz jej listę jako część żądania uzupełnienia (zazwyczaj zdefiniowaną jako funkcje przy użyciu wariantu schematu JSON).

Dokładna składnia definicji narzędzi jest zależna od modelu.

Oto jak definiuje się funkcję `get_weather` w Claude 3:

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

A oto jak zdefiniowałbyś tę samą funkcję dla GPT-4, przekazując ją jako wartość parametru `tools`:

```
{
    "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"],
```

```
    },
}
```

> Prawie to samo, tylko inne bez wyraźnego powodu! Jakie to irytujące.

Definicje funkcji określają nazwę, opis i parametry wejściowe. Parametry wejściowe można dodatkowo zdefiniować za pomocą atrybutów, takich jak wyliczenia ograniczające dopuszczalne wartości, oraz określić, czy dany parametr jest wymagany, czy nie.

Oprócz właściwych definicji funkcji, możesz również zawrzeć w dyrektywie systemowej instrukcje lub kontekst wyjaśniający, dlaczego i jak używać funkcji.

Na przykład, moje narzędzie wyszukiwania internetowego w Olympii zawiera tę dyrektywę systemową, która przypomina SI, że ma do dyspozycji wspomniane narzędzia:

```
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.
```

Dostarczanie szczegółowych opisów jest uważane za najważniejszy czynnik wpływający na wydajność narzędzia. Twoje opisy powinny wyjaśniać każdy szczegół dotyczący narzędzia, w tym:

- Co narzędzie robi
- Kiedy powinno być używane (a kiedy nie)
- Co oznacza każdy parametr i jak wpływa na działanie narzędzia
- Wszelkie istotne zastrzeżenia lub ograniczenia dotyczące implementacji narzędzia

Im więcej kontekstu możesz dostarczyć AI na temat swoich narzędzi, tym lepiej będzie ono w stanie decydować, kiedy i jak ich używać. Na przykład, Anthropic zaleca co najmniej 3-4 zdania opisu na każde narzędzie dla swojej serii Claude 3, więcej jeśli narzędzie jest złożone.

Nie jest to może intuicyjne, ale opisy są również uważane za ważniejsze niż przykłady. Chociaż możesz zawrzeć przykłady użycia narzędzia w jego opisie lub w towarzyszącym prompcie, jest to mniej istotne niż posiadanie jasnego i kompleksowego wyjaśnienia celu i parametrów narzędzia. Dodawaj przykłady dopiero po pełnym opracowaniu opisu.

Oto przykład specyfikacji funkcji API w stylu Stripe:

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

W praktyce niektóre modele mają problemy z obsługą zagnieżdżonych specyfikacji funkcji i złożonych typów danych wyjściowych, takich jak tablice, słowniki itp. Ale teoretycznie powinieneś być w stanie dostarczać specyfikacje JSON Schema o dowolnej głębokości!

Dynamiczny Wybór Narzędzi

Gdy wykonujesz uzupełnienie czatu, które zawiera definicje narzędzi, model LLM dynamicznie wybiera najbardziej odpowiednie narzędzie(-a) do użycia i generuje wymagane parametry wejściowe dla każdego narzędzia.

W praktyce zdolność AI do wywoływania *dokładnie* właściwej funkcji i *dokładnego* przestrzegania twojej specyfikacji dla parametrów wejściowych bywa różna.

Ustawienie parametru temperature na 0.0 znacznie pomaga, ale z mojego doświadczenia wynika, że nadal będziesz napotykać sporadyczne błędy. Te niepowodzenia obejmują halucynowane nazwy funkcji, błędnie nazwane lub po prostu brakujące parametry wejściowe. Parametry są przekazywane jako JSON, co oznacza, że czasami zobaczysz błędy spowodowane obciętym, błędnie zacytowanym lub w inny sposób uszkodzonym kodem JSON.

Wzorce Samouzdrawiających się Danych mogą pomóc w automatycznym naprawianiu wywołań funkcji, które przestają działać z powodu błędów składni.

Wymuszony (tzw. Jawny) Wybór Narzędzi

Niektóre modele dają możliwość wymuszenia wywołania konkretnej funkcji jako parametr w żądaniu. W przeciwnym razie, decyzja o tym, czy wywołać funkcję, czy nie, należy całkowicie do uznania AI.

Możliwość wymuszenia wywołania funkcji jest kluczowa w niektórych scenariuszach, gdzie chcesz zapewnić wykonanie konkretnego narzędzia lub funkcji, niezależnie od procesu dynamicznego wyboru AI. Istnieje kilka powodów, dla których ta funkcjonalność jest ważna:

1. **Jawna Kontrola**: Możesz używać AI jako *Komponentu Dyskretnego* lub w predefiniowanym przepływie pracy, który wymaga wykonania konkretnej funkcji w konkretnym momencie. Wymuszając wywołanie, możesz zagwarantować, że pożądana funkcja zostanie wywołana, zamiast musieć grzecznie prosić AI o jej wykonanie.
2. **Debugowanie i Testowanie**: Podczas rozwijania i testowania aplikacji opartych na AI, możliwość wymuszania wywołań funkcji jest nieoceniona do celów debugowania. Poprzez jawne wyzwalanie określonych funkcji możesz izolować

i testować pojedyncze komponenty swojej aplikacji. Pozwala to na weryfikację poprawności implementacji funkcji, walidację parametrów wejściowych i upewnienie się, że zwracane są oczekiwane wyniki.

3. **Obsługa przypadków brzegowych**: Mogą wystąpić przypadki brzegowe lub wyjątkowe scenariusze, w których dynamiczny proces wyboru AI może nie zdecydować się na wykonanie funkcji, którą powinien wykonać, co wiesz na podstawie procesów zewnętrznych. W takich sytuacjach możliwość wymuszenia wywołania funkcji pozwala na jawną obsługę tych przypadków. Zdefiniuj reguły lub warunki w logice aplikacji, aby określić, kiedy należy przełamać decyzję AI.
4. **Spójność i odtwarzalność**: Jeśli masz określoną sekwencję funkcji, które muszą być wykonane w konkretnej kolejności, wymuszenie wywołań gwarantuje, że ta sama sekwencja będzie przestrzegana za każdym razem. Jest to szczególnie ważne w aplikacjach, gdzie spójność i przewidywalne zachowanie są kluczowe, jak na przykład w systemach finansowych czy symulacjach naukowych.
5. **Optymalizacja wydajności**: W niektórych przypadkach wymuszenie wywołania funkcji może prowadzić do optymalizacji wydajności. Jeśli wiesz, że konkretna funkcja jest wymagana do określonego zadania, a dynamiczny proces wyboru AI mógłby wprowadzić niepotrzebne obciążenie, możesz pominąć proces wyboru i bezpośrednio wywołać wymaganą funkcję. Może to pomóc zmniejszyć opóźnienia i poprawić ogólną wydajność aplikacji.

Podsumowując, możliwość wymuszania wywołań funkcji w aplikacjach opartych na AI zapewnia jawną kontrolę, pomaga w debugowaniu i testowaniu, obsługuje przypadki brzegowe, zapewnia spójność i odtwarzalność. To potężne narzędzie w twoim arsenale, ale musimy omówić jeszcze jeden aspekt tej ważnej funkcjonalności.

W wielu przypadkach użycia związanych z podejmowaniem decyzji, zawsze chcemy, aby model wykonał wywołanie funkcji i może nigdy nie chcemy, aby model odpowiadał tylko swoją wewnętrzną wiedzą. Na przykład, jeśli kierujesz ruch między wieloma modelami wyspecjalizowanymi w różnych zadaniach (dane wielojęzyczne, matematyka itp.), możesz użyć modelu wywołującego funkcje do delegowania żądań do jednego z modeli pomocniczych i nigdy nie odpowiadać samodzielnie.

Parametr wyboru narzędzia

GPT-4 i inne modele językowe obsługujące wywoływanie funkcji dają ci parametr `tool_choice` do kontrolowania, czy użycie narzędzia jest wymagane jako część uzupełnienia. Ten parametr ma trzy możliwe wartości:

- `auto` daje AI pełną swobodę w używaniu narzędzia lub po prostu odpowiadaniu
- `required` mówi AI, że *musi* wywołać narzędzie *zamiast* odpowiadać, ale pozostawia wybór narzędzia w gestii AI
- Trzecią opcją jest ustawienie parametru `name_of_function`, który chcesz wymusić. Więcej na ten temat w następnej sekcji.

Zauważ, że jeśli ustawisz wybór narzędzia (tool choice) na `required`, model będzie zmuszony wybrać najbardziej odpowiednią funkcję do wywołania spośród dostępnych, nawet jeśli żadna tak naprawdę nie pasuje do polecenia. W momencie publikacji nie znam modelu, który zwróciłby pustą odpowiedź `tool_calls` lub w inny sposób poinformował, że nie znalazł odpowiedniej funkcji do wywołania.

Wymuszanie Wywołania Funkcji dla Uzyskania Ustrukturyzowanych Danych

Możliwość wymuszenia wywołania funkcji daje sposób na wydobycie ustrukturyzowanych danych z uzupełnienia czatu, zamiast konieczności samodzielnego wyodrębniania ich z odpowiedzi tekstowej.

> Dlaczego wymuszanie funkcji w celu uzyskania ustrukturyzowanych danych jest tak ważne? Mówiąc prosto, ponieważ ekstrakcja ustrukturyzowanych danych z wyników LLM jest uciążliwa. Możesz nieco ułatwić sobie życie, prosząc o dane w formacie XML, ale wtedy musisz parsować XML. A co zrobić, gdy tego XML-a brakuje, bo AI odpowiedziało: "Przepraszam, ale nie mogę wygenerować żądanych danych, ponieważ bla, bla, bla..."

Podczas używania narzędzi w ten sposób:

- Prawdopodobnie powinieneś zdefiniować pojedyncze narzędzie w swoim żądaniu
- Pamiętaj o wymuszeniu użycia jego funkcji za pomocą parametru `tool_choice`.
- Pamiętaj, że model przekaże dane wejściowe do narzędzia, więc nazwa narzędzia i opis powinny być z perspektywy modelu, nie twojej.

Ten ostatni punkt zasługuje na przykład dla jasności. Powiedzmy, że prosisz AI o przeprowadzenie analizy sentymentu tekstu użytkownika. Nazwa funkcji nie byłaby `analyze_sentiment`, ale raczej coś w rodzaju `save_sentiment_analysis`. To AI przeprowadza analizę sentymentu, a *nie narzędzie*. Wszystko, co robi narzędzie (z perspektywy AI), to zapisywanie wyników analizy.

Oto przykład użycia Claude 3 do zapisania podsumowania obrazu w dobrze ustrukturyzowanym formacie JSON, tym razem z wiersza poleceń przy użyciu `curl`:

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

W przedstawionym przykładzie wykorzystujemy model Claude 3 firmy Anthropic do wygenerowania ustrukturyzowanego podsumowania obrazu w formacie JSON. Oto jak to działa:

1. Definiujemy pojedyncze narzędzie o nazwie `record_summary` w tablicy `tools` w ładunku żądania. To narzędzie odpowiada za zapisanie podsumowania obrazu w dobrze ustrukturyzowanym formacie JSON.
2. Narzędzie `record_summary` posiada `input_schema`, który określa oczekiwaną strukturę wyjściowego JSONa. Definiuje on trzy właściwości:

    - `key_colors`: Tablica obiektów reprezentujących kluczowe kolory w obrazie. Każdy obiekt koloru posiada właściwości określające wartości czerwonego, zielonego i niebieskiego (w zakresie od 0.0 do 1.0) oraz czytelną dla człowieka nazwę koloru w formacie snake_case.
    - `description`: Właściwość typu string zawierająca krótki opis obrazu, ograniczony do 1-2 zdań.
    - `estimated_year`: Opcjonalna właściwość typu integer określająca szacowany rok wykonania zdjęcia, jeśli wydaje się ono być rzeczywistą fotografią.

3. W tablicy `messages` dostarczamy dane obrazu jako ciąg zakodowany w base64 wraz z typem medium. Pozwala to modelowi na przetworzenie obrazu jako części danych wejściowych.
4. Polecamy również Claude'owi użycie narzędzia `record_summary` do opisania obrazu.
5. Gdy żądanie zostaje wysłane do modelu Claude 3, analizuje on obraz i generuje podsumowanie JSON w oparciu o określony `input_schema`. Model wyodrębnia kluczowe kolory, dostarcza krótki opis i szacuje rok wykonania zdjęcia (jeśli ma to zastosowanie).
6. Wygenerowane podsumowanie JSON jest przekazywane jako parametry do narzędzia `record_summary`, zapewniając ustrukturyzowaną reprezentację kluczowych cech obrazu.

Wykorzystując narzędzie `record_summary` z dobrze zdefiniowanym `input_schema`, możemy uzyskać ustrukturyzowane podsumowanie obrazu w formacie JSON bez

polegania na ekstrakcji zwykłego tekstu. To podejście zapewnia, że dane wyjściowe mają spójny format i mogą być łatwo analizowane i przetwarzane przez komponenty końcowe aplikacji.

Możliwość wymuszenia wywołania funkcji i określenia oczekiwanej struktury wyjściowej jest potężną funkcją wykorzystania narzędzi w aplikacjach opartych na AI. Pozwala to programistom na większą kontrolę nad generowanymi danymi wyjściowymi i upraszcza integrację danych generowanych przez AI z przepływem pracy ich aplikacji.

Wykonywanie Funkcji

Zdefiniowałeś funkcje i wysłałeś zapytanie do swojej sztucznej inteligencji, która zdecydowała, że powinna wywołać jedną z twoich funkcji. Teraz nadszedł czas, aby twój kod aplikacji lub biblioteka, jeśli używasz gema Ruby takiego jak `raix-rails`, przekazała wywołanie funkcji i jej parametry do odpowiedniej implementacji *w kodzie twojej aplikacji.*

Twój kod aplikacji decyduje, co zrobić z wynikami wykonania funkcji. Może to być pojedyncza linia kodu w lambdzie, albo może wymagać wywołania zewnętrznego API. Może wiązać się z wywołaniem innego komponentu AI, lub może obejmować setki czy nawet tysiące linii kodu w pozostałej części twojego systemu. To całkowicie zależy od ciebie.

Czasami wywołanie funkcji jest końcem operacji, ale jeśli wyniki reprezentują informacje w łańcuchu myślowym, które mają być kontynuowane przez AI, wtedy twój kod aplikacji musi wstawić wyniki wykonania do transkryptu czatu i pozwolić AI kontynuować przetwarzanie.

Na przykład, oto deklaracja funkcji Raix używana przez `AccountManager` Olympii do komunikacji z naszymi klientami jako część inteligentnej orkiestracji przepływu pracy w obsłudze klienta.

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

  # lots of other functions...

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

Może nie być od razu jasne, co się tutaj dzieje, więc wyjaśnię to krok po kroku.

1. Klasa `AccountManager` definiuje wiele funkcji związanych z zarządzaniem kontem. Może zmieniać plan, dodawać i usuwać członków zespołu, między innymi.
2. Jej instrukcje najwyższego poziomu informują `AccountManager`, że powinien powiadomić właściciela konta o wynikach żądania zmiany konta, używając funkcji `notify_account_owner`.
3. Zwięzła definicja funkcji zawiera jej:

- nazwę
- opis
- parametry `message: { type: "string" }`
- blok do wykonania, gdy funkcja jest wywoływana

Po zaktualizowaniu transkryptu wynikami bloku funkcji, metoda `chat_completion` jest wywoływana ponownie. Ta metoda jest odpowiedzialna za wysłanie zaktualizowanego transkryptu rozmowy z powrotem do modelu AI do dalszego przetwarzania. Nazywamy ten proces *pętlą konwersacji*.

Gdy model AI otrzymuje nowe żądanie chat completion z zaktualizowanym transkryptem, ma dostęp do wyników poprzednio wykonanej funkcji. Może analizować te wyniki, włączyć je do swojego procesu decyzyjnego i wygenerować następną odpowiedź lub działanie na podstawie skumulowanego kontekstu rozmowy. Może zdecydować się na wykonanie dodatkowych funkcji w oparciu o zaktualizowany kontekst lub wygenerować końcową odpowiedź na pierwotny prompt, jeśli uzna, że dalsze wywołania funkcji nie są konieczne.

Opcjonalna kontynuacja pierwotnego promptu

Gdy wysyłasz wyniki narzędzia z powrotem do LLM i kontynuujesz przetwarzanie pierwotnego promptu, AI wykorzystuje te wyniki do wywołania dodatkowych funkcji lub wygenerowania końcowej odpowiedzi w formie zwykłego tekstu.

Niektóre modele, takie jak Command-R od Cohere, potrafią cytować konkretne narzędzia użyte w swoich odpowiedziach, zapewniając dodatkową przejrzystość i możliwość śledzenia.

W zależności od używanego modelu, wyniki wywołania funkcji będą znajdować się w wiadomościach transkryptu, które mają swoją własną specjalną rolę, lub będą odzwierciedlone w innej składni. Jednak najważniejsze jest, aby te dane znajdowały się w transkrypcie, żeby AI mogło je uwzględnić przy podejmowaniu decyzji o kolejnych krokach.

Częstym (i potencjalnie kosztownym) błędem jest zapomnienie o dodaniu wyników funkcji do transkryptu przed kontynuacją czatu. W rezultacie AI otrzyma prompt zasadniczo taki sam, jak przed pierwszym wywołaniem funkcji. Innymi słowy, z perspektywy AI, funkcja jeszcze nie została wywołana. Więc wywołuje ją ponownie. I ponownie. I ponownie, w nieskończoność, dopóki nie przerwiesz. Miejmy nadzieję, że kontekst nie był zbyt duży, a model nie był zbyt drogi!

Najlepsze praktyki używania narzędzi

Aby jak najlepiej wykorzystać narzędzia, weź pod uwagę następujące najlepsze praktyki.

Opisowe definicje

Zapewnij jasne i opisowe nazwy oraz opisy dla każdego narzędzia i jego parametrów wejściowych. Pomaga to modelowi językowemu lepiej zrozumieć cel i możliwości każdego narzędzia.

> Z doświadczenia mogę powiedzieć, że popularne powiedzenie "nazewnictwo jest trudne" ma tu zastosowanie; widziałem dramatycznie różne rezultaty w działaniu modeli językowych tylko poprzez zmianę nazw funkcji czy sformułowania opisów. Czasami usunięcie opisów *poprawia* wydajność.

Przetwarzanie wyników narzędzi

Przekazując wyniki narzędzi z powrotem do modelu językowego, upewnij się, że są dobrze ustrukturyzowane i kompleksowe. Używaj znaczących kluczy i wartości do reprezentowania wyników każdego narzędzia. Eksperymentuj z różnymi formatami i sprawdź, który działa najlepiej, od JSON po zwykły tekst.

Interpreter wyników odpowiada na to wyzwanie, wykorzystując AI do analizy wyników i dostarczania przyjaznych dla człowieka wyjaśnień, podsumowań lub kluczowych wniosków.

Obsługa błędów

Zaimplementuj solidne mechanizmy obsługi błędów, aby poradzić sobie z przypadkami, gdy model językowy może wygenerować nieprawidłowe lub nieobsługiwane parametry wejściowe dla wywołań narzędzi. Elegancko obsługuj i odzyskuj sprawność po wszelkich błędach, które mogą wystąpić podczas wykonywania narzędzia.

Jedną niezwykle pozytywną cechą AI jest to, że rozumie komunikaty o błędach! Oznacza to, że jeśli pracujesz w trybie szybkim i niezbyt eleganckim, możesz po prostu przechwycić wszelkie wyjątki generowane podczas implementacji narzędzia i przekazać je z powrotem do AI, aby wiedziała, co się stało!

Na przykład, oto uproszczona wersja implementacji wyszukiwania Google w Olympii:

```
  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
```

Wyszukiwania Google w Olympii to proces dwuetapowy. Najpierw wykonujesz wyszukiwanie, a następnie podsumowujesz wyniki. Jeśli wystąpi błąd, niezależnie od jego rodzaju, komunikat o błędzie jest pakowany i odsyłany do SI. Ta technika stanowi podstawę praktycznie wszystkich wzorców *Inteligentnej Obsługi Błędów*.

Na przykład, powiedzmy, że wywołanie API `GoogleSearch` nie powiedzie się z powodu błędu 503 Usługa Niedostępna. Ten błąd przechodzi do głównej obsługi wyjątków, a opis błędu jest odsyłany do SI jako wynik wywołania funkcji. Zamiast pokazywać użytkownikowi pusty ekran lub techniczny błąd, SI mówi coś w rodzaju

“Przepraszam, ale w tej chwili nie mogę uzyskać dostępu do moich możliwości wyszukiwania Google. Mogę spróbować ponownie później, jeśli chcesz.”

Może się to wydawać jedynie sprytnym trikiem, ale rozważ inny rodzaj błędu, taki gdzie SI wywołuje zewnętrzne API i ma bezpośrednią kontrolę nad parametrami przekazywanymi do API. Może popełniła błąd w sposobie generowania tych parametrów? Zakładając, że komunikat o błędzie z zewnętrznego API jest wystarczająco szczegółowy, przekazanie komunikatu o błędzie z powrotem do wywołującej SI oznacza, że może ona przemyśleć te parametry i spróbować ponownie. Automatycznie. Niezależnie od tego, jaki był błąd.

Teraz pomyśl, ile wysiłku wymagałoby odtworzenie tego rodzaju solidnej obsługi błędów w *normalnym* kodzie. To praktycznie niemożliwe.

Iteracyjne Udoskonalanie

Jeśli LLM nie zaleca odpowiednich narzędzi lub generuje nieoptymalne odpowiedzi, należy iteracyjnie dopracowywać definicje narzędzi, opisy i parametry wejściowe. Ciągle udoskonalaj i ulepszaj konfigurację narzędzi w oparciu o zaobserwowane zachowanie i pożądane rezultaty.

1. Zacznij od prostych definicji narzędzi: Rozpocznij od zdefiniowania narzędzi z jasnymi i zwięzłymi nazwami, opisami i parametrami wejściowymi. Unikaj początkowo nadmiernego komplikowania konfiguracji narzędzi i skup się na podstawowej funkcjonalności. Na przykład, jeśli chcesz zapisać wyniki analizy sentymentu, zacznij od podstawowej definicji, takiej jak:

```
{
  "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. Testuj i obserwuj: Po utworzeniu początkowych definicji narzędzi, przetestuj je różnymi poleceniami i obserwuj, jak LLM wchodzi w interakcję z narzędziem. Zwróć uwagę na jakość i trafność generowanych odpowiedzi. Jeśli LLM generuje nieoptymalne odpowiedzi, czas na doprecyzowanie definicji narzędzi.
3. Udoskonal opisy: Jeśli LLM błędnie interpretuje przeznaczenie narzędzia, spróbuj doprecyzować jego opis. Dostarcz więcej kontekstu, przykładów lub wyjaśnień, aby pokierować LLM w efektywnym wykorzystaniu narzędzia. Na przykład, możesz zaktualizować opis narzędzia do analizy sentymentu, aby bardziej szczegółowo odnosił się do *tonu emocjonalnego* analizowanego tekstu:

```
{
  "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. Dostosuj parametry wejściowe: Jeśli LLM generuje nieprawidłowe lub nieistotne parametry wejściowe dla narzędzia, rozważ dostosowanie definicji parametrów. Dodaj bardziej szczegółowe ograniczenia, reguły walidacji lub przykłady, aby wyjaśnić oczekiwany format danych wejściowych.

5. Wprowadzaj zmiany na podstawie informacji zwrotnej: Stale monitoruj wydajność swoich narzędzi i zbieraj opinie od użytkowników lub interesariuszy. Wykorzystuj te informacje do identyfikacji obszarów wymagających poprawy i wprowadzaj iteracyjne udoskonalenia w definicjach narzędzi. Na przykład, jeśli użytkownicy zgłaszają, że analiza nie radzi sobie dobrze z sarkazmem, możesz dodać odpowiednią uwagę w opisie:

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

Poprzez iteracyjne udoskonalanie definicji narzędzi w oparciu o zaobserwowane zachowania i informacje zwrotne, możesz stopniowo poprawiać wydajność i skuteczność swojej aplikacji opartej na AI. Pamiętaj, aby definicje narzędzi były jasne, zwięzłe i skoncentrowane na konkretnym zadaniu. Regularnie testuj i waliduj interakcje narzędzi, aby upewnić się, że są zgodne z oczekiwanymi rezultatami.

Komponowanie i łączenie narzędzi

Jednym z najpotężniejszych aspektów wykorzystania narzędzi, o którym dotychczas tylko wspominano, jest możliwość komponowania i łączenia wielu narzędzi w celu realizacji złożonych zadań. Poprzez staranne projektowanie definicji narzędzi i ich formatów wejścia/wyjścia, możesz tworzyć wielokrotnego użytku komponenty, które można łączyć na różne sposoby.

Rozważmy przykład, w którym budujesz potok analizy danych dla swojej aplikacji opartej na AI. Możesz mieć następujące narzędzia:

1. **`DataRetrieval`**: Narzędzie, które pobiera dane z bazy danych lub API na podstawie określonych kryteriów.
2. **`DataProcessing`**: Narzędzie, które wykonuje obliczenia, transformacje lub agregacje na pobranych danych.
3. **`DataVisualization`**: Narzędzie, które przedstawia przetworzone dane w przyjaznym dla użytkownika formacie, takim jak wykresy lub grafy.

Łącząc te narzędzia ze sobą, możesz stworzyć potężny przepływ pracy, który pobiera odpowiednie dane, przetwarza je i przedstawia wyniki w znaczący sposób. Oto jak może wyglądać przepływ pracy z wykorzystaniem narzędzi:

1. LLM otrzymuje zapytanie użytkownika o wgląd w dane sprzedażowe dla określonej kategorii produktów.
2. LLM wybiera narzędzie `DataRetrieval` i generuje odpowiednie parametry wejściowe, aby pobrać właściwe dane sprzedażowe z bazy danych.
3. Pobrane dane są "przekazywane" do narzędzia `DataProcessing`, które oblicza metryki takie jak całkowity przychód, średnia cena sprzedaży i wskaźnik wzrostu.
4. Przetworzone dane są następnie przetwarzane przez narzędzie `DataVisualization`, które tworzy atrakcyjny wizualnie wykres lub graf do reprezentacji wyników, przekazując URL wykresu z powrotem do LLM.
5. Na koniec, LLM generuje sformatowaną odpowiedź na zapytanie użytkownika przy użyciu markdown, zawierającą zwizualizowane dane i podsumowanie kluczowych ustaleń.

Komponując te narzędzia razem, możesz stworzyć płynny przepływ analizy danych, który można łatwo zintegrować z twoją aplikacją. Piękno tego podejścia polega na tym, że każde narzędzie może być rozwijane i testowane niezależnie, a następnie łączone na różne sposoby w celu rozwiązywania różnorodnych problemów.

Aby umożliwić płynne komponowanie i łączenie narzędzi, ważne jest zdefiniowanie jasnych formatów wejściowych i wyjściowych dla każdego narzędzia.

Na przykład, narzędzie `DataRetrieval` może przyjmować parametry takie jak szczegóły połączenia z bazą danych, nazwa tabeli i warunki zapytania, a zwracać zbiór wyników jako ustrukturyzowany obiekt JSON. Narzędzie `DataProcessing` może następnie oczekiwać tego obiektu JSON jako danych wejściowych i generować przekształcony obiekt JSON jako wynik. Poprzez standaryzację przepływu danych między narzędziami można zapewnić kompatybilność i możliwość ponownego wykorzystania.

Projektując swój ekosystem narzędzi, zastanów się, jak różne narzędzia można łączyć w celu obsługi typowych przypadków użycia w twojej aplikacji. Rozważ stworzenie narzędzi wysokiego poziomu, które enkapsulują często występujące przepływy pracy lub logikę biznesową, ułatwiając LLM ich efektywny wybór i wykorzystanie.

Pamiętaj, że siła wykorzystania narzędzi tkwi w elastyczności i modularności, jaką zapewnia. Dzieląc złożone zadania na mniejsze, wielokrotnego użytku narzędzia, możesz stworzyć solidną i adaptowalną aplikację opartą na AI, która może sprostać szerokiemu zakresowi wyzwań.

Przyszłe Kierunki

Wraz z rozwojem dziedziny tworzenia aplikacji opartych na AI, możemy spodziewać się dalszych postępów w możliwościach wykorzystania narzędzi. Niektóre potencjalne kierunki rozwoju obejmują:

1. **Wieloetapowe Wykorzystanie Narzędzi**: LLM-y mogą być w stanie decydować, ile razy muszą użyć narzędzi, aby wygenerować satysfakcjonującą odpowiedź. Może to obejmować wielokrotne rundy wyboru i wykonania narzędzi w oparciu o wyniki pośrednie.
2. **Predefiniowane Narzędzia**: Platformy AI mogą dostarczać zestaw predefiniowanych narzędzi, które programiści mogą wykorzystać od razu

po instalacji, takich jak interpretery Pythona, narzędzia do wyszukiwania w sieci czy popularne funkcje użytkowe.

3. **Bezproblemowa Integracja**: Wraz z rosnącą popularnością wykorzystania narzędzi, możemy spodziewać się lepszej integracji między platformami AI a popularnymi frameworkami programistycznymi, ułatwiając programistom włączanie wykorzystania narzędzi do ich aplikacji.

Wykorzystanie narzędzi jest potężną techniką, która umożliwia programistom wykorzystanie pełnego potencjału LLM-ów w aplikacjach opartych na AI. Łącząc LLM-y z zewnętrznymi narzędziami i zasobami, możesz tworzyć bardziej dynamiczne, inteligentne i świadome kontekstu systemy, które mogą adaptować się do potrzeb użytkowników oraz dostarczać wartościowych spostrzeżeń i działań.

Chociaż wykorzystanie narzędzi oferuje ogromne możliwości, ważne jest, aby być świadomym potencjalnych wyzwań i kwestii do rozważenia. Jednym z kluczowych aspektów jest zarządzanie złożonością interakcji między narzędziami i zapewnienie stabilności oraz niezawodności całego systemu. Musisz obsłużyć scenariusze, w których wywołania narzędzi mogą się nie powieść, zwrócić nieoczekiwane wyniki lub mieć wpływ na wydajność. Dodatkowo powinieneś rozważyć środki bezpieczeństwa i kontroli dostępu, aby zapobiec nieuprawnionemu lub złośliwemu wykorzystaniu narzędzi. Odpowiednia obsługa błędów, logowanie i mechanizmy monitorowania są kluczowe dla utrzymania integralności i wydajności aplikacji opartej na AI.

Podczas eksplorowania możliwości wykorzystania narzędzi w Twoich własnych projektach, pamiętaj, aby zacząć od jasnych celów, projektować dobrze ustrukturyzowane definicje narzędzi i iterować na podstawie informacji zwrotnej i wyników. Przy właściwym podejściu i nastawieniu, wykorzystanie narzędzi może odblokować nowe poziomy innowacji i wartości w Twoich aplikacjach opartych na sztucznej inteligencji

Przetwarzanie strumieniowe

Strumieniowanie danych przez HTTP, znane również jako zdarzenia wysyłane przez serwer (SSE), to mechanizm, w którym serwer nieprzerwanie wysyła dane do klienta w miarę ich dostępności, bez potrzeby jawnego żądania ze strony klienta. Ponieważ odpowiedź AI jest generowana przyrostowo, sensowne jest zapewnienie responsywnego doświadczenia użytkownika poprzez wyświetlanie wyników AI w miarę ich generowania. I faktycznie, wszystkie znane mi interfejsy API dostawców AI oferują strumieniowe odpowiedzi jako opcję w ich punktach końcowych do uzupełniania.

Powód, dla którego ten rozdział pojawia się w tym miejscu książki, zaraz po Using Tools, wynika z tego, jak potężne może być połączenie używania narzędzi z odpowiedziami AI w czasie rzeczywistym dla użytkowników. Umożliwia to tworzenie dynamicznych i interaktywnych doświadczeń, gdzie AI może przetwarzać dane wejściowe użytkownika, wykorzystywać różne narzędzia i funkcje według

własnego uznania, a następnie dostarczać odpowiedzi w czasie rzeczywistym.

Aby osiągnąć tę płynną interakcję, musisz napisać procedury obsługi strumienia, które mogą wysyłać zarówno wywołania funkcji narzędziowych AI, jak i zwykły tekst wyjściowy do użytkownika końcowego. Konieczność wykonywania pętli po przetworzeniu funkcji narzędziowej dodaje do tego zadania interesujące wyzwanie.

Implementacja ReplyStream

Aby zademonstrować, jak można zaimplementować przetwarzanie strumieniowe, ten rozdział szczegółowo omówi uproszczoną wersję klasy `ReplyStream`, która jest używana w Olympii. Instancje tej klasy mogą być przekazywane jako parametr `stream` w bibliotekach klienckich AI, takich jak ruby-openai i openrouter

Oto jak używam `ReplyStream` w `PromptSubscriber` Olympii, który nasłuchuje poprzez Wisper tworzenia nowych wiadomości użytkownika.

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

  # many other declarations omitted...

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

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

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

Oprócz referencji `context` do subskrybenta promptu, który ją zainicjował, klasa `ReplyStream` posiada również zmienne instancyjne do przechowywania bufora otrzymanych danych oraz tablice do śledzenia nazw funkcji i argumentów wywołanych podczas przetwarzania strumienia.

```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
```

Metoda `initialize` konfiguruje stan początkowy instancji `ReplyStream`, inicjalizując bufor, kontekst i inne zmienne.

Metoda `call` jest głównym punktem wejścia do przetwarzania danych strumieniowych. Przyjmuje ona `chunk` danych (reprezentowany jako tablica asocjacyjna) oraz opcjonalny parametr `bytesize`, który w naszym przykładzie nie jest wykorzystywany. Wewnątrz tej metody klasa wykorzystuje dopasowywanie wzorców do obsługi różnych scenariuszy w zależności od struktury otrzymanego fragmentu danych.

Wywołanie `deep_symbolize_keys` na fragmencie danych pomaga uczynić dopasowywanie wzorców bardziej eleganckim, pozwalając nam operować na symbolach zamiast na ciągach znaków.

```
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
```

Pierwszym wzorcem, który dopasowujemy, jest wywołanie narzędzia wraz z powiązaną nazwą funkcji. Jeśli je wykryjemy, umieszczamy je w tablicy `f_name`. Przechowujemy nazwy funkcji w tablicy indeksowanej, ponieważ model jest zdolny do równoległego wywoływania funkcji, wysyłając więcej niż jedną funkcję do wykonania jednocześnie.

> Równoległe wywoływanie funkcji to zdolność modelu AI do wykonywania wielu wywołań funkcji jednocześnie, pozwalająca na równoległe rozwiązywanie efektów i wyników tych wywołań. Jest to szczególnie przydatne, gdy funkcje zajmują dużo czasu, i redukuje komunikację dwustronną z API, co z kolei może znacząco zmniejszyć zużycie tokenów.

Następnie musimy dopasować argumenty odpowiadające wywołaniom funkcji.

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

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

Podobnie jak w przypadku nazw funkcji, umieszczamy argumenty w tablicy indeksowanej.

Następnie zajmujemy się zwykłymi komunikatami dla użytkownika, które będą przychodzić z serwera token po tokenie i będą przypisywane do zmiennej `new_content`. Musimy również obserwować `finish_reason`. Będzie on miał wartość `nil` aż do ostatniego fragmentu sekwencji wyjściowej.

```
  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
```

Co istotne, dodajemy wyrażenie dopasowania wzorca do obsługi komunikatów błędów

wysyłanych przez dostawcę modelu AI. W lokalnych środowiskach programistycznych zgłaszamy wyjątek, ale w środowisku produkcyjnym logujemy błąd i finalizujemy.

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

Ostatnia klauzula else w instrukcji case zostanie wykonana, jeśli żaden z poprzednich wzorców nie zostanie dopasowany. To po prostu zabezpieczenie, dzięki któremu dowiemy się, gdy model AI zacznie wysyłać nam nierozpoznane fragmenty.

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

Metoda `send_to_client` odpowiada za wysyłanie zbuforowanej zawartości do klienta. Sprawdza, czy bufor nie jest pusty, aktualizuje zawartość wiadomości bota, renderuje wiadomość bota i zapisuje zawartość w bazie danych w celu zapewnienia trwałości danych.

```
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
```

Metoda `finalize` jest wywoływana po zakończeniu przetwarzania strumienia. Przekazuje wywołania funkcji, jeśli jakiekolwiek zostały otrzymane podczas strumienia, aktualizuje wiadomość bota z końcową zawartością i innymi istotnymi informacjami oraz resetuje historię wywołań funkcji

```
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
```

Jeśli model zdecyduje się wywołać funkcję, musisz “wysłać” to wywołanie funkcji

(nazwę i argumenty) w taki sposób, aby zostało ono wykonane, a komunikaty `function_call` i `function_result` zostały dodane do transkryptu rozmowy

Z mojego doświadczenia wynika, że lepiej jest obsługiwać tworzenie komunikatów funkcji w jednym miejscu w bazie kodu, zamiast polegać na implementacjach narzędzi. Jest to nie tylko bardziej przejrzyste, ale ma też bardzo ważny praktyczny powód: jeśli model AI wywoła funkcję i nie zobaczy wynikających z tego wywołania komunikatów w transkrypcie podczas pętli, *wywoła tę samą funkcję ponownie*. Potencjalnie w nieskończoność. Pamiętaj, że AI jest całkowicie bezstanowe, więc dopóki nie przekażesz mu z powrotem tych wywołań funkcji, to tak jakby się nie wydarzyły.

```
# 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
```

Czyszczenie historii wywołań funkcji po ich wykonaniu jest równie ważne jak upewnienie się, że wywołanie i jego wyniki trafiają do transkryptu, dzięki czemu nie będziesz ciągle wywoływać tych samych funkcji w kółko przy każdym przebiegu pętli.

“Pętla konwersacji”

W klasie `PromptSubscriber` używamy metody `prompt` z modułu `PromptDeclarations` do zdefiniowania zachowania pętli konwersacji. Parametr

`until` jest ustawiony na `-> { bot_message.complete? }`, co oznacza, że pętla będzie kontynuowana dopóki `bot_message` nie zostanie oznaczony jako zakończony.

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

A kiedy `bot_message` jest oznaczana jako zakończona? Jeśli nie pamiętasz, sprawdź linię 13 metody `finalize`.

Przeanalizujmy całą logikę przetwarzania strumieniowego.

1. `PromptSubscriber` otrzymuje nową wiadomość użytkownika przez metodę `message_created`, która jest wywoływana przez system pub/sub Wisper za każdym razem, gdy użytkownik końcowy tworzy nowe zapytanie.
2. Metoda klasowa `prompt` w sposób deklaratywny definiuje zachowanie logiki uzupełniania czatu dla `PromptSubscriber`. Model AI wykona uzupełnienie czatu z treścią wiadomości użytkownika, nową instancją `ReplyStream` jako parametrem strumieniowym oraz określonym warunkiem pętli.
3. Model AI przetwarza zapytanie i zaczyna generować odpowiedź. W miarę strumieniowania odpowiedzi, metoda `call` instancji `ReplyStream` jest wywoływana dla każdego fragmentu danych.
4. Jeśli model AI zdecyduje się wywołać funkcję narzędziową, nazwa funkcji i argumenty są wyodrębniane z fragmentu i zapisywane odpowiednio w tablicach `f_name` i `f_arguments`.
5. Jeśli model AI generuje treść przeznaczoną dla użytkownika, jest ona buforowana i wysyłana do klienta za pomocą metody `send_to_client`.
6. Po zakończeniu przetwarzania strumienia, wywoływana jest metoda `finalize`. Jeśli podczas strumieniowania zostały wywołane jakiekolwiek funkcje narzędziowe, są one dystrybuowane przy użyciu metody `dispatch` klasy `PromptSubscriber`.

7. Metoda `dispatch` dodaje wiadomość `function_call` do transkryptu rozmowy, wykonuje odpowiednią funkcję narzędziową i dodaje wiadomość `function_result` do transkryptu z wynikiem wywołania funkcji.
8. Po dystrybuowaniu funkcji narzędziowych, historia wywołań funkcji jest czyszczona, aby zapobiec duplikatom wywołań funkcji w kolejnych pętlach.
9. Jeśli nie wywołano żadnych funkcji narzędziowych, metoda `finalize` aktualizuje `bot_message` końcową treścią, oznacza ją jako zakończoną i wysyła zaktualizowaną wiadomość do klienta.
10. Warunek pętli `-> { bot_message.complete? }` jest sprawdzany. Jeśli `bot_message` nie jest oznaczona jako zakończona, pętla jest kontynuowana, a pierwotne zapytanie jest ponownie przesyłane z zaktualizowanym transkryptem rozmowy.
11. Kroki 3-10 są powtarzane do momentu oznaczenia `bot_message` jako kompletnej, co wskazuje, że model AI zakończył generowanie odpowiedzi i nie ma potrzeby wykonywania kolejnych funkcji narzędziowych.

Poprzez implementację pętli konwersacji, umożliwiasz modelowi AI prowadzenie interakcji tam i z powrotem z aplikacją, wykonywanie funkcji narzędziowych według potrzeb i generowanie odpowiedzi dla użytkownika, aż do naturalnego zakończenia konwersacji.

Połączenie przetwarzania strumieniowego i pętli konwersacji pozwala na dynamiczne i interaktywne doświadczenia oparte na AI, gdzie model AI może przetwarzać dane wejściowe użytkownika, wykorzystywać różne narzędzia i funkcje oraz dostarczać odpowiedzi w czasie rzeczywistym na podstawie rozwijającego się kontekstu konwersacji.

Automatyczna Kontynuacja

Należy pamiętać o ograniczeniach wyjścia AI. Większość modeli ma maksymalną liczbę tokenów, które mogą wygenerować w pojedynczej odpowiedzi, co jest określone

przez parametr `max_tokens`. Jeśli model AI osiągnie ten limit podczas generowania odpowiedzi, nagle się zatrzyma i wskaże, że wyjście zostało przerwane.

W strumieniowej odpowiedzi z API platformy AI można wykryć tę sytuację, sprawdzając zmienną `finish_reason` w fragmencie. Jeśli `finish_reason` jest ustawiony na `"length"` (lub inną wartość kluczową specyficzną dla modelu), oznacza to, że model osiągnął swój maksymalny limit tokenów podczas generowania i wyjście zostało przerwane.

Jednym ze sposobów na eleganckie obsłużenie tego scenariusza i zapewnienie płynnego doświadczenia użytkownika jest zaimplementowanie mechanizmu automatycznej kontynuacji w logice przetwarzania strumieniowego. Dodając dopasowanie wzorca dla powodów zakończenia związanych z długością, możesz wybrać zapętlenie i automatyczne kontynuowanie wyjścia od miejsca, w którym zostało przerwane.

Oto celowo uproszczony przykład tego, jak można zmodyfikować metodę `call` w klasie `ReplyStream`, aby obsługiwać automatyczną kontynuację:

```
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
```

```
21
22      # ...
23    end
24  end
25
26  private
27
28  def continue_cutoff
29    conversation.bot_message!(buffer.join, visible: false)
30    conversation.user_message!("please continue", visible: false)
31    bot_message.update_column(:created_at, Time.current)
32  end
```

W tej zmodyfikowanej wersji, gdy `finish_reason` wskazuje na przerwane dane wyjściowe, zamiast finalizować strumień, dodajemy parę wiadomości do transkryptu bez finalizacji, przenosimy oryginalną wiadomość widoczną dla użytkownika na "spód" transkryptu poprzez aktualizację jej atrybutu `created_at`, a następnie pozwalamy pętli się wykonać, dzięki czemu SI kontynuuje generowanie w miejscu, w którym przerwała.

Pamiętaj, że punkt końcowy uzupełniania SI jest bezstanowy. "Wie" on tylko to, co przekażesz mu przez transkrypt. W tym przypadku, sposób w jaki komunikujemy SI, że została przerwana, polega na dodaniu "niewidocznych" (dla końcowego użytkownika) wiadomości do transkryptu. Pamiętaj jednak, że jest to celowo uproszczony przykład. Rzeczywista implementacja wymagałaby dalszego zarządzania transkryptem, aby upewnić się, że nie marnujemy tokenów i/lub nie dezorientujemy SI zduplikowanymi wiadomościami asystenta w transkrypcie.

Prawdziwa implementacja auto-kontynuacji powinna również zawierać tak zwaną "logikę bezpiecznika", aby zapobiec niekontrolowanemu zapętlaniu. Powodem jest to, że przy określonych rodzajach monitów użytkownika i niskich ustawieniach `max_tokens`, SI mogłaby w nieskończoność generować dane wyjściowe widoczne dla użytkownika.

> Pamiętaj, że każda pętla wymaga osobnego żądania, a każde żądanie ponownie wykorzystuje cały twój transkrypt. Powinieneś dokładnie rozważyć kompromis między doświadczeniem użytkownika a wykorzystaniem API, podejmując decyzję o implementacji auto-kontynuacji w swojej aplikacji. Auto-kontynuacja może być szczególnie niebezpiecznie kosztowna, zwłaszcza przy korzystaniu z premium komercyjnych modeli.

Podsumowanie

Przetwarzanie strumieniowe jest kluczowym aspektem budowania aplikacji wykorzystujących sztuczną inteligencję, które łączą wykorzystanie narzędzi z odpowiedziami SI w czasie rzeczywistym. Poprzez efektywne zarządzanie danymi strumieniowymi z API platform SI, możesz zapewnić płynne i interaktywne doświadczenie użytkownika, obsługiwać duże odpowiedzi, optymalizować wykorzystanie zasobów i elegancko obsługiwać błędy.

Zaprezentowana klasa `Conversation::ReplyStream` pokazuje, jak przetwarzanie strumieniowe może być zaimplementowane w aplikacji Ruby przy użyciu dopasowania wzorców i architektury sterowanej zdarzeniami. Poprzez zrozumienie i wykorzystanie technik przetwarzania strumieniowego, możesz uwolnić pełny potencjał integracji SI w swoich aplikacjach i dostarczyć potężne i angażujące doświadczenia użytkownika.

Samonaprawiające się dane

Samonaprawiające się dane to skuteczne podejście do zapewnienia integralności, spójności i jakości danych w aplikacjach poprzez wykorzystanie możliwości dużych modeli językowych (DMJ). Ta kategoria wzorców koncentruje się na wykorzystaniu AI do automatycznego wykrywania, diagnozowania i korygowania anomalii, niespójności lub błędów w danych, zmniejszając tym samym obciążenie programistów i utrzymując wysoki poziom niezawodności danych.

U podstaw wzorców samonaprawiających się danych leży przekonanie, że dane są życiodajną siłą każdej aplikacji, a zapewnienie ich dokładności i integralności ma kluczowe znaczenie dla prawidłowego funkcjonowania i doświadczenia użytkownika aplikacji. Jednak zarządzanie i utrzymywanie jakości danych może być złożonym i czasochłonnym zadaniem, szczególnie gdy aplikacje rosną pod względem rozmiaru i złożoności. W tym miejscu do gry wkracza moc AI.

We wzorcach samonaprawiających się danych, komponenty AI są wykorzystywane do ciągłego monitorowania i analizowania danych aplikacji. Modele te mają zdolność rozumienia i interpretowania wzorców, zależności i anomalii w danych. Wykorzystując możliwości przetwarzania i rozumienia języka naturalnego, mogą identyfikować potencjalne problemy lub niespójności w danych i podejmować odpowiednie działania w celu ich naprawy.

Proces samonaprawiania danych zazwyczaj obejmuje kilka kluczowych kroków:

1. **Monitorowanie danych**: Komponenty AI nieustannie monitorują strumienie danych aplikacji, bazy danych lub systemy przechowywania, poszukując wszelkich oznak anomalii, niespójności lub błędów. Alternatywnie możesz aktywować komponent AI w reakcji na wyjątek.
2. **Wykrywanie anomalii**: Gdy problem zostanie wykryty, komponent AI szczegółowo analizuje dane, aby zidentyfikować konkretny charakter i zakres problemu. Może to obejmować wykrywanie brakujących wartości, niespójnych formatów lub danych naruszających predefiniowane reguły czy ograniczenia.
3. **Diagnoza i korekta**: Po zidentyfikowaniu problemu, komponent AI wykorzystuje swoją wiedzę i zrozumienie domeny danych do określenia odpowiedniego działania. Może to obejmować automatyczne korygowanie danych, uzupełnianie brakujących wartości lub oznaczanie problemu do interwencji człowieka, jeśli jest to konieczne.
4. **Ciągłe uczenie się (opcjonalne, w zależności od przypadku użycia)**: Gdy komponent AI napotyka i rozwiązuje różne problemy z danymi, może generować opisy tego, co się wydarzyło i jak zareagował. Te metadane mogą być przekazywane do procesów uczenia, co pozwala (i potencjalnie bazowemu modelowi, poprzez dostrajanie) na coraz skuteczniejsze i efektywniejsze identyfikowanie i rozwiązywanie anomalii w danych.

Dzięki automatycznemu wykrywaniu i korygowaniu problemów z danymi możesz

zapewnić, że Twoja aplikacja działa na wysokiej jakości, niezawodnych danych. Zmniejsza to ryzyko wystąpienia błędów, niespójności lub usterek związanych z danymi, które mogłyby wpłynąć na funkcjonalność aplikacji lub doświadczenie użytkownika.

Gdy pracownicy AI zajmują się zadaniem monitorowania i korygowania danych, możesz skupić swoje wysiłki na innych kluczowych aspektach aplikacji. Oszczędza to czas i zasoby, które w przeciwnym razie zostałyby poświęcone na ręczne czyszczenie i utrzymanie danych. W rzeczywistości, wraz ze wzrostem rozmiaru i złożoności aplikacji, ręczne zarządzanie jakością danych staje się coraz trudniejsze. Wzorce "Samouzdrawiających się danych" skalują się efektywnie, wykorzystując moc AI do obsługi dużych ilości danych i wykrywania problemów w czasie rzeczywistym.

Ze względu na swoją naturę, modele AI mogą adaptować się do zmieniających się wzorców danych, schematów lub wymagań w czasie, przy niewielkim nadzorze lub bez niego. Dopóki ich dyrektywy zapewniają odpowiednie wskazówki, szczególnie w zakresie zamierzonych rezultatów, Twoja aplikacja może ewoluować i obsługiwać nowe scenariusze danych bez konieczności intensywnej interwencji ręcznej lub zmian w kodzie.

Wzorce samouzdrawiających się danych dobrze komponują się z innymi kategoriami wzorców, o których mówiliśmy, takimi jak "Multitude of Workers". Możliwość samouzdrawiania danych można postrzegać jako wyspecjalizowany rodzaj pracownika, który koncentruje się szczególnie na zapewnieniu jakości i integralności danych. Ten rodzaj pracownika działa równolegle z innymi pracownikami AI, z których każdy przyczynia się do różnych aspektów funkcjonalności aplikacji.

Wdrażanie wzorców samouzdrawiających się danych w praktyce wymaga starannego zaprojektowania i integracji modeli AI z architekturą aplikacji. Ze względu na ryzyko utraty i uszkodzenia danych, powinieneś określić jasne wytyczne dotyczące sposobu wykorzystania tej techniki. Powinieneś również wziąć pod uwagę takie czynniki jak wydajność, skalowalność i bezpieczeństwo danych.

Praktyczne studium przypadku: Naprawianie uszkodzonego JSON-a

Jednym z najbardziej praktycznych i wygodnych sposobów wykorzystania samouzdrawiających się danych jest również bardzo prosty do wyjaśnienia: naprawianie uszkodzonego JSON-a.

Tę technikę można zastosować do powszechnego wyzwania, jakim jest radzenie sobie z niedoskonałymi lub niespójnymi danymi generowanymi przez LLM-y, takimi jak uszkodzony JSON, i zapewnia podejście do automatycznego wykrywania i korygowania tych problemów.

W Olympii regularnie spotykam się z sytuacjami, w których LLM-y generują dane JSON, które nie są w pełni poprawne. Może się to zdarzyć z różnych powodów, takich jak dodawanie przez LLM komentarzy przed lub po właściwym kodzie JSON lub wprowadzanie błędów składniowych, jak brakujące przecinki czy nieucieczkowane cudzysłowy. Te problemy mogą prowadzić do błędów parsowania i powodować zakłócenia w funkcjonowaniu aplikacji.

Aby rozwiązać ten problem, zaimplementowałem praktyczne rozwiązanie w postaci klasy JsonFixer. Klasa ta ucieleśnia wzorzec “Self-Healing Data”, przyjmując uszkodzony JSON jako dane wejściowe i wykorzystując LLM do jego naprawy, zachowując przy tym jak najwięcej informacji i pierwotnych intencji.

```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
```

Zwróć uwagę, jak `JsonFixer` wykorzystuje Ventriloquist do kierowania odpowiedziami AI.

Proces samouzdrawiania danych JSON działa następująco:

1. **Generowanie JSON**: LLM jest wykorzystywany do generowania danych JSON na podstawie określonych poleceń lub wymagań. Jednak ze względu na naturę LLM, wygenerowany JSON nie zawsze może być idealnie poprawny. Parser JSON oczywiście zgłosi błąd `ParserError`, jeśli podamy mu niepoprawny JSON.

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

Należy zauważyć, że komunikat o błędzie jest również przekazywany do wywołania `JSONFixer`, dzięki czemu nie musi on w pełni zakładać, co jest nie tak z danymi, szczególnie że parser często dokładnie wskazuje, co jest nieprawidłowe.

2. **Korekta oparta na LLM**: Klasa `JSONFixer` wysyła uszkodzony JSON z powrotem do LLM, wraz z konkretnym poleceniem lub instrukcją naprawy JSONa, zachowując przy tym jak najwięcej oryginalnych informacji i intencji. LLM, wytrenowany na ogromnych ilościach danych i rozumiejący składnię JSON, próbuje poprawić błędy i wygenerować prawidłowy ciąg JSON. Ograniczanie odpowiedzi jest używane do ograniczenia wyjścia LLM, a jako model AI wybieramy Mixtral 8x7B, ponieważ jest szczególnie dobry do tego rodzaju zadań.
3. **Walidacja i integracja**: Naprawiony ciąg JSON zwrócony przez LLM jest parsowany przez samą klasę `JSONFixer`, ponieważ wywołała ona `chat_completion(json: true)`. Jeśli naprawiony JSON przejdzie walidację, zostaje zintegrowany z powrotem do przepływu pracy aplikacji, pozwalając aplikacji na płynne kontynuowanie przetwarzania danych. Uszkodzony JSON został "uzdrowiony".

Mimo że napisałem i przepisałem własną implementację `JSONFixer` wiele razy, wątpię, czy całkowity czas zainwestowany we wszystkie te wersje przekracza godzinę lub dwie.

Należy zauważyć, że zachowanie intencji jest kluczowym elementem każdego wzorca samouzdrawiających się danych. Proces korekty oparty na LLM ma na celu zachowanie oryginalnych informacji i intencji wygenerowanego JSONa w możliwie największym

stopniu. Zapewnia to, że naprawiony JSON zachowuje swoje znaczenie semantyczne i może być skutecznie wykorzystany w kontekście aplikacji.

Ta praktyczna implementacja podejścia "Samouzdrawiających się danych" w Olympii jasno pokazuje, jak AI, a w szczególności LLM, mogą być wykorzystane do rozwiązywania rzeczywistych wyzwań związanych z danymi. Demonstruje to moc łączenia tradycyjnych technik programowania z możliwościami AI w celu budowania solidnych i wydajnych aplikacji.

Prawo Postela a wzorzec "Samouzdrawiających się danych"

"Samouzdrawiające się dane", czego przykładem jest klasa JSONFixer, dobrze wpisują się w zasadę znaną jako Prawo Postela, nazywaną również Zasadą solidności. Prawo Postela mówi:

"Bądź konserwatywny w tym, co robisz, bądź liberalny w tym, co przyjmujesz od innych."

Ta zasada, pierwotnie sformułowana przez Jona Postela, pioniera wczesnego Internetu, podkreśla znaczenie budowania systemów, które są tolerancyjne wobec różnorodnych lub nawet lekko niepoprawnych danych wejściowych, zachowując jednocześnie ścisłe przestrzeganie określonych protokołów przy wysyłaniu danych wyjściowych.

W kontekście "Self-Healing Data" (samouzdrawiających się danych), klasa JSONFixer ucieleśnia Prawo Postela poprzez liberalne podejście do przyjmowania uszkodzonych lub niedoskonałych danych JSON generowanych przez LLM (duże modele językowe). Nie odrzuca natychmiast ani nie kończy się niepowodzeniem w przypadku napotkania kodu JSON, który nie jest ściśle zgodny z oczekiwanym formatem. Zamiast tego przyjmuje tolerancyjne podejście i próbuje naprawić JSON,

wykorzystując możliwości modeli LLM.

Poprzez liberalne przyjmowanie niedoskonałego JSONa, klasa JSONFixer demonstruje odporność i elastyczność. Uznaje, że dane w rzeczywistym świecie często występują w różnych formach i nie zawsze mogą być zgodne ze ścisłymi specyfikacjami. Dzięki umiejętnemu obsługiwaniu i korygowaniu tych odchyleń, klasa zapewnia, że aplikacja może nadal sprawnie funkcjonować, nawet w obecności niedoskonałych danych.

Z drugiej strony, klasa JSONFixer przestrzega również konserwatywnego aspektu Prawa Postela w odniesieniu do danych wyjściowych. Po naprawieniu JSONa za pomocą LLM, klasa waliduje poprawiony JSON, aby upewnić się, że ściśle odpowiada oczekiwanemu formatowi. Zachowuje integralność i poprawność danych przed przekazaniem ich do innych części aplikacji. To konserwatywne podejście gwarantuje, że dane wyjściowe klasy JSONFixer są niezawodne i spójne, promując interoperacyjność i zapobiegając propagacji błędów.

Ciekawostki o Jonie Postelu:

- Jon Postel (1943-1998) był amerykańskim informatykiem, który odegrał kluczową rolę w rozwoju Internetu. Był znany jako "Bóg Internetu" ze względu na swój znaczący wkład w podstawowe protokoły i standardy.
- Postel był redaktorem serii dokumentów Request for Comments (RFC), która jest zbiorem technicznych i organizacyjnych notatek na temat Internetu. Był autorem lub współautorem ponad 200 RFC, w tym fundamentalnych protokołów takich jak TCP, IP i SMTP.
- Oprócz wkładu technicznego, Postel był znany ze swojego skromnego i opartego na współpracy podejścia. Wierzył w znaczenie osiągania konsensusu i wspólnej pracy nad budową solidnej i interoperacyjnej sieci.
- Postel pełnił funkcję Dyrektora Wydziału Sieci Komputerowych w Information Sciences Institute (ISI) Uniwersytetu Południowej Kalifornii (USC) od 1977 roku aż do swojej przedwczesnej śmierci w 1998 roku.

- W uznaniu jego ogromnego wkładu, Postel został pośmiertnie uhonorowany prestiżową Nagrodą Turinga w 1998 roku, często określaną jako "Nagroda Nobla w dziedzinie informatyki."

Klasa `JSONFixer` promuje odporność, elastyczność i interoperacyjność, które były podstawowymi wartościami, jakich Postel przestrzegał przez całą swoją karierę. Budując systemy, które są tolerancyjne wobec niedoskonałości, jednocześnie zachowując ścisłe przestrzeganie protokołów, możemy tworzyć aplikacje, które są bardziej odporne i adaptacyjne w obliczu rzeczywistych wyzwań.

Zastrzeżenia i Przeciwwskazania

Możliwość zastosowania podejść związanych z samoleczącymi się danymi jest całkowicie zależna od rodzaju danych, jakimi operuje twoja aplikacja. Istnieje powód, dla którego możesz nie chcieć po prostu stosować monkeypatch na `JSON.parse`, aby automatycznie korygować *wszystkie błędy parsowania JSON* w swojej aplikacji: nie wszystkie błędy mogą lub powinny być automatycznie korygowane.

Samoleczenie jest szczególnie problematyczne w połączeniu z wymogami regulacyjnymi lub zgodnościowymi związanymi z przetwarzaniem i obsługą danych. Niektóre branże, takie jak ochrona zdrowia i finanse, mają tak surowe przepisy dotyczące integralności danych i możliwości audytu, że wykonywanie jakiejkolwiek korekcji danych w formie "czarnej skrzynki" bez odpowiedniego nadzoru lub rejestrowania może naruszać te regulacje. Kluczowe jest zapewnienie, że wszystkie opracowane techniki samoleczenia danych są zgodne z odpowiednimi ramami prawnymi i regulacyjnymi.

Stosowanie technik samoleczenia danych, szczególnie tych wykorzystujących modele AI, może również mieć duży wpływ na wydajność aplikacji i wykorzystanie

zasobów. Przetwarzanie dużych ilości danych przez modele AI w celu wykrywania i korygowania błędów może być obliczeniowo intensywne. Ważne jest, aby ocenić kompromisy między korzyściami płynącymi z samoleczenia danych a związanymi z tym kosztami wydajnościowymi i zasobowymi.

To powiedziawszy, zagłębmy się w czynniki związane z podejmowaniem decyzji o tym, kiedy i gdzie stosować to potężne podejście.

Krytyczność Danych

Przy rozważaniu zastosowania technik samoleczenia danych, kluczowa jest ocena krytyczności przetwarzanych danych. Poziom krytyczności odnosi się do ważności i wrażliwości danych w kontekście twojej aplikacji i jej domeny biznesowej.

W niektórych przypadkach automatyczne korygowanie błędów w danych może nie być odpowiednie, szczególnie jeśli dane są wysoce wrażliwe lub mają implikacje prawne. Rozważmy następujące scenariusze:

1. **Transakcje Finansowe**: W aplikacjach finansowych, takich jak systemy bankowe czy platformy tradingowe, dokładność danych jest najwyższej wagi. Nawet drobne błędy w danych finansowych mogą mieć poważne konsekwencje, takie jak nieprawidłowe salda kont, błędnie przekierowane środki czy błędne decyzje handlowe. W takich przypadkach automatyczne korekty bez dokładnej weryfikacji i audytu mogą wprowadzać niedopuszczalne ryzyko.
2. **Dokumentacja Medyczna**: Aplikacje związane z opieką zdrowotną operują na wysoce wrażliwych i poufnych danych pacjentów. Niedokładności w dokumentacji medycznej mogą mieć poważne implikacje dla bezpieczeństwa pacjenta i decyzji dotyczących leczenia. Automatyczna modyfikacja danych medycznych bez odpowiedniego nadzoru i walidacji przez wykwalifikowanych pracowników służby zdrowia może naruszać wymogi regulacyjne i zagrażać dobru pacjenta.

3. **Dokumenty Prawne**: Aplikacje obsługujące dokumenty prawne, takie jak kontrakty, umowy czy dokumenty sądowe, wymagają ścisłej dokładności i integralności. Nawet drobne błędy w danych prawnych mogą mieć znaczące konsekwencje prawne. Automatyczne korekty w tej dziedzinie mogą nie być odpowiednie, ponieważ dane często wymagają ręcznego przeglądu i weryfikacji przez ekspertów prawnych w celu zapewnienia ich ważności i wykonalności.

W tych krytycznych scenariuszach dotyczących danych, ryzyko związane z automatycznymi korektami często przewyższa potencjalne korzyści. Konsekwencje wprowadzenia błędów lub nieprawidłowej modyfikacji danych mogą być poważne, prowadząc do strat finansowych, odpowiedzialności prawnej, a nawet szkód dla osób.

Przy pracy z danymi o wysokim stopniu krytyczności, kluczowe jest priorytetowe traktowanie procesów ręcznej weryfikacji i walidacji. Nadzór ludzki i wiedza ekspercka są niezbędne do zapewnienia dokładności i integralności danych. Techniki automatycznego samouzdrawiania mogą być nadal stosowane do oznaczania potencjalnych błędów lub niespójności, ale ostateczna decyzja dotycząca korekt powinna uwzględniać ludzką ocenę i zatwierdzenie.

Należy jednak zauważyć, że nie wszystkie dane w aplikacji muszą mieć ten sam poziom krytyczności. W ramach tej samej aplikacji mogą występować podzbiory danych, które są mniej wrażliwe lub których błędy mają mniejszy wpływ. W takich przypadkach techniki samouzdrawiania danych można selektywnie stosować do tych konkretnych podzbiorów danych, podczas gdy dane krytyczne pozostają przedmiotem ręcznej weryfikacji.

Kluczowe jest staranne oszacowanie krytyczności każdej kategorii danych w aplikacji oraz określenie jasnych wytycznych i procesów obsługi korekt w oparciu o związane z nimi ryzyko i konsekwencje. Rozróżniając dane krytyczne (np. księgi rachunkowe, dokumentacja medyczna) i niekrytyczne (np. adresy pocztowe, ostrzeżenia o zasobach), można znaleźć równowagę między wykorzystaniem zalet technik samouzdrawiania danych tam, gdzie jest to właściwe, a utrzymaniem ścisłej kontroli i nadzoru tam, gdzie

jest to konieczne.

Ostatecznie, decyzja o zastosowaniu technik samouzdrawiania danych do danych krytycznych powinna być podejmowana w porozumieniu z ekspertami dziedzinowymi, doradcami prawnymi i innymi odpowiednimi interesariuszami. Istotne jest uwzględnienie konkretnych wymagań, przepisów i ryzyka związanego z danymi aplikacji oraz odpowiednie dostosowanie strategii korygowania danych.

Poziom Krytyczności Błędów

Przy stosowaniu technik samouzdrawiania danych ważne jest ocenienie poziomu krytyczności i wpływu błędów w danych. Nie wszystkie błędy są sobie równe, a odpowiedni sposób działania może się różnić w zależności od powagi problemu.

Drobne niespójności lub problemy z formatowaniem mogą nadawać się do automatycznej korekty. Na przykład, proces samouzdrawiania danych zajmujący się naprawą uszkodzonego JSONa może obsłużyć brakujące przecinki lub nieescapowane cudzysłowy bez znaczącej zmiany znaczenia lub struktury danych. Tego typu błędy są często proste do naprawienia i mają minimalny wpływ na ogólną integralność danych.

Jednak poważniejsze błędy, które fundamentalnie zmieniają znaczenie lub integralność danych, mogą wymagać innego podejścia. W takich przypadkach automatyczne korekty mogą nie być wystarczające i może być konieczna interwencja człowieka w celu zapewnienia dokładności i poprawności danych.

W tym miejscu pojawia się koncepcja wykorzystania samej sztucznej inteligencji do pomocy w określaniu poziomu krytyczności błędów. Wykorzystując możliwości modeli AI, możemy zaprojektować samonaprawiających się pracowników danych, którzy nie tylko korygują błędy, ale także oceniają ich krytyczność i podejmują świadome decyzje dotyczące sposobu ich obsługi.

Weźmy na przykład samonaprawiającego się pracownika danych odpowiedzialnego za korygowanie niespójności w danych wpływających do bazy danych klientów.

Pracownik może zostać zaprojektowany tak, aby analizował dane i identyfikował potencjalne błędy, takie jak brakujące lub sprzeczne informacje. Jednak zamiast automatycznie poprawiać wszystkie błędy, pracownik może zostać wyposażony w dodatkowe wywołania narzędzi, które pozwalają mu oznaczać poważne błędy do weryfikacji przez człowieka.

Oto przykład, jak można to zaimplementować:

```
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
```

W tym przykładzie, worker `CustomerDataHealer` został zaprojektowany do identyfikacji i korygowania niespójności w danych klientów. Ponownie używamy wzorców Response Fencing i Ventriloquist do uzyskania ustrukturyzowanego wyjścia. Co istotne, dyrektywa systemowa workera zawiera instrukcje użycia funkcji `flag_for_review` w przypadku napotkania poważnych błędów.

Gdy worker przetwarza dane klienta, analizuje je i próbuje skorygować wszelkie niespójności. Jeśli worker stwierdzi, że błędy są poważne i wymagają interwencji człowieka, może użyć narzędzia `flag_for_review` do oznaczenia danych i podania powodu tego oznaczenia.

Metoda `chat_completion` jest wywoływana z parametrem `json: true`, aby przetworzyć skorygowane dane klienta jako JSON. Nie ma możliwości zapętlenia po wywołaniu funkcji, więc wynik będzie pusty, jeśli została wywołana funkcja `flag_for_review`. W przeciwnym razie dane klienta są aktualizowane sprawdzonymi i potencjalnie skorygowanymi informacjami.

Poprzez włączenie oceny powagi błędów i opcji oznaczania danych do przeglądu przez człowieka, worker do samonaprawiania danych staje się bardziej inteligentny i adaptowalny. Może automatycznie obsługiwać drobne błędy, jednocześnie eskalując poważne błędy do ekspertów w celu ręcznej interwencji.

Konkretne kryteria określania powagi błędów mogą być zdefiniowane w dyrektywie workera w oparciu o wiedzę domenową i wymagania biznesowe. Podczas oceny powagi można wziąć pod uwagę takie czynniki jak wpływ na integralność danych, potencjalne ryzyko utraty lub uszkodzenia danych oraz konsekwencje nieprawidłowych danych.

Wykorzystując sztuczną inteligencję do oceny powagi błędów i zapewniając możliwość interwencji człowieka, techniki samonaprawiania danych mogą znaleźć równowagę

między automatyzacją a zachowaniem dokładności danych. Takie podejście zapewnia, że drobne błędy są efektywnie korygowane, podczas gdy poważne błędy otrzymują niezbędną uwagę i wiedzę ekspercką od ludzkich recenzentów.

Złożoność Domeny

Rozważając zastosowanie technik samonaprawiania danych, ważne jest ocenienie złożoności domeny danych oraz reguł określających ich strukturę i zależności. Złożoność domeny może znacząco wpływać na skuteczność i wykonalność zautomatyzowanych podejść do korekty danych.

Techniki samonaprawiania danych działają dobrze, gdy dane podążają za dobrze zdefiniowanymi wzorcami i ograniczeniami. W domenach, gdzie struktura danych jest stosunkowo prosta, a relacje między elementami danych są jednoznaczne, automatyczne korekty mogą być stosowane z wysokim stopniem pewności. Na przykład, korygowanie problemów z formatowaniem lub wymuszanie podstawowych ograniczeń typów danych może być często skutecznie obsługiwane przez workery samonaprawiające dane.

Jednak wraz ze wzrostem złożoności domeny danych, rosną również wyzwania związane z automatyczną korektą danych. W domenach o złożonej logice biznesowej, skomplikowanych relacjach między encjami danych czy specyficznych regułach i wyjątkach domenowych, techniki samoleczącej się danych mogą nie zawsze uchwycić wszystkie niuanse i mogą prowadzić do nieprzewidzianych konsekwencji.

Rozważmy przykład złożonej domeny: system transakcji finansowych. W tej domenie dane obejmują różne instrumenty finansowe, dane rynkowe, reguły handlowe i wymogi regulacyjne. Relacje między różnymi elementami danych mogą być skomplikowane, a reguły określające poprawność i spójność danych mogą być wysoce specyficzne dla danej domeny.

W tak złożonej domenie, proces samoleczenia danych odpowiedzialny za korygowanie niespójności w danych transakcyjnych musiałby posiadać głębokie zrozumienie reguł

i ograniczeń specyficznych dla domeny. Musiałby uwzględniać takie czynniki jak regulacje rynkowe, limity transakcyjne, obliczenia ryzyka i procedury rozliczeniowe. Automatyczne korekty w tym kontekście mogą nie zawsze uchwycić pełną złożoność domeny i mogą nieumyślnie wprowadzać błędy lub naruszać reguły specyficzne dla domeny.

Aby sprostać wyzwaniom związanym ze złożonością domeny, techniki samoleczenia danych można ulepszyć poprzez włączenie wiedzy i reguł specyficznych dla domeny do modeli AI i procesów roboczych. Można to osiągnąć poprzez następujące techniki:

1. **Szkolenie Specyficzne dla Domeny**: Modele AI wykorzystywane do samoleczenia danych mogą być kierowane lub nawet dostrajane na zbiorach danych specyficznych dla domeny, które uwzględniają złożoności i reguły danej domeny. Poprzez ekspozycję modeli na reprezentatywne dane i scenariusze, mogą one nauczyć się wzorców, ograniczeń i wyjątków specyficznych dla domeny.
2. **Ograniczenia Oparte na Regułach**: Procesy samoleczenia danych mogą być wzbogacone o jawne ograniczenia oparte na regułach, które kodyfikują wiedzę specyficzną dla domeny. Reguły te mogą być definiowane przez ekspertów domenowych i integrowane w proces korekty danych. Modele AI mogą następnie wykorzystywać te reguły do kierowania swoimi decyzjami i zapewnienia zgodności z wymaganiami specyficznymi dla domeny.
3. **Współpraca z Ekspertami Domenowymi**: W złożonych domenach kluczowe jest zaangażowanie ekspertów domenowych w projektowanie i rozwój technik samoleczenia danych. Eksperci domenowi mogą dostarczyć cennych spostrzeżeń dotyczących złożoności danych, reguł biznesowych i potencjalnych przypadków brzegowych. Ich wiedza może być włączona do modeli AI i procesów roboczych w celu poprawy dokładności i niezawodności automatycznych korekt danych przy użyciu wzorców Human In The Loop.
4. **Podejście Przyrostowe i Iteracyjne**: W przypadku złożonych domen często korzystne jest przyjęcie przyrostowego i iteracyjnego podejścia do samoleczenia

danych. Zamiast próbować automatyzować korekty dla całej domeny na raz, należy skupić się na konkretnych poddomenach lub kategoriach danych, gdzie reguły i ograniczenia są dobrze zrozumiane. Stopniowo rozszerzać zakres technik samoleczenia w miarę pogłębiania zrozumienia domeny i potwierdzania skuteczności technik.

Biorąc pod uwagę złożoność domeny danych i włączając wiedzę dziedzinową do technik samouzdrawiających się danych, można osiągnąć równowagę między automatyzacją a dokładnością. Ważne jest, aby zdawać sobie sprawę, że samouzdrawiające się dane nie są uniwersalnym rozwiązaniem i że podejście powinno być dostosowane do konkretnych wymagań i wyzwań każdej domeny.

W złożonych domenach najbardziej efektywne może być podejście hybrydowe, łączące techniki samouzdrawiających się danych z wiedzą ekspercką i nadzorem człowieka. Automatyczne korekty mogą obsługiwać rutynowe i dobrze zdefiniowane przypadki, podczas gdy złożone scenariusze lub wyjątki mogą być oznaczane do przeglądu i interwencji człowieka. Takie współpracujące podejście zapewnia realizację korzyści z automatyzacji przy jednoczesnym zachowaniu niezbędnej kontroli i dokładności w złożonych domenach danych.

Wytłumaczalność i Przejrzystość

Wytłumaczalność odnosi się do zdolności zrozumienia i interpretacji rozumowania stojącego za decyzjami podejmowanymi przez modele AI, podczas gdy przejrzystość polega na zapewnieniu jasnego wglądu w proces korygowania danych.

W wielu kontekstach modyfikacje danych muszą być możliwe do skontrolowania i uzasadnienia. Interesariusze, w tym użytkownicy biznesowi, audytorzy i organy regulacyjne, mogą wymagać wyjaśnień, dlaczego wprowadzono określone korekty danych i jak modele AI doszły do tych decyzji. Jest to szczególnie istotne w dziedzinach, gdzie dokładność i integralność danych mają znaczące implikacje, takich jak finanse, ochrona zdrowia i kwestie prawne.

Aby sprostać potrzebie wytłumaczalności i przejrzystości, techniki samouzdrawiających się danych powinny zawierać mechanizmy zapewniające wgląd w proces podejmowania decyzji przez modele AI. Można to osiągnąć poprzez różne podejścia:

1. **Łańcuch Rozumowania**: Proszenie modelu o wyjaśnienie swojego toku myślenia "na głos" przed wprowadzeniem zmian w danych może ułatwić zrozumienie procesu podejmowania decyzji i może generować zrozumiałe dla człowieka wyjaśnienia wprowadzonych korekt. Kompromisem jest nieco większa złożoność w oddzielaniu wyjaśnienia od ustrukturyzowanych danych wyjściowych, co można rozwiązać poprzez…
2. **Generowanie Wyjaśnień**: Pracownicy zajmujący się samouzdrawiającymi się danymi mogą być wyposażeni w możliwość generowania zrozumiałych dla człowieka wyjaśnień wprowadzanych przez nich korekt. Można to osiągnąć, prosząc model o przedstawienie procesu podejmowania decyzji w formie łatwo zrozumiałych wyjaśnień *zintegrowanych z samymi danymi*. Na przykład, pracownik zajmujący się samouzdrawiającymi się danymi mógłby generować raport, który podkreśla konkretne niespójności w danych, które zidentyfikował, wprowadzone korekty i uzasadnienie tych korekt.
3. **Istotność Cech**: Modele AI mogą być instruowane informacjami o znaczeniu różnych cech lub atrybutów w procesie korygowania danych jako część ich dyrektyw. Te dyrektywy z kolei mogą być udostępniane ludzkim interesariuszom. Poprzez identyfikację kluczowych czynników wpływających na decyzje modelu, interesariusze mogą uzyskać wgląd w rozumowanie stojące za korektami i ocenić ich zasadność.
4. **Rejestrowanie i Audytowanie**: Wdrożenie kompleksowych mechanizmów rejestrowania i audytowania jest kluczowe dla zachowania przejrzystości w procesie samoleczenia danych. Każda korekta danych dokonana przez modele AI powinna być rejestrowana, włączając w to dane oryginalne, dane skorygowane oraz podjęte działania szczegółowe. Taka ścieżka audytu umożliwia

analizę retrospektywną i zapewnia jasny zapis modyfikacji wprowadzonych do danych.

5. **Podejście z Człowiekiem w Pętli**: Włączenie podejścia z człowiekiem w pętli może zwiększyć wytłumaczalność i przejrzystość technik samoleczenia danych. Poprzez zaangażowanie ekspertów w proces przeglądu i walidacji korekt generowanych przez AI, organizacje mogą zapewnić, że korekty są zgodne z wiedzą dziedzinową i wymaganiami biznesowymi. Nadzór ludzki dodaje dodatkową warstwę odpowiedzialności i pozwala na identyfikację potencjalnych uprzedzeń lub błędów w modelach AI.
6. **Ciągłe Monitorowanie i Ocena**: Regularne monitorowanie i ocena wydajności technik samoleczenia danych są niezbędne dla utrzymania przejrzystości i zaufania. Poprzez ocenę dokładności i skuteczności modeli AI w czasie, organizacje mogą identyfikować wszelkie odchylenia lub anomalie i podejmować działania korygujące. Ciągłe monitorowanie pomaga zapewnić, że proces samoleczenia danych pozostaje niezawodny i zgodny z oczekiwanymi rezultatami.

Wytłumaczalność i przejrzystość są kluczowymi aspektami przy wdrażaniu technik samoleczenia danych. Poprzez zapewnienie jasnych wyjaśnień dla korekt danych, utrzymywanie kompleksowych ścieżek audytu i angażowanie nadzoru ludzkiego, organizacje mogą budować zaufanie do procesu samoleczenia danych i zapewnić, że modyfikacje wprowadzane do danych są uzasadnione i zgodne z celami biznesowymi.

Ważne jest znalezienie równowagi między korzyściami z automatyzacji a potrzebą zachowania przejrzystości. Podczas gdy techniki samoleczenia danych mogą znacząco poprawić jakość i efektywność danych, nie powinno się to odbywać kosztem utraty widoczności i kontroli nad procesem korekty danych. Projektując systemy samoleczenia danych z myślą o wytłumaczalności i przejrzystości, organizacje mogą wykorzystać moc AI przy jednoczesnym zachowaniu niezbędnego poziomu odpowiedzialności i zaufania do danych.

Niezamierzone Konsekwencje

Podczas gdy techniki samoleczenia danych mają na celu poprawę jakości i spójności danych, kluczowe jest, aby być świadomym potencjalnych niezamierzonych konsekwencji. Automatyczne korekty, jeśli nie są starannie zaprojektowane i monitorowane, mogą nieumyślnie zmienić znaczenie lub kontekst danych, prowadząc do problemów w dalszych procesach.

Jednym z głównych zagrożeń związanych z samoleczeniem danych jest wprowadzenie uprzedzeń lub błędów w procesie korekty danych. Modele AI, jak każdy inny system informatyczny, mogą podlegać uprzedzeniom obecnym w danych treningowych lub wprowadzonym poprzez projekt algorytmów. Jeśli te uprzedzenia nie zostaną zidentyfikowane i złagodzone, mogą rozprzestrzeniać się poprzez proces samoleczenia danych i skutkować zniekształconymi lub nieprawidłowymi modyfikacjami danych.

Weźmy na przykład samouzdrawiającego pracownika danych, którego zadaniem jest korygowanie niespójności w danych demograficznych klientów. Jeśli model AI nauczył się stronniczości z danych historycznych, takich jak kojarzenie określonych zawodów lub poziomów dochodów z konkretnymi płciami czy grupami etnicznymi, może dokonywać nieprawidłowych założeń i modyfikować dane w sposób, który wzmacnia te uprzedzenia. Może to prowadzić do niedokładnych profili klientów, błędnych decyzji biznesowych i potencjalnie dyskryminujących rezultatów.

Inną potencjalną niezamierzoną konsekwencją jest utrata cennych informacji lub kontekstu podczas procesu korygowania danych. Techniki samouzdrawiających się danych często koncentrują się na standaryzacji i normalizacji danych w celu zapewnienia spójności. Jednak w niektórych przypadkach oryginalne dane mogą zawierać niuanse, wyjątki lub informacje kontekstowe, które są istotne dla zrozumienia pełnego obrazu. Automatyczne korekty, które ślepo wymuszają standaryzację, mogą nieumyślnie usunąć lub zaciemnić te cenne informacje.

Na przykład, wyobraźmy sobie samouzdrawiającego pracownika danych

odpowiedzialnego za korygowanie niespójności w dokumentacji medycznej. Jeśli pracownik napotka historię medyczną pacjenta z rzadką chorobą lub nietypowym planem leczenia, może próbować znormalizować dane tak, aby pasowały do bardziej powszechnego wzorca. Jednak robiąc to, może utracić szczegółowe informacje i kontekst, które są kluczowe dla dokładnego przedstawienia wyjątkowej sytuacji pacjenta. Ta utrata informacji może mieć poważne konsekwencje dla opieki nad pacjentem i podejmowania decyzji medycznych.

Aby złagodzić ryzyko niezamierzonych konsekwencji, konieczne jest przyjęcie proaktywnego podejścia podczas projektowania i wdrażania technik samouzdrawiających się danych:

1. **Dokładne Testowanie i Walidacja**: Przed wdrożeniem samouzdrawiających pracowników danych w środowisku produkcyjnym, kluczowe jest dokładne przetestowanie i zwalidowanie ich zachowania w różnorodnych scenariuszach. Obejmuje to testowanie na reprezentatywnych zbiorach danych, które uwzględniają różne przypadki brzegowe, wyjątki i potencjalne stronniczości. Rygorystyczne testowanie pomaga zidentyfikować i rozwiązać wszelkie niezamierzone konsekwencje, zanim wpłyną one na rzeczywiste dane.
2. **Ciągłe Monitorowanie i Ocena**: Wdrożenie mechanizmów ciągłego monitorowania i oceny jest niezbędne do wykrywania i łagodzenia niezamierzonych konsekwencji w czasie. Regularne przeglądanie wyników procesów samouzdrawiających się danych, analizowanie wpływu na systemy podrzędne i procesy decyzyjne oraz zbieranie informacji zwrotnych od interesariuszy może pomóc w identyfikacji wszelkich niepożądanych efektów i spowodować szybkie działania naprawcze. Jeśli Twoja organizacja posiada pulpity operacyjne, dodanie wyraźnie widocznych metryk związanych z automatycznymi zmianami danych jest prawdopodobnie dobrym pomysłem. Dodanie alarmów połączonych z dużymi odchyleniami od normalnej aktywności zmian danych jest prawdopodobnie jeszcze lepszym pomysłem!

3. **Nadzór i Interwencja Ludzka**: Utrzymanie nadzoru ludzkiego i możliwości interwencji w procesie samouzdrawiania danych jest kluczowe. Chociaż automatyzacja może znacznie poprawić wydajność, ważne jest, aby eksperci kontrolowali i walidowali korekty dokonywane przez modele AI, szczególnie w krytycznych lub wrażliwych dziedzinach. Ludzki osąd i wiedza ekspercka mogą pomóc w identyfikacji i rozwiązywaniu wszelkich niezamierzonych konsekwencji, które mogą się pojawić.

4. **Wytłumaczalna SI (XAI) i transparentność**: Jak omówiono w poprzedniej sekcji, włączenie technik wytłumaczalnej sztucznej inteligencji i zapewnienie transparentności w procesie samouzdrawiania danych może pomóc w złagodzeniu niezamierzonych konsekwencji. Poprzez dostarczanie jasnych wyjaśnień dla korekt danych i utrzymywanie kompleksowych ścieżek audytu, organizacje mogą lepiej zrozumieć i prześledzić rozumowanie stojące za modyfikacjami dokonywanymi przez modele SI.
5. **Podejście przyrostowe i iteracyjne**: Przyjęcie przyrostowego i iteracyjnego podejścia do samouzdrawiających się danych może pomóc zminimalizować ryzyko niezamierzonych konsekwencji. Zamiast stosować automatyczne korekty do całego zbioru danych naraz, należy zacząć od podzbioru danych i stopniowo rozszerzać zakres w miarę jak techniki okazują się skuteczne i niezawodne. Pozwala to na dokładne monitorowanie i dostosowywanie procesu po drodze, zmniejszając wpływ ewentualnych niezamierzonych konsekwencji.
6. **Współpraca i informacja zwrotna**: Angażowanie interesariuszy z różnych dziedzin oraz zachęcanie do współpracy i przekazywania informacji zwrotnych w trakcie procesu samouzdrawiania danych może pomóc w identyfikacji i rozwiązywaniu niezamierzonych konsekwencji. Regularne zasięganie opinii ekspertów dziedzinowych, konsumentów danych i użytkowników końcowych może dostarczyć cennych spostrzeżeń na temat rzeczywistego wpływu korekt danych i zwrócić uwagę na problemy, które mogły zostać przeoczone.

Poprzez proaktywne podejście do ryzyka niezamierzonych konsekwencji i wdrożenie odpowiednich zabezpieczeń, organizacje mogą wykorzystać korzyści płynące z technik samouzdrawiania danych, minimalizując jednocześnie potencjalne negatywne skutki. Ważne jest, aby podchodzić do samouzdrawiania danych jako do procesu iteracyjnego i opartego na współpracy, stale monitorując, oceniając i udoskonalając techniki, aby zapewnić ich zgodność z pożądanymi rezultatami oraz zachować integralność i wiarygodność danych.

Rozważając wykorzystanie wzorców samouzdrawiających się danych, należy dokładnie ocenić te czynniki i wyważyć korzyści względem potencjalnych zagrożeń i ograniczeń. W niektórych przypadkach najodpowiedniejszym rozwiązaniem może być podejście hybrydowe, łączące automatyczne korekty z nadzorem i interwencją człowieka.

Warto również zauważyć, że techniki samouzdrawiania danych nie powinny być traktowane jako zamiennik dla solidnej walidacji danych, sanityzacji danych wejściowych i mechanizmów obsługi błędów. Te podstawowe praktyki pozostają kluczowe dla zapewnienia integralności i bezpieczeństwa danych. Samouzdrawianie danych powinno być postrzegane jako podejście komplementarne, które może rozszerzyć i ulepszyć te istniejące środki.

Ostatecznie, decyzja o zastosowaniu wzorców samouzdrawiających się danych zależy od konkretnych wymagań, ograniczeń i priorytetów Twojej aplikacji. Poprzez staranne rozważenie przedstawionych powyżej aspektów i dostosowanie ich do celów i architektury Twojej aplikacji, możesz podejmować świadome decyzje dotyczące tego, kiedy i jak efektywnie wykorzystywać techniki samouzdrawiania danych.

Generowanie Treści Kontekstowej

Wzorce Generowania Treści Kontekstowej wykorzystują możliwości dużych modeli językowych (LLMs) do generowania dynamicznej i kontekstowej treści w aplikacjach. Ta kategoria wzorców uznaje znaczenie dostarczania spersonalizowanych i odpowiednich treści użytkownikom w oparciu o ich konkretne potrzeby, preferencje, a nawet poprzednie i obecne interakcje z aplikacją.

> W kontekście tego podejścia, "treść" odnosi się zarówno do treści podstawowej (tj. wpisów na blogu, artykułów itp.), jak i meta-treści, takich jak rekomendacje do treści podstawowej.

Wzorce Generowania Treści Kontekstowej mogą odgrywać kluczową rolę w zwiększaniu poziomu zaangażowania użytkowników, zapewnianiu spersonalizowanych doświadczeń i automatyzacji zadań związanych z tworzeniem treści zarówno dla Ciebie, jak i Twoich użytkowników. Wykorzystując wzorce opisane w tym rozdziale, możesz tworzyć aplikacje, które generują treści dynamicznie, dostosowując się do kontekstu i danych wejściowych w czasie rzeczywistym.

Wzorce działają poprzez integrację LLMs z elementami wyjściowymi aplikacji, począwszy od interfejsu użytkownika (czasami nazywanego "chrome"), przez e-maile i inne formy powiadomień, aż po wszelkie procesy generowania treści.

Gdy użytkownik wchodzi w interakcję z aplikacją lub wywołuje określone żądanie treści, aplikacja przechwytuje odpowiedni kontekst, taki jak preferencje użytkownika, poprzednie interakcje lub konkretne polecenia. Te informacje kontekstowe są następnie przekazywane do LLM wraz z niezbędnymi szablonami lub wytycznymi i wykorzystywane do tworzenia tekstu, który w przeciwnym razie musiałby być albo zakodowany na stałe, przechowywany w bazie danych, albo generowany algorytmicznie.

Treści generowane przez LLM mogą przybierać różne formy, takie jak spersonalizowane rekomendacje, dynamiczne opisy produktów, spersonalizowane odpowiedzi e-mail, a nawet całe artykuły czy wpisy na blogu. Jednym z najbardziej radykalnych zastosowań tej treści, które zapoczątkowałem ponad rok temu, jest dynamiczne generowanie elementów UI, takich jak etykiety formularzy, dymki podpowiedzi i inne rodzaje tekstu objaśniającego.

Personalizacja

Jedną z kluczowych korzyści wzorców Generowania Treści Kontekstowej jest możliwość dostarczania wysoce spersonalizowanych doświadczeń użytkownikom. Poprzez generowanie treści w oparciu o kontekst specyficzny dla użytkownika, wzorce

te umożliwiają aplikacjom dostosowywanie treści do indywidualnych zainteresowań, preferencji i interakcji użytkowników.

Personalizacja wykracza poza zwykłe wstawianie imienia użytkownika do ogólnej treści. Polega na wykorzystaniu bogatego kontekstu dostępnego dla każdego użytkownika do generowania treści, które odpowiadają jego konkretnym potrzebom i pragnieniom. Kontekst ten może obejmować szeroki zakres czynników, takich jak:

1. **Informacje z Profilu Użytkownika**: Na najbardziej ogólnym poziomie stosowania tej techniki, dane demograficzne, zainteresowania, preferencje i inne atrybuty profilu mogą być wykorzystywane do generowania treści, które są zgodne z pochodzeniem i charakterystyką użytkownika.
2. **Dane Behawioralne**: Wcześniejsze interakcje użytkownika z aplikacją, takie jak przeglądane strony, kliknięte linki czy zakupione produkty, mogą dostarczyć cennych informacji o jego zachowaniu i zainteresowaniach. Dane te można wykorzystać do generowania sugestii treści, które odzwierciedlają wzorce zaangażowania i przewidują przyszłe potrzeby.
3. **Czynniki Kontekstowe**: Aktualny kontekst użytkownika, taki jak jego lokalizacja, urządzenie, pora dnia, czy nawet pogoda, może wpływać na proces generowania treści. Na przykład, aplikacja turystyczna może posiadać moduł AI, który jest w stanie generować spersonalizowane rekomendacje w oparciu o aktualną lokalizację użytkownika i panujące warunki pogodowe.

Wykorzystując te czynniki kontekstowe, wzorce Kontekstowego Generowania Treści umożliwiają aplikacjom dostarczanie treści, które wydają się być stworzone specjalnie dla każdego indywidualnego użytkownika. Ten poziom personalizacji ma kilka istotnych korzyści:

1. **Zwiększone Zaangażowanie**: Spersonalizowana treść przyciąga uwagę użytkowników i utrzymuje ich zaangażowanie w aplikacji. Gdy użytkownicy czują, że treść jest odpowiednia i bezpośrednio odpowiada ich potrzebom, są

bardziej skłonni spędzać więcej czasu na interakcji z aplikacją i odkrywaniu jej funkcji.

2. **Lepsza Satysfakcja Użytkownika**: Spersonalizowana treść pokazuje, że aplikacja rozumie i dba o unikalne wymagania użytkownika. Dostarczając treści, które są pomocne, informacyjne i zgodne z ich zainteresowaniami, aplikacja może zwiększyć satysfakcję użytkowników i zbudować z nimi silniejszą więź.
3. **Wyższe Współczynniki Konwersji**: W kontekście aplikacji e-commerce lub marketingowych, spersonalizowana treść może znacząco wpłynąć na współczynniki konwersji. Przedstawiając użytkownikom produkty, oferty lub rekomendacje dostosowane do ich preferencji i zachowań, aplikacja może zwiększyć prawdopodobieństwo podejmowania przez nich pożądanych działań, takich jak dokonanie zakupu lub zapisanie się do usługi.

Produktywność

Wzorce Kontekstowego Generowania Treści mogą znacząco zwiększyć pewne rodzaje produktywności poprzez zmniejszenie potrzeby ręcznego generowania i edytowania treści w procesach kreatywnych. Wykorzystując moc Dużych Modeli Językowych, możesz generować wysokiej jakości treści na dużą skalę, oszczędzając czas i wysiłek, który twórcy treści i programiści musieliby w przeciwnym razie poświęcić na żmudną pracę ręczną.

Tradycyjnie, twórcy treści muszą prowadzić badania, pisać, redagować i formatować treści, aby spełniały wymagania aplikacji i oczekiwania użytkowników. Ten proces może być czasochłonny i wymagać znacznych zasobów, szczególnie gdy objętość treści rośnie.

Jednak dzięki wzorcom Kontekstowego Generowania Treści, proces tworzenia treści może być w dużej mierze zautomatyzowany. Duże Modele Językowe (LLM) mogą generować spójne, poprawne gramatycznie i kontekstowo trafne treści na podstawie dostarczonych poleceń i wytycznych. Ta automatyzacja oferuje kilka korzyści w zakresie produktywności:

1. **Zmniejszony Wysiłek Manualny**: Poprzez delegowanie zadań generowania treści do LLM, twórcy treści mogą skupić się na zadaniach wyższego poziomu, takich jak strategia treści, generowanie pomysłów i kontrola jakości. Mogą dostarczyć LLM niezbędny kontekst, szablony i wytyczne, pozwalając mu zająć się właściwym generowaniem treści. Zmniejsza to manualny wysiłek wymagany przy pisaniu i redagowaniu, pozwalając twórcom treści być bardziej produktywnymi i wydajnymi.
2. **Szybsze Tworzenie Treści**: LLM mogą generować treści znacznie szybciej niż ludzcy autorzy. Przy odpowiednich poleceniach i wytycznych, LLM może stworzyć wiele fragmentów treści w ciągu sekund lub minut. Ta szybkość umożliwia aplikacjom generowanie treści w znacznie szybszym tempie, nadążając za potrzebami użytkowników i stale zmieniającym się cyfrowym krajobrazem.

> Czy szybsze tworzenie treści prowadzi do sytuacji "tragedii wspólnego pastwiska", gdzie internet tonie w treściach, których nikt nie czyta? Niestety, podejrzewam, że odpowiedź brzmi tak.

3. **Spójność i Jakość**: LLM mogą z łatwością dostosowywać treści tak, aby zachować spójność stylu, tonu i jakości. Przy jasnych wytycznych i przykładach, określone rodzaje aplikacji (np. redakcje, PR itp.) mogą zapewnić, że ich treści tworzone przez ludzi są zgodne z tożsamością marki i spełniają pożądane standardy jakości. Ta spójność zmniejsza potrzebę intensywnego edytowania i poprawek, oszczędzając czas i wysiłek w procesie tworzenia treści.
4. **Iteracja i Optymalizacja**: Wzorce Kontekstowego Generowania Treści umożliwiają szybką iterację i optymalizację treści. Poprzez dostosowanie poleceń, szablonów lub wytycznych dostarczanych do LLM, aplikacje mogą

szybko generować warianty treści i testować różne podejścia w zautomatyzowany sposób, co nigdy wcześniej nie było możliwe. Ten iteracyjny proces pozwala na szybsze eksperymentowanie i doskonalenie strategii treści, prowadząc do bardziej efektywnych i angażujących treści w miarę upływu czasu. Ta konkretna technika może całkowicie zmienić reguły gry dla aplikacji takich jak e-commerce, których życie i śmierć zależy od współczynników odrzuceń i zaangażowania

Należy pamiętać, że choć wzorce generowania treści kontekstowej mogą znacznie zwiększyć produktywność, nie eliminują całkowicie potrzeby zaangażowania człowieka. Twórcy treści i redaktorzy nadal odgrywają kluczową rolę w definiowaniu ogólnej strategii treści, zapewnianiu wskazówek dla modelu LLM oraz dbaniu o jakość i odpowiedniość wygenerowanych treści.

Automatyzując bardziej powtarzalne i czasochłonne aspekty tworzenia treści, wzorce generowania treści kontekstowej uwalniają cenny czas i zasoby ludzkie, które można przekierować na zadania o większej wartości. Ten wzrost produktywności pozwala dostarczać użytkownikom bardziej spersonalizowane i angażujące treści, optymalizując jednocześnie procesy tworzenia contentu.

Szybka iteracja i eksperymentowanie

Wzorce generowania treści kontekstowej umożliwiają szybką iterację i eksperymentowanie z różnymi wariantami treści, pozwalając na szybszą optymalizację i dopracowanie strategii contentowej. Możesz generować wiele wersji treści w ciągu kilku sekund, po prostu dostosowując kontekst, szablony lub wytyczne przekazywane modelowi.

Ta możliwość szybkiej iteracji oferuje kilka kluczowych korzyści:

1. **Testowanie i optymalizacja**: Dzięki możliwości szybkiego generowania wariantów treści, można łatwo testować różne podejścia i mierzyć ich skuteczność. Na przykład, możesz wygenerować wiele wersji opisu produktu lub komunikatu marketingowego, dostosowanych do konkretnego segmentu użytkowników lub kontekstu. Analizując wskaźniki zaangażowania użytkowników, takie jak współczynniki klikalności czy konwersji, możesz zidentyfikować najbardziej skuteczne warianty treści i odpowiednio zoptymalizować swoją strategię.

2. **Testy A/B**: Wzorce generowania treści kontekstowej umożliwiają płynne przeprowadzanie testów A/B. Możesz wygenerować dwie lub więcej wersji treści i losowo prezentować je różnym grupom użytkowników. Porównując wyniki każdego wariantu, możesz określić, która treść najlepiej trafia do Twojej grupy docelowej. To oparte na danych podejście pozwala podejmować świadome decyzje i stale doskonalić treści, aby maksymalizować zaangażowanie użytkowników i osiągać pożądane rezultaty.
3. **Eksperymenty z personalizacją**: Szybka iteracja i eksperymentowanie są szczególnie cenne w przypadku personalizacji. Dzięki wzorcom generowania treści kontekstowej, możesz szybko generować spersonalizowane warianty treści w oparciu o różne segmenty użytkowników, preferencje czy zachowania. Eksperymentując z różnymi strategiami personalizacji, możesz zidentyfikować najskuteczniejsze podejścia do angażowania poszczególnych użytkowników i dostarczania dostosowanych doświadczeń.
4. **Adaptacja do zmieniających się trendów**: Możliwość szybkiej iteracji i eksperymentowania pozwala zachować elastyczność i dostosowywać się do zmieniających się trendów i preferencji użytkowników. Gdy pojawiają się nowe tematy, słowa kluczowe lub zachowania użytkowników, możesz szybko generować treści zgodne z tymi trendami. Poprzez ciągłe eksperymentowanie i doskonalenie treści, możesz pozostać aktualnym i zachować przewagę konkurencyjną w stale ewoluującym krajobrazie cyfrowym.

5. **Eksperymentowanie efektywne kosztowo**: Tradycyjne eksperymenty z treścią często wymagają znacznego nakładu czasu i zasobów, ponieważ twórcy treści muszą ręcznie opracowywać i testować różne warianty. Jednak dzięki wzorcom Generowania Treści Kontekstowej, koszt eksperymentowania jest znacznie zredukowany. Duże modele językowe mogą szybko generować warianty treści na dużą skalę, pozwalając na eksplorację szerokiego zakresu pomysłów i podejść bez ponoszenia znaczących kosztów.

Aby w pełni wykorzystać możliwości szybkiej iteracji i eksperymentowania, ważne jest posiadanie dobrze zdefiniowanego frameworka eksperymentalnego. Framework ten powinien zawierać:

- Jasno określone cele i hipotezy dla każdego eksperymentu
- Odpowiednie metryki i mechanizmy śledzenia do pomiaru efektywności treści
- Strategie segmentacji i targetowania, aby zapewnić dostarczanie odpowiednich wariantów treści właściwym użytkownikom
- Narzędzia do analizy i raportowania w celu wyciągania wniosków z danych eksperymentalnych
- Proces włączania zdobytej wiedzy i optymalizacji do strategii treści

Przyjmując podejście szybkiej iteracji i eksperymentowania, możesz stale udoskonalać i optymalizować swoją treść, zapewniając, że pozostaje ona angażująca, odpowiednia i skuteczna w osiąganiu celów Twojej aplikacji. To zwinne podejście do tworzenia treści pozwala wyprzedzać trendy i dostarczać wyjątkowe doświadczenia użytkownika.

Skalowalność i wydajność

Wraz z rozwojem aplikacji i rosnącym zapotrzebowaniem na spersonalizowane treści, wzorce generowania treści kontekstowej umożliwiają efektywne skalowanie procesu tworzenia treści. Duże modele językowe mogą generować treści dla dużej liczby

użytkowników i kontekstów jednocześnie, bez potrzeby proporcjonalnego zwiększania zasobów ludzkich. Ta skalowalność pozwala aplikacjom dostarczać spersonalizowane doświadczenia rosnącej bazie użytkowników bez przeciążania możliwości tworzenia treści.

Warto zauważyć, że generowanie treści kontekstowej może być skutecznie wykorzystane do internacjonalizacji aplikacji "w locie". W rzeczywistości, dokładnie to zrobiłem używając mojego gemu Instant18n do dostarczenia Olympii w ponad pół tuzinie języków, mimo że mamy mniej niż rok.

Lokalizacja wspierana przez AI

Jeśli pozwolicie mi się przez chwilę pochwalić, uważam, że moja biblioteka Instant18n dla aplikacji Rails jest przełomowym przykładem wzorca "Generowania Treści Kontekstowej" w działaniu, pokazującym transformacyjny potencjał AI w rozwoju aplikacji. Ten gem wykorzystuje moc dużego modelu językowego GPT od OpenAI, aby zrewolucjonizować sposób, w jaki internacjonalizacja i lokalizacja są obsługiwane w aplikacjach Rails.

Tradycyjnie, internacjonalizacja aplikacji Rails wymaga ręcznego definiowania kluczy tłumaczeń i dostarczania odpowiednich tłumaczeń dla każdego obsługiwanego języka. Ten proces może być czasochłonny, wymagający zasobów i podatny na niespójności. Jednak dzięki gemowi Instant18n, paradygmat lokalizacji jest całkowicie przedefiniowany.

Poprzez integrację modelu języka wielkoskalowego, gem Instant18n umożliwia generowanie tłumaczeń w czasie rzeczywistym, bazując na kontekście i znaczeniu tekstu. Zamiast polegać na predefiniowanych kluczach tłumaczeń i statycznych przekładach, gem dynamicznie tłumaczy tekst wykorzystując moc sztucznej inteligencji. Takie podejście oferuje kilka kluczowych korzyści:

1. **Płynna lokalizacja:** Dzięki gemowi Instant18n, programiści nie muszą już ręcznie definiować i zarządzać plikami tłumaczeń dla każdego obsługiwanego języka. Gem automatycznie generuje tłumaczenia na podstawie dostarczonego tekstu i docelowego języka, sprawiając, że proces lokalizacji jest bezwysiłkowy i płynny.
2. **Dokładność kontekstowa:** SI może otrzymać wystarczająco dużo kontekstu, aby zrozumieć niuanse tłumaczonego tekstu. Może uwzględnić otaczający kontekst, idiomy i odniesienia kulturowe, aby generować tłumaczenia, które są dokładne, naturalne i odpowiednie kontekstowo.
3. **Szeroka obsługa języków**: Gem Instant18n wykorzystuje rozległą wiedzę i możliwości językowe GPT, umożliwiając tłumaczenia na szeroki zakres języków. Od popularnych języków jak hiszpański i francuski, po bardziej obscuryczne lub fikcyjne języki jak klingoński i elficki, gem może obsłużyć różnorodne wymagania tłumaczeniowe.
4. **Elastyczność i kreatywność**: Gem wykracza poza tradycyjne tłumaczenia językowe i umożliwia kreatywne i niekonwencjonalne opcje lokalizacji. Programiści mogą tłumaczyć tekst na różne style, dialekty, a nawet języki fikcyjne, otwierając nowe możliwości dla unikalnych doświadczeń użytkownika i angażujących treści.
5. **Optymalizacja wydajności:** Gem Instant18n zawiera mechanizmy buforowania w celu poprawy wydajności i zmniejszenia obciążenia związanego z powtarzającymi się tłumaczeniami. Przetłumaczony tekst jest buforowany, co pozwala na szybką obsługę kolejnych żądań tego samego tłumaczenia bez potrzeby redundantnych wywołań API.

Gem Instant18n jest przykładem mocy wzorca "Kontekstowego Generowania Treści" poprzez wykorzystanie SI do dynamicznego generowania zlokalizowanych treści. Pokazuje, jak SI może być zintegrowana z podstawową funkcjonalnością aplikacji Rails, transformując sposób, w jaki programiści podchodzą do internacjonalizacji i lokalizacji.

Poprzez wyeliminowanie potrzeby ręcznego zarządzania tłumaczeniami i umożliwienie tłumaczeń w czasie rzeczywistym na podstawie kontekstu, gem Instant18n oszczędza

programistom znaczną ilość czasu i wysiłku. Pozwala im skupić się na budowaniu podstawowych funkcji ich aplikacji, jednocześnie zapewniając, że aspekt lokalizacji jest obsługiwany płynnie i dokładnie.

Znaczenie testów użytkowników i opinii zwrotnej

Na koniec, zawsze pamiętaj o znaczeniu testów użytkowników i opinii zwrotnej. Kluczowe jest zweryfikowanie, czy kontekstowe generowanie treści spełnia oczekiwania użytkowników i jest zgodne z celami aplikacji. Nieustannie iteruj i udoskonalaj generowane treści w oparciu o spostrzeżenia użytkowników i analitykę. Jeśli generujesz dynamiczne treści na dużą skalę, których ręczna walidacja przez Ciebie i Twój zespół byłaby niemożliwa, rozważ dodanie mechanizmów zbierania opinii, które pozwolą użytkownikom zgłaszać dziwne lub niepoprawne treści wraz z wyjaśnieniem dlaczego. Ta cenna informacja zwrotna może nawet zostać przekazana pracownikowi SI, którego zadaniem jest wprowadzanie poprawek do komponentu generującego treść!

Generative UI

W dzisiejszych czasach uwaga jest tak cennym zasobem, że skuteczne angażowanie użytkowników wymaga, aby oprogramowanie oferowało nie tylko płynne i intuicyjne, ale także wysoce spersonalizowane doświadczenia, dostosowane do indywidualnych potrzeb, preferencji i kontekstów. W rezultacie projektanci i programiści stają przed wyzwaniem tworzenia interfejsów użytkownika, które mogą adaptować się i odpowiadać na unikalne wymagania każdego użytkownika *na dużą skalę*.

Generative UI (GenUI) to prawdziwie rewolucyjne podejście do projektowania interfejsów użytkownika, wykorzystujące moc dużych modeli językowych (LLM) do tworzenia wysoce spersonalizowanych i dynamicznych doświadczeń użytkownika w czasie rzeczywistym. Chciałem koniecznie przedstawić w tej książce przynajmniej podstawy GenUI, ponieważ uważam, że jest to jedna z najbardziej obiecujących, niezagospodarowanych możliwości, jakie obecnie istnieją w dziedzinie projektowania aplikacji i frameworków. Jestem przekonany, że w tej konkretnej niszy pojawi się

dziesiątki lub więcej nowych udanych projektów komercyjnych i open-source.

U podstaw GenUI leży połączenie zasad Generowania Treści Kontekstowych z zaawansowanymi technikami AI do dynamicznego generowania elementów interfejsu użytkownika, takich jak tekst, obrazy i układy, w oparciu o głębokie zrozumienie kontekstu, preferencji i celów użytkownika. GenUI umożliwia projektantom i programistom tworzenie interfejsów, które dostosowują się i ewoluują w odpowiedzi na interakcje użytkownika, zapewniając poziom personalizacji, który wcześniej był nieosiągalny.

GenUI reprezentuje fundamentalną zmianę w sposobie, w jaki podchodzimy do projektowania interfejsów użytkownika. Zamiast projektować dla mas, GenUI pozwala nam projektować dla jednostki. Spersonalizowana treść i interfejsy mają potencjał tworzenia doświadczeń użytkownika, które rezonują z każdym użytkownikiem na głębszym poziomie, zwiększając zaangażowanie, satysfakcję i lojalność.

Jako najnowocześniejsza technologia, przejście na GenUI jest pełne wyzwań koncepcyjnych i praktycznych. Integracja AI w procesie projektowania, zapewnienie, że generowane interfejsy są nie tylko spersonalizowane, ale także użyteczne, dostępne i zgodne z ogólnym wizerunkiem marki i doświadczeniem użytkownika - wszystkie te wyzwania sprawiają, że GenUI jest domeną nielicznych, a nie wielu. Dodatkowo, zaangażowanie AI rodzi pytania dotyczące prywatności danych, transparentności, a nawet implikacji etycznych.

Pomimo wyzwań, spersonalizowane doświadczenia na dużą skalę mają moc całkowitego przekształcenia sposobu, w jaki wchodzimy w interakcję z cyfrowymi produktami i usługami. Otwiera to możliwości tworzenia inkluzywnych i dostępnych interfejsów, które zaspokajają różnorodne potrzeby użytkowników, niezależnie od ich możliwości, pochodzenia czy preferencji.

W tym rozdziale zbadamy koncepcję GenUI, analizując niektóre charakterystyczne cechy, kluczowe korzyści i potencjalne wyzwania. Zaczynamy od rozważenia najbardziej podstawowej i dostępnej formy GenUI: generowania tekstu dla tradycyjnie

zaprojektowanych i zaimplementowanych interfejsów użytkownika.

Generowanie Tekstu dla Interfejsów Użytkownika

Elementy tekstowe występujące w chrome aplikacji, takie jak etykiety formularzy, podpowiedzi i tekst objaśniający, są zazwyczaj na stałe zakodowane w szablonach lub komponentach UI, zapewniając spójne, ale ogólne doświadczenie dla wszystkich użytkowników. Wykorzystując wzorce generowania kontekstowego treści, możesz przekształcić te statyczne elementy w dynamiczne, świadome kontekstu i spersonalizowane komponenty.

Spersonalizowane Formularze

Formularze są wszechobecną częścią aplikacji internetowych i mobilnych, służąc jako podstawowy sposób zbierania danych od użytkowników. Jednak tradycyjne formularze często przedstawiają ogólne i bezosobowe doświadczenie, ze standardowymi etykietami i polami, które nie zawsze odpowiadają konkretnemu kontekstowi lub potrzebom użytkownika. Użytkownicy chętniej wypełniają formularze, które wydają się dostosowane do ich potrzeb i preferencji, co prowadzi do wyższych współczynników konwersji i zadowolenia użytkowników.

Jednak ważne jest zachowanie równowagi między personalizacją a spójnością. Podczas gdy dostosowywanie formularzy do indywidualnych użytkowników może być korzystne, kluczowe jest utrzymanie pewnego poziomu znajomości i przewidywalności. Użytkownicy powinni nadal być w stanie łatwo rozpoznawać formularze i poruszać się po nich, nawet z spersonalizowanymi elementami.

Oto kilka pomysłów na spersonalizowane formularze dla inspiracji:

Kontekstowe Sugestie Pól

GenUI może analizować poprzednie interakcje użytkownika, preferencje i dane, aby zapewniać inteligentne sugestie pól jako przewidywania. Na przykład, jeśli użytkownik wcześniej wprowadził swój adres wysyłki, formularz może automatycznie wypełnić odpowiednie pola zapisanymi informacjami. Nie tylko oszczędza to czas, ale także pokazuje, że aplikacja rozumie i pamięta preferencje użytkownika.

Chwileczkę, czy ta technika nie mogłaby być zrealizowana bez wykorzystania AI? Oczywiście, ale piękno napędzania tego rodzaju funkcjonalności przez AI jest dwojakie: 1) jak łatwe może być jej wdrożenie i 2) jak odporna może być w miarę zmian i ewolucji twojego UI w czasie.

Stwórzmy usługę dla naszego teoretycznego systemu obsługi zamówień, która będzie proaktywnie próbować wypełnić właściwy adres wysyłki dla użytkownika.

```
class OrderShippingAddressSubscriber
  include Raix::ChatCompletion

  attr_accessor :order

  delegate :customer, to: :order

  DIRECTIVE = "You are a smart order processing assistant. Given the
  customer's order history, guess the most likely shipping address
  for the current order."

  def order_created(order)
    return unless order.pending? && order.shipping_address.blank?

    self.order = order

    transcript.clear
    transcript << { system: DIRECTIVE }
    transcript << { user: "Order History: #{order_history.to_json}" }
    transcript << { user: "Current Order: #{order.to_json}" }

    response = chat_completion
```

```
    apply_predicted_shipping_address(order, response)
  end

  private

  def apply_predicted_shipping_address(order, response)
    # extract the shipping address from the response...
    # ...and assume there's some sort of live update of the address fields
    order.update(shipping_address:)
  end

  def order_history
    customer.orders.successful.limit(100).map do |order|
      {
        date: order.date,
        description: order.description,
        shipping_address: order.shipping_address
      }
    end
  end
end
```

Ten przykład jest bardzo uproszczony, ale powinien sprawdzić się w większości przypadków. Chodzi o to, aby pozwolić SI zgadywać w ten sam sposób, w jaki zrobiłby to człowiek. Aby wyjaśnić, o czym mówię, rozważmy pewne dane przykładowe:

```
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"}
]
```

Czy zauważyłeś schemat w tych danych? Gwarantuję ci, że to pestka dla LLM. Aby to zademonstrować, zapytajmy GPT-4, jaki jest najbardziej prawdopodobny adres wysyłki dla “termometru”.

```
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.
```

Jeśli myślisz, że używanie drogiego modelu takiego jak GPT-4 do tego zadania to przesada, masz rację! Wypróbowałem to samo polecenie na Mistral 7B Instruct i wygenerował odpowiedź z prędkością 75 tokenów na sekundę, przy znikomo niskim koszcie 0,000218 USD.

```
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.
```

Czy nakład pracy i koszty związane z tą techniką są warte tego, aby uczynić doświadczenie zakupowe bardziej magicznym? Dla wielu sprzedawców internetowych, absolutnie tak. A patrząc na obecną sytuację, koszty obliczeń AI będą tylko spadać, szczególnie w przypadku dostawców hostingu modeli open source konkurujących cenowo.

Użyj Szablonu Podpowiedzi i Ustrukturyzowanego WE/WY wraz z Ograniczaniem Odpowiedzi, aby zoptymalizować tego rodzaju uzupełnianie czatu.

Adaptacyjne Porządkowanie Pól

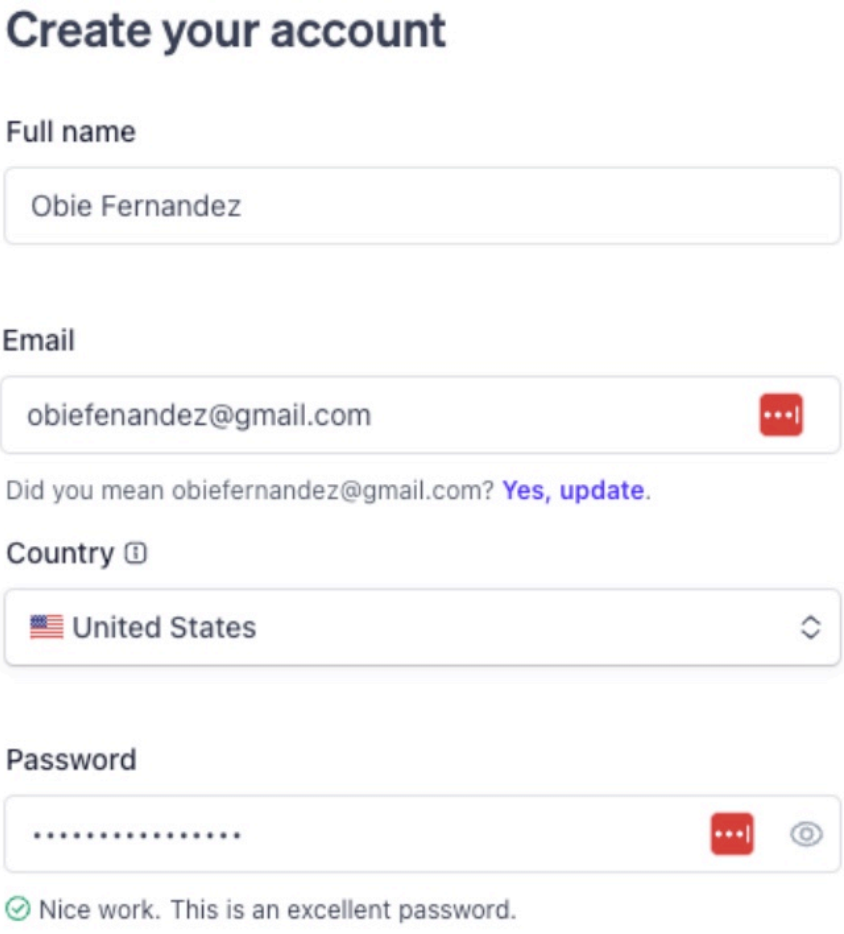
Kolejność, w jakiej pola formularza są prezentowane, może znacząco wpływać na doświadczenie użytkownika i wskaźniki ukończenia. Dzięki GenUI, możesz dynamicznie dostosowywać kolejność pól w oparciu o kontekst użytkownika i wagę każdego pola. Na przykład, jeśli użytkownik wypełnia formularz rejestracyjny w aplikacji fitness, formularz może priorytetowo traktować pola związane z ich celami fitness i preferencjami, sprawiając, że proces staje się bardziej trafny i angażujący.

Spersonalizowane Mikroteksty

Tekst instruktażowy, komunikaty o błędach i inne mikroteksty związane z formularzami również mogą być personalizowane przy użyciu GenUI. Zamiast wyświetlać ogólne komunikaty o błędach, takie jak "Nieprawidłowy adres email", możesz generować bardziej pomocne i kontekstowe wiadomości, na przykład "Proszę podać prawidłowy adres email, aby otrzymać potwierdzenie zamówienia". Te spersonalizowane akcenty mogą sprawić, że korzystanie z formularza będzie bardziej przyjazne dla użytkownika i mniej frustrujące.

Spersonalizowana Walidacja

Podobnie jak w przypadku Spersonalizowanych Mikrotekstów, możesz użyć AI do walidacji formularza w sposób, który wydaje się magiczny. Wyobraź sobie pozwolenie AI na walidację formularza profilu użytkownika, szukając potencjalnych błędów na poziomie *semantycznym*.



Rycina 9. Czy zauważasz zachodzącą walidację semantyczną?

Progresywne ujawnianie

GenUI może inteligentnie określić, które pola formularza są niezbędne w oparciu o kontekst użytkownika i stopniowo odkrywać dodatkowe pola w miarę potrzeb. Ta technika progresywnego ujawniania pomaga zmniejszyć obciążenie poznawcze i sprawia, że proces wypełniania formularzy jest bardziej przystępny. Na przykład,

jeśli użytkownik rejestruje się w podstawowej subskrypcji, formularz może początkowo prezentować tylko niezbędne pola, a w miarę postępu lub wyboru określonych opcji, dodatkowe istotne pola mogą być dynamicznie wprowadzane.

Kontekstowe teksty objaśniające

Dymki podpowiedzi są często używane do dostarczania dodatkowych informacji lub wskazówek użytkownikom, gdy najeżdżają kursorem lub wchodzą w interakcję z określonymi elementami. Dzięki podejściu "Generowania treści kontekstowej" możesz generować dymki podpowiedzi, które dostosowują się do kontekstu użytkownika i dostarczają odpowiednich informacji. Na przykład, jeśli użytkownik eksploruje złożoną funkcję, dymek może oferować spersonalizowane wskazówki lub przykłady bazujące na jego wcześniejszych interakcjach lub poziomie umiejętności.

Teksty objaśniające, takie jak instrukcje, opisy czy komunikaty pomocy, mogą być dynamicznie generowane w oparciu o kontekst użytkownika. Zamiast prezentować ogólne wyjaśnienia, możesz użyć modeli LLM do generowania tekstu dostosowanego do konkretnych potrzeb lub pytań użytkownika. Na przykład, jeśli użytkownik ma trudności z konkretnym krokiem w procesie, tekst objaśniający może dostarczyć spersonalizowanych wskazówek lub porad dotyczących rozwiązywania problemów.

Mikroteksty odnoszą się do małych fragmentów tekstu, które prowadzą użytkowników przez aplikację, takich jak etykiety przycisków, komunikaty o błędach czy monity potwierdzające. Stosując podejście Generowania treści kontekstowej do mikrotekstów, możesz stworzyć adaptacyjny interfejs użytkownika, który reaguje na działania użytkownika i dostarcza odpowiedni i pomocny tekst. Na przykład, jeśli użytkownik ma wykonać krytyczną akcję, monit potwierdzający może być generowany dynamicznie, aby przedstawić jasny i spersonalizowany komunikat.

Spersonalizowane teksty objaśniające i dymki podpowiedzi mogą znacząco usprawnić proces wdrażania nowych użytkowników. Dostarczając wskazówki i przykłady

dostosowane do kontekstu, możesz pomóc użytkownikom szybko zrozumieć i nawigować po aplikacji, zmniejszając krzywą uczenia się i zwiększając adopcję.

Dynamiczne i kontekstowe elementy interfejsu mogą również sprawić, że aplikacja będzie bardziej intuicyjna i angażująca. Użytkownicy chętniej wchodzą w interakcję i eksplorują funkcje, gdy towarzyszący im tekst jest dostosowany do ich konkretnych potrzeb i zainteresowań.

Do tej pory omówiliśmy pomysły na ulepszenie istniejących paradygmatów interfejsu użytkownika za pomocą AI, ale co z przemyśleniem na nowo sposobu projektowania i implementacji interfejsów użytkownika w bardziej radykalny sposób?

Definiowanie Generatywnego UI

W przeciwieństwie do tradycyjnego projektowania UI, gdzie projektanci tworzą ustalone, statyczne interfejsy, GenUI zapowiada przyszłość, w której nasze oprogramowanie będzie mogło pochwalić się elastycznymi, spersonalizowanymi doświadczeniami, które mogą ewoluować i adaptować się w czasie rzeczywistym. Za każdym razem, gdy korzystamy z konwersacyjnego interfejsu opartego na AI, pozwalamy sztucznej inteligencji dostosować się do konkretnych potrzeb użytkownika. GenUI idzie o krok dalej, stosując ten poziom adaptacyjności do *wizualnego* interfejsu oprogramowania.

Powodem, dla którego możliwe jest eksperymentowanie z koncepcjami GenUI już dziś, jest to, że duże modele językowe już rozumieją programowanie, a ich podstawowa wiedza obejmuje technologie i frameworki UI. Pytanie brzmi więc, czy duże modele językowe mogą być wykorzystane do generowania elementów UI, takich jak tekst, obrazy, układy, a nawet całe interfejsy, które są dostosowane do każdego indywidualnego użytkownika. Model mógłby zostać poinstruowany, aby uwzględniał

różne czynniki, takie jak poprzednie interakcje użytkownika, określone preferencje, informacje demograficzne i aktualny kontekst użycia, w celu tworzenia wysoce spersonalizowanych i odpowiednich interfejsów.

GenUI różni się od tradycyjnego projektowania interfejsu użytkownika na kilka kluczowych sposobów:

1. **Dynamiczność i Adaptacyjność**: Tradycyjne projektowanie UI polega na tworzeniu ustalonych, statycznych interfejsów, które pozostają takie same dla wszystkich użytkowników. W przeciwieństwie do tego, GenUI umożliwia tworzenie interfejsów, które mogą dynamicznie adaptować się i zmieniać w zależności od potrzeb użytkownika i kontekstu. Oznacza to, że ta sama aplikacja może prezentować różne interfejsy różnym użytkownikom, a nawet temu samemu użytkownikowi w różnych sytuacjach.
2. **Personalizacja na Dużą Skalę**: W tradycyjnym projektowaniu tworzenie spersonalizowanych doświadczeń dla każdego użytkownika jest często niepraktyczne ze względu na wymagany czas i zasoby. GenUI natomiast umożliwia personalizację na dużą skalę. Wykorzystując AI, projektanci mogą tworzyć interfejsy, które automatycznie dostosowują się do unikalnych potrzeb i preferencji każdego użytkownika, bez konieczności ręcznego projektowania i rozwijania osobnych interfejsów dla każdego segmentu użytkowników.
3. **Koncentracja na Rezultatach**: Tradycyjne projektowanie UI często skupia się na tworzeniu wizualnie atrakcyjnych i funkcjonalnych interfejsów. Choć te aspekty są nadal ważne w GenUI, główny nacisk przesuwa się w kierunku osiągania pożądanych rezultatów dla użytkownika. GenUI ma na celu tworzenie interfejsów zoptymalizowanych pod kątem konkretnych celów i zadań każdego użytkownika, przedkładając użyteczność i efektywność nad czysto estetyczne względy.
4. **Ciągłe Uczenie się i Doskonalenie**: Systemy GenUI mogą nieustannie uczyć się i doskonalić w czasie na podstawie interakcji użytkowników i otrzymywanej informacji zwrotnej. Gdy użytkownicy korzystają z wygenerowanych

interfejsów, modele AI mogą gromadzić dane na temat zachowań użytkowników, ich preferencji i osiąganych rezultatów, wykorzystując te informacje do udoskonalania i optymalizacji przyszłych generacji interfejsów. Ten iteracyjny proces uczenia pozwala systemom GenUI stawać się coraz skuteczniejszymi w zaspokajaniu potrzeb użytkowników wraz z upływem czasu.

Należy zauważyć, że GenUI nie jest tym samym co narzędzia projektowe wspomagane przez SI, takie jak te, które oferują sugestie lub automatyzują określone zadania projektowe. Podczas gdy narzędzia te mogą być pomocne w usprawnieniu procesu projektowania, nadal polegają na projektantach, którzy podejmują ostateczne decyzje i tworzą statyczne interfejsy. GenUI natomiast zakłada, że system SI przyjmuje bardziej aktywną rolę w faktycznym generowaniu i adaptacji interfejsów w oparciu o dane użytkownika i kontekst.

GenUI reprezentuje znaczącą zmianę w podejściu do projektowania interfejsów użytkownika, odchodząc od rozwiązań uniwersalnych na rzecz wysoce spersonalizowanych, adaptacyjnych doświadczeń. Wykorzystując moc SI, GenUI ma potencjał zrewolucjonizować sposób, w jaki wchodzimy w interakcję z cyfrowymi produktami i usługami, tworząc interfejsy, które są bardziej intuicyjne, angażujące i efektywne dla każdego indywidualnego użytkownika.

Przykład

Aby zilustrować koncepcję GenUI, rozważmy hipotetyczną aplikację fitness o nazwie “FitAI”. Ta aplikacja ma na celu dostarczanie spersonalizowanych planów treningowych i porad żywieniowych użytkownikom w oparciu o ich indywidualne cele, poziom sprawności i preferencje.

W tradycyjnym podejściu do projektowania UI, FitAI mogłaby mieć ustalony zestaw ekranów i elementów, które są takie same dla wszystkich użytkowników. Jednak

z GenUI interfejs aplikacji mógłby dynamicznie dostosowywać się do unikalnych potrzeb i kontekstu każdego użytkownika.

To podejście jest dość trudne do wyobrażenia sobie w kontekście wdrożenia w 2024 roku i może nawet nie mieć odpowiedniego ROI, ale jest możliwe.

Oto jak mogłoby to działać:

1. **Wdrożenie początkowe**:

    - Zamiast standardowego kwestionariusza, FitAI wykorzystuje SI konwersacyjną do zbierania informacji o celach użytkownika, aktualnym poziomie sprawności i preferencjach.
    - Na podstawie tej początkowej interakcji, SI generuje spersonalizowany układ panelu, podkreślając funkcje i informacje najbardziej istotne dla celów użytkownika.
    - Obecna technologia SI mogłaby mieć do dyspozycji wybór komponentów ekranu do wykorzystania w tworzeniu spersonalizowanego panelu.
    - Przyszła technologia SI mogłaby przyjąć rolę doświadczonego projektanta UI i faktycznie tworzyć panel *od podstaw*.

2. **Planer treningowy**:

    - Interfejs planera treningowego jest dostosowywany przez SI specjalnie do poziomu doświadczenia użytkownika i dostępnego sprzętu.
    - Dla początkującego bez sprzętu może pokazywać proste ćwiczenia z masą własnego ciała ze szczegółowymi instrukcjami i filmami.
    - Dla zaawansowanego użytkownika z dostępem do siłowni może wyświetlać bardziej złożone rutyny z mniejszą ilością treści objaśniających.
    - Zawartość planera treningowego nie jest po prostu filtrowana z większego zbioru. Może być generowana *w czasie rzeczywistym* w oparciu o bazę wiedzy, która jest odpytywana z uwzględnieniem kontekstu zawierającego wszystkie znane informacje o użytkowniku.

3. **Śledzenie postępów**:

    - Interfejs śledzenia postępów ewoluuje w oparciu o cele użytkownika i wzorce zaangażowania.
    - Jeśli użytkownik koncentruje się głównie na utracie wagi, interfejs może wyraźnie wyświetlać wykres trendu wagi i statystyki spalonych kalorii.
    - Dla użytkownika budującego masę mięśniową, może podkreślać przyrosty siły i zmiany w kompozycji ciała.
    - SI może dostosować tę część aplikacji do rzeczywistych postępów użytkownika. Jeśli postęp zatrzyma się na pewien czas, aplikacja może przejść w tryb, w którym próbuje nakłonić użytkownika do ujawnienia przyczyn niepowodzenia, aby je złagodzić.

4. **Porady żywieniowe**:

    - Sekcja żywieniowa dostosowuje się do preferencji i ograniczeń dietetycznych użytkownika.
    - Dla użytkownika wegańskiego może pokazywać sugestie posiłków roślinnych i źródła białka.
    - Dla użytkownika z nietolerancją glutenu automatycznie filtruje produkty zawierające gluten z rekomendacji.
    - Ponownie, treść nie jest pobierana z ogromnego zbioru danych o posiłkach, który ma zastosowanie do wszystkich użytkowników, ale jest syntezowana z bazy wiedzy zawierającej informacje dostosowywalne do konkretnej sytuacji i ograniczeń użytkownika.
    - Na przykład, przepisy są generowane ze specyfikacjami składników, które odpowiadają stale zmieniającym się potrzebom kalorycznym użytkownika w miarę ewolucji jego poziomu sprawności i parametrów ciała.

5. **Elementy motywacyjne**:

- Treści motywacyjne i powiadomienia w aplikacji są personalizowane w oparciu o typ osobowości użytkownika i reakcję na różne strategie motywacyjne.
- Niektórzy użytkownicy mogą otrzymywać zachęcające wiadomości, podczas gdy inni dostają bardziej oparte na danych informacje zwrotne.

W tym przykładzie GenUI umożliwia FitAI tworzenie wysoce spersonalizowanego doświadczenia dla każdego użytkownika, potencjalnie zwiększając zaangażowanie, satysfakcję i prawdopodobieństwo osiągnięcia celów fitness. Elementy interfejsu, treść, a nawet "osobowość" aplikacji dostosowują się, aby najlepiej służyć potrzebom i preferencjom każdego indywidualnego użytkownika.

Przejście do projektowania zorientowanego na rezultaty

GenUI reprezentuje fundamentalną zmianę w podejściu do projektowania interfejsu użytkownika!, przechodząc od koncentracji na tworzeniu konkretnych elementów interfejsu do bardziej holistycznego podejścia zorientowanego na rezultaty. Ta zmiana ma kilka ważnych implikacji:

1. **Koncentracja na celach użytkownika**:
    - Projektanci będą musieli głębiej zastanowić się nad celami użytkowników i pożądanymi rezultatami, zamiast nad konkretnymi komponentami interfejsu.
    - Nacisk zostanie położony na tworzenie systemów, które mogą generować interfejsy pomagające użytkownikom efektywnie i skutecznie osiągać ich cele.

- Pojawią się nowe frameworki UI, które zapewnią projektantom opartym na SI narzędzia potrzebne do generowania doświadczeń użytkownika *w locie* i *od podstaw*, zamiast bazować na predefiniowanych specyfikacjach ekranów.

2. **Zmieniająca się rola projektantów**:

   - Projektanci przejdą od tworzenia stałych układów do definiowania reguł, ograniczeń i wytycznych, którymi systemy AI będą się kierować podczas generowania interfejsów.
   - Będą musieli rozwinąć umiejętności w takich obszarach jak analiza danych, inżynieria promptów i myślenie systemowe, aby skutecznie kierować systemami GenUI.

3. **Znaczenie badań użytkowników**:

   - Badania użytkowników stają się jeszcze bardziej kluczowe w kontekście GenUI, ponieważ projektanci muszą zrozumieć nie tylko preferencje użytkowników, ale także to, jak te preferencje i potrzeby zmieniają się w różnych kontekstach.
   - Ciągłe testy z użytkownikami i pętle informacji zwrotnej będą niezbędne do udoskonalania i poprawy zdolności AI do generowania efektywnych interfejsów.

4. **Projektowanie z uwzględnieniem zmienności**:

   - Zamiast tworzyć jeden "idealny" interfejs, projektanci będą musieli uwzględnić wiele możliwych wariantów i zapewnić, że system może generować odpowiednie interfejsy dla różnorodnych potrzeb użytkowników.
   - Obejmuje to projektowanie z myślą o przypadkach brzegowych i zapewnienie, że generowane interfejsy zachowują użyteczność i dostępność w różnych konfiguracjach.

- Różnicowanie produktów nabiera nowych wymiarów, obejmując różne perspektywy w zakresie psychologii użytkownika oraz wykorzystanie unikalnych zbiorów danych i baz wiedzy niedostępnych dla konkurencji.

Wyzwania i aspekty do rozważenia

Choć GenUI oferuje ekscytujące możliwości, wiąże się również z kilkoma wyzwaniami i kwestiami do rozważenia:

1. **Ograniczenia techniczne**:
   - Obecna technologia AI, choć zaawansowana, wciąż ma ograniczenia w rozumieniu złożonych intencji użytkowników i generowaniu interfejsów prawdziwie świadomych kontekstu.
   - Problemy z wydajnością związane z generowaniem elementów interfejsu w czasie rzeczywistym, szczególnie na mniej wydajnych urządzeniach.
2. **Wymagania dotyczące danych**:
   - W zależności od przypadku użycia, efektywne systemy GenUI mogą wymagać znacznych ilości danych użytkowników do generowania spersonalizowanych interfejsów.
   - Wyzwania związane z etycznym pozyskiwaniem autentycznych danych użytkowników rodzą obawy dotyczące prywatności i bezpieczeństwa danych, a także potencjalnych uprzedzeń w danych używanych do trenowania modeli GenUI.
3. **Użyteczność i spójność**:
   - Przynajmniej do czasu, gdy praktyka stanie się powszechna, aplikacja ze stale zmieniającymi się interfejsami może prowadzić do problemów z użytecznością, ponieważ użytkownicy mogą mieć trudności ze znalezieniem znajomych elementów lub efektywną nawigacją.

- Kluczowe będzie znalezienie równowagi między personalizacją a utrzymaniem spójnego, możliwego do nauczenia się interfejsu.

4. **Nadmierne poleganie na AI**:

   - Istnieje ryzyko nadmiernego delegowania decyzji projektowych do systemów AI, co potencjalnie może prowadzić do nieinspirujących, problematycznych lub po prostu wadliwych wyborów interfejsu.
   - Nadzór ludzki i możliwość nadpisywania projektów generowanych przez AI pozostaną ważne w dającej się przewidzieć przyszłości.

5. **Kwestie dostępności**:

   - Zapewnienie, aby dynamicznie generowane interfejsy pozostały dostępne dla użytkowników z niepełnosprawnościami, stanowi zupełnie nowe wyzwania, co jest niepokojące biorąc pod uwagę niski poziom zgodności z zasadami dostępności wykazywany przez typowe systemy.
   - Z drugiej strony, projektanci AI mogą być wdrażani z *wbudowaną* dbałością o dostępność oraz możliwościami tworzenia dostępnych interfejsów w locie, tak jak tworzą interfejsy dla użytkowników bez niepełnosprawności.
   - W każdym przypadku systemy GenUI powinny być projektowane z uwzględnieniem solidnych wytycznych dotyczących dostępności i procesów testowych.

6. **Zaufanie i transparentność użytkowników**:

   - Użytkownicy mogą czuć się niekomfortowo z interfejsami, które wydają się "wiedzieć zbyt wiele" o nich lub zmieniają się w sposób dla nich niezrozumiały.
   - Zapewnienie przejrzystości w kwestii tego, jak i dlaczego interfejsy są personalizowane, będzie istotne dla budowania zaufania użytkowników.

Perspektywy i możliwości na przyszłość

Przyszłość Generative UI (GenUI) niesie ze sobą ogromne możliwości zrewolucjonizowania sposobu, w jaki wchodzimy w interakcję z cyfrowymi produktami i usługami. W miarę rozwoju tej technologii możemy spodziewać się sejsmicznej zmiany w sposobie projektowania, wdrażania i doświadczania interfejsów użytkownika. Uważam, że GenUI jest zjawiskiem, które wreszcie wprowadzi nasze oprogramowanie w obszar tego, co obecnie uznawane jest za science fiction.

Jedną z najbardziej ekscytujących perspektyw GenUI jest jego potencjał do zwiększenia dostępności na wielką skalę, wykraczającą poza zwykłe upewnienie się, że osoby z poważnymi niepełnosprawnościami nie są całkowicie wykluczone z korzystania z oprogramowania. Dzięki automatycznemu dostosowywaniu interfejsów do indywidualnych potrzeb użytkowników, GenUI może uczynić cyfrowe doświadczenia bardziej inkluzywnymi niż kiedykolwiek wcześniej. Wyobraźmy sobie interfejsy, które płynnie dostosowują się, zapewniając większy tekst dla młodszych użytkowników lub osób z wadami wzroku, czy uproszczone układy dla osób z niepełnosprawnościami poznawczymi - wszystko to bez konieczności ręcznej konfiguracji lub oddzielnych "dostępnych" wersji aplikacji.

Możliwości personalizacji GenUI prawdopodobnie przyczynią się do zwiększenia zaangażowania użytkowników, ich satysfakcji i lojalności w szerokim zakresie produktów cyfrowych. Gdy interfejsy staną się bardziej dostrojone do indywidualnych preferencji i zachowań, użytkownicy będą postrzegać cyfrowe doświadczenia jako bardziej intuicyjne i przyjemne, co potencjalnie doprowadzi do głębszych i bardziej znaczących interakcji z technologią.

GenUI ma również potencjał do przekształcenia procesu wdrażania nowych użytkowników. Poprzez tworzenie intuicyjnych, spersonalizowanych doświadczeń dla nowych użytkowników, które szybko dostosowują się do poziomu wiedzy każdego użytkownika, GenUI może znacząco zmniejszyć krzywą uczenia się związaną z nowymi

aplikacjami. Może to prowadzić do szybszego tempa adopcji i zwiększonej pewności użytkowników w odkrywaniu nowych funkcji i funkcjonalności.

Kolejną ekscytującą możliwością jest zdolność GenUI do utrzymania spójnego doświadczenia użytkownika na różnych urządzeniach i platformach, przy jednoczesnej optymalizacji dla każdego konkretnego kontekstu użytkowania. Mogłoby to rozwiązać długotrwały problem zapewnienia spójnych doświadczeń w coraz bardziej rozdrobnionym krajobrazie urządzeń, od smartfonów i tabletów po komputery stacjonarne i powstające technologie, takie jak okulary rzeczywistości rozszerzonej.

Charakter GenUI oparty na danych otwiera możliwości szybkiej iteracji i doskonalenia projektowania interfejsu użytkownika. Dzięki zbieraniu danych w czasie rzeczywistym o tym, jak użytkownicy wchodzą w interakcję z generowanymi interfejsami, projektanci i programiści mogą uzyskać bezprecedensowy wgląd w zachowania i preferencje użytkowników. Ta pętla sprzężenia zwrotnego może prowadzić do ciągłych ulepszeń w projektowaniu interfejsu użytkownika, napędzanych rzeczywistymi wzorcami użytkowania, a nie założeniami czy ograniczonymi testami użytkowników.

Aby przygotować się na tę zmianę, projektanci będą musieli rozwinąć swoje umiejętności i sposób myślenia. Nacisk przesunie się z tworzenia stałych układów na rozwijanie kompleksowych systemów projektowych i wytycznych, które mogą kierować generowaniem interfejsów przez AI. Projektanci będą musieli rozwinąć głębokie zrozumienie analizy danych, technologii AI i myślenia systemowego, aby skutecznie kierować systemami GenUI.

Co więcej, ponieważ GenUI zaciera granice między projektowaniem a technologią, projektanci będą musieli ściślej współpracować z programistami i naukowcami zajmującymi się danymi. To interdyscyplinarne podejście będzie kluczowe w tworzeniu systemów GenUI, które są nie tylko atrakcyjne wizualnie i przyjazne dla użytkownika, ale także technicznie solidne i etycznie poprawne.

Implikacje etyczne GenUI również wysuną się na pierwszy plan wraz z dojrzewaniem tej technologii. Projektanci będą odgrywać kluczową rolę w opracowywaniu ram

odpowiedzialnego wykorzystania AI w projektowaniu interfejsów, zapewniając, że personalizacja ulepsza doświadczenia użytkowników bez naruszania prywatności czy manipulowania zachowaniem użytkowników w nieetyczny sposób.

Patrząc w przyszłość, GenUI przedstawia zarówno ekscytujące możliwości, jak i znaczące wyzwania. Ma potencjał tworzenia bardziej intuicyjnych, wydajnych i satysfakcjonujących doświadczeń cyfrowych dla użytkowników na całym świecie. Choć będzie wymagać od projektantów adaptacji i zdobycia nowych umiejętności, oferuje również bezprecedensową możliwość kształtowania przyszłości interakcji człowiek-komputer w głęboki i znaczący sposób. Droga do w pełni rozwiniętych systemów GenUI będzie niewątpliwie złożona, ale potencjalne korzyści w zakresie poprawy doświadczeń użytkownika i dostępności cyfrowej czynią ją przyszłością, o którą warto zabiegać.

Inteligentna Orkiestracja Przepływu Pracy

W dziedzinie tworzenia aplikacji, przepływy pracy odgrywają kluczową rolę w definiowaniu sposobu, w jaki zadania, procesy i interakcje użytkownika są strukturyzowane i wykonywane. W miarę jak aplikacje stają się coraz bardziej złożone, a oczekiwania użytkowników rosną, potrzeba inteligentnej i adaptacyjnej orkiestracji przepływu pracy staje się coraz bardziej widoczna.

Podejście “Inteligentnej Orkiestracji Przepływu Pracy” koncentruje się na wykorzystaniu komponentów AI do dynamicznej orkiestracji i optymalizacji złożonych przepływów pracy w aplikacjach. Celem jest tworzenie aplikacji, które są bardziej wydajne, responsywne i zdolne do adaptacji w oparciu o dane i kontekst w czasie rzeczywistym.

W tym rozdziale zbadamy kluczowe zasady i wzorce, które stanowią podstawę podejścia inteligentnej orkiestracji przepływu pracy. Przyjrzymy się, jak AI może być wykorzystana do inteligentnego kierowania zadaniami, automatyzacji podejmowania decyzji i dynamicznego dostosowywania przepływów pracy w oparciu o różne czynniki, takie jak zachowanie użytkownika, wydajność systemu i reguły biznesowe. Poprzez praktyczne przykłady i scenariusze z rzeczywistego świata, pokażemy transformacyjny potencjał AI w usprawnianiu i optymalizacji przepływów pracy w aplikacjach.

Niezależnie od tego, czy tworzysz aplikacje korporacyjne ze złożonymi procesami biznesowymi, czy aplikacje konsumenckie z dynamicznymi ścieżkami użytkownika, wzorce i techniki omówione w tym rozdziale wyposażą Cię w wiedzę i narzędzia do tworzenia inteligentnych i wydajnych przepływów pracy, które poprawiają ogólne doświadczenie użytkownika i zwiększają wartość biznesową.

Potrzeba Biznesowa

Tradycyjne podejścia do zarządzania przepływem pracy często opierają się na predefiniowanych regułach i statycznych drzewach decyzyjnych, które mogą być sztywne, nieelastyczne i niezdolne do radzenia sobie z dynamicznym charakterem nowoczesnych aplikacji.

Rozważmy scenariusz, w którym aplikacja e-commerce musi obsłużyć złożony proces realizacji zamówień. Przepływ pracy może obejmować wiele kroków, takich jak walidacja zamówienia, sprawdzenie stanu magazynowego, przetwarzanie płatności, wysyłka i powiadomienia dla klientów. Każdy krok może mieć własny zestaw reguł, zależności, integracji zewnętrznych i mechanizmów obsługi wyjątków. Zarządzanie takim przepływem pracy ręcznie lub poprzez zakodowaną na stałe logikę może szybko stać się uciążliwe, podatne na błędy i trudne w utrzymaniu.

Ponadto, w miarę jak aplikacja się rozwija, a liczba jednoczesnych użytkowników rośnie, przepływ pracy może wymagać dostosowania i optymalizacji w oparciu o dane

w czasie rzeczywistym i wydajność systemu. Na przykład, w okresach szczytowego ruchu, aplikacja może potrzebować dynamicznie dostosować przepływ pracy, aby priorytetyzować określone zadania, efektywnie alokować zasoby i zapewnić płynne doświadczenie użytkownika.

W tym miejscu pojawia się podejście "Inteligentnej Orkiestracji Przepływu Pracy". Wykorzystując komponenty AI, programiści mogą tworzyć przepływy pracy, które są inteligentne, adaptacyjne i samooptymalizujące się. AI może analizować ogromne ilości danych, uczyć się z przeszłych doświadczeń i podejmować świadome decyzje w czasie rzeczywistym, aby efektywnie orkiestrować przepływ pracy.

Kluczowe Korzyści

1. **Zwiększona Wydajność**: AI może optymalizować przydzielanie zadań, wykorzystanie zasobów i wykonywanie przepływu pracy, prowadząc do szybszego przetwarzania i poprawy ogólnej efektywności.
2. **Adaptowalność**: Przepływy pracy oparte na AI mogą dynamicznie dostosowywać się do zmieniających się warunków, takich jak wahania w zapotrzebowaniu użytkowników, wydajności systemu czy wymaganiach biznesowych, zapewniając, że aplikacja pozostaje responsywna i odporna.
3. **Zautomatyzowane Podejmowanie Decyzji**: AI może automatyzować złożone procesy decyzyjne w ramach przepływu pracy, zmniejszając potrzebę ręcznej interwencji i minimalizując ryzyko błędów ludzkich.
4. **Personalizacja**: AI może analizować zachowanie użytkowników, preferencje i kontekst, aby personalizować przepływ pracy i dostarczać spersonalizowane doświadczenia poszczególnym użytkownikom.
5. **Skalowalność**: Przepływy pracy wspierane przez AI mogą się płynnie skalować, aby obsługiwać rosnące wolumeny danych i interakcji użytkowników, bez kompromisów w zakresie wydajności czy niezawodności.

W kolejnych sekcjach zbadamy kluczowe wzorce i techniki, które umożliwiają implementację inteligentnych przepływów pracy i pokażemy rzeczywiste przykłady tego, jak AI transformuje zarządzanie przepływem pracy w nowoczesnych aplikacjach.

Kluczowe Wzorce

Aby zaimplementować inteligentną orkiestrację przepływu pracy w aplikacjach, programiści mogą wykorzystać kilka kluczowych wzorców, które wykorzystują moc AI. Wzorce te zapewniają ustrukturyzowane podejście do projektowania i zarządzania przepływami pracy, umożliwiając aplikacjom adaptację, optymalizację i automatyzację procesów w oparciu o dane w czasie rzeczywistym i kontekst. Przyjrzyjmy się niektórym z fundamentalnych wzorców w inteligentnej orkiestracji przepływu pracy.

Dynamiczne Trasowanie Zadań

Ten wzorzec polega na wykorzystaniu AI do inteligentnego trasowania zadań w ramach przepływu pracy w oparciu o różne czynniki, takie jak priorytet zadania, dostępność zasobów i wydajność systemu. Algorytmy AI mogą analizować charakterystykę każdego zadania, uwzględniać aktualny stan systemu i podejmować świadome decyzje w celu przypisania zadań do najbardziej odpowiednich zasobów lub ścieżek przetwarzania. Dynamiczne trasowanie zadań zapewnia efektywną dystrybucję i wykonanie zadań, optymalizując ogólną wydajność przepływu pracy.

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

  attr_accessor :task

  # list of functions that can be called by the AI entirely at its
  # discretion depending on the task received

  function :analyze_task_priority do
    TaskPriorityAnalyzer.perform(task)
  end

  function :check_resource_availability, # ...
  function :assess_system_performance, # ...
  function :assign_task_to_resource, # ...

  DIRECTIVE = "You are a task router, responsible for intelligently
   assigning tasks to available resources based on priority, resource
   availability, and system performance..."

  def initialize(task)
    self.task = task
    transcript << { system: DIRECTIVE }
    transcript << { user: task.to_json }
  end

  def perform
    while task.unassigned?
      chat_completion

      # todo: add max loop counter and break
    end

    # capture the transcript for later analysis
    task.update(routing_transcript: transcript)
  end
end
```

Zwróć uwagę na pętlę utworzoną przez wyrażenie `while` w linii 29, która kontynuuje wysyłanie zapytań do AI, dopóki zadanie nie zostanie przypisane. W linii 35 zapisujemy

transkrypt zadania do późniejszej analizy i debugowania, jeśli okaże się to konieczne.

Podejmowanie Decyzji Kontekstowych

Możesz użyć bardzo podobnego kodu do podejmowania decyzji uwzględniających kontekst w przepływie pracy. Analizując istotne dane, takie jak preferencje użytkownika, wzorce historyczne i dane w czasie rzeczywistym, komponenty AI mogą określić najbardziej odpowiedni przebieg działań w każdym punkcie decyzyjnym przepływu pracy. Dostosuj zachowanie przepływu pracy na podstawie konkretnego kontekstu każdego użytkownika lub scenariusza, zapewniając spersonalizowane i zoptymalizowane doświadczenia.

Adaptacyjna Kompozycja Przepływu Pracy

Ten wzorzec koncentruje się na dynamicznym komponowaniu i dostosowywaniu przepływów pracy w oparciu o zmieniające się wymagania lub warunki. AI może analizować aktualny stan przepływu pracy, identyfikować wąskie gardła lub nieefektywności i automatycznie modyfikować strukturę przepływu pracy w celu optymalizacji wydajności. Adaptacyjna kompozycja przepływu pracy pozwala aplikacjom na ciągłą ewolucję i usprawnianie procesów bez konieczności ręcznej interwencji.

Obsługa i Odzyskiwanie po Wyjątkach

Obsługa i odzyskiwanie po wyjątkach są kluczowymi aspektami inteligentnej orkiestracji przepływu pracy. Podczas pracy z komponentami AI i złożonymi przepływami pracy istotne jest przewidywanie i eleganckie obsługiwanie wyjątków w celu zapewnienia stabilności i niezawodności systemu.

Oto kluczowe aspekty i techniki obsługi i odzyskiwania po wyjątkach w inteligentnych przepływach pracy:

1. **Propagacja Wyjątków**: Zaimplementuj spójne podejście do propagacji wyjątków między komponentami przepływu pracy. Gdy wystąpi wyjątek w komponencie, powinien zostać przechwycony, zarejestrowany i przekazany do orkiestratora lub oddzielnego komponentu odpowiedzialnego za obsługę wyjątków. Celem jest centralizacja obsługi wyjątków i zapobieganie ich cichemu pochłanianiu, a także otwieranie możliwości dla Inteligentnej Obsługi Błędów.
2. **Mechanizmy Ponawiania**: Mechanizmy ponawiania pomagają zwiększyć odporność przepływu pracy i elegancko obsługiwać tymczasowe awarie. Zdecydowanie warto zaimplementować mechanizmy ponawiania dla przejściowych lub możliwych do naprawienia wyjątków, takich jak problemy z łącznością sieciową lub niedostępność zasobów, które mogą być automatycznie ponawiane po określonym opóźnieniu. Posiadanie orkiestratora lub mechanizmu obsługi wyjątków opartego na AI oznacza, że strategie ponawiania nie muszą być mechaniczne, opierające się na stałych algorytmach, takich jak wykładnicze wycofywanie. Możesz pozostawić obsługę ponowień "do uznania" komponentu AI odpowiedzialnego za decydowanie o sposobie obsługi wyjątku.
3. **Strategie awaryjne**: Jeśli komponent AI nie dostarczy prawidłowej odpowiedzi lub napotka błąd—co jest częstym zjawiskiem, biorąc pod uwagę jego nowatorski charakter—należy mieć przygotowany mechanizm awaryjny, aby zapewnić ciągłość przepływu pracy. Może to obejmować użycie wartości domyślnych, alternatywnych algorytmów lub Human In The Loop do podejmowania decyzji i utrzymania ciągłości przepływu pracy.
4. **Działania kompensacyjne**: Dyrektywy orkiestratora powinny zawierać instrukcje dotyczące działań kompensacyjnych w celu obsługi wyjątków, których nie można rozwiązać automatycznie. Działania kompensacyjne to kroki podejmowane w celu cofnięcia lub złagodzenia skutków nieudanej operacji. Na przykład, jeśli nie powiedzie się etap przetwarzania płatności, działaniem kompensacyjnym może być wycofanie transakcji i powiadomienie użytkownika. Działania kompensacyjne pomagają utrzymać spójność i integralność danych

w obliczu wyjątków.

5. **Monitorowanie i alarmowanie o wyjątkach**: Skonfiguruj mechanizmy monitorowania i alarmowania w celu wykrywania i powiadamiania odpowiednich interesariuszy o krytycznych wyjątkach. Orkiestrator może zostać wyposażony w progi i reguły wyzwalające alarmy, gdy wyjątki przekroczą określone limity lub gdy wystąpią określone typy wyjątków. Umożliwia to proaktywną identyfikację i rozwiązywanie problemów, zanim wpłyną one na cały system.

Oto przykład obsługi wyjątków i odzyskiwania w komponencie przepływu pracy w Ruby:

```ruby
class InventoryManager
  def check_availability(order)
    begin
      # Perform inventory check logic
      inventory = Inventory.find_by(product_id: order.product_id)
      if inventory.available_quantity >= order.quantity
        return true
      else
        raise InsufficientInventoryError,
              "Insufficient inventory for product #{order.product_id}"
      end
    rescue InsufficientInventoryError => e
      # Log the exception
      logger.error("Inventory check failed: #{e.message}")

      # Retry the operation after a delay
      retry_count ||= 0
      if retry_count < MAX_RETRIES
        retry_count += 1
        sleep(RETRY_DELAY)
        retry
      else
        # Fallback to manual intervention
        NotificationService.admin("Inventory check failed: Order #{order.id}")
        return false
      end
```

```
    end
  end
end
```

W tym przykładzie, komponent `InventoryManager` sprawdza dostępność produktu dla danego zamówienia. Jeśli dostępna ilość jest niewystarczająca, zgłasza wyjątek `InsufficientInventoryError`. Wyjątek jest przechwytywany, rejestrowany, a następnie uruchamiany jest mechanizm ponownych prób. Jeśli limit ponownych prób zostanie przekroczony, komponent przechodzi do interwencji ręcznej poprzez powiadomienie administratora.

Poprzez wdrożenie solidnych mechanizmów obsługi i odzyskiwania po wystąpieniu wyjątków, możesz zapewnić, że twoje inteligentne przepływy pracy są odporne, łatwe w utrzymaniu i zdolne do elegackiego radzenia sobie z nieoczekiwanymi sytuacjami.

Te wzorce stanowią fundament inteligentnej orkiestracji przepływu pracy i mogą być łączone oraz dostosowywane do konkretnych wymagań różnych aplikacji. Wykorzystując te wzorce, programiści mogą tworzyć przepływy pracy, które są elastyczne, odporne i zoptymalizowane pod kątem wydajności oraz doświadczenia użytkownika.

W następnej sekcji zbadamy, jak te wzorce mogą być implementowane w praktyce, używając przykładów z rzeczywistych zastosowań i fragmentów kodu, aby zilustrować integrację komponentów AI w zarządzaniu przepływem pracy.

Praktyczna implementacja inteligentnej orkiestracji przepływu pracy

Teraz, gdy poznaliśmy kluczowe wzorce w inteligentnej orkiestracji przepływu pracy, zagłębmy się w to, jak te wzorce mogą być implementowane w aplikacjach

rzeczywistych. Przedstawimy praktyczne przykłady i fragmenty kodu, aby zilustrować integrację komponentów AI w zarządzaniu przepływem pracy.

Inteligentny procesor zamówień

Przyjrzyjmy się praktycznemu przykładowi implementacji inteligentnej orkiestracji przepływu pracy przy użyciu wspomaganego przez AI komponentu `OrderProcessor` w aplikacji e-commerce napisanej w Ruby on Rails. `OrderProcessor` realizuje koncepcję Process Manager Enterprise Integration, którą po raz pierwszy spotkaliśmy w Rozdziale 3 podczas omawiania Wielości Pracowników. Komponent będzie odpowiedzialny za zarządzanie przepływem realizacji zamówień, podejmowanie decyzji dotyczących routingu na podstawie wyników pośrednich oraz orkiestrację wykonania różnych kroków przetwarzania.

Proces realizacji zamówienia obejmuje wiele kroków, takich jak walidacja zamówienia, sprawdzenie stanu magazynu, przetwarzanie płatności i wysyłka. Każdy krok jest zaimplementowany jako osobny proces roboczy, który wykonuje określone zadanie i zwraca wynik do `OrderProcessor`. Kroki nie są obowiązkowe i nie muszą być nawet wykonywane w określonej kolejności.

Oto przykładowa implementacja `OrderProcessor`. Zawiera ona dwa mixiny z Raix. Pierwszy (`ChatCompletion`) daje mu możliwość wykonywania chat completion, co czyni go komponentem AI. Drugi (`FunctionDispatch`) umożliwia wywoływanie funkcji przez AI, pozwalając na odpowiadanie na prompt wywołaniem funkcji zamiast wiadomością tekstową.

Funkcje robocze (`validate_order`, `check_inventory` i inne) delegują zadania do swoich odpowiednich klas roboczych, które mogą być komponentami SI lub innymi, z jedynym wymogiem, aby zwracały wyniki swojej pracy w formacie, który można przedstawić jako ciąg znaków.

Podobnie jak wszystkie inne przykłady w tej części książki, ten kod jest praktycznie pseudokodem i ma na celu jedynie przekazanie znaczenia wzorca oraz zainspirowanie do własnych rozwiązań. Pełne opisy wzorców i kompletne przykłady kodu znajdują się w Części 2.

```
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

```

```
36    def validate_order
37      OrderValidationWorker.perform(@order)
38    end
39
40    def check_inventory
41      InventoryCheckWorker.perform(@order)
42    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
```

W przykładzie, OrderProcessor jest inicjalizowany z obiektem zamówienia i utrzymuje transkrypt wykonania przepływu pracy, w typowym formacie transkryptu konwersacji, który jest charakterystyczny dla dużych modeli językowych. AI otrzymuje pełną kontrolę nad organizacją wykonania różnych kroków przetwarzania, takich jak walidacja zamówienia, sprawdzenie stanu magazynowego, przetwarzanie płatności i wysyłka.

Za każdym razem, gdy wywoływana jest metoda `chat_completion`, transkrypt jest wysyłany do AI w celu uzyskania odpowiedzi w postaci wywołania funkcji. To całkowicie zależy od AI, aby przeanalizować wynik poprzedniego kroku i określić odpowiednie działanie do podjęcia. Na przykład, jeśli sprawdzenie stanu magazynowego wykaże niski poziom zapasów, `OrderProcessor` może zaplanować

zadanie uzupełnienia. Jeśli przetwarzanie płatności nie powiedzie się, może zainicjować ponowną próbę lub powiadomić obsługę klienta.

> Powyższy przykład nie ma zdefiniowanych funkcji do uzupełniania zapasów czy powiadamiania obsługi klienta, ale absolutnie mógłby je mieć.

Transkrypt rośnie za każdym razem, gdy funkcja jest wywoływana i służy jako zapis wykonania przepływu pracy, zawierający wyniki każdego kroku oraz wygenerowane przez AI instrukcje dla kolejnych kroków. Ten transkrypt może być wykorzystywany do debugowania, audytu i zapewnienia widoczności procesu realizacji zamówienia.

Wykorzystując AI w `OrderProcessor`, aplikacja e-commerce może dynamicznie dostosowywać przepływ pracy w oparciu o dane w czasie rzeczywistym i inteligentnie obsługiwać wyjątki. Komponent AI może podejmować świadome decyzje, optymalizować przepływ pracy i zapewniać sprawną realizację zamówień nawet w złożonych scenariuszach.

Fakt, że jedynym wymogiem dla procesów roboczych jest zwrócenie zrozumiałego wyniku, który AI może wziąć pod uwagę przy podejmowaniu decyzji o kolejnych krokach, może uświadomić ci, jak to podejście może zmniejszyć nakład pracy związany z mapowaniem wejścia/wyjścia, które zazwyczaj jest wymagane przy integracji różnych systemów ze sobą.

Inteligentny Moderator Treści

Aplikacje mediów społecznościowych zazwyczaj wymagają przynajmniej minimalnej moderacji treści, aby zapewnić bezpieczną i zdrową społeczność. Ten przykładowy komponent `ContentModerator` wykorzystuje AI do inteligentnego organizowania przepływu moderacji, podejmując decyzje na podstawie charakterystyki treści i wyników różnych kroków moderacji.

Proces moderacji obejmuje wiele kroków, takich jak analiza tekstu, rozpoznawanie obrazów, ocena reputacji użytkownika i ręczny przegląd. Każdy krok jest zaimplementowany jako oddzielny proces roboczy, który wykonuje określone zadanie i zwraca wynik do `ContentModerator`.

Oto przykładowa implementacja `ContentModerator`:

```
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
```

W tym przykładzie `ContentModerator` jest inicjalizowany z obiektem treści i utrzymuje transkrypt moderacji w formacie konwersacji. Komponent AI ma pełną kontrolę nad procesem moderacji, decydując które kroki wykonać na podstawie charakterystyki treści i wyników każdego etapu.

Dostępne funkcje robocze, które AI może wywołać, obejmują `analyze_text`, `recognize_image`, `assess_user_reputation` oraz `escalate_to_manual_review`. Każda funkcja deleguje zadanie do odpowiedniego procesu roboczego (`TextAnalysisWorker`, `ImageRecognitionWorker`, itd.) i dodaje wynik do transkryptu moderacji, z wyjątkiem funkcji eskalacji, która działa jako stan końcowy. Wreszcie, funkcje `approve_content` i `reject_content` również działają jako stany końcowe.

Komponent AI analizuje treść i określa odpowiednie działanie do podjęcia. Jeśli treść zawiera odniesienia do obrazów, może wywołać proces roboczy `recognize_image` w celu pomocy w wizualnej weryfikacji. Jeśli którykolwiek z procesów roboczych ostrzeże o potencjalnie szkodliwej treści, AI może zdecydować o eskalacji treści

do ręcznej weryfikacji lub po prostu ją odrzucić. Jednak w zależności od powagi ostrzeżenia, AI może zdecydować się wykorzystać wyniki oceny reputacji użytkownika przy podejmowaniu decyzji o tym, jak obsłużyć treść, co do której nie ma pewności. W zależności od przypadku użycia, być może zaufani użytkownicy mają większą swobodę w tym, co mogą publikować. I tak dalej…

Podobnie jak w poprzednim przykładzie menedżera procesów, transkrypt moderacji służy jako zapis wykonania przepływu pracy, zawierający wyniki każdego kroku i decyzje generowane przez AI. Ten transkrypt może być wykorzystywany do audytu, przejrzystości i doskonalenia procesu moderacji w czasie.

Wykorzystując AI w `ContentModerator`, aplikacja mediów społecznościowych może dynamicznie dostosowywać proces moderacji w oparciu o charakterystykę treści i inteligentnie obsługiwać złożone scenariusze moderacji. Komponent AI może podejmować świadome decyzje, optymalizować przepływ pracy i zapewniać bezpieczne i zdrowe doświadczenie społeczności.

Przyjrzyjmy się dwóm kolejnym przykładom, które demonstrują predykcyjne harmonogramowanie zadań oraz obsługę wyjątków i odzyskiwanie w kontekście inteligentnej orkiestracji przepływu pracy.

Predykcyjne Harmonogramowanie Zadań w Systemie Obsługi Klienta

W aplikacji obsługi klienta zbudowanej przy użyciu Ruby on Rails, efektywne zarządzanie i priorytetyzacja zgłoszeń pomocy technicznej jest kluczowa dla zapewnienia terminowej pomocy klientom. Komponent `SupportTicketScheduler` wykorzystuje AI do predykcyjnego harmonogramowania i przydzielania zgłoszeń pomocy technicznej dostępnym agentom w oparciu o różne czynniki, takie jak pilność zgłoszenia, wiedza agenta i obciążenie pracą.

```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
```

W tym przykładzie, `SupportTicketScheduler` jest inicjalizowany z obiektem zgłoszenia serwisowego i utrzymuje transkrypt harmonogramowania. Komponent AI analizuje szczegóły zgłoszenia i predykcyjnie planuje jego przydzielenie w oparciu o czynniki takie jak pilność zgłoszenia, kompetencje agenta i przewidywane obciążenie agenta.

Dostępne funkcje, które AI może wywołać, obejmują `analyze_ticket_urgency`, `list_available_agents`, `predict_agent_workload` i `assign_ticket_to_agent`. Każda funkcja deleguje zadanie do odpowiedniego komponentu analizującego lub przewidującego i dodaje wynik do transkryptu harmonogramowania. AI ma również możliwość opóźnienia przydzielenia za pomocą funkcji `delay_assignment`.

Komponent AI bada transkrypt harmonogramowania i podejmuje świadome decyzje dotyczące przydzielania zgłoszeń. Bierze pod uwagę pilność zgłoszenia, kompetencje dostępnych agentów oraz przewidywane obciążenie każdego agenta, aby określić najbardziej odpowiedniego agenta do obsługi zgłoszenia.

Wykorzystując predykcyjne planowanie zadań, aplikacja obsługi klienta może zoptymalizować przydzielanie zgłoszeń, skrócić czas reakcji i poprawić ogólny poziom zadowolenia klientów. Proaktywne i efektywne zarządzanie zgłoszeniami serwisowymi zapewnia, że odpowiednie zgłoszenia są przydzielane odpowiednim agentom we właściwym czasie.

Obsługa Wyjątków i Odzyskiwanie w Potoku Przetwarzania Danych

Obsługa wyjątków i odzyskiwanie po awariach są niezbędne do zapewnienia integralności danych i zapobiegania ich utracie. Komponent `DataProcessingOrchestrator` wykorzystuje AI do inteligentnej obsługi wyjątków i orkiestracji procesu odzyskiwania w potoku przetwarzania danych

```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",
```

```
43          # ...
44        },
45        {
46          name: "finished",
47          # ...
48        }
49      ]
50    end
51
52    # implementation of functions that can be called by the AI
53    # entirely at its discretion, depending on the needs of the order
54
55    def validate_data
56      result = DataValidator.perform(@data_batch)
57      continue_with(result)
58    rescue ValidationException => e
59      handle_validation_exception(e)
60    end
61
62    def process_data
63      result = DataProcessor.perform(@data_batch)
64      continue_with(result)
65    rescue ProcessingException => e
66      handle_processing_exception(e)
67    end
68
69    def request_fix(description_of_fix)
70      result = SmartDataFixer.new(description_of_fix, @data_batch)
71      continue_with(result)
72    end
73
74    def retry_processing(timeout_in_seconds)
75      wait(timeout_in_seconds)
76      process_data
77    end
78
79    def mark_data_as_failed
80      @data_batch.update!(status: 'failed', transcript: @transcript)
81    end
82
83    def finished
84      @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
```

W tym przykładzie, `DataProcessingOrchestrator` jest inicjalizowany z obiektem wsadu danych i utrzymuje transkrypt przetwarzania. Komponent AI orkiestruje potokiem przetwarzania danych, obsługując wyjątki i przywracając system po awariach według potrzeb.

Dostępne funkcje, które AI może wywołać, obejmują `validate_data`, `process_data`, `request_fix`, `retry_processing` oraz `mark_data_as_failed`. Każda funkcja deleguje zadanie do odpowiedniego komponentu przetwarzania danych i dodaje wynik lub szczegóły wyjątku do transkryptu przetwarzania.

Jeśli podczas kroku `validate_data` wystąpi wyjątek walidacji, funkcja `handle_validation_exception` dodaje dane wyjątku do transkryptu i przekazuje kontrolę z powrotem do AI. Podobnie, jeśli podczas kroku `process_data` wystąpi wyjątek przetwarzania, AI może zdecydować o strategii przywracania.

W zależności od charakteru napotkanego wyjątku, AI może według własnego uznania zdecydować o wywołaniu `request_fix`, które deleguje zadanie do napędzanego

sztuczną inteligencją komponentu `SmartDataFixer` (zobacz rozdział o Samoleczącej się Danych). Moduł naprawy danych otrzymuje w prostym języku angielskim opis tego, jak powinien zmodyfikować `@data_batch`, aby można było ponowić przetwarzanie. Być może udane ponowienie próby wymagałoby usunięcia z wsadu danych rekordów, które nie przeszły walidacji i/lub skopiowania ich do innego potoku przetwarzania do przeglądu przez człowieka? Możliwości są praktycznie nieograniczone.

Poprzez włączenie obsługi wyjątków i przywracania systemu sterowanego przez AI, aplikacja przetwarzająca dane staje się bardziej odporna i tolerancyjna na błędy. `DataProcessingOrchestrator` inteligentnie zarządza wyjątkami, minimalizuje utratę danych i zapewnia płynne wykonanie przepływu przetwarzania danych.

Monitorowanie i Rejestrowanie

Monitorowanie i rejestrowanie zapewniają wgląd w postęp, wydajność i kondycję komponentów przepływu pracy napędzanych przez AI, umożliwiając programistom śledzenie i analizowanie zachowania systemu. Wdrożenie efektywnych mechanizmów monitorowania i rejestrowania jest niezbędne do debugowania, audytu i ciągłego doskonalenia inteligentnych przepływów pracy.

Monitorowanie Postępu i Wydajności Przepływu Pracy

Aby zapewnić płynne wykonanie inteligentnych przepływów pracy, ważne jest monitorowanie postępu i wydajności każdego komponentu przepływu pracy. Obejmuje to śledzenie kluczowych metryk i zdarzeń w całym cyklu życia przepływu pracy.

Ważne aspekty do monitorowania obejmują:

1. Czas Wykonania Przepływu Pracy: Pomiar czasu potrzebnego każdemu komponentowi przepływu pracy na wykonanie swojego zadania. Pomaga to zidentyfikować wąskie gardła wydajności i zoptymalizować ogólną efektywność przepływu pracy.

2. **Wykorzystanie Zasobów**: Monitorowanie wykorzystania zasobów systemowych, takich jak CPU, pamięć i przestrzeń dyskowa, przez każdy komponent przepływu pracy. Pomaga to zapewnić, że system działa w ramach swoich możliwości i może efektywnie obsłużyć obciążenie.

3. **Współczynniki błędów i wyjątki**: Śledź występowanie błędów i wyjątków w komponentach przepływu pracy. Pomaga to zidentyfikować potencjalne problemy i umożliwia proaktywną obsługę błędów oraz przywracanie systemu.

4. **Punkty decyzyjne i rezultaty**: Monitoruj punkty decyzyjne w przepływie pracy oraz rezultaty decyzji podejmowanych przez systemy AI. Dostarcza to wglądu w zachowanie i skuteczność komponentów AI.

Dane przechwycone przez procesy monitorowania mogą być wyświetlane w panelach kontrolnych lub wykorzystywane jako dane wejściowe do zaplanowanych raportów, które informują administratorów systemu o jego kondycji.

Dane z monitorowania mogą być przekazywane do procesu administratora systemu opartego na AI w celu przeglądu i potencjalnego działania!

Rejestrowanie kluczowych zdarzeń i decyzji

Rejestrowanie jest kluczową praktyką, która polega na przechwytywaniu i przechowywaniu istotnych informacji o kluczowych zdarzeniach, decyzjach i wyjątkach występujących podczas wykonywania przepływu pracy.

Ważne aspekty do rejestrowania obejmują:

1. **Inicjację i zakończenie przepływu pracy**: Rejestruj czas rozpoczęcia i zakończenia każdej instancji przepływu pracy, wraz z odpowiednimi metadanymi, takimi jak dane wejściowe i kontekst użytkownika.

2. **Wykonanie komponentów**: Rejestruj szczegóły wykonania każdego komponentu przepływu pracy, w tym parametry wejściowe, wyniki wyjściowe i wszelkie wygenerowane dane pośrednie.

3. **Decyzje AI i rozumowanie**: Rejestruj decyzje podejmowane przez komponenty AI wraz z podstawowym rozumowaniem lub poziomami pewności. Zapewnia to przejrzystość i umożliwia audyt decyzji podejmowanych przez AI.

4. **Wyjątki i komunikaty o błędach**: Rejestruj wszelkie wyjątki lub komunikaty o błędach napotkane podczas wykonywania przepływu pracy, w tym ślad stosu i istotne informacje kontekstowe.

Rejestrowanie może być implementowane przy użyciu różnych technik, takich jak zapisywanie do plików logów, przechowywanie logów w bazie danych lub wysyłanie logów do scentralizowanej usługi rejestrowania. Ważne jest, aby wybrać framework rejestrowania, który zapewnia elastyczność, skalowalność i łatwą integrację z architekturą aplikacji.

Oto przykład implementacji rejestrowania w aplikacji Ruby on Rails przy użyciu klasy `ActiveSupport::Logger`:

```
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)
```

Poprzez strategiczne rozmieszczenie instrukcji rejestrowania w komponentach

przepływu pracy i punktach decyzyjnych AI, programiści mogą gromadzić cenne informacje do debugowania, audytu i analizy.

Korzyści z Monitorowania i Rejestrowania

Wdrożenie monitorowania i rejestrowania w inteligentnej orkiestracji przepływu pracy oferuje kilka korzyści:

1. **Debugowanie i Rozwiązywanie Problemów**: Szczegółowe logi i dane z monitorowania pomagają programistom szybko identyfikować i diagnozować problemy. Zapewniają wgląd w przebieg wykonania przepływu pracy, interakcje między komponentami oraz napotkane błędy i wyjątki.

2. **Optymalizacja Wydajności**: Monitorowanie metryk wydajności pozwala programistom identyfikować wąskie gardła i optymalizować komponenty przepływu pracy dla lepszej efektywności. Analizując czasy wykonania, wykorzystanie zasobów i inne metryki, programiści mogą podejmować świadome decyzje w celu poprawy ogólnej wydajności systemu.

3. **Audyt i Zgodność**: Rejestrowanie kluczowych zdarzeń i decyzji zapewnia ścieżkę audytu dla zgodności regulacyjnej i odpowiedzialności. Umożliwia organizacjom śledzenie i weryfikację działań podejmowanych przez komponenty AI oraz zapewnienie zgodności z regułami biznesowymi i wymogami prawnymi.

4. **Ciągłe Doskonalenie**: Dane z monitorowania i rejestrowania służą jako cenne źródła informacji do ciągłego doskonalenia inteligentnych przepływów pracy. Analizując dane historyczne, identyfikując wzorce i mierząc efektywność decyzji AI, programiści mogą iteracyjnie udoskonalać i ulepszać logikę orkiestracji przepływu pracy.

Uwagi i Najlepsze Praktyki

Przy wdrażaniu monitorowania i rejestrowania w inteligentnej orkiestracji przepływu pracy, należy wziąć pod uwagę następujące najlepsze praktyki:

1. **Zdefiniowanie Jasnych Metryk Monitorowania**: Zidentyfikuj kluczowe metryki i zdarzenia, które należy monitorować w oparciu o konkretne wymagania przepływu pracy. Skup się na metrykach, które dostarczają znaczących informacji o wydajności, kondycji i zachowaniu systemu.

2. **Wdrożenie Szczegółowego Rejestrowania**: Upewnij się, że instrukcje rejestrowania są umieszczone w odpowiednich punktach w komponentach przepływu pracy i punktach decyzyjnych AI. Zbieraj istotne informacje kontekstowe, takie jak parametry wejściowe, wyniki wyjściowe i wszelkie generowane dane pośrednie.

3. **Stosowanie Rejestrowania Strukturalnego**: Przyjmij strukturalny format rejestrowania, aby ułatwić analizę i przetwarzanie danych logów. Rejestrowanie strukturalne umożliwia lepsze wyszukiwanie, filtrowanie i agregację wpisów w logach.

4. **Zarządzanie Przechowywaniem i Rotacją Logów**: Wdróż polityki przechowywania i rotacji logów, aby zarządzać przechowywaniem i cyklem życia plików logów. Określ odpowiedni okres przechowywania w oparciu o wymagania prawne, ograniczenia pamięci i potrzeby analityczne. Jeśli to możliwe, przekieruj rejestrowanie do usługi zewnętrznej, takiej jak Papertrail.

5. **Zabezpiecz wrażliwe informacje**: Zachowaj ostrożność podczas rejestrowania wrażliwych informacji, takich jak dane osobowe umożliwiające identyfikację (PII) lub poufne dane biznesowe. Wdróż odpowiednie środki bezpieczeństwa, takie jak maskowanie danych lub szyfrowanie, aby chronić wrażliwe informacje w plikach dziennika.

6. **Zintegruj z narzędziami do monitorowania i alertów**: Wykorzystaj narzędzia do monitorowania i alertów w celu scentralizowania zbierania, analizy i wizualizacji danych z monitorowania i rejestrowania. Narzędzia te mogą zapewnić wgląd w czasie rzeczywistym, generować alerty na podstawie predefiniowanych progów oraz ułatwiać proaktywne wykrywanie i rozwiązywanie problemów. Moim ulubionym z tych narzędzi jest Datadog.

Wdrażając kompleksowe mechanizmy monitorowania i rejestrowania, programiści

mogą uzyskać cenne informacje na temat zachowania i wydajności inteligentnych przepływów pracy. Informacje te umożliwiają efektywne debugowanie, optymalizację i ciągłe doskonalenie systemów orkiestracji przepływów pracy opartych na sztucznej inteligencji.

Aspekty skalowalności i wydajności

Skalowalność i wydajność to kluczowe aspekty, które należy wziąć pod uwagę podczas projektowania i implementacji inteligentnych systemów orkiestracji przepływów pracy. Wraz ze wzrostem liczby współbieżnych przepływów pracy i złożoności komponentów opartych na sztucznej inteligencji, kluczowe staje się zapewnienie, że system może efektywnie obsługiwać obciążenie i płynnie skalować się w odpowiedzi na rosnące wymagania.

Obsługa dużej liczby współbieżnych przepływów pracy

Inteligentne systemy orkiestracji przepływów pracy często muszą obsługiwać dużą liczbę współbieżnych przepływów. Aby zapewnić skalowalność, rozważ następujące strategie:

1. **Przetwarzanie asynchroniczne**: Zaimplementuj mechanizmy przetwarzania asynchronicznego, aby rozdzielić wykonywanie komponentów przepływu pracy. Pozwala to systemowi na obsługę wielu przepływów pracy jednocześnie bez blokowania lub oczekiwania na zakończenie każdego komponentu. Przetwarzanie asynchroniczne można osiągnąć za pomocą kolejek komunikatów, architektur sterowanych zdarzeniami lub frameworków do przetwarzania zadań w tle, takich jak Sidekiq.

2. **Architektura rozproszona**: Zaprojektuj architekturę systemu tak, aby wykorzystywała komponenty bezserwerowe (takie jak AWS Lambda) lub po prostu rozdzielała obciążenie między wiele węzłów lub serwerów obok głównego serwera

aplikacji. Umożliwia to skalowalność poziomą, gdzie można dodawać dodatkowe węzły w celu obsługi zwiększonej liczby przepływów pracy.

3. **Wykonywanie równoległe**: Zidentyfikuj możliwości wykonywania równoległego w ramach przepływów pracy. Niektóre komponenty przepływu pracy mogą być od siebie niezależne i mogą być wykonywane jednocześnie. Wykorzystując techniki przetwarzania równoległego, takie jak wielowątkowość lub rozproszone kolejki zadań, system może zoptymalizować wykorzystanie zasobów i skrócić całkowity czas wykonywania przepływu pracy.

Optymalizacja wydajności komponentów opartych na sztucznej inteligencji

Komponenty oparte na sztucznej inteligencji, takie jak modele uczenia maszynowego czy silniki przetwarzania języka naturalnego, mogą być obliczeniowo intensywne i wpływać na ogólną wydajność systemu orkiestracji przepływu pracy. Aby zoptymalizować wydajność komponentów AI, rozważ następujące techniki:

1. **Buforowanie**: Jeśli twoje przetwarzanie AI jest czysto generatywne i nie wymaga wyszukiwania informacji w czasie rzeczywistym ani integracji zewnętrznych w celu generowania odpowiedzi czatu, możesz rozważyć mechanizmy buforowania do przechowywania i ponownego wykorzystania wyników często używanych lub obliczeniowo kosztownych operacji.

2. **Optymalizacja Modelu**: Ciągle optymalizuj sposób wykorzystania modeli AI w komponentach przepływu pracy. Może to obejmować techniki takie jak *Destylacja Promptów* lub może to być po prostu kwestia testowania nowych modeli w miarę ich pojawiania się.

3. **Przetwarzanie Wsadowe**: Jeśli pracujesz z modelami klasy GPT-4, możesz wykorzystać techniki przetwarzania wsadowego do obsługi wielu punktów danych lub żądań w jednej partii, zamiast przetwarzać je pojedynczo. Przetwarzając dane

wsadowo, system może zoptymalizować wykorzystanie zasobów i zmniejszyć narzut związany z powtarzającymi się żądaniami do modelu.

Monitorowanie i Profilowanie Wydajności

Aby zidentyfikować wąskie gardła wydajności i zoptymalizować skalowalność inteligentnego systemu orkiestracji przepływu pracy, kluczowe jest wdrożenie mechanizmów monitorowania i profilowania. Rozważ następujące podejścia:

1. **Metryki Wydajności**: Zdefiniuj i śledź kluczowe metryki wydajności, takie jak czas odpowiedzi, przepustowość, wykorzystanie zasobów i opóźnienie. Te metryki dostarczają wglądu w wydajność systemu i pomagają zidentyfikować obszary do optymalizacji. Popularny agregator modeli AI OpenRouter zawiera metryki Host[1] i Speed[2] w każdej odpowiedzi API, co ułatwia śledzenie tych kluczowych metryk.

2. **Narzędzia Profilujące**: Wykorzystuj narzędzia profilujące do analizy wydajności poszczególnych komponentów przepływu pracy i operacji AI. Narzędzia profilujące mogą pomóc zidentyfikować wąskie gardła wydajności, nieefektywne ścieżki kodu lub operacje intensywnie wykorzystujące zasoby. Popularne narzędzia profilujące to New Relic, Scout lub wbudowane profilery dostarczone przez język programowania lub framework.

3. **Testy obciążeniowe**: Przeprowadzaj testy obciążeniowe, aby ocenić wydajność systemu pod różnymi poziomami współbieżnych obciążeń. Testy obciążeniowe pomagają zidentyfikować limity skalowalności systemu, wykryć pogorszenie wydajności i upewnić się, że system może obsłużyć oczekiwany ruch bez pogorszenia wydajności.

4. **Monitoring ciągły**: Wdróż mechanizmy ciągłego monitorowania i alertowania, aby proaktywnie wykrywać problemy z wydajnością i wąskie gardła. Skonfiguruj

[1] Host to czas potrzebny na otrzymanie pierwszego bajtu strumieniowanej generacji od hosta modelu, tzw. "czas do pierwszego bajtu."

[2] Speed jest obliczana jako liczba tokenów uzupełnienia podzielona przez całkowity czas generowania. Dla żądań niestrumieniowanych opóźnienie jest uznawane za część czasu generowania.

pulpity monitorowania i alerty do śledzenia kluczowych wskaźników wydajności (KPI) oraz otrzymywania powiadomień, gdy przekroczone zostaną predefiniowane progi. Umożliwia to szybką identyfikację i rozwiązywanie problemów z wydajnością.

Strategie skalowania

Aby obsłużyć rosnące obciążenia i zapewnić skalowalność inteligentnego systemu orkiestracji przepływu pracy, rozważ następujące strategie skalowania:

1. **Skalowanie pionowe**: Skalowanie pionowe polega na zwiększaniu zasobów (np. CPU, pamięci) pojedynczych węzłów lub serwerów w celu obsługi większych obciążeń. To podejście jest odpowiednie, gdy system wymaga większej mocy obliczeniowej lub pamięci do obsługi złożonych przepływów pracy lub operacji AI.

2. **Skalowanie poziome**: Skalowanie poziome polega na dodawaniu większej liczby węzłów lub serwerów do systemu w celu rozłożenia obciążenia. To podejście jest skuteczne, gdy system musi obsłużyć dużą liczbę współbieżnych przepływów pracy lub gdy obciążenie może być łatwo rozdzielone między wiele węzłów. Skalowanie poziome wymaga architektury rozproszonej i mechanizmów równoważenia obciążenia, aby zapewnić równomierne rozłożenie ruchu.

3. **Automatyczne skalowanie**: Wdróż mechanizmy automatycznego skalowania, aby automatycznie dostosowywać liczbę węzłów lub zasobów w zależności od zapotrzebowania na obciążenie. Automatyczne skalowanie pozwala systemowi dynamicznie skalować się w górę lub w dół w zależności od napływającego ruchu, zapewniając optymalne wykorzystanie zasobów i efektywność kosztową. Platformy chmurowe jak Amazon Web Services (AWS) czy Google Cloud Platform (GCP) zapewniają możliwości automatycznego skalowania, które można wykorzystać w inteligentnych systemach orkiestracji przepływu pracy.

Techniki optymalizacji wydajności

Oprócz strategii skalowania, rozważ następujące techniki optymalizacji wydajności, aby zwiększyć efektywność inteligentnego systemu orkiestracji przepływu pracy:

1. **Efektywne przechowywanie i pobieranie danych**: Zoptymalizuj mechanizmy przechowywania i pobierania danych używane przez komponenty przepływu pracy. Wykorzystaj wydajne indeksowanie baz danych, techniki optymalizacji zapytań i buforowanie danych, aby zminimalizować opóźnienia i poprawić wydajność operacji intensywnie wykorzystujących dane.

2. **Asynchroniczne operacje we/wy**: Wykorzystaj asynchroniczne operacje we/wy, aby zapobiec blokowaniu i poprawić responsywność systemu. Asynchroniczne operacje we/wy pozwalają systemowi obsługiwać wiele żądań jednocześnie bez oczekiwania na zakończenie operacji we/wy, maksymalizując tym samym wykorzystanie zasobów.

3. **Wydajna serializacja i deserializacja**: Zoptymalizuj procesy serializacji i deserializacji używane do wymiany danych między komponentami przepływu pracy. Wykorzystuj wydajne formaty serializacji, takie jak Protocol Buffers lub MessagePack, aby zmniejszyć narzut serializacji danych i poprawić wydajność komunikacji między komponentami.

W przypadku aplikacji opartych na Ruby, rozważ użycie Universal ID. Universal ID wykorzystuje zarówno MessagePack, jak i Brotli (połączenie stworzone z myślą o szybkości i najlepszej w swojej klasie kompresji danych). Połączenie tych bibliotek jest do 30% szybsze i osiąga współczynniki kompresji w zakresie 2-5% w porównaniu do Protocol Buffers.

4. **Kompresja i kodowanie**: Zastosuj techniki kompresji i kodowania, aby zmniejszyć rozmiar danych przesyłanych między komponentami przepływu pracy. Algorytmy kompresji, takie jak gzip lub Brotli, mogą znacząco zmniejszyć wykorzystanie przepustowości sieci i poprawić ogólną wydajność systemu.

Biorąc pod uwagę aspekty skalowalności i wydajności podczas projektowania i implementacji systemów inteligentnej orkiestracji przepływów pracy, możesz zapewnić, że twój system będzie w stanie obsłużyć dużą liczbę współbieżnych przepływów pracy, zoptymalizować wydajność komponentów opartych na AI i płynnie skalować się, aby sprostać rosnącym wymaganiom. Ciągłe monitorowanie, profilowanie i wysiłki optymalizacyjne są niezbędne do utrzymania wydajności i responsywności systemu w miarę wzrostu obciążenia i złożoności w czasie.

Testowanie i walidacja przepływów pracy

Testowanie i walidacja są kluczowymi aspektami rozwoju i utrzymania systemów inteligentnej orkiestracji przepływów pracy. Biorąc pod uwagę złożoną naturę przepływów pracy opartych na AI, istotne jest zapewnienie, że każdy komponent działa zgodnie z oczekiwaniami, ogólny przepływ pracy zachowuje się poprawnie, a decyzje AI są dokładne i niezawodne. W tej sekcji przyjrzymy się różnym technikom i aspektom testowania i walidacji inteligentnych przepływów pracy.

Testy jednostkowe komponentów przepływu pracy

Testy jednostkowe polegają na testowaniu pojedynczych komponentów przepływu pracy w izolacji, aby zweryfikować ich poprawność i niezawodność. Podczas przeprowadzania testów jednostkowych komponentów opartych na AI, weź pod uwagę następujące aspekty:

1. **Walidacja danych wejściowych**: Przetestuj zdolność komponentu do obsługi różnych typów danych wejściowych, w tym danych poprawnych i niepoprawnych. Zweryfikuj, czy komponent odpowiednio obsługuje przypadki brzegowe i dostarcza odpowiednie komunikaty o błędach lub wyjątki.

2. **Weryfikacja danych wyjściowych**: Upewnij się, że komponent generuje oczekiwane dane wyjściowe dla danego zestawu danych wejściowych. Porównaj rzeczywiste wyniki

z oczekiwanymi, aby zapewnić poprawność.

3. **Obsługa błędów**: Przetestuj mechanizmy obsługi błędów komponentu poprzez symulowanie różnych scenariuszy błędów, takich jak nieprawidłowe dane wejściowe, niedostępność zasobów czy nieoczekiwane wyjątki. Sprawdź, czy komponent prawidłowo wychwytuje i obsługuje błędy.

4. **Warunki brzegowe**: Przetestuj zachowanie komponentu w warunkach brzegowych, takich jak puste dane wejściowe, maksymalny rozmiar danych wejściowych czy wartości ekstremalne. Upewnij się, że komponent obsługuje te warunki poprawnie, bez zawieszania się czy generowania nieprawidłowych wyników.

Oto przykład testu jednostkowego dla komponentu przepływu pracy w Ruby z wykorzystaniem frameworka testowego RSpec:

```
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
```

W tym przykładzie, komponent `OrderValidator` jest testowany przy użyciu dwóch przypadków testowych: jednego dla prawidłowego zamówienia i drugiego dla

nieprawidłowego zamówienia. Przypadki testowe weryfikują, czy metoda `validate` zwraca oczekiwaną wartość logiczną w zależności od poprawności zamówienia.

Testowanie integracyjne interakcji w przepływie pracy

Testowanie integracyjne koncentruje się na weryfikacji interakcji i przepływu danych między różnymi komponentami przepływu pracy. Zapewnia, że komponenty współpracują ze sobą płynnie i generują oczekiwane rezultaty. Podczas testowania integracyjnego inteligentnych przepływów pracy należy wziąć pod uwagę następujące aspekty:

1. **Interakcja komponentów**: Testowanie komunikacji i wymiany danych między komponentami przepływu pracy. Weryfikacja, czy dane wyjściowe jednego komponentu są prawidłowo przekazywane jako dane wejściowe do następnego komponentu w przepływie pracy.

2. **Spójność danych**: Zapewnienie, że dane pozostają spójne i dokładne podczas przepływu przez workflow. Weryfikacja, czy transformacje danych, obliczenia i agregacje są wykonywane prawidłowo.

3. **Propagacja wyjątków**: Testowanie sposobu propagacji i obsługi wyjątków oraz błędów między komponentami przepływu pracy. Weryfikacja, czy wyjątki są przechwytywane, rejestrowane i odpowiednio obsługiwane, aby zapobiec zakłóceniom w przepływie pracy.

4**. Zachowanie asynchroniczne:** Jeśli przepływ pracy obejmuje komponenty asynchroniczne lub wykonanie równoległe, należy przetestować mechanizmy koordynacji i synchronizacji. Upewnienie się, że przepływ pracy zachowuje się prawidłowo w scenariuszach współbieżnych i asynchronicznych.

Oto przykład testu integracyjnego dla przepływu pracy w Ruby z wykorzystaniem frameworka testowego RSpec:

```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
```

W tym przykładzie, `OrderProcessingWorkflow` jest testowany poprzez weryfikację interakcji między różnymi komponentami przepływu pracy. Przypadek testowy ustala oczekiwania dotyczące zachowania każdego komponentu i zapewnia, że przepływ pracy przetwarza zamówienie poprawnie, odpowiednio aktualizując status zamówienia.

Testowanie punktów decyzyjnych AI

Testowanie punktów decyzyjnych AI jest kluczowe dla zapewnienia dokładności i niezawodności przepływów pracy opartych na sztucznej inteligencji. Podczas testowania punktów decyzyjnych AI, należy wziąć pod uwagę następujące aspekty:

1. **Dokładność decyzji**: Zweryfikuj, czy komponent AI podejmuje trafne decyzje na podstawie danych wejściowych i wytrenowanego modelu. Porównaj decyzje AI z oczekiwanymi wynikami lub danymi referencyjnymi.

2. **Przypadki brzegowe**: Przetestuj zachowanie komponentu AI w przypadkach brzegowych i nietypowych scenariuszach. Zweryfikuj, czy komponent AI obsługuje te przypadki płynnie i podejmuje rozsądne decyzje.

3. **Stronniczość i sprawiedliwość**: Oceń komponent AI pod kątem potencjalnej stronniczości i upewnij się, że podejmuje sprawiedliwe i bezstronne decyzje. Przetestuj komponent z różnorodnymi danymi wejściowymi i przeanalizuj wyniki pod kątem wzorców dyskryminacyjnych.

4. **Wytłumaczalność**: Jeśli komponent AI dostarcza wyjaśnienia lub uzasadnienia swoich decyzji, zweryfikuj poprawność i jasność tych wyjaśnień. Upewnij się, że wyjaśnienia są zgodne z podstawowym procesem podejmowania decyzji.

Oto przykład testowania punktu decyzyjnego AI w Ruby przy użyciu frameworka testowego RSpec:

```
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
```

W tym przykładzie komponent AI `FraudDetector` jest testowany przy użyciu dwóch przypadków testowych: jednego dla transakcji oszukańczej i drugiego dla transakcji

prawidłowej. Przypadki testowe weryfikują, czy metoda `detect_fraud` zwraca oczekiwaną wartość logiczną na podstawie charakterystyki transakcji.

Testy End-to-End

Testy end-to-end obejmują testowanie całego przepływu pracy od początku do końca, symulując rzeczywiste scenariusze i interakcje użytkownika. Zapewniają one, że przepływ pracy działa poprawnie i przynosi oczekiwane rezultaty. Podczas przeprowadzania testów end-to-end dla inteligentnych przepływów pracy, należy wziąć pod uwagę następujące aspekty:

1. **Scenariusze użytkownika**: Zidentyfikuj typowe scenariusze użytkownika i przetestuj zachowanie przepływu pracy w tych scenariuszach. Zweryfikuj, czy przepływ pracy prawidłowo obsługuje dane wejściowe użytkownika, podejmuje odpowiednie decyzje i generuje oczekiwane wyniki.

2. **Walidacja danych**: Upewnij się, że przepływ pracy waliduje i oczyszcza dane wejściowe użytkownika, aby zapobiec niespójnościom danych lub lukom w zabezpieczeniach. Przetestuj przepływ pracy z różnymi typami danych wejściowych, w tym z danymi prawidłowymi i nieprawidłowymi.

3. **Obsługa błędów**: Przetestuj zdolność przepływu pracy do przywracania sprawności po błędach i wyjątkach. Zasymuluj scenariusze błędów i zweryfikuj, czy przepływ pracy obsługuje je prawidłowo, rejestruje błędy i podejmuje odpowiednie działania naprawcze.

4. **Wydajność i skalowalność**: Oceń wydajność i skalowalność przepływu pracy w różnych warunkach obciążenia. Przetestuj przepływ pracy z dużą ilością równoczesnych żądań i zmierz czasy odpowiedzi, wykorzystanie zasobów i ogólną stabilność systemu.

Oto przykład testu end-to-end dla przepływu pracy w Ruby z wykorzystaniem frameworka testowego RSpec i biblioteki Capybara do symulowania interakcji użytkownika:

```ruby
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
```

W tym przykładzie, test end-to-end symuluje użytkownika składającego zamówienie przez interfejs internetowy. Wypełnia wymagane pola formularza, przesyła zamówienie i weryfikuje, czy zamówienie zostało pomyślnie przetworzone, wyświetlając odpowiedni komunikat potwierdzający i aktualizując status zamówienia w bazie danych.

Ciągła Integracja i Wdrażanie

Aby zapewnić niezawodność i łatwość utrzymania inteligentnych przepływów pracy, zaleca się zintegrowanie testowania i walidacji z potokiem ciągłej integracji i wdrażania (CI/CD). Umożliwia to zautomatyzowane testowanie i walidację zmian w przepływie pracy przed ich wdrożeniem na środowisku produkcyjnym. Warto rozważyć następujące praktyki:

1. **Automatyczne Wykonywanie Testów**: Skonfiguruj potok CI/CD tak, aby automatycznie uruchamiał zestaw testów przy każdej zmianie w kodzie przepływu pracy. Zapewnia to wczesne wykrywanie regresji lub błędów w procesie rozwoju.

2. **Monitorowanie Pokrycia Testami**: Mierz i monitoruj pokrycie testami komponentów przepływu pracy i punktów decyzyjnych AI. Dąż do wysokiego pokrycia testami, aby zapewnić dokładne przetestowanie krytycznych ścieżek i scenariuszy.

3. **Ciągła Informacja Zwrotna**: Zintegruj wyniki testów i metryki jakości kodu z procesem rozwoju. Zapewnij programistom ciągłą informację zwrotną na temat statusu testów, jakości kodu i wszelkich problemów wykrytych podczas procesu CI/CD.

4. **Środowiska Stagingowe**: Wdrażaj przepływ pracy w środowiskach stagingowych, które dokładnie odzwierciedlają środowisko produkcyjne. Przeprowadzaj dodatkowe testy i walidację w środowisku stagingowym, aby wychwycić wszelkie problemy związane z infrastrukturą, konfiguracją lub integracją danych.

5. **Mechanizmy Wycofywania Zmian**: Zaimplementuj mechanizmy wycofywania zmian na wypadek niepowodzeń wdrożenia lub wykrycia krytycznych problemów na produkcji. Upewnij się, że przepływ pracy może zostać szybko przywrócony do poprzedniej stabilnej wersji, aby zminimalizować przestoje i wpływ na użytkowników.

Poprzez włączenie testowania i walidacji w cały cykl rozwoju inteligentnych przepływów pracy, organizacje mogą zapewnić niezawodność, dokładność i łatwość utrzymania swoich systemów opartych na AI. Regularne testowanie i walidacja pomagają wykryć błędy, zapobiec regresjom i budować zaufanie do zachowania i rezultatów przepływu pracy.

Część 2: Wzorce

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Inżynieria promptów

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Łańcuch myślowy

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykłady

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Generowanie Treści

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Tworzenie Jednostek Strukturalnych

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Instruktaż Agentów LLM

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Korzyści i Uwagi

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przełączanie Trybu

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kiedy tego używać

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przypisanie Roli

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kiedy to stosować

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykłady

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Obiekt Promptu

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Prompt Template

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Korzyści i uwagi

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kiedy go używać:

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Structured IO

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Skalowanie Structured IO

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Korzyści i Aspekty do Rozważenia

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Łańcuchowanie promptów

Jak to działa

Kiedy tego używać

Przykład: Proces wdrażania Olympii

Moduł przepisywania promptów

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Response Fencing

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Korzyści i uwagi

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Obsługa Błędów

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Analizator Zapytań

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Implementacja

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Znakowanie Części Mowy (POS) i Rozpoznawanie Jednostek Nazewniczych (NER)

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Klasyfikacja intencji

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Ekstrakcja słów kluczowych

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Korzyści

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Modyfikator Zapytań

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak To Działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Korzyści

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Ventriloquist

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kiedy stosować

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Komponenty dyskretne

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Predykat

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kiedy go używać

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Fasada API

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kluczowe Korzyści

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kiedy Stosować

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Uwierzytelnianie i Autoryzacja

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Obsługa Żądań

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Formatowanie Odpowiedzi

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Obsługa Błędów i Przypadków Brzegowych

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kwestie skalowalności i wydajności

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Porównanie z innymi wzorcami projektowymi

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Interpreter Wyników

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kiedy go używać

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Maszyna Wirtualna

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kiedy tego użyć

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Za Kulisami Magii

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Specyfikacja i Testowanie

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Określanie Zachowania

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Pisanie Przypadków Testowych

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład: Testowanie komponentu Translator

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Odtwarzanie interakcji HTTP

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Human In The Loop (HITL)

Wzorce wysokiego poziomu

Inteligencja hybrydowa

Odpowiedź adaptacyjna

Zamiana ról między człowiekiem a SI

Eskalacja

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kluczowe korzyści

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Zastosowanie w rzeczywistym świecie: Ochrona zdrowia

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Pętla sprzężenia zwrotnego

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Zastosowania i Przykłady

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Zaawansowane Techniki Integracji Informacji Zwrotnej od Człowieka

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Pasywna radiacja informacji

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kontekstowy wyświetlacz informacji

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Powiadomienia Proaktywne

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Wyjaśniające Spostrzeżenia

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Interaktywna Eksploracja

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kluczowe Korzyści

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Zastosowania i Przykłady

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Collaborative Decision Making (CDM)

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Ciągłe uczenie się

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Zastosowania i przykłady

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Względy etyczne

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Rola HITL w ograniczaniu ryzyka związanego z AI

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Postęp technologiczny i perspektywy na przyszłość

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Wyzwania i ograniczenia systemów HITL

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Inteligentna obsługa błędów

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Tradycyjne podejścia do obsługi błędów

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kontekstowa Diagnostyka Błędów

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Inżynieria promptów dla kontekstowej diagnostyki błędów

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Generowanie ze Wspomaganiem Wyszukiwania w Diagnostyce Kontekstowej Błędów

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Inteligentne Raportowanie Błędów

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Predykcyjne Zapobieganie Błędom

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Inteligentne przywracanie po błędach

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Spersonalizowana Komunikacja Błędów

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Adaptacyjny Przepływ Obsługi Błędów

Jak to działa

Kontrola Jakości

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Eval

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Problem

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Rozwiązanie

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kwestie do rozważenia

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Zrozumienie Wzorców Referencyjnych

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak działają ewaluacje bezwzorcowe

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Zabezpieczenie

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Problem

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Rozwiązanie

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Jak to działa

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Przykład

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Kwestie do rozważenia

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Zabezpieczenia i ewaluacje: Dwie strony tego samego medalu

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Wymienność zabezpieczeń i ewaluacji bez referencji

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Wdrażanie dwufunkcyjnych guardrails i ewaluacji

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Słownik pojęć

Słownik pojęć

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

A

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

B

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

C

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

D

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

E

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

F

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

G

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

H

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

I

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

J

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

K

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

L

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

M

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

N

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

O

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

P

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Q

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

R

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

S

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

T

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

U

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

V

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

W

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Z

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/patterns-of-application-development-using-ai-pl.

Index