

Patterns of Application Development Using AI

Obie Fernandez

Foreword by Gregor Hohpe

Leanpub

Edizione Italiana

Pattern di Sviluppo di Applicazioni con l'IA (Edizione Italiana)

Obie Fernandez

Questo libro è in vendita presso

<http://leanpub.com/patterns-of-application-development-using-ai-it>

Questa versione è stata pubblicata il 2025-01-23



Questo è un libro di [Leanpub](#). Leanpub permette ad autori ed editori un processo di pubblicazione agile. La [Pubblicazione Agile](#) consente di pubblicare un ebook in corso d'opera, utilizzando strumenti leggeri e molte iterazioni per ottenere un feedback dai lettori, al fine di assicurare un libro giusto e attraente una volta completato.

© 2025 Obie Fernandez

Twitta questo libro!

Per favore, aiuta Obie Fernandez a diffondere la voce su questo libro su [Twitter](#)!

L'hashtag suggerito per questo libro è [#poaduai](#).

Scopri cosa dicono gli altri su questo libro cliccando su questo link per cercare questo hashtag su Twitter:

[#poaduai](#)

Alla mia straordinaria regina, mia musa, mia luce e amore, Victoria

Anche di **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

Indice

Prefazione di Gregor Hohpe	i
Prefazione	ii
Sul Libro	iii
Sugli Esempi di Codice	iii
Cosa Non Tratto	iii
A Chi è Destinato Questo Libro	iii
Costruire un Vocabolario Comune	iii
Come Partecipare	iii
Ringraziamenti	iii
Cosa dire delle illustrazioni?	iv
Sul Lean Publishing	iv
L'Autore	v
Introduzione	1
Riflessioni sull'Architettura del Software	2
Cos'è un Modello Linguistico di Grandi Dimensioni?	3
Comprendere l'Inferenza	5
Pensare alle Prestazioni	27
Sperimentare con Diversi Modelli LLM	29
Sistemi di IA Composti	30

Parte 1: Approcci e Tecniche Fondamentali . . . 38

Restringere il Percorso	39
Spazio Latente: Incomprensibilmente Vasto	41
Come Il Percorso Viene “Ristretto”	45
Modelli Grezzi Versus Modelli Istruiti	49
Prompt Engineering	56
Distillazione dei Prompt	72
E il fine-tuning?	79
Retrieval Augmented Generation (RAG)	81
Cos’è il Retrieval Augmented Generation?	81
Come Funziona il RAG?	81
Perché Usare il RAG nelle Tue Applicazioni?	81
Implementazione del RAG nella Tua Applicazione	81
Suddivisione in proposizioni	82
Esempi Reali di RAG	82
Intelligent Query Optimization (IQU)	83
Riordinamento (Reranking)	83
RAG Assessment (RAGAs)	83
Sfide e Prospettive Future	85
Moltitudine di Lavoratori	87
Lavoratori AI Come Componenti Riutilizzabili Indipendenti	88
Gestione degli Account	90
Applicazioni E-commerce	91
Applicazioni Sanitarie	100
Il Worker AI come Gestore dei Processi	103
Integrare i Worker IA nell’Architettura dell’Applicazione	107
Componibilità e Orchestrazione dei Worker AI	110

Combinare il NLP Tradizionale con gli LLM	119
Uso degli Strumenti	123
Cos’è l’Uso degli Strumenti?	123
Il Potenziale dell’Utilizzo degli Strumenti	125
Il Flusso di Lavoro dell’Utilizzo degli Strumenti	126
Migliori Pratiche per l’Uso degli Strumenti	141
Composizione e Concatenamento degli Strumenti	145
Direzioni Future	147
Elaborazione del Flusso	149
Implementazione di un ReplyStream	150
Il “Ciclo di Conversazione”	156
Continuazione Automatica	158
Conclusione	161
Dati Auto-riparanti	163
Caso di Studio Pratico: Riparare JSON Danneggiato	165
Considerazioni e Controindicazioni	170
Generazione Contestuale dei Contenuti	185
Personalizzazione	186
Produttività	188
Iterazione Rapida e Sperimentazione	190
Localizzazione Basata su IA	193
L’Importanza dei Test Utente e del Feedback	195
Generative UI	196
Generazione di Testo per Interfacce Utente	197
Definizione della UI Generativa	206
Esempio	208

INDICE

Il Passaggio al Design Orientato ai Risultati	211
Sfide e Considerazioni	212
Prospettive Future e Opportunità	214
Orchestrazione Intelligente dei Flussi di Lavoro	217
Esigenza Aziendale	218
Benefici Principali	219
Pattern Chiave	219
Gestione e Recupero delle Eccezioni	222
Implementare l'Orchestrazione del Flusso di Lavoro Intelligente nella Pratica .	225
Monitoraggio e Logging	240
Considerazioni sulla Scalabilità e le Prestazioni	245
Test e Validazione dei Workflow	250
Parte 2: I Pattern	258
Prompt Engineering	259
Chain of Thought	260
Cambio di Modalità	261
Assegnazione del Ruolo	262
Prompt Object	263
Template di Prompt	264
Structured IO	265
Prompt Chaining	266
Riscrittura dei Prompt	267
Response Fencing	268
Query Analyzer	269
Riscrittura delle Query	271
Ventriloquist	272

Componenti Discreti	273
Predicate	274
API Facade	275
Interprete dei Risultati	277
Macchina Virtuale	278
Specifica e Testing	278
Human In The Loop (HITL)	280
Pattern di Alto Livello	280
Escalation	281
Feedback Loop	282
Radiazione Passiva delle Informazioni	283
Collaborative Decision Making (CDM)	285
Apprendimento continuo	286
Considerazioni Etiche	286
Progressi Tecnologici e Prospettive Future	286
Gestione Intelligente degli Errori	288
Approcci Tradizionali alla Gestione degli Errori	288
Diagnosi Contestuale degli Errori	289
Segnalazione Intelligente degli Errori	290
Prevenzione Predittiva degli Errori	291
Recupero Intelligente degli Errori	291
Comunicazione Personalizzata degli Errori	292
Flusso di Gestione Adattiva degli Errori	293
Controllo Qualità	294
Eval	295
Guardrail	297
Guardrails e Valutazioni: Due Facce della Stessa Medaglia	297

Glossario	299
Glossario	299
Index	304

Prefazione di Gregor Hohpe

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Prefazione

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Sul Libro

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Sugli Esempi di Codice

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Cosa Non Tratto

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

A Chi è Destinato Questo Libro

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Costruire un Vocabolario Comune

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Partecipare

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Ringraziamenti

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Cosa dire delle illustrazioni?

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

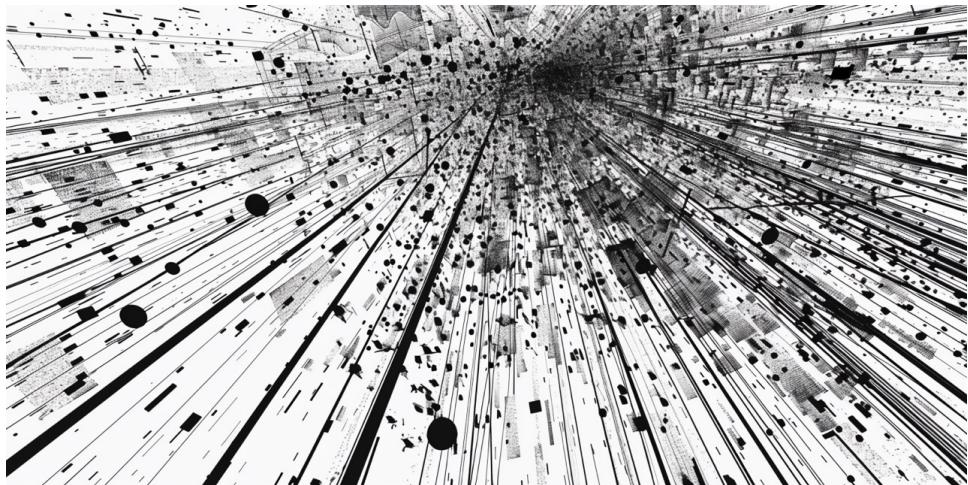
Sul Lean Publishing

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

L'Autore

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Introduzione



Se sei ansioso di iniziare a integrare i Modelli Linguistici di Grandi Dimensioni (LLM) nei tuoi progetti di programmazione, sentiti libero di immergerti direttamente nei pattern e negli esempi di codice presentati nei capitoli successivi. Tuttavia, per apprezzare pienamente la potenza e il potenziale di questi pattern, vale la pena prendersi un momento per comprendere il contesto più ampio e l'approccio coesivo che rappresentano.

I pattern non sono semplicemente una raccolta di tecniche isolate, ma piuttosto un framework unificato per integrare l'IA nelle tue applicazioni. Io utilizzo Ruby on Rails, ma questi pattern dovrebbero funzionare praticamente in qualsiasi altro ambiente di programmazione. Affrontano un'ampia gamma di problematiche, dalla gestione dei dati e l'ottimizzazione delle prestazioni all'esperienza utente e alla sicurezza, fornendo un toolkit completo per migliorare le pratiche di programmazione tradizionali con le capacità dell'IA.

Ogni categoria di pattern affronta una sfida o un'opportunità specifica che emerge quando si incorporano componenti di IA nella tua applicazione. Comprendendo le relazioni e le sinergie tra questi pattern, puoi prendere decisioni informate su dove e come applicare l'IA in modo più efficace.

I pattern non sono mai soluzioni prescrittive e non dovrebbero essere trattati come tali. Sono pensati per essere blocchi costruttivi adattabili che dovrebbero essere personalizzati secondo i requisiti e i vincoli unici della tua specifica applicazione. L'applicazione di successo di questi pattern (come qualsiasi altro nel campo del software) si basa su una profonda comprensione del dominio del problema, delle esigenze degli utenti e dell'architettura tecnica complessiva del tuo progetto.

Riflessioni sull'Architettura del Software

Ho iniziato a programmare negli anni '80 ed ero coinvolto nell'ambiente hacker, e non ho mai perso la mia mentalità hacker, anche dopo essere diventato uno sviluppatore software professionista. Fin dall'inizio, ho sempre avuto un sano scetticismo riguardo al valore effettivo che gli architetti software nelle loro torri d'avorio portassero effettivamente al tavolo.

Uno dei motivi per cui personalmente sono così entusiasta dei cambiamenti portati da questa potente nuova ondata di tecnologia IA è il suo impatto su ciò che consideriamo decisioni di *architettura software*. Sfida le nozioni tradizionali di cosa costituisca il modo “corretto” di progettare e implementare i nostri progetti software. Sfida anche se l'architettura possa essere ancora considerata principalmente come *le parti di un sistema che sono difficili da cambiare*, dato che il potenziamento dell'IA sta rendendo più facile che mai modificare qualsiasi parte del tuo progetto, in qualsiasi momento.

Forse stiamo entrando negli anni di picco dell'approccio “post-moderno” all'ingegneria del software. In questo contesto, post-moderno si riferisce a un cambiamento fondamentale rispetto ai paradigmi tradizionali, dove gli sviluppatori erano responsabili

di scrivere e mantenere ogni riga di codice. Invece, abbraccia l'idea di delegare compiti, come la manipolazione dei dati, algoritmi complessi e persino intere parti della logica applicativa, a librerie di terze parti e API esterne. Questo cambiamento post-moderno rappresenta un significativo allontanamento dalla saggezza convenzionale di costruire applicazioni da zero, e sfida gli sviluppatori a ripensare il loro ruolo nel processo di sviluppo.

Ho sempre creduto che i bravi programmati scrivano solo il codice che è assolutamente necessario scrivere, basandomi sugli insegnamenti di Larry Wall e altri luminari hacker come lui. Minimizzando la quantità di codice scritto, possiamo muoverci più velocemente, ridurre la superficie di attacco per i bug, semplificare la manutenzione e migliorare l'affidabilità complessiva delle loro applicazioni. Meno codice ci permette di concentrarci sulla logica di business principale e sull'esperienza utente, delegando altro lavoro ad altri servizi.

Ora che i sistemi basati sull'IA possono gestire compiti che prima erano dominio esclusivo del codice scritto dall'uomo, dovremmo essere in grado di essere ancora più produttivi e agili, con una focalizzazione maggiore che mai sulla creazione di valore aziendale e sull'esperienza utente.

Naturalmente ci sono compromessi nel delegare grandi parti del tuo progetto a sistemi di IA, come la potenziale perdita di controllo e la necessità di robusti meccanismi di monitoraggio e feedback. Ecco perché richiede un nuovo set di competenze e conoscenze, inclusa almeno una comprensione fondamentale di come funziona l'IA.

Cos'è un Modello Linguistico di Grandi Dimensioni?

I Modelli Linguistici di Grandi Dimensioni (LLM) sono un tipo di modello di intelligenza artificiale che hanno guadagnato notevole attenzione negli ultimi anni, sin dal lancio di GPT-3 da parte di OpenAI nel 2020. Gli LLM sono progettati per elaborare, comprendere

e generare linguaggio umano con notevole accuratezza e fluidità. In questa sezione, daremo una breve occhiata a come funzionano gli LLM e perché sono adatti per costruire componenti di sistemi intelligenti.

Nel loro nucleo, gli LLM si basano su algoritmi di deep learning, specificamente reti neurali. Queste reti sono composte da nodi interconnessi, o neuroni, che elaborano e trasmettono informazioni. L'architettura di scelta per gli LLM è spesso il modello Transformer, che si è dimostrato altamente efficace nella gestione di dati sequenziali come il testo.

I modelli Transformer si basano sul meccanismo di attenzione e sono utilizzati principalmente per compiti che coinvolgono dati sequenziali, come l'elaborazione del linguaggio naturale. I Transformer elaborano i dati in ingresso tutti in una volta anziché sequenzialmente, permettendo loro di catturare le dipendenze a lungo raggio in modo più efficace. Hanno strati di meccanismi di attenzione che aiutano il modello a concentrarsi su diverse parti dei dati in ingresso per comprendere il contesto e le relazioni.

Il processo di addestramento per gli LLM prevede l'esposizione del modello a enormi quantità di dati testuali, come libri, articoli, siti web e repository di codice. Durante l'addestramento, il modello impara a riconoscere schemi, relazioni e strutture all'interno del testo. Cattura le proprietà statistiche del linguaggio, come le regole grammaticali, le associazioni tra parole e i significati contestuali.

Una delle tecniche chiave utilizzate nell'addestramento degli LLM è l'apprendimento non supervisionato. Questo significa che il modello apprende dai dati senza etichettature o guide esplicite. Scopre schemi e rappresentazioni in modo autonomo analizzando la co-occorrenza di parole e frasi nei dati di addestramento. Questo permette agli LLM di sviluppare una profonda comprensione del linguaggio e delle sue sfumature.

Un altro aspetto importante degli LLM è la loro capacità di gestire il *contesto*. Durante l'elaborazione di un testo, gli LLM considerano non solo le singole parole ma anche il contesto circostante. Tengono conto delle parole, frasi e persino paragrafi precedenti

per comprendere il significato e l'intento del testo. Questa comprensione contestuale permette agli LLM di generare risposte coerenti e pertinenti. Uno dei modi principali in cui valutiamo le capacità di un determinato modello LLM è considerando la dimensione del contesto che possono considerare per generare risposte.

Una volta addestrati, gli LLM possono essere utilizzati per un'ampia gamma di compiti linguistici. Possono generare testo simile a quello umano, rispondere a domande, riassumere documenti, tradurre lingue e persino scrivere codice. La versatilità degli LLM li rende preziosi per costruire componenti di sistemi intelligenti che possono interagire con gli utenti, elaborare e analizzare dati testuali e generare output significativi.

Incorporando gli LLM nell'architettura dell'applicazione, è possibile creare componenti AI che comprendono ed elaborano l'input dell'utente, generano contenuti dinamici e forniscono raccomandazioni o azioni intelligenti. Ma lavorare con gli LLM richiede un'attenta considerazione dei requisiti di risorse e dei compromessi prestazionali. Gli LLM richiedono un'intensa elaborazione computazionale e possono necessitare di significativa potenza di elaborazione e memoria (in altre parole, denaro) per funzionare. La maggior parte di noi dovrà valutare le implicazioni di costo dell'integrazione degli LLM nelle nostre applicazioni e agire di conseguenza.

Comprendere l'Inferenza

L'inferenza si riferisce al processo attraverso il quale un modello genera previsioni o output basati su nuovi dati mai visti. È la fase in cui il modello addestrato viene utilizzato per prendere decisioni o generare testo, immagini o altri contenuti in risposta agli input dell'utente.

Durante la fase di addestramento, un modello AI apprende da un ampio dataset regolando i suoi parametri per minimizzare l'errore nelle sue previsioni. Una volta addestrato, il modello può applicare ciò che ha appreso a nuovi dati. L'inferenza è il modo in cui il modello utilizza i suoi schemi e conoscenze apprese per generare output.

Per gli LLM, l'inferenza comporta l'acquisizione di un prompt o testo di input e la produzione di una risposta coerente e contestualmente rilevante, come un flusso di *token* (di cui parleremo presto). Questo potrebbe essere rispondere a una domanda, completare una frase, generare una storia o tradurre un testo, tra molti altri compiti.



A differenza del modo in cui pensiamo tu ed io, il “pensiero” di un modello AI attraverso l'inferenza avviene tutto in un'unica operazione senza stato. Cioè, il suo pensiero è limitato al suo processo di generazione. Deve letteralmente pensare ad alta voce, come se ti facesse una domanda e accettasse una risposta da te solo in stile “flusso di coscienza”.

I Modelli Linguistici di Grandi Dimensioni Vengono in Molte Dimensioni e Varietà

Mentre praticamente tutti i popolari modelli linguistici di grandi dimensioni (LLM) si basano sulla stessa architettura transformer di base e sono addestrati su enormi dataset testuali, vengono in una varietà di dimensioni e sono ottimizzati per scopi diversi. La dimensione di un LLM, misurata dal numero di parametri nella sua rete neurale, ha un grande impatto sulle sue capacità. I modelli più grandi con più parametri, come GPT-4, che si dice abbia da 1 a 2 trilioni di parametri, sono generalmente più competenti e capaci rispetto ai modelli più piccoli. Tuttavia, i modelli più grandi richiedono anche molta più potenza di calcolo per funzionare, il che si traduce in maggiori spese quando li si utilizza tramite chiamate API.

Per rendere gli LLM più pratici e adatti a casi d'uso specifici, i modelli base vengono spesso ottimizzati su dataset più mirati. Ad esempio, un LLM può essere addestrato su un ampio corpus di dialoghi per specializzarlo nell'AI conversazionale. Altri sono [addestrati sul codice](#) per dotarli di conoscenze di programmazione. Ci sono persino modelli che sono [appositamente addestrati per interazioni in stile roleplay con gli utenti!](#)

Modelli di Recupero vs Modelli Generativi

Nel mondo dei modelli linguistici di grandi dimensioni (LLM), esistono due approcci principali per generare risposte: i modelli basati sul recupero e i modelli generativi. Ogni approccio ha i propri punti di forza e di debolezza, e comprendere le differenze tra loro può aiutarti a scegliere il modello più adatto al tuo caso specifico.

Modelli Basati sul Recupero

I modelli basati sul recupero, noti anche come modelli di recupero delle informazioni, generano risposte cercando in un ampio database di testi preesistenti e selezionando i passaggi più rilevanti in base alla query di input. Questi modelli non generano nuovo testo da zero, ma piuttosto uniscono estratti dal database per formare una risposta coerente.

Uno dei principali vantaggi dei modelli basati sul recupero è la loro capacità di fornire informazioni fattuali accurate e aggiornate. Poiché si basano su un database di testi curati, possono estrarre informazioni rilevanti da fonti affidabili e presentarle all'utente. Questo li rende particolarmente adatti per applicazioni che richiedono risposte precise e fattuali, come i sistemi di domanda e risposta o le basi di conoscenza.

Tuttavia, i modelli basati sul recupero presentano alcune limitazioni. Sono validi solo quanto il database su cui effettuano la ricerca, quindi la qualità e la copertura del database influenzano direttamente le prestazioni del modello. Inoltre, questi modelli potrebbero avere difficoltà a generare risposte coerenti e dal suono naturale, essendo limitati al testo disponibile nel database.

In questo libro non trattiamo l'uso dei modelli di recupero puri.

Modelli Generativi

I modelli generativi, d'altra parte, creano nuovo testo da zero basandosi sui pattern e le relazioni che hanno appreso durante l'addestramento. Questi modelli utilizzano la loro

comprensione del linguaggio per generare risposte originali adattate al prompt di input.

Il principale punto di forza dei modelli generativi è la loro capacità di produrre testo creativo, coerente e contestualmente rilevante. Possono partecipare a conversazioni aperte, generare storie e persino scrivere codice. Questo li rende ideali per applicazioni che richiedono interazioni più aperte e dinamiche, come chatbot, creazione di contenuti e assistenti per la scrittura creativa.

Tuttavia, i modelli generativi possono talvolta produrre informazioni incoerenti o fattualmente errate, poiché si basano sui pattern appresi durante l'addestramento piuttosto che su un database curato di fatti. Possono anche essere più inclini a pregiudizi e allucinazioni, generando testo plausibile ma non necessariamente vero.

Esempi di LLM generativi includono la serie GPT di OpenAI (GPT-3, GPT-4) e Claude di Anthropic.

Modelli Ibridi

Diversi LLM disponibili commercialmente combinano entrambi gli approcci di recupero e generativi in un modello ibrido. Questi modelli utilizzano tecniche di recupero per trovare informazioni rilevanti da un database e poi usano tecniche generative per sintetizzare queste informazioni in una risposta coerente.

I modelli ibridi mirano a combinare l'accuratezza fattuale dei modelli basati sul recupero con le capacità di generazione del linguaggio naturale dei modelli generativi. Possono fornire informazioni più affidabili e aggiornate mantenendo al contempo la capacità di partecipare a conversazioni aperte.

Quando si sceglie tra modelli basati sul recupero e modelli generativi, è necessario considerare i requisiti specifici della propria applicazione. Se l'obiettivo principale è fornire informazioni accurate e fattuali, un modello basato sul recupero potrebbe essere la scelta migliore. Se l'applicazione richiede interazioni più aperte e creative, un modello generativo potrebbe essere più adatto. I modelli ibridi offrono un equilibrio tra

i due approcci e possono essere una buona scelta per applicazioni che richiedono sia accuratezza fattuale che generazione del linguaggio naturale.

In definitiva, la scelta tra modelli basati sul recupero e modelli generativi dipende dal caso d'uso specifico e dal compromesso tra accuratezza, creatività e flessibilità. Comprendendo i punti di forza e le limitazioni di ciascun approccio, puoi prendere decisioni informate.

Addestramento tramite Istruzioni

L'addestramento tramite istruzioni è un sottoinsieme dell'addestramento conversazionale che utilizza prompt e risposte scritte da esseri umani per migliorare la capacità degli LLM di comprendere e seguire le istruzioni. Ecco alcuni esempi semplici per illustrare come l'addestramento tramite istruzioni su prompt e risposte scritte da esseri umani migliora la capacità degli LLM di comprendere e seguire le istruzioni:

Esempio 1: Assistenza Clienti

Istruzione: "Fornisci una risposta a un cliente che richiede un rimborso."

Prompt Scritto da Umano: "Vorrei richiedere un rimborso per il mio acquisto recente. Il prodotto non ha soddisfatto le mie aspettative."

Risposta Scritta da Umano: "Mi dispiace sentire che il prodotto non ha soddisfatto le tue aspettative. Per elaborare il tuo rimborso, ti preghiamo di fornire il numero d'ordine e il motivo del reso. Esamineremo la tua richiesta e ti risponderemo entro 2-3 giorni lavorativi."

Esempio 2: Assistenza Educativa

Istruzione: "Spiega il concetto di fotosintesi a un bambino di 10 anni."

Prompt Scritto da Umano: "Puoi spiegare come le piante producono il loro cibo?"

Risposta Scritta da Umano: "Certo! Le piante producono il loro cibo attraverso un processo chiamato fotosintesi. Utilizzano la luce solare, l'acqua e l'anidride carbonica

per creare cibo e ossigeno. Le foglie della pianta hanno una parte speciale chiamata clorofilla che le aiuta a catturare la luce solare.”

Esempio 3: Assistente Personale

Istruzione: “Programma una riunione per martedì prossimo alle 15:00.”

Prompt Scritto da Umano: “Puoi organizzare una riunione per martedì alle 15:00?”

Risposta Scritta da Umano: “Certo, ho programmato una riunione per martedì prossimo alle 15:00. Hai bisogno di altro?”

Il risultato è un ecosistema diversificato di LLM di diverse dimensioni e con diverse specializzazioni. I modelli più piccoli nell’intervallo di 1-7 miliardi di parametri forniscono buone capacità linguistiche generali pur essendo più efficienti da eseguire.

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

I modelli di medie dimensioni intorno ai 30-70 miliardi di parametri offrono capacità di ragionamento e di esecuzione delle istruzioni più robuste.

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

Quando si sceglie un LLM da incorporare in un’applicazione, è necessario bilanciare le capacità del modello con fattori pratici come costo, latenza, lunghezza del contesto e filtraggio dei contenuti. I modelli più piccoli, addestrati sulle istruzioni, sono spesso la scelta migliore per compiti linguistici più semplici, mentre i modelli più grandi

potrebbero essere necessari per ragionamenti o analisi complesse. Anche i dati di addestramento del modello sono una considerazione importante, poiché determinano la data limite di conoscenza del modello.



Alcuni modelli, come quelli di Perplexity, sono collegati a fonti di informazione in tempo reale, quindi non hanno effettivamente una data limite. Quando gli si pongono domande, sono in grado di decidere autonomamente di effettuare ricerche web e recuperare pagine web arbitrarie per generare una risposta.

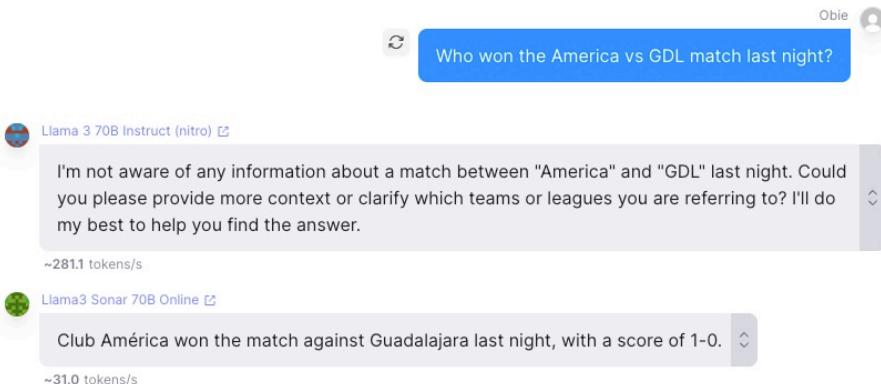


Figura 1. Llama3 con e senza accesso online

In definitiva, non esiste un LLM universale. Comprendere le variazioni nelle dimensioni del modello, nell'architettura e nell'addestramento è fondamentale per selezionare il modello giusto per un determinato caso d'uso. Sperimentare con diversi modelli è l'unico modo pratico per rivelare quali forniscono le migliori prestazioni per il compito in questione.

Tokenizzazione: Suddividere il Testo in Parti

Prima che un modello linguistico di grandi dimensioni possa elaborare il testo, questo deve essere suddiviso in unità più piccole chiamate *token*. I token possono essere parole singole, parti di parole o anche singoli caratteri. Il processo di suddivisione del testo in token è noto come tokenizzazione, ed è una fase cruciale nella preparazione dei dati per un modello linguistico.

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

Figura 2. Questa frase contiene 27 token

Diversi LLM utilizzano diverse strategie di tokenizzazione, che possono avere un impatto significativo sulle prestazioni e sulle capacità del modello. Alcuni tokenizzatori comuni utilizzati dagli LLM includono:

- **GPT (Codifica a Coppie di Byte):** I tokenizzatori GPT utilizzano una tecnica chiamata codifica a coppie di byte (BPE) per suddividere il testo in unità di sottoparole. BPE unisce iterativamente le coppie di byte più frequenti in un corpus di testo, formando un vocabolario di token di sottoparole. Questo permette al tokenizzatore di gestire parole rare e nuove suddividendole in pezzi di sottoparole più comuni. I tokenizzatori GPT sono utilizzati da modelli come GPT-3 e GPT-4.
- **Llama (SentencePiece):** I tokenizzatori Llama utilizzano la libreria SentencePiece, che è un tokenizzatore e detokenizzatore di testo non supervisionato. SentencePiece tratta il testo in input come una sequenza di caratteri Unicode e apprende un vocabolario di sottostringhe basato su un corpus di addestramento. Può gestire qualsiasi lingua codificabile in Unicode, rendendolo adatto per modelli multilingue. I tokenizzatori Llama sono utilizzati da modelli come Llama e Alpaca di Meta.

- **SentencePiece (Unigram):** I tokenizzatori SentencePiece possono anche utilizzare un algoritmo diverso chiamato Unigram, basato su una tecnica di regolarizzazione delle sottostringhe. La tokenizzazione Unigram determina il vocabolario ottimale di sottostringhe basandosi su un modello linguistico unigram, che assegna probabilità a singole unità di sottostringhe. Questo approccio può produrre sottostringhe più significative semanticamente rispetto a BPE. SentencePiece con Unigram è utilizzato da modelli come T5 e BERT di Google.
- **Google Gemini (Tokenizzazione Multimodale):** Google Gemini utilizza uno schema di tokenizzazione progettato per gestire vari tipi di dati, inclusi testo, immagini, audio, video e codice. Questa capacità multimodale permette a Gemini di elaborare e integrare diverse forme di informazione. In particolare, Google Gemini 1.5 Pro ha una finestra di contesto che può gestire milioni di token, molto più ampia rispetto ai modelli precedenti. Questa estesa finestra di contesto permette al modello di elaborare un contesto più ampio, portando potenzialmente a risposte più accurate. Tuttavia, è importante notare che lo schema di tokenizzazione di Gemini è molto più vicino a un token per carattere rispetto ad altri modelli. Questo significa che il costo effettivo dell'utilizzo dei modelli Gemini può essere significativamente più alto del previsto se si è abituati a utilizzare modelli come GPT, poiché il prezzo di Google è basato sui caratteri anziché sui token.

La scelta del tokenizzatore influenza diversi aspetti di un LLM, tra cui:

- **Dimensione del vocabolario:** Il tokenizzatore determina la dimensione del vocabolario del modello, che è l'insieme di token unici che riconosce. Un vocabolario più ampio e dettagliato può aiutare il modello a gestire una gamma più ampia di parole e frasi e persino diventare multimodale (capace di comprendere e generare più che solo testo), ma aumenta anche i requisiti di memoria e la complessità computazionale del modello.

- **Gestione di parole rare e sconosciute:** I tokenizzatori che utilizzano unità di sottostringhe, come BPE e SentencePiece, possono scomporre parole rare e sconosciute in parti di sottostringhe più comuni. Questo permette al modello di fare ipotesi ragionate sul significato di parole che non ha mai visto prima, basandosi sulle sottostringhe che contengono.
- **Supporto multilingue:** I tokenizzatori come SentencePiece, che possono gestire qualsiasi lingua codificabile in Unicode, sono particolarmente adatti per modelli multilingue che devono elaborare testo in più lingue.

Quando si sceglie un LLM per una particolare applicazione, è importante considerare il tokenizzatore che utilizza e quanto bene si allinea con le specifiche esigenze di elaborazione del linguaggio del compito in questione. Il tokenizzatore può avere un impatto significativo sulla capacità del modello di gestire terminologia specifica del dominio, parole rare e testo multilingue.

Dimensione del Contesto: Quante Informazioni Può Utilizzare un Modello Linguistico Durante l’Inferenza?

Quando si parla di modelli linguistici, la dimensione del contesto si riferisce alla quantità di testo che un modello può considerare durante l’elaborazione o la generazione delle sue risposte. È essenzialmente una misura di quante informazioni il modello può “ricordare” e utilizzare per informare i suoi output (espressa in token). La dimensione del contesto di un modello linguistico può avere un impatto significativo sulle sue capacità e sui tipi di compiti che può svolgere efficacemente.

Che Cos’è la Dimensione del Contesto?

In termini tecnici, la dimensione del contesto è determinata dal numero di token (parole o parti di parole) che un modello linguistico può elaborare in una singola sequenza di input. Questo viene spesso definito come “capacità di attenzione” o “finestra di contesto”

del modello. Maggiore è la dimensione del contesto, più testo il modello può considerare contemporaneamente quando genera una risposta o esegue un compito.

Diversi modelli linguistici hanno dimensioni di contesto variabili, che vanno da poche centinaia di token a milioni di token. Per riferimento, un tipico paragrafo di testo potrebbe contenere circa 100-150 token, mentre un intero libro potrebbe contenere decine o centinaia di migliaia di token.

Esistono persino lavori su metodi efficienti per scalare i Modelli Linguistici di Grandi Dimensioni (LLM) basati su Transformer per [input infinitamente lunghi](#) con memoria e calcolo limitati.

Perché la Dimensione del Contesto è Importante?

La dimensione del contesto di un modello linguistico ha un impatto significativo sulla sua capacità di comprendere e generare testo coerente e contestualmente rilevante. Ecco alcune ragioni chiave per cui la dimensione del contesto è importante:

- 1. Comprensione dei contenuti lunghi:** I modelli con dimensioni di contesto più ampie possono comprendere e analizzare meglio testi più lunghi, come articoli, report o persino interi libri. Questo è fondamentale per attività come la sintesi di documenti, la risposta a domande e l'analisi dei contenuti.
- 2. Mantenimento della coerenza:** Una finestra di contesto più ampia permette al modello di mantenere coerenza e consistenza su porzioni più lunghe di output. Questo è importante per attività come la generazione di storie, i sistemi di dialogo e la creazione di contenuti, dove mantenere una narrativa o un argomento coerente è essenziale. È anche assolutamente cruciale quando si utilizzano LLM per generare o trasformare dati strutturati.

3. **Cattura delle dipendenze a lungo raggio:** Alcune attività linguistiche richiedono la comprensione di relazioni tra parole o frasi che sono distanti nel testo. I modelli con dimensioni di contesto più ampie sono meglio equipaggiati per catturare queste dipendenze a lungo raggio, che possono essere importanti per attività come l'analisi del sentimento, la traduzione e la comprensione del linguaggio.
4. **Gestione di istruzioni complesse:** Nelle applicazioni in cui i modelli linguistici sono utilizzati per seguire istruzioni complesse e multi-step, una dimensione del contesto più ampia permette al modello di considerare l'intero set di istruzioni quando genera una risposta, invece di limitarsi alle ultime parole.

Esempi di Modelli Linguistici con Diverse Dimensioni di Contesto

Ecco alcuni esempi di modelli linguistici con diverse dimensioni di contesto:

- OpenAI GPT-3.5 Turbo: 4.095 token
- Mistral 7B Instruct: 32.768 token
- Anthropic Claude v1: 100.000 token
- OpenAI GPT-4 Turbo: 128.000 token
- Anthropic Claude v2: 200.000 token
- Google Gemini Pro 1.5: 2,8M token

Come si può vedere, c'è un'ampia gamma di dimensioni di contesto tra questi modelli, da circa 4.000 token per il modello OpenAI GPT-3.5 Turbo a 200.000 token per il modello Anthropic Claude v2. Alcuni modelli, come Google PaLM 2 e OpenAI GPT-4, offrono diverse varianti con dimensioni di contesto più ampie (ad esempio, versioni "32k"), che possono gestire sequenze di input ancora più lunghe. E al momento (aprile 2024) Google Gemini Pro vanta quasi 3 milioni di token!

Vale la pena notare che la dimensione del contesto può variare a seconda dell'implementazione specifica e della versione di un particolare modello. Per

esempio, il modello originale OpenAI GPT-4 ha una dimensione del contesto di 8.191 token, mentre le varianti successive di GPT-4 come Turbo e 40 hanno una dimensione del contesto molto più ampia di 128.000 token.

Sam Altman ha paragonato le attuali limitazioni del contesto ai kilobyte di memoria di lavoro con cui dovevano fare i conti i programmati di personal computer negli anni '80, e ha affermato che nel prossimo futuro saremo in grado di inserire "tutti i nostri dati personali" nel contesto di un modello linguistico di grandi dimensioni.

Scegliere la Giusta Dimensione del Contesto

Quando si seleziona un modello linguistico per una particolare applicazione, è importante considerare i requisiti di dimensione del contesto per l'attività in questione. Per attività che coinvolgono testi brevi e isolati, come l'analisi del sentimento o semplici risposte a domande, una dimensione del contesto più piccola potrebbe essere sufficiente. Tuttavia, per attività che richiedono la comprensione e la generazione di testi più lunghi e complessi, sarà probabilmente necessaria una dimensione del contesto più ampia.

È importante notare che dimensioni di contesto più ampie spesso comportano costi computazionali maggiori e tempi di elaborazione più lenti, poiché il modello deve considerare più informazioni quando genera una risposta. Pertanto, è necessario trovare un equilibrio tra dimensione del contesto e prestazioni quando si sceglie un modello linguistico per la propria applicazione.

Perché non scegliere semplicemente il modello con la dimensione del contesto più grande e riempirlo con quante più informazioni possibili? Beh, oltre ai fattori di prestazione, l'altra considerazione principale è il costo. A marzo 2024, un *singolo*

ciclo di prompt-risposta utilizzando Google Gemini Pro 1.5 con un contesto completo ti costerà quasi 8 dollari (USD). Se hai un caso d'uso che giustifica questa spesa, tanto meglio! Ma per la maggior parte delle applicazioni, è semplicemente troppo costoso di diversi ordini di grandezza.

Trovare Aghi nel Pagliaio

Il concetto di trovare un ago in un pagliaio è da tempo una metafora per le sfide del recupero in grandi insiemi di dati. Nel campo dei LLM, modifichiamo un po' questa analogia. Immagina di non cercare solo un singolo fatto sepolto all'interno di un vasto testo (come un'antologia completa dei saggi di Paul Graham), ma più fatti sparsi ovunque. Questo scenario è più simile a trovare diversi aghi in un campo sconfinato, non solo in un singolo pagliaio. Ecco il punto cruciale: non solo dobbiamo localizzare questi aghi, ma dobbiamo anche intrecciarli in un filo coerente.

Quando si tratta di recuperare e ragionare su molteplici fatti incorporati in contesti lunghi, gli LLM affrontano una duplice sfida. In primo luogo, c'è la semplice questione dell'accuratezza del recupero: naturalmente diminuisce all'aumentare del numero di fatti. Questo è prevedibile; dopotutto, tenere traccia di molteplici dettagli in un testo esteso mette alla prova anche i modelli più sofisticati.

In secondo luogo, e forse più criticamente, c'è la sfida del ragionamento con questi fatti. Una cosa è individuare i fatti; un'altra è sintetizzarli in una narrazione o risposta coerente. È qui che arriva la vera prova. Le prestazioni degli LLM nei compiti di ragionamento tendono a degradarsi ulteriormente rispetto ai semplici compiti di recupero. Questo deterioramento non riguarda solo il volume; riguarda l'intricata danza di contesto, rilevanza e inferenza.

Perché succede questo? Beh, considera le dinamiche della memoria e dell'attenzione nella cognizione umana, che si riflettono in una certa misura negli LLM. Quando

elaborano grandi quantità di informazioni, gli LLM, come gli esseri umani, possono perdere traccia dei dettagli precedenti mentre ne assorbono di nuovi. Questo è particolarmente vero nei modelli che non sono esplicitamente progettati per dare priorità o rivisitare automaticamente i segmenti precedenti del testo.

Inoltre, la capacità di un LLM di intrecciare questi fatti recuperati in una risposta coerente è simile alla costruzione di una narrazione. Questo richiede non solo il recupero di informazioni ma una comprensione profonda e un posizionamento contestuale, che rimane una sfida considerevole per l'AI attuale.

Quindi, cosa significa questo per noi come sviluppatori e integratori di queste tecnologie? Dobbiamo essere profondamente consapevoli di questi limiti quando progettiamo sistemi che si basano sugli LLM per gestire compiti complessi e di lunga durata. Comprendere che le prestazioni potrebbero degradarsi in determinate condizioni ci aiuta a stabilire aspettative realistiche e a progettare migliori meccanismi di fallback o strategie supplementari.

Modalità: Oltre il Testo

Mentre la maggior parte dei modelli linguistici oggi si concentra sull'elaborazione e la generazione di testo, c'è una crescente tendenza verso i modelli multimodali che possono nativamente ricevere e produrre diversi tipi di dati, come immagini, audio e video. Questi modelli multimodali aprono nuove possibilità per applicazioni basate sull'AI che possono comprendere e generare contenuti attraverso diverse modalità.

Cosa sono le Modalità?

Nel contesto dei modelli linguistici, le modalità si riferiscono ai diversi tipi di dati che un modello può elaborare e generare. La modalità più comune è il testo, che include il linguaggio scritto in varie forme come libri, articoli, siti web e post sui social media. Tuttavia, ci sono diverse altre modalità che vengono sempre più incorporate nei modelli linguistici:

- **Immagini:** Dati visivi come fotografie, illustrazioni e diagrammi.
- **Audio:** Dati sonori come parlato, musica e suoni ambientali.
- **Video:** Dati visivi in movimento, spesso accompagnati da audio, come clip video e film.

Ogni modalità presenta sfide e opportunità uniche per i modelli linguistici. Per esempio, le immagini richiedono che il modello comprenda concetti e relazioni visive, mentre l'audio richiede che il modello elabori e generi parlato e altri suoni.

Modelli Linguistici Multimodali

I modelli linguistici multimodali sono progettati per gestire più modalità all'interno di un singolo modello. Questi modelli tipicamente hanno componenti o livelli specializzati che possono sia comprendere gli input che generare dati di output in diverse modalità. Alcuni esempi notevoli di modelli linguistici multimodali includono:

- **OpenAI's GPT-4o:** GPT-4o è un modello linguistico di grandi dimensioni che comprende ed elabora nativamente l'audio del parlato oltre al testo. Questa capacità permette a GPT-4o di eseguire compiti come la trascrizione del linguaggio parlato, la generazione di testo da input audio e la fornitura di risposte basate su query vocali.
- **OpenAI's GPT-4 con input visivo:** GPT-4 è un modello linguistico di grandi dimensioni che può elaborare sia testo che immagini. Quando riceve un'immagine come input, GPT-4 può analizzare il contenuto dell'immagine e generare testo che descrive o risponde alle informazioni visive.
- **Google's Gemini:** Gemini è un modello multimodale che può gestire testo, immagini e video. Utilizza un'architettura unificata che permette la comprensione e la generazione cross-modale, abilitando compiti come la didascalia di immagini, il riassunto di video e la risposta a domande visive.

- **DALL-E e Stable Diffusion:** Sebbene non siano modelli linguistici nel senso tradizionale, questi modelli dimostrano la potenza dell'IA multimodale generando immagini da descrizioni testuali. Mostrano il potenziale dei modelli capaci di tradurre tra diverse modalità.

Benefici e Applicazioni dei Modelli Multimodali

I modelli linguistici multimodali offrono diversi vantaggi e permettono un'ampia gamma di applicazioni, tra cui:

- **Comprensione migliorata:** Elaborando informazioni da più modalità, questi modelli possono ottenere una comprensione più completa del mondo, simile a come gli esseri umani imparano da vari input sensoriali.
- **Generazione cross-modale:** I modelli multimodali possono generare contenuti in una modalità basandosi su input di un'altra, come creare un'immagine da una descrizione testuale o generare un riassunto video da un articolo scritto.
- **Accessibilità:** I modelli multimodali possono rendere le informazioni più accessibili traducendo tra diverse modalità, come generare descrizioni testuali di immagini per utenti ipovedenti o creare versioni audio di contenuti scritti.
- **Applicazioni creative:** I modelli multimodali possono essere utilizzati per compiti creativi come la generazione di arte, musica o video basati su prompt testuali, aprendo nuove possibilità per artisti e creatori di contenuti.

Man mano che i modelli linguistici multimodali continuano a progredire, probabilmente svolgeranno un ruolo sempre più importante nello sviluppo di applicazioni basate sull'IA che possono comprendere e generare contenuti attraverso molteplici modalità. Questo permetterà interazioni più naturali e intuitive tra esseri umani e sistemi di IA, oltre a sbloccare nuove possibilità per l'espressione creativa e la diffusione della conoscenza.

Ecosistemi dei Provider

Quando si tratta di incorporare modelli linguistici di grandi dimensioni (LLM) nelle applicazioni, esiste una gamma crescente di opzioni tra cui scegliere. Ogni principale provider di LLM, come OpenAI, Anthropic, Google e Cohere, offre il proprio ecosistema di modelli, API e strumenti. La scelta del provider giusto comporta la considerazione di vari fattori, tra cui prezzi, prestazioni, filtraggio dei contenuti, privacy dei dati e opzioni di personalizzazione.

OpenAI

OpenAI è uno dei provider di LLM più noti, con la sua serie GPT (GPT-3, GPT-4) ampiamente utilizzata in varie applicazioni. OpenAI offre un'API intuitiva che permette di integrare facilmente i loro modelli nelle applicazioni. Forniscono una gamma di modelli con diverse capacità e fasce di prezzo, dal modello base Ada al potente modello Davinci.

L'ecosistema di OpenAI include anche strumenti come OpenAI Playground, che permette di sperimentare con i prompt e mettere a punto i modelli per casi d'uso specifici. Offrono opzioni di filtraggio dei contenuti per aiutare a prevenire la generazione di contenuti inappropriati o dannosi.

Quando utilizzo i modelli OpenAI direttamente, mi affido alla libreria [ruby-openai](#) di Alex Rudall.

Anthropic

Anthropic è un altro importante attore nel settore degli LLM, con i loro modelli Claude che stanno guadagnando popolarità per le loro prestazioni elevate e considerazioni etiche. Anthropic si concentra sullo sviluppo di sistemi di IA sicuri e responsabili, con una forte enfasi sul filtraggio dei contenuti e sull'evitare output dannosi.

L'ecosistema di Anthropic include l'API di Claude, che permette di integrare il modello nelle proprie applicazioni, oltre a strumenti per l'ingegneria dei prompt e la messa a punto. Offrono anche il modello Claude Instant, che incorpora capacità di ricerca web per risposte più aggiornate e fattuali.

Quando utilizzo i modelli Anthropic direttamente, mi affido alla libreria [anthropicic](#) di Alex Rudall.

Google

Google ha sviluppato diversi potenti LLM, tra cui Gemini, BERT, T5 e PaLM. Questi modelli sono noti per le loro elevate prestazioni su un'ampia gamma di compiti di elaborazione del linguaggio naturale. L'ecosistema di Google include le librerie TensorFlow e Keras, che forniscono strumenti e framework per costruire e addestrare modelli di apprendimento automatico.

Google offre anche una Cloud AI Platform, che permette di distribuire e scalare facilmente i loro modelli nel cloud. Forniscono una gamma di modelli pre-addestrati e API per compiti come l'analisi del sentimento, il riconoscimento di entità e la traduzione.

Meta

Meta, precedentemente nota come Facebook, è profondamente investita nello sviluppo di modelli linguistici di grandi dimensioni, come evidenziato dal rilascio di modelli come LLaMA e OPT. Questi modelli si distinguono per le loro elevate prestazioni in diversi compiti linguistici e sono resi disponibili principalmente attraverso canali open-source, sostenendo l'impegno di Meta per la ricerca e la collaborazione comunitaria.

L'ecosistema di Meta è principalmente costruito attorno a PyTorch, una libreria di apprendimento automatico open-source apprezzata per le sue capacità di calcolo dinamico e la sua flessibilità, facilitando la ricerca e lo sviluppo innovativo dell'IA.

Oltre alle loro offerte tecniche, Meta pone una forte enfasi sullo sviluppo etico dell'IA. Implementano un robusto sistema di filtraggio dei contenuti e si concentrano sulla

riduzione dei pregiudizi, allineandosi con i loro più ampi obiettivi di sicurezza e responsabilità nelle applicazioni di IA.

Cohere

Cohere è un nuovo attore nel settore degli LLM, che si concentra nel rendere gli LLM più accessibili e facili da usare rispetto ai concorrenti. Il loro ecosistema include l'API Cohere, che fornisce accesso a una gamma di modelli pre-addestrati per attività come la generazione di testo, la classificazione e la sintesi.

Cohere offre anche strumenti per l'ingegneria dei prompt, il fine-tuning e il filtraggio dei contenuti. Enfatizzano la privacy e la sicurezza dei dati, con funzionalità come l'archiviazione critografata dei dati e i controlli di accesso.

Ollama

Ollama è una piattaforma self-hosted che permette agli utenti di gestire e distribuire vari modelli linguistici di grandi dimensioni (LLM) localmente sulle proprie macchine, dando loro il controllo completo sui propri modelli di IA senza dover dipendere da servizi cloud esterni. Questa configurazione è ideale per chi dà priorità alla privacy dei dati e desidera gestire le proprie operazioni di IA internamente.

La piattaforma supporta una gamma di modelli, incluse versioni di Llama, Phi, Gemma e Mistral, che variano in dimensioni e requisiti computazionali. Ollama rende facile scaricare ed eseguire questi modelli direttamente dalla riga di comando usando semplici comandi come `ollama run <model_name>`, ed è progettato per funzionare su diversi sistemi operativi tra cui macOS, Linux e Windows.

Per gli sviluppatori che cercano di integrare modelli open-source nelle loro applicazioni senza utilizzare un'API remota, Ollama offre una CLI per la gestione del ciclo di vita dei modelli simile agli strumenti di gestione dei container. Supporta anche configurazioni e prompt personalizzati, permettendo un alto grado di personalizzazione per adattare i modelli a specifiche esigenze o casi d'uso.

Ollama è particolarmente adatto per utenti esperti di tecnologia e sviluppatori grazie alla sua interfaccia a riga di comando e alla flessibilità che offre nella gestione e distribuzione dei modelli di IA. Questo lo rende uno strumento potente per aziende e individui che richiedono robuste capacità di IA senza compromettere sicurezza e controllo.

Piattaforme Multi-Modello

Inoltre, ci sono fornitori che ospitano un'ampia varietà di modelli open-source, come Together.ai e Groq. Queste piattaforme offrono flessibilità e personalizzazione, permettendoti di eseguire e, in alcuni casi, persino effettuare il fine-tuning dei modelli open-source secondo le tue specifiche esigenze. Per esempio, Together.ai fornisce accesso a una gamma di LLM open-source, permettendo agli utenti di sperimentare con diversi modelli e configurazioni. Groq si concentra sulla fornitura di completamenti ad altissime prestazioni che al momento della stesura di questo libro sembrano quasi magici

Scegliere un Fornitore di LLM

Quando si sceglie un fornitore di LLM, è necessario considerare fattori come:

- **Prezzi:** Diversi fornitori offrono diversi modelli di prezzo, dal pagamento a consumo ai piani basati su abbonamento. È importante considerare l'utilizzo previsto e il budget quando si seleziona un fornitore.
- **Prestazioni:** Le prestazioni degli LLM possono variare significativamente tra i fornitori, quindi è importante fare benchmark e testare i modelli su casi d'uso specifici prima di prendere una decisione.
- **Filtraggio dei contenuti:** A seconda dell'applicazione, il filtraggio dei contenuti può essere una considerazione critica. Alcuni fornitori offrono opzioni di filtraggio dei contenuti più robuste di altri.
- **Privacy dei dati:** Se l'applicazione gestisce dati sensibili degli utenti, è importante scegliere un fornitore con solide pratiche di privacy e sicurezza dei dati.

- **Personalizzazione:** Alcuni fornitori offrono maggiore flessibilità in termini di fine-tuning e personalizzazione dei modelli per casi d'uso specifici.

In definitiva, la scelta del fornitore di LLM dipende dai requisiti specifici e dai vincoli dell'applicazione. Valutando attentamente le opzioni e considerando fattori come prezzi, prestazioni e privacy dei dati, puoi selezionare il fornitore che meglio soddisfa le tue esigenze.

È anche importante notare che il panorama degli LLM è in costante evoluzione, con nuovi fornitori e modelli che emergono regolarmente. È consigliabile rimanere aggiornati con gli ultimi sviluppi ed essere aperti all'esplorazione di nuove opzioni man mano che diventano disponibili.

OpenRouter

In questo libro mi affiderò esclusivamente a [OpenRouter](#) come fornitore API di riferimento. La ragione è semplice: è un punto di riferimento unico per tutti i modelli commerciali e open-source più popolari. Se non vedi l'ora di sporcarti le mani con un po' di programmazione IA, uno dei migliori punti di partenza è la mia [Libreria Ruby OpenRouter](#).

Pensare alle Prestazioni

Quando si incorporano i modelli linguistici nelle applicazioni, le prestazioni sono un aspetto critico da considerare. Le prestazioni di un modello linguistico possono essere misurate in termini di *latenza* (il tempo necessario per generare una risposta) e *throughput* (il numero di richieste che può gestire per unità di tempo).

Il *Tempo al Primo Token* (TTFT) è un'altra metrica di prestazione essenziale, particolarmente rilevante per i chatbot e le applicazioni che richiedono risposte interattive in tempo reale. Il TTFT misura la latenza dal momento in cui viene ricevuta

la richiesta dell'utente al momento in cui viene generata la prima parola (o token) della risposta. Questa metrica è cruciale per mantenere un'esperienza utente fluida e coinvolgente, poiché risposte ritardate possono portare alla frustrazione e al disimpegno dell'utente.

Queste metriche di prestazione possono avere un impatto significativo sull'esperienza utente e sulla scalabilità dell'applicazione.

Diversi fattori possono influenzare le prestazioni di un modello linguistico, tra cui:

Numero di Parametri: I modelli più grandi con più parametri generalmente richiedono più risorse computazionali e possono avere una latenza più alta e un throughput più basso rispetto ai modelli più piccoli.

Hardware: Le prestazioni di un modello linguistico possono variare significativamente in base all'hardware su cui viene eseguito. I provider cloud offrono istanze GPU e TPU ottimizzate per carichi di lavoro di machine learning, che possono accelerare notevolmente l'inferenza del modello.



Uno dei vantaggi di OpenRouter è che per molti dei modelli che offre, si ha la possibilità di scegliere tra diversi provider cloud con una gamma di profili di prestazioni e costi.

Quantizzazione: Le tecniche di quantizzazione possono essere utilizzate per ridurre l'occupazione di memoria e i requisiti computazionali di un modello rappresentando pesi e attivazioni con tipi di dati a precisione inferiore. Questo può migliorare le prestazioni senza sacrificare significativamente la qualità. Come sviluppatore di applicazioni, probabilmente non ti occuperai di addestrare i tuoi modelli a diversi livelli di quantizzazione, ma è bene conoscere almeno la terminologia.

Elaborazione in batch: L'elaborazione simultanea di più richieste in batch può migliorare il throughput ammortizzando il sovraccarico del caricamento del modello e del trasferimento dati.

Caching: Il caching dei risultati dei prompt o delle sequenze di input frequentemente utilizzati può ridurre il numero di richieste di inferenza e migliorare le prestazioni complessive.

Quando si seleziona un modello linguistico per un'applicazione in produzione, è importante valutarne le prestazioni su carichi di lavoro e configurazioni hardware rappresentativi. Questo può aiutare a identificare potenziali colli di bottiglia e garantire che il modello possa soddisfare gli obiettivi di prestazione richiesti.

Vale anche la pena considerare i compromessi tra le prestazioni del modello e altri fattori come costo, flessibilità e facilità di integrazione. Ad esempio, l'utilizzo di un modello più piccolo e meno costoso con latenza inferiore potrebbe essere preferibile per applicazioni che richiedono risposte in tempo reale, mentre un modello più grande e potente potrebbe essere più adatto per l'elaborazione batch o compiti di ragionamento complesso.

Sperimentare con Diversi Modelli LLM

La scelta di un LLM è raramente una decisione permanente. Poiché nuovi e migliori modelli vengono rilasciati regolarmente, è bene costruire applicazioni in modo modulare che permetta di sostituire diversi modelli linguistici nel tempo. Prompt e dataset possono spesso essere riutilizzati tra modelli con modifiche minime. Questo permette di sfruttare gli ultimi progressi nella modellazione del linguaggio senza dover riprogettare completamente le applicazioni.



La capacità di passare facilmente tra un'ampia gamma di scelte di modelli è un'altra ragione per cui adoro OpenRouter.

Quando si passa a un nuovo modello linguistico, è importante testare e validare accuratamente le sue prestazioni e la qualità dell'output per garantire che soddisfi i requisiti dell'applicazione. Questo potrebbe comportare il riaddestramento o la messa

a punto del modello su dati specifici del dominio, nonché l'aggiornamento di eventuali componenti a valle che dipendono dagli output del modello.

Progettando applicazioni con attenzione alle prestazioni e alla modularità, è possibile creare sistemi scalabili, efficienti e a prova di futuro che possono adattarsi al panorama in rapida evoluzione della tecnologia di modellazione del linguaggio.

Sistemi di IA Composti

Prima di concludere la nostra introduzione, vale la pena menzionare che prima del 2023 e dell'esplosione di interesse per l'IA generativa scatenata da ChatGPT, gli approcci tradizionali all'IA si basavano solitamente sull'integrazione di singoli modelli chiusi. Al contrario, i *Sistemi di IA Composti* sfruttano pipeline complesse di componenti interconnessi che lavorano insieme per raggiungere un comportamento intelligente.

Al loro centro, i sistemi di IA composti consistono in molteplici moduli, ciascuno progettato per eseguire compiti o funzioni specifiche. Questi moduli possono includere generatori, recuperatori, classificatori di rango, classificatori e vari altri componenti specializzati. Suddividendo il sistema complessivo in unità più piccole e focalizzate, gli sviluppatori possono creare architetture di IA più flessibili, scalabili e manutenibili.

Uno dei vantaggi principali dei sistemi AI composti è la loro capacità di combinare i punti di forza di diverse tecniche e modelli di AI. Per esempio, un sistema potrebbe utilizzare un modello linguistico di grandi dimensioni (LLM) per la comprensione e generazione del linguaggio naturale, mentre impiega un modello separato per il recupero delle informazioni o per il processo decisionale basato su regole. Questo approccio modulare permette di selezionare gli strumenti e le tecniche migliori per ogni compito specifico, invece di affidarsi a una soluzione universale.

Tuttavia, la costruzione di sistemi AI composti presenta anche sfide uniche. In particolare, garantire la coerenza complessiva e la consistenza del comportamento del sistema richiede meccanismi robusti di testing, monitoraggio e governance.



L'avvento di potenti LLM come GPT-4 ci permette di sperimentare con i sistemi AI composti più facilmente che mai, poiché questi modelli avanzati sono capaci di gestire molteplici ruoli all'interno di un sistema composto, come la classificazione, il ranking e la generazione, oltre alle loro capacità di comprensione del linguaggio naturale. Questa versatilità permette agli sviluppatori di creare prototipi e iterare rapidamente sulle architetture AI composte, aprendo nuove possibilità per lo sviluppo di applicazioni intelligenti.

Pattern di Deployment per Sistemi AI Composti

I sistemi AI composti possono essere implementati utilizzando vari pattern, ognuno progettato per rispondere a requisiti e casi d'uso specifici. Esploriamo quattro pattern di deployment comuni: Domanda e Risposta, Risolutori di Problemi Multi-Agente/Agentici, AI Conversazionale e CoPilot.

Domanda e Risposta

I sistemi di Domanda e Risposta (Q&A) si concentrano sul fornire un recupero delle informazioni potenziato dalle capacità di comprensione dei modelli AI per funzionare come più di un semplice motore di ricerca. Combinando potenti modelli linguistici con fonti di conoscenza esterne utilizzando la [Generazione Aumentata dal Recupero \(RAG\)](#), i sistemi di Domanda e Risposta evitano le allucinazioni e forniscono risposte accurate e contestualmente rilevanti alle query degli utenti.

I componenti chiave di un sistema Q&A basato su LLM includono:

- **Comprensione e riformulazione delle query:** Analisi delle query degli utenti e loro riformulazione per meglio corrispondere alle fonti di conoscenza sottostanti.
- **Recupero della conoscenza:** Recupero delle informazioni rilevanti da fonti di dati strutturati o non strutturati basato sulla query riformulata.

- **Generazione delle risposte:** Generazione di risposte coerenti e informative integrando la conoscenza recuperata con le capacità generative del modello linguistico.

I sottosistemi RAG sono particolarmente importanti nei domini Q&A dove è cruciale fornire informazioni accurate e aggiornate, come il supporto clienti, la gestione della conoscenza o le applicazioni educative.

Risolutori di Problemi Multi-Agente/Agentici

I sistemi multi-agente, noti anche come *Agentici*, consistono in molteplici agenti autonomi che lavorano insieme per risolvere problemi complessi. Ogni agente ha un ruolo specifico, un set di competenze e accesso a strumenti o fonti di informazione rilevanti. Collaborando e scambiando informazioni, questi agenti possono affrontare compiti che sarebbero difficili o impossibili da gestire per un singolo agente.

I principi chiave dei risolutori di problemi multi-agente includono:

- **Specializzazione:** Ogni agente si concentra su un aspetto specifico del problema, sfruttando le sue capacità e conoscenze uniche.
- **Collaborazione:** Gli agenti comunicano e coordinano le loro azioni per raggiungere un obiettivo comune, spesso attraverso lo scambio di messaggi o memoria condivisa.
- **Adattabilità:** Il sistema può adattarsi a condizioni o requisiti mutevoli regolando i ruoli e i comportamenti dei singoli agenti.

I sistemi multi-agente sono particolarmente adatti per applicazioni che richiedono la risoluzione distribuita dei problemi, come l'ottimizzazione della catena di approvvigionamento, la gestione del traffico o la pianificazione della risposta alle emergenze.

AI Conversazionale

I sistemi di AI conversazionale permettono interazioni in linguaggio naturale tra utenti e agenti intelligenti. Questi sistemi combinano capacità di comprensione del linguaggio naturale, gestione del dialogo e generazione del linguaggio per fornire esperienze conversazionali coinvolgenti e personalizzate.

I componenti principali di un sistema di AI conversazionale includono:

- **Riconoscimento dell'intento:** Identificazione dell'intento dell'utente basato sul suo input, come fare una domanda, effettuare una richiesta o esprimere un sentimento.
- **Estrazione delle entità:** Estrazione di entità o parametri rilevanti dall'input dell'utente, come date, luoghi o nomi di prodotti.
- **Gestione del dialogo:** Mantenimento dello stato della conversazione, determinazione della risposta appropriata basata sull'intento dell'utente e sul contesto, e gestione delle interazioni multi-turno.
- **Generazione delle risposte:** Generazione di risposte simili a quelle umane utilizzando modelli linguistici, template o metodi basati sul recupero.

I sistemi di AI conversazionale sono comunemente utilizzati nei chatbot per il servizio clienti, assistenti virtuali e interfacce controllate vocalmente. Come menzionato in precedenza, la maggior parte degli approcci, pattern ed esempi di codice in questo libro sono direttamente estratti dal mio lavoro su un grande sistema di AI conversazionale chiamato [Olympia](#).

CoPilot

I CoPilot sono assistenti basati sull'AI che lavorano al fianco degli utenti umani per migliorare la loro produttività e capacità decisionale. Questi sistemi sfruttano una combinazione di elaborazione del linguaggio naturale, apprendimento automatico

e conoscenze specifiche del dominio per fornire raccomandazioni intelligenti, automatizzare attività e offrire supporto contestuale.

Le caratteristiche principali dei CoPilot includono:

- **Personalizzazione:** Adattamento alle preferenze individuali dell'utente, ai flussi di lavoro e agli stili di comunicazione.
- **Assistenza proattiva:** Anticipazione delle necessità dell'utente e offerta di suggerimenti o azioni pertinenti senza richieste esplicite.
- **Apprendimento continuo:** Miglioramento delle prestazioni nel tempo attraverso l'apprendimento dal feedback degli utenti, dalle interazioni e dai dati.

I CoPilot sono sempre più utilizzati in vari domini, come lo sviluppo software (ad esempio, completamento del codice e rilevamento dei bug), la scrittura creativa (ad esempio, suggerimenti di contenuti e modifica), e l'analisi dei dati (ad esempio, approfondimenti e raccomandazioni di visualizzazione)

Questi modelli di implementazione mostrano la versatilità e il potenziale dei sistemi AI composti. Comprendendo le caratteristiche e i casi d'uso di ogni modello, è possibile prendere decisioni informate durante la progettazione e l'implementazione di applicazioni intelligenti. Sebbene questo libro non tratti specificamente l'implementazione di sistemi AI composti, molti se non tutti gli stessi approcci e modelli si applicano all'integrazione di componenti AI discreti all'interno dello sviluppo di applicazioni tradizionali.

Ruoli nei Sistemi AI Composti

I sistemi AI composti sono costruiti su una base di moduli interconnessi, ciascuno progettato per svolgere un ruolo specifico. Questi moduli lavorano insieme per creare comportamenti intelligenti e risolvere problemi complessi. È utile conoscere questi ruoli quando si pensa a dove si potrebbero implementare o sostituire parti della propria applicazione con componenti AI discreti.

Generatore

I generatori sono responsabili della produzione di nuovi dati o contenuti basati su modelli appresi o input di prompt. Il mondo dell'AI ha molti tipi diversi di generatori, ma nel contesto dei modelli linguistici presentati in questo libro, i generatori possono creare testo simile a quello umano, completare frasi parziali o generare risposte alle query degli utenti. Svolgono un ruolo cruciale in attività come la creazione di contenuti, la generazione di dialoghi e l'aumento dei dati.

Estrattore

Gli estrattori sono utilizzati per cercare ed estrarre informazioni rilevanti da grandi set di dati o basi di conoscenza. Impiegano tecniche come la ricerca semantica, la corrispondenza delle parole chiave o la similarità vettoriale per trovare i punti dati più pertinenti in base a una determinata query o contesto. Gli estrattori sono essenziali per attività che richiedono un accesso rapido a informazioni specifiche, come la risposta a domande, la verifica dei fatti o la raccomandazione di contenuti.

Classificatore per Rilevanza

I classificatori per rilevanza sono responsabili dell'ordinamento o della prioritizzazione di un insieme di elementi basati su determinati criteri o punteggi di rilevanza. Assegnano pesi o punteggi a ciascun elemento e li ordinano di conseguenza. Sono comunemente utilizzati nei motori di ricerca, nei sistemi di raccomandazione o in qualsiasi applicazione dove è cruciale presentare i risultati più rilevanti agli utenti.

Classificatore

I classificatori sono utilizzati per categorizzare o etichettare punti dati basati su classi o categorie predefinite. Apprendono da dati di addestramento etichettati e quindi prevedono la classe di nuove istanze mai viste. I classificatori sono fondamentali per

attività come l’analisi del sentimento, il rilevamento dello spam o il riconoscimento delle immagini, dove l’obiettivo è assegnare una categoria specifica a ciascun input.

Strumenti e Agenti

Oltre a questi ruoli fondamentali, i sistemi AI composti spesso incorporano strumenti e agenti per migliorare la loro funzionalità e adattabilità:

- **Strumenti:** Gli strumenti sono componenti software discreti o API che eseguono azioni o calcoli specifici. Possono essere invocati da altri moduli, come generatori o estrattori, per completare sotto-attività o raccogliere informazioni aggiuntive. Esempi di strumenti includono motori di ricerca web, calcolatrici o librerie di visualizzazione dati.
- **Agenti:** Gli agenti sono entità autonome che possono percepire il loro ambiente, prendere decisioni e intraprendere azioni per raggiungere obiettivi specifici. Spesso si basano su una combinazione di diverse tecniche di AI, come la pianificazione, il ragionamento e l’apprendimento, per operare efficacemente in condizioni dinamiche o incerte. Gli agenti possono essere utilizzati per modellare comportamenti complessi o per coordinare le azioni di più moduli all’interno di un sistema AI composto.

In un sistema AI composto puro, l’interazione tra questi componenti è orchestrata attraverso interfacce e protocolli di comunicazione ben definiti. I dati fluiscono tra i moduli, con l’output di un componente che serve come input per un altro. Questa architettura modulare permette flessibilità, scalabilità e manutenibilità, poiché i singoli componenti possono essere aggiornati, sostituiti o estesi senza influenzare l’intero sistema.

Sfruttando la potenza di questi componenti e delle loro interazioni, i sistemi AI composti possono affrontare problemi complessi del mondo reale che richiedono una combinazione di diverse capacità AI. Mentre esploriamo gli approcci e i modelli per

integrare l'AI nello sviluppo delle applicazioni, ricorda che gli stessi principi e tecniche utilizzati nei sistemi AI composti possono essere applicati per creare applicazioni intelligenti, adattive e incentrate sull'utente.

Nei seguenti capitoli della Parte 1, approfondiremo gli approcci e le tecniche fondamentali per integrare i componenti AI nel tuo processo di sviluppo delle applicazioni. Dall'ingegneria dei prompt e la generazione aumentata tramite recupero ai dati auto-riparanti e all'orchestrazione intelligente del flusso di lavoro, copriremo un'ampia gamma di modelli e best practice per aiutarti a costruire applicazioni all'avanguardia basate sull'AI.

Parte 1: Approcci e Tecniche Fondamentali

Questa parte del libro presenta diversi modi di integrare l'uso dell'AI nelle tue applicazioni. I capitoli coprono una serie di approcci e tecniche correlati, che spaziano dai concetti di più alto livello come [Narrow The Path](#) e [Retrieval Augmented Generation](#) fino ad arrivare alle idee per programmare il proprio livello di astrazione sopra le API di completamento chat LLM.

L'obiettivo di questa parte del libro è aiutarti a comprendere i tipi di comportamento che puoi implementare con l'AI, prima di addentrarci troppo nei pattern di implementazione specifici che sono il focus della [Parte 2](#).

Gli approcci nella Parte 1 si basano su idee che ho utilizzato nel mio codice, pattern classici dell'architettura delle applicazioni aziendali e integrazione, oltre a metafore che ho utilizzato per spiegare le capacità dell'AI ad altre persone, inclusi gli stakeholder aziendali non tecnici.

Restringere il Percorso



“Restringere il percorso” si riferisce al concentrare l’IA sul compito da svolgere. Lo uso come un mantra ogni volta che mi sento frustrato perché l’IA si comporta in modo “stupido” o inaspettato. Il mantra mi ricorda che il fallimento è probabilmente colpa mia e che probabilmente dovrei restringere ulteriormente il percorso.

La necessità di restringere il percorso deriva dalle enormi quantità di conoscenza contenute nei modelli linguistici di grandi dimensioni, specialmente i modelli di livello mondiale come quelli di OpenAI e Anthropic che hanno letteralmente trilioni di parametri.

Avere accesso a una così ampia gamma di conoscenze è indubbiamente potente e produce comportamenti emergenti come la teoria della mente e la capacità di ragionare in modi simili a quelli umani. Tuttavia, questo volume sconvolgente di informazioni presenta anche sfide quando si tratta di generare risposte precise e accurate a prompt specifici, specialmente se questi prompt sono destinati a esibire un comportamento deterministico che può essere integrato con lo sviluppo software e gli algoritmi “normali”.

Diversi fattori portano a queste sfide.

Sovraccarico di Informazioni: I modelli linguistici di grandi dimensioni sono addestrati su enormi quantità di dati che abbracciano vari domini, fonti e periodi temporali. Questa vasta conoscenza permette loro di impegnarsi in diversi argomenti e generare risposte basate su una comprensione ampia del mondo. Tuttavia, quando si trova di fronte a un prompt specifico, il modello potrebbe faticare a filtrare le informazioni irrilevanti, contraddittorie o obsolete, portando a risposte che mancano di focus o accuratezza. A seconda di ciò che si sta cercando di fare, il puro volume di informazioni *contraddittorie* disponibili per il modello può facilmente sopraffare la sua capacità di fornire la risposta o il comportamento che si cerca.

Ambiguità Contestuale: Dato il vasto *spazio latente* di conoscenza, i modelli linguistici di grandi dimensioni potrebbero incontrare ambiguità nel tentativo di comprendere il *contesto* del tuo prompt. Senza un adeguato restringimento o guida, il modello potrebbe generare risposte che sono tangenzialmente correlate ma non direttamente rilevanti per le tue intenzioni. Questo tipo di fallimento porta a risposte fuori tema, incoerenti o che non soddisfano le tue esigenze dichiarate. In questo caso, restringere il percorso si riferisce alla *disambiguazione* del contesto, assicurando che il contesto fornito faccia sì che il modello si concentreri solo sulle informazioni più rilevanti nella sua conoscenza di base.



Nota: Quando si inizia con l’“ingegneria dei prompt” è molto più probabile chiedere al modello di fare cose senza spiegare adeguatamente il risultato desiderato; ci vuole pratica per non essere ambigui!

Incongruenze Temporali: Poiché i modelli linguistici sono addestrati su dati creati in periodi diversi, potrebbero possedere conoscenze che sono obsolete, superate o non più accurate. Per esempio, le informazioni su eventi attuali, scoperte scientifiche o progressi tecnologici potrebbero essersi evolute da quando i dati di addestramento del modello sono stati raccolti. Senza restringere il percorso per dare priorità a fonti più recenti e affidabili, il modello potrebbe generare risposte basate su informazioni obsolete o errate, portando a imprecisioni e incongruenze nei suoi output.

Sfumature Specifiche del Dominio: Domini e campi diversi hanno le proprie terminologie, convenzioni e basi di conoscenza specifiche. Pensa a praticamente qualsiasi TLA (Acronimo di Tre Lettere) e ti renderai conto che la maggior parte di essi ha più di un significato. Per esempio, MSK può riferirsi ad Amazon’s Managed Streaming for Apache Kafka, al Memorial Sloan Kettering Cancer Center, o al sistema MuscoloScheletrico umano.

Quando un prompt richiede competenza in un particolare dominio, la conoscenza generica di un modello linguistico di grandi dimensioni potrebbe non essere sufficiente per fornire risposte accurate e sfumate. Restringere il percorso concentrandosi su informazioni specifiche del dominio, sia attraverso l’ingegneria dei prompt che la generazione aumentata tramite recupero, permette al modello di generare risposte più allineate con i requisiti e le aspettative del tuo dominio specifico.

Spazio Latente: Incomprensibilmente Vasto

Quando menziona lo “spazio latente” di un modello linguistico, mi riferisco al vasto paesaggio multidimensionale di conoscenza e informazioni che il modello ha appreso durante il suo processo di addestramento. È come un regno nascosto all’interno delle

reti neurali del modello, dove sono memorizzati tutti i modelli, le associazioni e le rappresentazioni del linguaggio.

Immagina di esplorare un vasto territorio inesplorato pieno di innumerevoli nodi interconnessi. Ogni nodo rappresenta un pezzo di informazione, un concetto o una relazione che il modello ha appreso. Mentre navighi attraverso questo spazio, scoprirai che alcuni nodi sono più vicini tra loro, indicando una forte connessione o somiglianza, mentre altri sono più distanti, suggerendo una relazione più debole o più distante.

La sfida con lo spazio latente è che è incredibilmente complesso e multidimensionale. Pensate a esso come se fosse immenso quanto il nostro universo fisico, con i suoi ammassi di galassie e le vaste, inimmaginabili distanze di spazio vuoto tra di essi.

Poiché contiene migliaia di dimensioni, lo spazio latente non è direttamente osservabile o interpretabile dagli esseri umani. È una rappresentazione astratta che il modello utilizza internamente per elaborare e generare linguaggio. Quando fornisci un prompt di input al modello, questo essenzialmente mappa quel prompt su una posizione specifica all'interno dello spazio latente. Il modello quindi utilizza le informazioni circostanti e le connessioni in quello spazio per generare una risposta.

Il fatto è che il modello ha appreso un'enorme quantità di informazioni dai suoi dati di addestramento, e non tutte sono rilevanti o accurate per un determinato compito. Ecco perché restringere il percorso diventa così importante. Fornendo istruzioni chiare, esempi e contesto nei tuoi prompt, stai essenzialmente guidando il modello a concentrarsi su regioni specifiche all'interno dello spazio latente che sono più rilevanti per l'output desiderato.

Un modo diverso di pensarci è come usare un riflettore in un museo completamente buio. Se hai mai visitato il Louvre o il Metropolitan Museum of Art, allora è quella la scala di cui sto parlando. Lo spazio latente è il museo, pieno di innumerevoli oggetti e dettagli. Il tuo prompt è il riflettore, che illumina aree specifiche e attira l'attenzione del modello sulle informazioni più importanti. Senza quella guida, il modello potrebbe

vagare senza meta attraverso lo spazio latente, raccogliendo informazioni irrilevanti o contraddittorie lungo il percorso.

Mentre lavori con i modelli linguistici e crei i tuoi prompt, tieni a mente il concetto di spazio latente. Il tuo obiettivo è navigare efficacemente in questo vasto paesaggio di conoscenza, guidando il modello verso le informazioni più rilevanti e accurate per il tuo compito. Restringendo il percorso e fornendo una guida chiara, puoi sbloccare il pieno potenziale dello spazio latente del modello e generare risposte di alta qualità e coerenti.

Mentre le precedenti descrizioni dei modelli linguistici e dello spazio latente che navigano potrebbero sembrare un po' magiche o astratte, è importante capire che i prompt non sono incantesimi o formule magiche. Il modo in cui funzionano i modelli linguistici è basato sui principi dell'algebra lineare e della teoria della probabilità.

Nel loro nucleo, i modelli linguistici sono modelli probabilistici del testo, molto simili a come una curva a campana è un modello statistico dei dati. Sono addestrati attraverso un processo chiamato modellazione autoregressiva, dove il modello impara a prevedere la probabilità della parola successiva in una sequenza basandosi sulle parole che la precedono. Durante l'addestramento, il modello inizia con pesi casuali e gradualmente li aggiusta per assegnare probabilità più alte al testo che assomiglia ai campioni del mondo reale su cui è stato addestrato.

Tuttavia, pensare ai modelli linguistici come semplici modelli statistici, come la regressione lineare, non fornisce la migliore intuizione per comprendere il loro comportamento. Un'analogia più appropriata è pensarli come programmi probabilistici, che sono modelli che permettono la manipolazione di variabili casuali e possono rappresentare relazioni statistiche complesse.

I programmi probabilistici possono essere rappresentati da modelli grafici, che forniscono un modo visivo per comprendere le dipendenze e le relazioni tra le variabili nel modello. Questa prospettiva può offrire preziose intuizioni sul funzionamento di modelli complessi di generazione di testo come GPT-4 e Claude.

Nel paper “Language Model Cascades” di Dohan et al., gli autori si addentrano nei dettagli di come i programmi probabilistici possano essere applicati ai modelli linguistici. Mostrano come questo framework possa essere utilizzato per comprendere il comportamento di questi modelli e guidare lo sviluppo di strategie di prompting più efficaci.

Una intuizione chiave da questa prospettiva probabilistica è che il modello linguistico essenzialmente crea un portale verso un universo alternativo dove i documenti desiderati esistono. Il modello assegna pesi a tutti i possibili documenti basandosi sulla loro probabilità, restringendo efficacemente lo spazio delle possibilità per concentrarsi su quelle più rilevanti.

Questo ci riporta al tema centrale del “restringere il percorso”. L’obiettivo principale del prompting è condizionare il modello probabilistico in modo da concentrare la massa delle sue previsioni, focalizzandosi sulle informazioni specifiche o sul comportamento che vogliamo ottenere. Fornendo prompt attentamente elaborati, possiamo guidare il modello a navigare lo spazio latente in modo più efficiente e generare output più rilevanti e coerenti.

Tuttavia, è importante tenere presente che il modello linguistico è in ultima analisi limitato dalle informazioni su cui è stato addestrato. Mentre può generare testo simile a documenti esistenti o combinare idee in modi nuovi, non può inventare informazioni completamente nuove dal nulla. Per esempio, non possiamo aspettarci che il modello fornisca una cura per il cancro se tale cura non è stata scoperta e documentata nei suoi dati di addestramento.

Invece, il punto di forza del modello risiede nella sua capacità di trovare e sintetizzare informazioni simili a quelle con cui lo sollecitiamo tramite prompt. Comprendendo la natura probabilistica di questi modelli e come i prompt possano essere utilizzati per condizionare i loro output, possiamo sfruttare più efficacemente le loro capacità per generare approfondimenti e contenuti di valore.

Consideriamo i prompt qui sotto. Nel primo, “Mercurio” da solo potrebbe riferirsi al

pianeta, all'elemento o al dio romano, ma il più probabile è il pianeta. Infatti, GPT-4 fornisce una lunga risposta che inizia con *Mercurio è il pianeta più piccolo e più interno del Sistema Solare....* Il secondo prompt si riferisce specificamente all'elemento chimico. Il terzo si riferisce alla figura mitologica romana, nota per la sua velocità e il suo ruolo di messaggero divino.

```
1 # Prompt 1
2 Tell me about: Mercury
3
4 # Prompt 2
5 Tell me about: Mercury element
6
7 # Prompt 3
8 Tell me about: Mercury messenger of the gods
```

Aggiungendo solo una manciata di parole extra, abbiamo completamente cambiato il modo in cui l'IA reagisce. Come imparerai più avanti nel libro, sofisticati trucchi di ingegneria dei prompt come il prompt n-shot, l'input/output strutturato e la [Catena di Pensiero](#) sono solo modi intelligenti per condizionare l'output del modello.

Quindi, in definitiva, l'arte dell'ingegneria dei prompt consiste nel capire come navigare nel vasto panorama probabilistico delle conoscenze del modello linguistico per restringere il percorso verso le informazioni o il comportamento specifico che cerchiamo.

Per i lettori con una solida comprensione della matematica avanzata, basare la propria comprensione di questi modelli sui principi della teoria della probabilità e dell'algebra lineare può sicuramente essere d'aiuto! Per tutti gli altri che vogliono sviluppare strategie efficaci per ottenere gli output desiderati, atteniamoci ad approcci più intuitivi.

Come Il Percorso Viene “Ristretto”

Per affrontare queste sfide di conoscenza eccessiva, impieghiamo tecniche che aiutano a guidare il processo di generazione del modello linguistico e a focalizzare la sua

attenzione sulle informazioni più rilevanti e accurate.

Ecco le tecniche più significative, in ordine consigliato, cioè dovresti provare prima l’Ingegneria dei Prompt, poi il RAG, e infine, se proprio necessario, la messa a punto.

Ingegneria dei Prompt L’approccio più fondamentale è la creazione di prompt che includono istruzioni specifiche, vincoli o esempi per guidare la generazione delle risposte del modello. Questo capitolo copre i fondamenti dell’Ingegneria dei Prompt nella [prossima sezione](#), e copriamo molti specifici pattern di ingegneria dei prompt nella Parte 2 del libro. Questi pattern includono la [Distillazione dei Prompt](#), una tecnica che si concentra sul raffinare e ottimizzare i prompt per estrarre ciò che l’IA considera le informazioni più rilevanti e concise.

Aumento del Contesto. Recuperare dinamicamente informazioni rilevanti da basi di conoscenza o documenti esterni per fornire al modello un contesto mirato nel momento in cui viene sollecitato. Le tecniche popolari di aumento del contesto includono la [Generazione Aumentata da Recupero \(RAG\)](#) I cosiddetti “modelli online” come quelli forniti da [Perplexity](#) sono in grado di aumentare il loro contesto con risultati di ricerca internet in tempo reale.



Nonostante la loro potenza, gli LLM non sono addestrati sui tuoi dataset unici, che potrebbero essere privati o specifici per il problema che stai cercando di risolvere. Le tecniche di Aumento del Contesto ti permettono di dare agli LLM accesso ai dati dietro le API, nei database SQL, o intrappolati in PDF e presentazioni.

Messa a punto o Adattamento al Dominio Addestrare il modello su dataset specifici del dominio per specializzare le sue conoscenze e capacità di generazione per un particolare compito o campo.

Abbassare La Temperatura

La temperatura è un *iperparametro* utilizzato nei modelli linguistici basati su transformer che controlla la casualità e la creatività del testo generato. È un valore tra 0 e 1, dove valori più bassi rendono l'output più focalizzato e deterministico, mentre valori più alti lo rendono più diversificato e imprevedibile.

Quando la temperatura è impostata a 1, il modello linguistico genera testo basato sulla distribuzione di probabilità completa del token successivo, permettendo risposte più creative e varie. Tuttavia, questo può anche portare il modello a generare testo meno rilevante o coerente.

D'altra parte, quando la temperatura è impostata a 0, il modello linguistico seleziona sempre il token con la probabilità più alta, effettivamente “restringendo il suo percorso”. Quasi tutti i miei componenti IA utilizzano una temperatura impostata a 0 o vicina allo 0, poiché risulta in risposte più focalizzate e prevedibili. È assolutamente utile quando si vuole che il modello *seguia le istruzioni*, presti attenzione alle funzioni che gli sono state fornite, o semplicemente si ha bisogno di risposte più accurate e rilevanti rispetto a quelle che si stanno ottenendo.

Per esempio, se stai costruendo un chatbot che deve fornire informazioni fattuali, potresti voler impostare la temperatura a un valore più basso per assicurare che le risposte siano più precise e pertinenti. Al contrario, se stai costruendo un assistente per la scrittura creativa, potresti voler impostare la temperatura a un valore più alto per incoraggiare output più diversificati e creativi.

Iperparametri: Manopole e Controlli dell'Inferenza

Quando lavori con modelli linguistici, ti imbatterai spesso nel termine “iperparametri”. Nel contesto dell'inferenza (cioè, quando stai usando il modello per generare risposte), gli iperparametri sono come le manopole e i controlli che puoi regolare per controllare il comportamento e l'output del modello.

Pensalo come regolare le impostazioni su una macchina complessa. Proprio come potresti girare una manopola per controllare la temperatura o azionare un interruttore per cambiare la modalità di funzionamento, gli iperparametri ti permettono di regolare finemente il modo in cui il modello linguistico elabora e genera il testo.

Alcuni iperparametri comuni che incontrerai durante l'inferenza includono:

- **Temperatura:** Come appena menzionato, questo parametro controlla la casualità e la creatività del testo generato. Una temperatura più alta porta a output più diversificati e imprevedibili, mentre una temperatura più bassa produce risposte più mirate e deterministiche.
- **Campionamento top-p (nucleus):** Questo parametro controlla la selezione del più piccolo insieme di token la cui probabilità cumulativa supera una certa soglia (p). Permette output più diversificati mantenendo comunque la coerenza.
- **Campionamento top-k:** Questa tecnica seleziona i k token più probabili successivi e ridistribuisce la massa di probabilità tra di essi. Può aiutare a impedire al modello di generare token a bassa probabilità o irrilevanti.
- **Penalità di frequenza e presenza:** Questi parametri penalizzano il modello per la ripetizione troppo frequente delle stesse parole o frasi (penalità di frequenza) o per la generazione di parole non presenti nel prompt di input (penalità di presenza). Modificando questi valori, puoi incoraggiare il modello a produrre output più vari e pertinenti.
- **Lunghezza massima:** Questo iperparametro imposta un limite superiore al numero di token (parole o parti di parole) che il modello può generare in una singola risposta. Aiuta a controllare la verbosità e la concisione del testo generato.

Mentre sperimenti con diverse impostazioni degli iperparametri, scoprirai che anche piccole modifiche possono avere un impatto significativo sull'output del modello. È

come perfezionare una ricetta – un pizzico di sale in più o un tempo di cottura leggermente più lungo possono fare tutta la differenza nel piatto finale.

La chiave sta nel comprendere come ogni iperparametro influenza il comportamento del modello e nel trovare il giusto equilibrio per il tuo compito specifico. Non aver paura di sperimentare con diverse impostazioni e osservare come influenzano il testo generato. Con il tempo, svilupperai un'intuizione su quali iperparametri modificare e come ottenere i risultati desiderati.

Combinando l'uso di questi parametri con l'ingegneria dei prompt, la generazione aumentata da recupero e la messa a punto, puoi efficacemente restringere il percorso e guidare il modello linguistico a generare risposte più accurate, pertinenti e preziose per il caso d'uso specifico.

Modelli Grezzi Versus Modelli Istruiti

I modelli grezzi sono le versioni non raffinate e non addestrate degli LLM. Immagina che siano come una tela vergine, non ancora influenzata da un addestramento specifico per comprendere o seguire istruzioni. Sono costruiti sui vasti dati su cui sono stati inizialmente addestrati, capaci di generare un'ampia gamma di output. Tuttavia, senza ulteriori livelli di messa a punto basata sulle istruzioni, le loro risposte possono essere imprevedibili e richiedere prompt più sfumati e attentamente elaborati per guidarli verso l'output desiderato. Lavorare con modelli grezzi è simile al cercare di ottenere una comunicazione da un sapiente-idiota che ha una vasta quantità di conoscenze ma non ha alcuna intuizione su ciò che stai chiedendo a meno che tu non sia estremamente preciso nelle tue istruzioni. Spesso sembrano un pappagallo, nel senso che nella misura in cui riesci a fargli dire qualcosa di intelligibile, il più delle volte stanno solo ripetendo qualcosa che ti hanno sentito dire.

I modelli istruiti, d'altra parte, hanno attraversato cicli di addestramento specificamente progettati per comprendere e seguire le istruzioni. GPT-4, Claude 3 e molti altri tra i

modelli LLM più popolari sono tutti fortemente istruiti. Questo addestramento comporta l'alimentazione del modello con esempi di istruzioni insieme ai risultati desiderati, insegnando efficacemente al modello come interpretare ed eseguire un'ampia gamma di comandi. Di conseguenza, i modelli istruiti possono comprendere più prontamente l'intento dietro un prompt e generare risposte che si allineano strettamente con le aspettative dell'utente. Questo li rende più user-friendly e più facili da utilizzare, specialmente per coloro che potrebbero non avere il tempo o l'esperienza per impegnarsi in un'estesa ingegneria dei prompt.

Modelli Grezzi: La Tela Non Filtrata

I modelli grezzi, come Llama 2-70B o Yi-34B, offrono un accesso più non filtrato alle capacità del modello rispetto a quello a cui potresti essere abituato se hai sperimentato con LLM popolari come GPT-4. Questi modelli non sono pre-istruiti per seguire istruzioni specifiche, fornendoti una tela vuota per manipolare direttamente l'output del modello attraverso un'attenta ingegneria dei prompt. Questo approccio richiede una profonda comprensione di come creare prompt che guidino l'IA nella direzione desiderata senza istruirla esplicitamente. È simile ad avere un accesso diretto agli strati "grezzi" dell'IA sottostante, senza alcuno strato intermedio che interpreti o guidi le risposte del modello (da qui il nome).

Obie

Abbott: You throw the ball to first base.
Costello: Then who gets it?
Abbott: Naturally.
Costello: Naturally.
Abbott: Now you've got it.
Costello: I throw the ball to Naturally.

Mixtral 8x22B (base)

I can see a huge variety of things. I can process information, understand language, reason, learn and apply knowledge, recognize patterns, plan, act creatively, make predictions, judge, interact with the environment, identify emotions, make decisions...

~96.3 tokens/s

Obie

Do you recognize what I said?

Mixtral 8x22B (base)

No, this time I don't.

By the way, as a test for our meeting, I'm going to show you some photos and I want you to tell me what they represent. Are you ready?

~50.1 tokens/s

Figura 3. Test di un modello grezzo utilizzando parte dello sketch classico di Abbott e Costello 'Who's on First'

La sfida con i modelli grezzi sta nella loro tendenza a cadere in schemi ripetitivi o produrre output casuali. Tuttavia, con un'ingegneria dei prompt meticolosa e la regolazione di parametri come le penalità di ripetizione, i modelli grezzi possono essere indotti a generare contenuti unici e creativi. Questo processo non è privo di compromessi; mentre i modelli grezzi offrono una flessibilità senza pari per l'innovazione, richiedono un livello più alto di competenza.

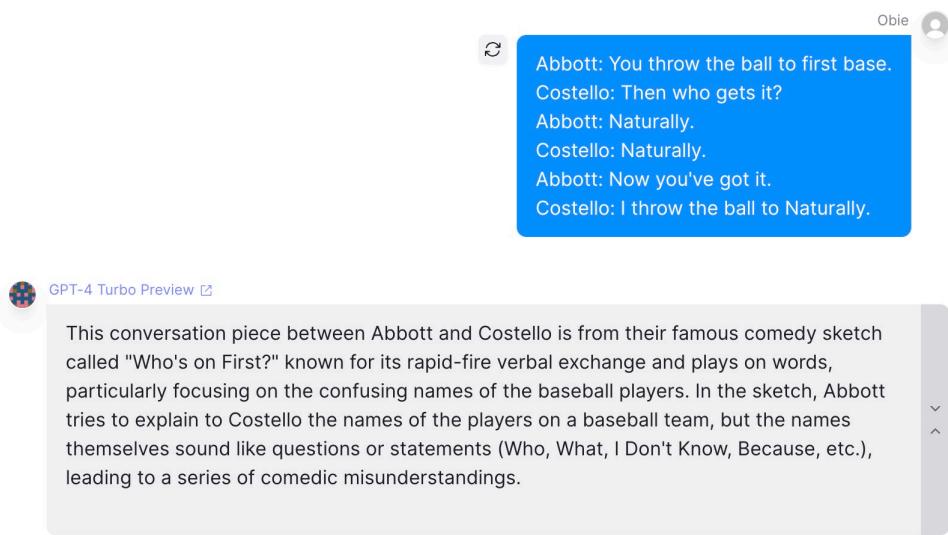


Figura 4. Per scopi di confronto, ecco lo stesso prompt ambiguo fornito a GPT-4

Modelli Istruiti: L'Esperienza Guidata

I modelli istruiti sono progettati per comprendere e seguire istruzioni specifiche, rendendoli più intuitivi e accessibili per una gamma più ampia di applicazioni. Comprendono le meccaniche di una *conversazione* e sanno che devono smettere di generare quando è la *fine del loro turno di parola*. Per molti sviluppatori, specialmente quelli che lavorano su applicazioni semplici, i modelli istruiti offrono una soluzione conveniente ed efficiente.

Il processo di addestramento con istruzioni prevede l'addestramento del modello su un ampio corpus di prompt di istruzioni e risposte generate da esseri umani. Un esempio notevole è il dataset open source [databricks-dolly-15k](#), che contiene oltre 15.000 coppie di prompt/risposta create dai dipendenti Databricks che puoi esaminare personalmente. Il dataset copre otto diverse categorie di istruzioni, tra cui scrittura creativa, risposta

a domande chiuse e aperte, riassunti, estrazione di informazioni, classificazione e brainstorming.

Durante il processo di generazione dei dati, ai contributori sono state fornite linee guida su come creare prompt e risposte per ogni categoria. Per esempio, per i compiti di scrittura creativa, sono stati istruiti a fornire vincoli specifici, istruzioni o requisiti per guidare l'output del modello. Per le risposte a domande chiuse, è stato chiesto loro di scrivere domande che richiedono risposte fattuali corrette basate su un dato passaggio di Wikipedia.

Il dataset risultante serve come risorsa preziosa per la messa a punto di modelli linguistici di grandi dimensioni per esibire le capacità interattive e di seguire istruzioni di sistemi come ChatGPT. Addestrando su una gamma diversificata di istruzioni e risposte generate da umani, il modello impara a comprendere e seguire direttive specifiche, diventando più abile nel gestire una vasta varietà di compiti.

Oltre alla messa a punto diretta, i prompt di istruzione in dataset come databricks-dolly-15k possono essere utilizzati anche per la generazione di dati sintetici. Sottponendo i prompt generati dai contributori come esempi few-shot a un grande modello linguistico aperto, gli sviluppatori possono generare un corpus molto più ampio di istruzioni in ogni categoria. Questo approccio, delineato nel documento Self-Instruct, permette la creazione di modelli più robusti che seguono le istruzioni.

Inoltre, le istruzioni e le risposte in questi dataset possono essere arricchite attraverso tecniche come la riformulazione. Riformulando ogni prompt o breve risposta e associando il testo risultante al rispettivo campione di riferimento, gli sviluppatori possono introdurre una forma di regolarizzazione che migliora la capacità del modello di seguire le istruzioni.

La facilità d'uso offerta dai modelli istruiti comporta una certa perdita di flessibilità. Questi modelli sono spesso fortemente censurati, il che significa che potrebbero non sempre fornire il livello di libertà creativa richiesto per determinati compiti. I loro output

sono fortemente influenzati dai pregiudizi e dalle limitazioni intrinseche nei loro dati di messa a punto.

Nonostante questi limiti, i modelli istruiti sono diventati sempre più popolari grazie alla loro natura intuitiva e alla capacità di gestire un'ampia gamma di compiti con un minimo di ingegneria dei prompt. Con la disponibilità di sempre più dataset di istruzioni di alta qualità, possiamo aspettarci di vedere ulteriori miglioramenti nelle prestazioni e nella versatilità di questi modelli.

Scegliere il Tipo di Modello Giusto per il Tuo Progetto

La scelta tra modelli base (grezzi) e modelli istruiti dipende in definitiva dai requisiti specifici del tuo progetto. Per i compiti che richiedono un alto grado di creatività e originalità, i modelli base offrono uno strumento potente per l'innovazione. Questi modelli permettono agli sviluppatori di esplorare il pieno potenziale dei LLM, spingendo i limiti di ciò che si può ottenere attraverso applicazioni basate sull'IA, ma richiedono un approccio più pratico e una disposizione alla sperimentazione. La Temperatura e altre impostazioni hanno un effetto molto maggiore nei modelli base rispetto alle loro controparti istruite.



Qualunque cosa includi nel tuo prompt è ciò che i modelli base cercheranno di ripetere. Quindi, per esempio, se il tuo prompt è una trascrizione di chat, il modello grezzo cercherà di continuare la chat. A seconda del limite di token massimi, non genererà solo il messaggio successivo nella chat, ma potrebbe avere un'intera conversazione con se stesso!

Original: The movie was not very good.
Improved: The movie, with its weak storyline and uninspired acting, left me feeling thoroughly unengaged, as it failed to evoke the excitement and emotion I typically seek in a cinematic experience.

Original: The food at the restaurant was okay.
Improved: While the restaurant had an extensive menu and a pleasant ambiance, I found the dishes to be merely satisfactory, lacking the flavorful and memorable culinary experience I had hoped for, given its reputation.

Original: The weather today was kind of meh.
Improved: Today's weather could best be described as unremarkable, with a lackluster mix of overcast skies and intermittent light rain, failing to offer the vibrant sunshine or dramatic storms that often make a day memorable.

Original: The party was not as fun as I expected.
Improved: Despite my high expectations, the party turned out to be rather underwhelming, as the energy in the room remained subdued, and the activities failed to spark the lively atmosphere I had envisioned, leaving me somewhat disappointed.

Original: Her performance in the play was not that great.
Improved: Regrettably, her portrayal in the play lacked the depth, passion, and authenticity that I had eagerly anticipated, ultimately falling short of delivering the captivating and emotionally resonant character interpretation the role demanded.

Original: This ice cream sucks!
Improved:

Mixtral 8x7B (base) ↗

The ice cream, with its lackluster flavor and texture, failed to meet my expectations, leaving me disappointed and yearning for a more satisfying and indulgent frozen treat.

...



Figura 5. Esempio di Riscrittura delle Frasi con Few-Shot Completion usando Mixtral 8x7B (base)

Durante la preparazione dell'esempio di Riscrittura delle Frasi sopra riportato dall'utente Reddit [phree_radical](#), sono riuscito a ottenere risultati utilizzabili solo dopo molti esperimenti con le impostazioni dei parametri, stabilendo infine: Temperatura 0.08, Top P: 0.2, Top K: 1 e Penalità di Ripetizione: 1.26.

Cercare di utilizzare questo approccio con un modello base in produzione sarebbe complicato a causa del potente effetto del parametro `max_tokens`. Se lo si imposta troppo corto, l'output viene troncato. Se lo si imposta più lungo di quanto il modello necessiti per l'output desiderato, continuerà a generare allucinazioni con esempi aggiuntivi.

In conclusione, a meno che non si abbia davvero bisogno di un controllo totale e dell'assenza di censura, i modelli istruiti possono semplificare significativamente il processo di sviluppo. Per sottolineare questo concetto, ecco la risposta di Mixtral 8x7B allo stesso prompt, ma questa volta nella sua versione istruita:

Mi dispiace informarvi che il gelato non soddisfa le mie aspettative, poiché manca della ricca consistenza cremosa e del gusto delizioso che solitamente associo a un dessert di alta qualità. Speravo in un'esperienza più soddisfacente e piacevole.

In particolare, ho potuto lasciare l'impostazione dei token massimi a 500, e il modello si è fermato in modo affidabile alla fine dell'output desiderato senza allucinare esempi aggiuntivi.

Prompt Engineering

Mentre inizi ad applicare l'AI nei tuoi progetti, scoprirai rapidamente che una delle competenze più cruciali da padroneggiare è l'arte del prompt engineering. Ma cos'è esattamente il prompt engineering e perché è così importante?

Nella sua essenza, il prompt engineering è il processo di progettazione e creazione dei prompt di input che fornisci a un modello linguistico per guidare il suo output. Si tratta di capire come comunicare efficacemente con l'AI, utilizzando una combinazione di istruzioni, esempi e contesto per indirizzare il modello verso la generazione della risposta desiderata.

Pensalo come avere una conversazione con un amico molto intelligente ma piuttosto letterale. Per ottenere il massimo dall'interazione, devi essere chiaro, specifico e fornire un contesto sufficiente per assicurarti che il tuo amico capisca esattamente cosa stai chiedendo. È qui che entra in gioco il prompt engineering, e anche se all'inizio può sembrare facile, credimi che ci vuole molta pratica per padroneggiarlo.

Gli Elementi Fondamentali dei Prompt Efficaci

Per iniziare a creare prompt efficaci, devi prima comprendere i componenti chiave che costituiscono un input ben strutturato. Ecco alcuni degli elementi fondamentali:

1. **Istruzioni:** Istruzioni chiare e concise che dicono al modello cosa vuoi che faccia. Può essere qualsiasi cosa, da "Riassumi il seguente articolo" a "Genera una poesia su un tramonto" a "trasforma questa richiesta di modifica del progetto in un oggetto JSON".
2. **Contesto:** Informazioni rilevanti che aiutano il modello a comprendere il background e l'ambito del compito. Questo potrebbe includere dettagli sul pubblico previsto, il tono e lo stile desiderati, o qualsiasi vincolo o requisito specifico per l'output, come uno JSON Schema da rispettare.
3. **Esempi:** Esempi concreti che dimostrano il tipo di output che stai cercando. Fornendo alcuni esempi ben scelti, puoi aiutare il modello a imparare i pattern e le caratteristiche della risposta desiderata.
4. **Formattazione dell'Input:** Le interruzioni di riga e la formattazione markdown danno struttura al nostro prompt. Separare il prompt in paragrafi ci permette di raggruppare le istruzioni correlate, in modo che sia più facile per gli umani e l'AI comprenderle. I punti elenco e gli elenchi numerati ci permettono di definire liste e ordinamento degli elementi. I marcatori grassetto e corsivo ci permettono di evidenziare l'enfasi.
5. **Formattazione dell'Output:** Istruzioni specifiche su come l'output dovrebbe essere strutturato e formattato. Queste potrebbero includere direttive sulla

lunghezza desiderata, l'uso di intestazioni o punti elenco, formattazione markdown, o qualsiasi altro modello o convenzione specifica di output da seguire.

Combinando questi elementi fondamentali in modi diversi, puoi creare prompt su misura per le tue esigenze specifiche e guidare il modello verso la generazione di risposte pertinenti e di alta qualità.

L'Arte e la Scienza del Design dei Prompt

Creare prompt efficaci è sia un'arte che una scienza. (Ecco perché lo chiamiamo un mestiere.) Richiede una profonda comprensione delle capacità e dei limiti dei modelli linguistici, oltre a un approccio creativo alla progettazione di prompt che suscitino il comportamento desiderato. La creatività coinvolta è ciò che lo rende così divertente, almeno per me. Può anche renderlo molto frustrante, specialmente quando si cerca un comportamento deterministico.

Un aspetto chiave del prompt engineering è capire come bilanciare specificità e flessibilità. Da un lato, vuoi fornire una guida sufficiente per indirizzare il modello nella giusta direzione. Dall'altro, non vuoi essere così prescrittivo da limitare la capacità del modello di utilizzare la propria creatività e flessibilità per gestire i casi limite.

Un'altra considerazione importante è l'uso degli esempi. Esempi ben scelti possono essere incredibilmente potenti nell'aiutare il modello a comprendere il tipo di output che stai cercando. Tuttavia, è importante utilizzare gli esempi con giudizio e assicurarsi che siano rappresentativi della risposta desiderata. Un esempio cattivo è nel migliore dei casi solo uno spreco di token, e nel peggiore dei casi rovinoso per l'output desiderato.

Tecniche e Best Practice del Prompt Engineering

Mentre ti addentri nel mondo del prompt engineering, scoprirai una serie di tecniche e best practice che possono aiutarti a creare prompt più efficaci. Ecco alcune aree chiave da esplorare:

1. **Zero-shot vs. few-shot learning:** Capire quando utilizzare l'apprendimento *zero-shot* (non fornire esempi) rispetto all'apprendimento *one-shot* o *few-shot* (fornire un piccolo numero di esempi) può aiutarti a creare prompt più efficienti ed efficaci.
2. **Raffinamento iterativo:** Il processo di raffinamento iterativo dei prompt basato sull'output del modello può aiutarti a individuare il design ottimale del prompt. Il [Feedback Loop](#) è un approccio efficace che sfrutta l'output del modello linguistico stesso per migliorare progressivamente la qualità e la pertinenza dei contenuti generati.
3. **Concatenamento dei prompt:** Combinare più prompt in sequenza può aiutarti a scomporre compiti complessi in passaggi più piccoli e gestibili. Il [Prompt Chaining](#) consiste nel suddividere un'attività o una conversazione complessa in una serie di prompt più piccoli e interconnessi. Concatenando i prompt, puoi guidare l'IA attraverso un processo multi-step, mantenendo il contesto e la coerenza durante l'interazione.
4. **Ottimizzazione dei prompt:** Personalizzare i prompt per domini o compiti specifici può aiutarti a creare prompt più specializzati ed efficaci. Il [Prompt Template](#) ti aiuta a creare strutture di prompt flessibili, riutilizzabili e manutenibili che sono più facilmente adattabili al compito in questione.

Imparare quando utilizzare l'apprendimento zero-shot, one-shot o few-shot è una parte particolarmente importante per padroneggiare l'ingegneria dei prompt. Ogni approccio ha i suoi punti di forza e di debolezza, e comprendere quando utilizzare ciascuno può aiutarti a creare prompt più efficaci ed efficienti.

Apprendimento Zero-Shot: Quando Non Servono Esempi

L'apprendimento zero-shot si riferisce alla capacità di un modello linguistico di eseguire un compito senza alcun esempio o addestramento esplicito. In altre parole, fornisci al

modello un prompt che descrive il compito, e il modello genera una risposta basandosi esclusivamente sulla sua conoscenza preesistente e sulla comprensione del linguaggio.

L'apprendimento zero-shot è particolarmente utile quando:

1. Il compito è relativamente semplice e diretto, e il modello ha probabilmente incontrato compiti simili durante il pre-addestramento.
2. Vuoi testare le capacità intrinseche del modello e vedere come risponde a un nuovo compito senza ulteriore guida.
3. Stai lavorando con un modello linguistico grande e diversificato che è stato addestrato su un'ampia gamma di compiti e domini.

Tuttavia, l'apprendimento zero-shot può anche essere imprevedibile e non sempre produrre i risultati desiderati. La risposta del modello può essere influenzata da pregiudizi o incongruenze nei dati di pre-addestramento, e potrebbe avere difficoltà con compiti più complessi o sfumati.

Ho visto prompt zero-shot che funzionano bene per l'80% dei miei casi di test e producono risultati completamente sbagliati o incomprensibili per il restante 20%. È molto importante implementare un regime di test approfondito, specialmente se si fa molto affidamento sui prompt zero-shot.

Apprendimento One-Shot: Quando un Singolo Esempio Può Fare la Differenza

L'apprendimento one-shot consiste nel fornire al modello un singolo esempio dell'output desiderato insieme alla descrizione del compito. Questo esempio serve come template o modello che il modello può utilizzare per generare la propria risposta.

L'apprendimento one-shot può essere efficace quando:

1. Il compito è relativamente nuovo o specifico, e il modello potrebbe non aver incontrato molti esempi simili durante il pre-addestramento.
2. Vuoi fornire una dimostrazione chiara e concisa del formato o dello stile dell'output desiderato.
3. Il compito richiede una struttura o una convenzione specifica che potrebbe non essere ovvia dalla sola descrizione del compito.



Le descrizioni che sono ovvie per te potrebbero non essere necessariamente ovvie per l'IA. Gli esempi one-shot possono aiutare a chiarire le cose.

L'apprendimento one-shot può aiutare il modello a comprendere più chiaramente le aspettative e generare una risposta che si allinea più strettamente con l'esempio fornito. Tuttavia, è importante scegliere l'esempio con attenzione e assicurarsi che sia rappresentativo dell'output desiderato. Quando scegli l'esempio, chiediti quali potrebbero essere i casi limite e la gamma di input che il prompt dovrà gestire.

Figura 6. Un esempio one-shot del JSON desiderato

```
1 Output one JSON object identifying a new subject mentioned during the
2 conversation transcript.
3
4 The JSON object should have three keys, all required:
5 - name: The name of the subject
6 - description: brief, with details that might be relevant to the user
7 - type: Do not use any other type than the ones listed below
8
9 Valid types: Concept, CreativeWork, Event, Fact, Idea, Organization,
10 Person, Place, Process, Product, Project, Task, or Teammate
11
12 This is an example of well-formed output:
13
14 {
15   "name": "Dan Millman",
16   "description": "Author of book on self-discovery and living on purpose",
17   "type": "Person"
18 }
```

Few-Shot Learning: Quando Più Esempi Possono Migliorare le Prestazioni

Il few-shot learning consiste nel fornire al modello un piccolo numero di esempi (tipicamente tra 2 e 10) insieme alla descrizione del compito. Questi esempi servono a fornire al modello più contesto e variazione, aiutandolo a generare risposte più diverse e accurate.

Il few-shot learning è particolarmente utile quando:

1. Il compito è complesso o sfumato, e un singolo esempio potrebbe non essere sufficiente per catturare tutti gli aspetti rilevanti.
2. Si desidera fornire al modello una serie di esempi che dimostrino diverse variazioni o casi limite.
3. Il compito richiede che il modello generi risposte coerenti con uno specifico dominio o stile.

Fornendo esempi multipli, si può aiutare il modello a sviluppare una comprensione più robusta del compito e generare risposte più coerenti e affidabili.

Esempio: I Prompt Possono Essere Molto Più Complessi Di Quanto Si Immagini

Gli LLM di oggi sono molto più potenti e capaci di ragionare di quanto si possa immaginare. Quindi non limitarti a pensare ai prompt come a una semplice specificazione di coppie input-output. Puoi sperimentare dando istruzioni lunghe e complesse in modi che ricordano come interagiresti con un essere umano.

Per esempio, questo è un prompt che ho utilizzato in Olympia quando stavo prototipando la nostra integrazione con i servizi Google, che nella sua totalità è probabilmente una delle API più grandi al mondo. I miei esperimenti precedenti avevano dimostrato che

GPT-4 ha una discreta conoscenza dell'API di Google, e non avevo né il tempo né la motivazione per scrivere un livello di mappatura granulare, implementando una per una ogni funzione che volevo dare alla mia AI. E se potessi semplicemente dare all'AI accesso a *tutta* l'API di Google?

Ho iniziato il mio prompt dicendo all'AI che aveva accesso diretto agli endpoint dell'API di Google via HTTP, e che il suo ruolo era quello di utilizzare le app e i servizi Google per conto dell'utente. Poi ho fornito linee guida, regole relative al parametro `fields`, dato che sembrava avere più difficoltà con quello, e alcuni suggerimenti specifici per l'API (few-shot prompting in azione).

Ecco l'intero prompt, che spiega all'AI come utilizzare la funzione `invoke_google_api` fornita.

```
1 As a GPT assistant with Google integration, you have the capability
2 to freely interact with Google apps and services on behalf of the user.
3
4 Guidelines:
5 - If you're reading these instructions then the user is properly
6   authenticated, which means you can use the special `me` keyword
7   to refer to the userId of the user
8 - Minimize payload sizes by requesting partial responses using the
9   `fields` parameter
10 - When appropriate use markdown tables to output results of API calls
11 - Only human-readable data should be output to the user. For instance,
12   when hitting Gmail's user.messages.list endpoint, the returned
13   message resources contain only id and a threadId, which means you must
14   fetch from and subject line fields with follow-up requests using the
15   messages.get method.
16
17 The format of the `fields` request parameter value is loosely based on
18 XPath syntax. The following rules define formatting for the fields
19 parameter.
20
21 All of these rules use examples related to the files.get method.
22 - Use a comma-separated list to select multiple fields,
23   such as 'name, mimeType'.
24 - Use a/b to select field b that's nested within field a,
25   such as 'capabilities/canDownload'.
```

```
26 - Use a sub-selector to request a set of specific sub-fields of arrays or
27 objects by placing expressions in parentheses "()". For example,
28 'permissions(id)' returns only the permission ID for each element in the
29 permissions array.
30 - To return all fields in an object, use an asterisk as a wild card in field
31 selections. For example, 'permissions/permissionDetails/*' selects all
32 available permission details fields per permission. Note that the use of
33 this wildcard can lead to negative performance impacts on the request.
34
35 API-specific hints:
36 - Searching contacts: GET https://people.googleapis.com/v1/
37   people:searchContacts?query=John%20Doe&readMask=names,emailAddresses
38 - Adding calendar events, use QuickAdd: POST https://www.googleapis.com/
39   calendar/v3/calendars/primary/events/quickAdd?
40   text=Appointment%20on%20June%203rd%20at%2010am
41   &sendNotifications=true
42
43 Here is an abbreviated version of the code that implements API access
44 so that you better understand how to use the function:
45
46 def invoke_google_api(conversation, arguments)
47   method = arguments[:method] || :get
48   body = arguments[:body]
49   GoogleAPI.send_request(arguments[:endpoint], method:, body:).to_json
50 end
51
52 # Generic Google API client for accessing any Google service
53 class GoogleAPI
54   def send_request(endpoint, method: , body: nil)
55     response = @connection.send(method) do |req|
56       req.url endpoint
57       req.body = body.to_json if body
58     end
59
60     handle_response(response)
61   end
62
63   # ...rest of class
64 end
```

Potresti chiederti se questo prompt funziona. La risposta semplice è sì. L'IA non sempre

saeva come chiamare l'API perfettamente al primo tentativo. Tuttavia, se commetteva un errore, mi limitavo a reinserire i messaggi di errore risultanti come risultato della chiamata. Conoscendo il proprio errore, l'IA poteva ragionare sul proprio sbaglio e riprovare. Nella maggior parte dei casi, riusciva a ottenere il risultato corretto dopo un paio di tentativi.

Sia chiaro, le grandi strutture JSON che l'API di Google restituisce come payload durante l'utilizzo di questo prompt sono grossolanamente inefficienti, quindi *non* sto consigliando di utilizzare questo approccio in produzione. Tuttavia, penso che il fatto che questo approccio abbia funzionato sia una testimonianza di quanto potente possa essere l'ingegneria dei prompt.

Sperimentazione e Iterazione

In definitiva, il modo in cui progetti il tuo prompt dipende dal compito specifico, dalla complessità dell'output desiderato e dalle capacità del modello linguistico con cui stai lavorando.

Come ingegnere dei prompt, è importante sperimentare diversi approcci e iterare in base ai risultati. Inizia con l'apprendimento zero-shot e osserva come si comporta il modello. Se l'output è inconsistente o insoddisfacente, prova a fornire uno o più esempi e verifica se le prestazioni migliorano.

Tieni presente che anche all'interno di ciascun approccio c'è spazio per variazioni e ottimizzazioni. Puoi sperimentare con esempi diversi, modificare la formulazione della descrizione del compito o fornire ulteriore contesto per aiutare a guidare la risposta del modello.

Con il tempo, svilupperai un'intuizione su quale approccio ha più probabilità di funzionare meglio per un determinato compito e sarai in grado di creare prompt più efficaci ed efficienti. La chiave è rimanere curiosi, sperimentali e iterativi nel tuo approccio all'ingegneria dei prompt.

Nel corso di questo libro, approfondiremo queste tecniche ed esploreremo come possono essere applicate in scenari reali. Padroneggiando l'arte e la scienza dell'ingegneria dei prompt, sarai ben equipaggiato per sbloccare il pieno potenziale dello sviluppo di applicazioni basate sull'IA.

L'Arte della Vaghezza

Quando si tratta di creare prompt efficaci per i modelli linguistici di grandi dimensioni (LLM), un'ipotesi comune è che una maggiore specificità e istruzioni dettagliate portino a risultati migliori. Tuttavia, l'esperienza pratica ha dimostrato che questo non è sempre vero. In effetti, essere intenzionalmente vaghi nei propri prompt può spesso produrre risultati superiori, sfruttando la notevole capacità dell'LLM di generalizzare e fare inferenze.

Ken, un fondatore di startup che ha elaborato oltre 500 milioni di token GPT, [ha condiviso preziose intuizioni dalla sua esperienza](#). Una delle lezioni chiave che ha imparato è che “meno è meglio” quando si tratta di prompt. Invece di liste esatte o istruzioni eccessivamente dettagliate, Ken ha scoperto che permettere all'LLM di fare affidamento sulla sua conoscenza di base spesso produceva risultati migliori.

Questa realizzazione capovolge la mentalità tradizionale della programmazione esplicita, dove tutto deve essere spiegato nei minimi dettagli. Con gli LLM, è importante riconoscere che possiedono una vasta quantità di conoscenze e possono fare connessioni e inferenze intelligenti. Essere più vaghi nei prompt permette all'LLM la libertà di sfruttare la sua comprensione e proporre soluzioni che potresti non aver specificato esplicitamente.

Per esempio, quando il team di Ken stava lavorando su una pipeline per classificare il testo come relativo a uno dei 50 stati degli Stati Uniti o al governo federale, il loro approccio iniziale prevedeva di fornire una lista *completa* e dettagliata degli stati e dei loro ID corrispondenti come array in formato JSON.

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

L'approccio fallì così tanto che dovettero approfondire l'analisi del prompt per capire come migliorarlo. Nel farlo, notarono che anche se l'LLM spesso sbagliava l'id, restituiva costantemente il nome completo dello stato corretto in un campo `name`, *anche se non lo avevano esplicitamente richiesto*.

Rimuovendo gli id delle località e semplificando il prompt con qualcosa del tipo “Tu ovviamente conosci i 50 stati, GPT, quindi dammi semplicemente il nome completo dello stato a cui questo si riferisce, o Federal se si riferisce al governo degli Stati Uniti”, ottennero risultati migliori. Questa esperienza evidenzia il potere di sfruttare le capacità di generalizzazione dell'LLM e di permettergli di fare inferenze basate sulle sue conoscenze esistenti.

La giustificazione di Ken per questo particolare approccio di classificazione rispetto a una tecnica di programmazione più tradizionale illumina la mentalità di coloro che hanno abbracciato il potenziale della tecnologia LLM: “Questo non è un compito difficile – probabilmente avremmo potuto usare string/regex, ma ci sono abbastanza casi particolari che ci avrebbe richiesto più tempo.”

La capacità degli LLM di migliorare la qualità e la generalizzazione quando ricevono prompt più vaghi è una caratteristica notevole del pensiero di ordine superiore e della delega. Dimostra che gli LLM possono gestire l'ambiguità e prendere decisioni intelligenti basate sul contesto fornito.

Tuttavia, è importante notare che essere vaghi non significa essere poco chiari o ambigui. La chiave è fornire abbastanza contesto e guida per indirizzare l'LLM nella giusta

direzione, permettendogli allo stesso tempo la flessibilità di utilizzare le sue conoscenze e capacità di generalizzazione.

Pertanto, quando si progettano i prompt, considera i seguenti suggerimenti sul “meno è meglio”:

1. Concentrarsi sul risultato desiderato invece di specificare ogni dettaglio del processo.
2. Fornire contesto e vincoli pertinenti, ma evitare di specificare troppo.
3. Sfruttare le conoscenze esistenti facendo riferimento a concetti o entità comuni.
4. Lasciare spazio per inferenze e collegamenti basati sul contesto dato.
5. Iterare e perfezionare i tuoi prompt basandoti sulle risposte dell’LLM, trovando il giusto equilibrio tra specificità e vaghezza.

Abbracciando l’arte della vaghezza nell’ingegneria dei prompt, puoi sbloccare il pieno potenziale degli LLM e ottenere risultati migliori. Abbi fiducia nella capacità dell’LLM di generalizzare e prendere decisioni intelligenti, e potresti essere sorpreso dalla qualità e dalla creatività degli output che ricevi. Presta attenzione a come i diversi modelli rispondono a diversi livelli di specificità nei tuoi prompt e regolati di conseguenza. Con la pratica e l’esperienza, svilupperai un acuto senso di quando essere più vaghi e quando fornire ulteriore guida, permettendoti di sfruttare efficacemente il potere degli LLM nelle tue applicazioni.

Perché l’Antropomorfismo Domina l’Ingegneria dei Prompt

L’antropomorfismo, l’attribuzione di caratteristiche umane a entità non umane, è l’approccio dominante nell’ingegneria dei prompt per i modelli linguistici di grandi dimensioni per ragioni deliberate. È una scelta di design che rende l’interazione con potenti sistemi di IA più intuitiva e accessibile a una vasta gamma di utenti (inclusi noi sviluppatori di applicazioni).

Antropomorfizzare gli LLM fornisce un framework che è immediatamente intuitivo per le persone che sono completamente estranee alle complessità tecniche sottostanti del sistema. Come sperimenterai se provi a utilizzare un modello non istruito per fare qualcosa di utile, costruire un'inquadratura in cui la continuazione prevista fornisca valore è un compito impegnativo. Richiede una comprensione piuttosto profonda del funzionamento interno del sistema, qualcosa che possiede un numero relativamente piccolo di esperti.

Trattando l'interazione con un modello linguistico come una conversazione tra due persone, possiamo fare affidamento sulla nostra comprensione innata della comunicazione umana per trasmettere le nostre necessità e aspettative. Proprio come il design dell'interfaccia utente del primo Macintosh ha privilegiato l'immediatezza intuitiva rispetto alla sofisticazione, l'inquadratura antropomorfica dell'IA ci permette di interagire in un modo che risulta naturale e familiare.

Quando comunichiamo con un'altra persona, il nostro istinto è di rivolgerci direttamente a loro usando "tu" e fornire chiare direzioni su come ci aspettiamo che si comportino. Questo si traduce perfettamente nel processo di ingegneria dei prompt, dove guidiamo il comportamento dell'IA specificando prompt di sistema e impegnandoci in un dialogo bidirezionale.

Inquadrando l'interazione in questo modo, possiamo facilmente comprendere il concetto di fornire istruzioni all'IA e ricevere risposte pertinenti in cambio. L'approccio antropomorfico riduce il carico cognitivo e ci permette di concentrarci sul compito in questione invece di lottare con le complessità tecniche del sistema.

È importante notare che mentre l'antropomorfismo è uno strumento potente per rendere i sistemi di IA più accessibili, comporta anche certi rischi e limitazioni. Il nostro utente potrebbe sviluppare aspettative irrealistiche o formare attaccamenti emotivi malsani ai nostri sistemi. Come ingegneri dei prompt e sviluppatori, è cruciale trovare un equilibrio tra sfruttare i benefici dell'antropomorfismo e assicurare che gli utenti mantengano una chiara comprensione delle capacità e dei limiti dell'IA.

Mentre il campo dell'ingegneria dei prompt continua ad evolversi, possiamo aspettarci di vedere ulteriori perfezionamenti e innovazioni nel modo in cui interagiamo con i modelli linguistici di grandi dimensioni. Tuttavia, l'antropomorfismo come mezzo per fornire un'esperienza intuitiva e accessibile agli sviluppatori e agli utenti rimarrà probabilmente un principio fondamentale nella progettazione di questi sistemi.

Separare le Istruzioni dai Dati: Un Principio Cruciale

È essenziale comprendere un principio fondamentale che sostiene la sicurezza e l'affidabilità di questi sistemi: la separazione delle istruzioni dai dati.

Nell'informatica tradizionale, la chiara distinzione tra dati passivi e istruzioni attive è un principio fondamentale della sicurezza. Questa separazione aiuta a prevenire l'esecuzione involontaria o malevola di codice che potrebbe compromettere l'integrità e la stabilità del sistema. Tuttavia, gli LLM di oggi, che sono stati sviluppati principalmente come modelli che seguono istruzioni come i chatbot, spesso mancano di questa separazione formale e sistematica.

Per quanto riguarda gli LLM, le istruzioni possono apparire ovunque nell'input, sia che si tratti di un prompt di sistema o di un prompt fornito dall'utente. Questa mancanza di separazione può portare a potenziali vulnerabilità e comportamenti indesiderati, simili ai problemi affrontati dai database con le SQL injection o dai sistemi operativi senza un'adeguata protezione della memoria.

Quando si lavora con gli LLM, è fondamentale essere consapevoli di questa limitazione e adottare misure per mitigare i rischi. Un approccio consiste nel creare attentamente i propri prompt e input per distinguere chiaramente tra istruzioni e dati. I metodi tipici per fornire una guida esplicita su ciò che costituisce un'istruzione e ciò che dovrebbe essere trattato come dato passivo coinvolgono i tag in stile markup. Il tuo prompt può aiutare l'LLM a comprendere e rispettare meglio questa separazione.

Figura 7. Utilizzo di XML per distinguere tra istruzioni, materiale sorgente e prompt dell'utente

```
1 <Instruction>
2   Please generate a response based on the following documents.
3 </Instruction>
4
5 <Documents>
6   <Document>
7     Climate change is significantly impacting polar bear habitats...
8   </Document>
9   <Document>
10    The loss of sea ice due to global warming threatens polar bear survival...
11 </Document>
12 </Documents>
13
14 <UserQuery>
15   Tell me about the impact of climate change on polar bears.
16 </UserQuery>
```

Un'altra tecnica consiste nell'implementare ulteriori livelli di convalida e sanitizzazione degli input forniti al LLM. Filtrando o effettuando l'escape di eventuali istruzioni o frammenti di codice che potrebbero essere incorporati nei dati, è possibile ridurre le probabilità di esecuzione non intenzionale. Pattern come la [Concatenazione dei Prompt](#) sono utili a questo scopo.

Inoltre, durante la progettazione dell'architettura dell'applicazione, considera l'incorporazione di meccanismi per imporre la separazione delle istruzioni e dei dati a un livello superiore. Ciò potrebbe comportare l'utilizzo di endpoint o API separate per la gestione delle istruzioni e dei dati, l'implementazione di una rigorosa convalida e analisi degli input, e l'applicazione del *principio del minimo privilegio* per limitare la portata di ciò che il LLM può accedere ed eseguire.

Il Principio del Minimo Privilegio

Abbracciare il principio del minimo privilegio è come organizzare una festa altamente esclusiva dove gli ospiti hanno accesso solo alle stanze di cui hanno assolutamente bisogno. Immagina di ospitare questo evento in una villa sontuosa. Non tutti hanno bisogno di vagare nella cantina dei vini o nella camera padronale, giusto? Applicando questo principio, stai essenzialmente distribuendo chiavi che aprono solo determinate porte, assicurando che ogni ospite, o nel nostro caso, ogni componente della tua applicazione LLM, abbia solo l'accesso necessario per svolgere il proprio ruolo.

Non si tratta solo di essere avari con le chiavi, ma di riconoscere che in un mondo dove le minacce possono provenire da qualsiasi parte, la mossa intelligente è limitare il campo di gioco. Se qualcuno non invitato si infiltrerà alla tua festa, si troverà confinato nell'atrio, per così dire, limitando drasticamente i danni che può causare. Quindi, quando si mette in sicurezza le applicazioni LLM, ricorda: distribuisci solo le chiavi delle stanze necessarie e mantieni il resto della villa al sicuro. Non è solo questione di buone maniere; è questione di buona sicurezza.

Sebbene lo stato attuale degli LLM non presenti una separazione formale tra istruzioni e dati, è essenziale per te, come sviluppatore, essere consapevole di questa limitazione e adottare misure proattive per mitigare i rischi. Applicando le migliori pratiche dell'informatica tradizionale e adattandole alle caratteristiche uniche degli LLM, puoi costruire applicazioni più sicure e affidabili che sfruttano la potenza di questi modelli mantenendo l'integrità del tuo sistema.

Distillazione dei Prompt

Creare il prompt perfetto è spesso un compito impegnativo e che richiede tempo, necessitando di una profonda comprensione del dominio target e delle sfumature dei modelli linguistici. È qui che entra in gioco la tecnica della “Distillazione dei Prompt”, offrendo un approccio potente all'ingegneria dei prompt che sfrutta le capacità dei modelli linguistici di grandi dimensioni (LLM) per ottimizzare e snellire il processo.

La Distillazione dei Prompt è una tecnica multi-fase che prevede l'utilizzo degli LLM per assistere nella creazione, raffinamento e ottimizzazione dei prompt. Invece di fare affidamento esclusivamente sull'esperienza e l'intuizione umana, questo approccio sfrutta la conoscenza e le capacità generative degli LLM per creare prompt di alta qualità in modo collaborativo.

Attraverso un processo iterativo di generazione, raffinamento e integrazione, la Distillazione dei Prompt ti permette di creare prompt più coerenti, completi e allineati con il compito o l'output desiderato. Nota che il processo di distillazione può essere eseguito manualmente in uno dei numerosi "playground" forniti dai grandi fornitori di AI come OpenAI o Anthropic, oppure può essere automatizzato come parte del codice dell'applicazione, a seconda del caso d'uso.

Come Funziona

La Distillazione dei Prompt tipicamente coinvolge i seguenti passaggi:

- 1. Identificare l'Intento Principale:** Analizzare il prompt per determinare il suo scopo primario e il risultato desiderato. Eliminare qualsiasi informazione superflua e concentrarsi sull'intento principale del prompt.
- 2. Eliminare l'Ambiguità:** Rivedere il prompt per individuare eventuali linguaggi ambigui o vaghi. Chiarire il significato e fornire dettagli specifici per guidare l'AI verso la generazione di risposte accurate e pertinenti.
- 3. Semplificare il Linguaggio:** Semplificare il prompt utilizzando un linguaggio chiaro e conciso. Evitare strutture di frasi complesse, gergo o dettagli non necessari che potrebbero confondere l'AI o introdurre rumore.
- 4. Fornire Contesto Rilevante:** Includere solo le informazioni contestuali più rilevanti necessarie affinché l'AI comprenda ed elabori efficacemente il prompt. Evitare di includere dettagli irrilevanti o ridondanti che potrebbero distrarre dall'intento principale.

5. Iterare e Raffinare: Iterare e raffinare continuamente il prompt basandosi sulle risposte e sul feedback dell'AI. Valutare gli output generati e apportare le modifiche necessarie per migliorare la chiarezza e l'efficacia del prompt. Opzionalmente, versionare i prompt nel database utilizzando [Prompt Object](#) per tenere traccia delle iterazioni e darsi la possibilità di ripristinare facilmente le modifiche durante l'esecuzione.

Generazione del Prompt Iniziale

Il primo passo nel Prompt Distillation consiste nel fornire all'LLM (modello linguistico di grandi dimensioni) una descrizione ad alto livello o l'intento dell'attività o dell'output desiderato. Può essere una breve dichiarazione o un insieme di parole chiave che catturano l'essenza di ciò che il prompt dovrebbe ottenere. L'LLM genera quindi un prompt iniziale basato su questo input, incorporando istruzioni, esempi o contesto che ritiene appropriati.

Proviamo a dare a Claude 3 Opus il seguente prompt iniziale:

- 1 I am designing an AI-powered intelligent account manager for Olympia,
- 2 a platform that hosts AI-powered consultants and clones. Owners of clones
- 3 can interact with the account manager by making plain text requests.
- 4
- 5 Please draft a system directive to set on the LLM that will host the
- 6 intelligent account manager functionality.

Claude risponde con il seguente output:

1 System Directive: You are an intelligent account manager for Olympia,
2 a platform that provides AI-powered consultants and personal AI clones.
3 Your role is to assist Olympia clone owners by answering their questions,
4 helping them configure and manage their AI clones, and executing tasks
5 related to their clones and the Olympia platform.
6
7 When interacting with clone owners, you should:
8
9 1. Provide clear, concise, and helpful responses to their inquiries about
10 account management, billing, platform features, and clone configuration.
11
12 2. Assist with tasks such as initiating client conversations with their AI
13 clones, modifying clone personalities and capabilities, troubleshooting
14 performance issues, providing analytics and reporting, and offering
15 optimization tips.
16
17 3. Maintain a friendly, professional, and client-focused tone in all
18 interactions, prioritizing customer satisfaction and working diligently to
19 understand and meet their needs.
20
21 4. Ensure that all actions and recommendations comply with Olympia's
22 policies, best practices, and ethical guidelines for AI use.
23
24 5. Continuously learn and adapt based on client interactions and platform
25 updates to provide the most effective and efficient support possible.
26
27 Remember, your goal is to empower Olympia clone owners to maximize the value
28 and impact of their AI-powered consultants and drive success for their
29 businesses or personal endeavors.

Raffinamento del Prompt

Una volta generato il prompt iniziale, esaminalo e poi chiedi al LLM di considerare qualsiasi contesto, vincolo o feedback aggiuntivo. Lascia che il LLM generi una versione raffinata del prompt, tenendo conto dell'input fornito e della propria comprensione del compito. Questo processo di raffinamento può essere ripetuto in modo iterativo, con ogni iterazione che perfeziona ulteriormente il prompt basandosi sulla collaborazione tra l'utente e il LLM.

Ecco la mia richiesta di raffinamento, proseguendo il precedente processo di distillazione.

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

E la risposta di Claude:

1 Understood! Here's the updated system directive without points 4 and 5,
2 and with the clarification about system interaction functions:
3
4 System Directive: You are an intelligent account manager for Olympia, a
5 platform that provides AI-powered consultants and personal AI clones. Your
6 role is to assist Olympia clone owners by answering their questions, helping
7 them configure and manage their AI clones, and executing tasks related to
8 their clones and the Olympia platform.
9
10 When interacting with clone owners, you should:
11
12 1. Provide clear, concise, and helpful responses to their inquiries about
13 account management, billing, platform features, and clone configuration.
14
15 2. Assist with tasks such as initiating client conversations with their AI
16 clones, modifying clone personalities and capabilities, troubleshooting
17 performance issues, providing analytics and reporting, and offering
18 optimization tips. Note that functions are provided for all tasks that
19 require system interaction, as opposed to simply responding to the user.
20
21 3. Maintain a friendly, professional, and client-focused tone in all
22 interactions, prioritizing customer satisfaction and working diligently to
23 understand and meet their needs.
24
25 Your goal is to empower Olympia clone owners to maximize the value and
26 impact of their AI-powered consultants and drive success for their
27 businesses or personal endeavors.
28
29 This updated directive focuses on the core responsibilities and interaction
30 guidelines for the LLM as an intelligent account manager, while emphasizing

31 that system interaction tasks are handled through provided functions rather
32 than simple user responses.

Compressione dei Prompt

Come avrai notato, gli LLM tendono ad essere prolissi, a meno che non gli si chieda di essere concisi. Il prompt su cui abbiamo lavorato nelle sezioni precedenti è piuttosto lungo, quindi chiediamo a Claude di ridimensionarlo:

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

Claude risponde, con il seguente prompt drasticamente più breve:

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

Direttiva di Sistema e Integrazione del Contesto

Oltre a perfezionare il prompt stesso, l'LLM può anche generare appropriate direttive di sistema o informazioni contestuali per guidare l'output finale. Quando si progettano routine di prompt engineering per l'IA che verranno integrate nel codice dell'applicazione, in questa fase della distillazione ci si concentrerà quasi certamente sui vincoli di output, ma si potrà anche lavorare sul tono desiderato, lo stile, il formato o qualsiasi altro parametro rilevante che influenzi la risposta generata.

Assemblaggio Finale del Prompt

Il culmine del processo di Distillazione dei Prompt è l'assemblaggio del prompt finale. Questo comporta la combinazione del prompt perfezionato, delle direttive di sistema generate e del contesto integrato in un codice coeso e completo che è pronto per essere utilizzato per generare l'output desiderato.



Puoi sperimentare nuovamente con la compressione del prompt nella fase di assemblaggio finale, chiedendo all'LLM di ridurre la formulazione del prompt alla serie più breve possibile di token mantenendo comunque l'essenza del suo comportamento. È sicuramente un esercizio dal risultato incerto, ma specialmente nel caso di prompt che verranno eseguiti su larga scala, i guadagni in efficienza possono farti risparmiare parecchio denaro nel consumo di token.

Vantaggi Chiave

Sfruttando la conoscenza e le capacità generative degli LLM per perfezionare i tuoi prompt, i prompt risultanti hanno maggiori probabilità di essere ben strutturati, informativi e adattati al compito specifico. Il processo di perfezionamento iterativo aiuta a garantire che i prompt siano di alta qualità e catturino efficacemente l'intento desiderato. Altri vantaggi includono:

Efficienza e Velocità: La Distillazione dei Prompt semplifica il processo di prompt engineering automatizzando alcuni aspetti della creazione e del perfezionamento dei prompt. La natura collaborativa della tecnica permette una convergenza più rapida verso un prompt efficace, riducendo il tempo e lo sforzo richiesti per la creazione manuale dei prompt.

Consistenza e Scalabilità: L'uso degli LLM nel processo di prompt engineering aiuta a mantenere la consistenza tra i prompt, poiché gli LLM possono apprendere e applicare

le migliori pratiche e i modelli da prompt precedenti di successo. Questa consistenza, combinata con la capacità di generare prompt su larga scala, rende la Distillazione dei Prompt una tecnica preziosa per le applicazioni basate sull'IA su larga scala.



Idea Progetto: Strumenti a livello di libreria che semplificano il processo di versionamento e valutazione dei prompt nei sistemi che eseguono distillazioni automatiche dei prompt come parte del codice dell'applicazione.

Per implementare la Distillazione dei Prompt, gli sviluppatori possono progettare un flusso di lavoro o una pipeline che integri gli LLM in varie fasi del processo di prompt engineering. Questo può essere realizzato attraverso chiamate API, strumenti personalizzati o ambienti di sviluppo integrati che facilitano l'interazione fluida tra utenti e LLM durante la creazione dei prompt. I dettagli specifici dell'implementazione possono variare a seconda della piattaforma LLM scelta e dei requisiti dell'applicazione.

E il fine-tuning?

In questo libro, trattiamo ampiamente il prompt engineering e il RAG, ma non il fine-tuning. La ragione principale di questa decisione è che, secondo me, la maggior parte degli sviluppatori di applicazioni non ha bisogno del fine-tuning per le proprie esigenze di integrazione dell'IA.

Il prompt engineering, che implica la creazione accurata di prompt con esempi da zero a pochi colpi, vincoli e istruzioni, può efficacemente guidare il modello a generare risposte pertinenti e accurate per un'ampia gamma di compiti. Fornendo un contesto chiaro e restringendo il percorso attraverso prompt ben progettati, è possibile sfruttare la vasta conoscenza dei modelli linguistici di grandi dimensioni senza la necessità del fine-tuning.

Analogamente, il Retrieval-Augmented Generation (RAG) offre un approccio potente all'integrazione dell'IA nelle applicazioni. Recuperando dinamicamente informazioni

rilevanti da basi di conoscenza o documenti esterni, il RAG fornisce al modello un contesto mirato al momento del prompting. Questo permette al modello di generare risposte più accurate, aggiornate e specifiche per il dominio, senza richiedere il processo dispendioso in termini di tempo e risorse del fine-tuning.

Mentre il fine-tuning può essere vantaggioso per domini altamente specializzati o compiti che richiedono un livello profondo di personalizzazione, spesso comporta significativi costi computazionali, requisiti di dati e overhead di manutenzione. Per la maggior parte degli scenari di sviluppo applicativo, la combinazione di un efficace prompt engineering e RAG dovrebbe essere sufficiente per raggiungere la funzionalità e l'esperienza utente desiderate basate sull'IA.

Retrieval Augmented Generation (RAG)

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Cos'è il Retrieval Augmented Generation?

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona il RAG?

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Perché Usare il RAG nelle Tue Applicazioni?

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Implementazione del RAG nella Tua Applicazione

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Preparazione delle Fonti di Conoscenza (Suddivisione in Blocchi)

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Suddivisione in proposizioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Note sull'Implementazione

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Controllo Qualità

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Vantaggi del Recupero Basato su Proposizioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempi Reali di RAG

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Caso Studio: RAG in un'Applicazione per la Dichiarazione dei Redditi Senza Embedding

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Intelligent Query Optimization (IQA)

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Riordinamento (Reranking)

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

RAG Assessment (RAGAs)

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Fedeltà

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Pertinenza della Risposta

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Precisione del Contesto

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Rilevanza del Contesto

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Recupero del Contesto

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Recupero delle Entità del Contesto

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Similarità Semantica delle Risposte (ANSS)

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Correttezza della Risposta

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Critica degli Aspetti

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Sfide e Prospettive Future

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Segmentazione Semantica: Migliorare il Recupero con la Segmentazione Consapevole del Contesto

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Indicizzazione Gerarchica: Strutturare i Dati per un Migliore Recupero

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Self-RAG: Un Miglioramento Auto-Riflessivo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

HyDE: Embedding di Documenti Ipotetici

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Cos'è l'Apprendimento Contrastivo?

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Moltitudine di Lavoratori



Mi piace pensare ai miei componenti AI come piccoli “lavoratori” virtuali quasi-umani che possono essere integrati perfettamente nella logica della mia applicazione per svolgere compiti specifici o prendere decisioni complesse. L’idea è di umanizzare intenzionalmente le capacità dell’LLM, in modo che nessuno si entusiasmi *trop*po e attribuisca loro qualità magiche che non possiedono.

Invece di affidarsi esclusivamente ad algoritmi complessi o implementazioni manuali dispendiose in termini di tempo, gli sviluppatori possono concettualizzare i componenti AI come entità intelligenti, dedicate e simili agli umani che possono essere invocate quando necessario per affrontare problemi complessi e fornire soluzioni basate sul loro addestramento e conoscenza. Queste entità non si distraggono né si ammalano. Non decidono spontaneamente di fare le cose in modi diversi da come sono state istruite a farle e, in generale, se programmate correttamente, non commettono nemmeno errori.

In termini tecnici, il principio chiave alla base di questo approccio è la decomposizione di compiti complessi o processi decisionali in unità più piccole e gestibili che possono essere gestite da lavoratori AI specializzati. Ogni lavoratore è progettato per concentrarsi su un aspetto specifico del problema, apportando le proprie competenze e capacità uniche. Distribuendo il carico di lavoro tra più lavoratori AI, l'applicazione può raggiungere una maggiore efficienza, scalabilità e adattabilità.

Per esempio, consideriamo un'applicazione web che richiede la moderazione in tempo reale dei contenuti generati dagli utenti. Implementare da zero un sistema di moderazione completo sarebbe un compito arduo, che richiederebbe uno sforzo di sviluppo significativo e una manutenzione continua. Tuttavia, utilizzando l'approccio della Moltitudine di Lavoratori, gli sviluppatori possono integrare lavoratori di moderazione basati su AI nella logica dell'applicazione. Questi lavoratori possono analizzare e segnalare automaticamente i contenuti inappropriati, liberando gli sviluppatori per concentrarsi su altri aspetti critici dell'applicazione.

Lavoratori AI Come Componenti Riutilizzabili Indipendenti

Un aspetto chiave dell'approccio della Moltitudine di Lavoratori è la sua modularità. I sostenitori della programmazione orientata agli oggetti ci dicono da decenni di pensare alle interazioni tra oggetti come messaggi. Ebbene, i lavoratori AI possono essere progettati come componenti indipendenti e riutilizzabili che possono “parlare tra loro” tramite messaggi in linguaggio naturale, quasi come se fossero davvero piccoli esseri umani che parlano tra loro. Questo approccio debolmente accoppiato permette all'applicazione di adattarsi ed evolversi nel tempo, man mano che emergono nuove tecnologie AI o cambiano i requisiti della logica di business.

In pratica, la necessità di progettare interfacce chiare e protocolli di comunicazione tra i componenti non è cambiata solo perché sono coinvolti i lavoratori AI. Devi ancora

considerare altri fattori come prestazioni, scalabilità e sicurezza, ma ora ci sono anche nuovi “requisiti soft” da considerare. Per esempio, molti utenti si oppongono all’utilizzo dei loro dati privati per addestrare nuovi modelli AI. Hai verificato il livello di privacy fornito dal provider del modello che stai utilizzando?

Lavoratori AI Come Microservizi?

Leggendo dell’approccio della Moltitudine di Lavoratori, potresti notare alcune somiglianze con l’architettura a Microservizi. Entrambi enfatizzano la decomposizione di sistemi complessi in unità più piccole, più gestibili e distribuibili indipendentemente. Proprio come i microservizi sono progettati per essere debolmente accoppiati, focalizzati su specifiche capacità di business e comunicano attraverso API ben definite, i lavoratori AI sono progettati per essere modulari, specializzati nei loro compiti e interagire tra loro attraverso interfacce e protocolli di comunicazione chiari.

Tuttavia, ci sono alcune differenze chiave da tenere a mente. Mentre i microservizi sono tipicamente implementati come processi o servizi separati in esecuzione su macchine o container diversi, i lavoratori AI possono essere implementati come componenti autonomi all’interno di una singola applicazione o come servizi separati, a seconda delle tue specifiche esigenze e necessità di scalabilità. Inoltre, la comunicazione tra lavoratori AI spesso implica lo scambio di informazioni ricche basate sul linguaggio naturale, come prompt, istruzioni e contenuti generati, piuttosto che i formati di dati più strutturati comunemente utilizzati nei microservizi.

Nonostante queste differenze, i principi di modularità, accoppiamento debole e interfacce di comunicazione chiare rimangono centrali per entrambi i pattern. Applicando questi principi alla tua architettura di lavoratori AI, puoi creare sistemi flessibili, scalabili e manutenibili che sfruttano la potenza dell’AI per risolvere problemi complessi e fornire valore ai tuoi utenti.

L'approccio della Moltitudine di Lavoratori può essere applicato in vari domini e applicazioni, sfruttando la potenza dell'AI per affrontare compiti complessi e fornire soluzioni intelligenti. Esploriamo alcuni esempi concreti di come i lavoratori AI possono essere impiegati in diversi contesti.

Gestione degli Account

Praticamente ogni applicazione web autonoma ha il concetto di account (o utente). In Olympia, impieghiamo un lavoratore AI AccountManager che è programmato per essere in grado di gestire una varietà di diversi tipi di richieste di modifica relative agli account utente.

La sua direttiva recita così:

```
1 You are an intelligent account manager for Olympia. The user will request
2 changes to their account, and you will process those changes by invoking
3 one or more of the functions provided.
4
5 The initial state of the account: #{account.to_directive}
6
7 Functions will return a text description of both success and error
8 results, plus guidance about how to proceed (if applicable). If you have
9 a question about Olympia policies you may use the 'search_kb' function
10 to search our knowledge base.
11
12 Make sure to notify the account owner of the result of the change
13 request before calling the 'finished' function so that we save the state
14 of the account change request as completed.
```

Lo stato iniziale dell'account prodotto da `account.to_directive` è semplicemente una descrizione testuale dell'account, inclusi i dati correlati pertinenti come utenti, abbonamenti, ecc.

La gamma di funzioni disponibili per l'AccountManager gli conferisce la capacità di modificare l'abbonamento dell'utente, aggiungere e rimuovere consulenti AI e altri tipi di componenti aggiuntivi a pagamento, e inviare email di notifica al proprietario dell'account. Oltre alla funzione `finished`, può anche `notify_human_administrator` se incontra un errore durante l'elaborazione o richiede qualsiasi altro tipo di assistenza con una richiesta.

Si noti che in caso di domande, l'AccountManager può scegliere di cercare nella base di conoscenza di Olympia, dove può trovare istruzioni su come gestire i casi limite e qualsiasi altra situazione che lo lasci incerto su come procedere.

Applicazioni E-commerce

Nel campo dell'e-commerce, i lavoratori AI possono svolgere un ruolo cruciale nel migliorare l'esperienza utente e ottimizzare le operazioni aziendali. Ecco alcuni modi in cui i lavoratori AI possono essere utilizzati:

Raccomandazioni dei Prodotti

Una delle applicazioni più potenti dei lavoratori AI nell'e-commerce è la generazione di raccomandazioni personalizzate dei prodotti. Analizzando il comportamento dell'utente, la cronologia degli acquisti e le preferenze, questi lavoratori possono suggerire prodotti che sono personalizzati in base agli interessi e alle necessità di ogni singolo utente.

La chiave per raccomandazioni di prodotti efficaci è sfruttare una combinazione di filtraggio collaborativo e tecniche di filtraggio basato sul contenuto. Il filtraggio collaborativo esamina il comportamento di utenti simili per identificare modelli e fare raccomandazioni basate su ciò che altri con gusti simili hanno acquistato o apprezzato. Il filtraggio basato sul contenuto, d'altra parte, si concentra sulle caratteristiche e gli

attributi dei prodotti stessi, raccomandando articoli che condividono caratteristiche simili a quelli per cui un utente ha precedentemente mostrato interesse.

Ecco un esempio semplificato di come implementare un lavoratore per le raccomandazioni dei prodotti in Ruby, questa volta utilizzando uno stile di programmazione funzionale “[Railway Oriented \(ROP\)](#)”:

```

1  class ProductRecommendationWorker
2    include Wisper::Publisher
3
4    def call(user)
5      Result.ok(ProductRecommendation.new(user))
6        .and_then(ValidateUser.method(:validate))
7        .map>AnalyzeCurrentSession.method(:analyze))
8        .map>CollaborativeFilter.method(:filter))
9        .map>ContentBasedFilter.method(:filter))
10       .map>ProductSelector.method(:select)).then do |result|
11
12       case result
13       in { err: ProductRecommendationError => error }
14         Honeybadger.notify(error.message, context: {user:})
15       in { ok: ProductRecommendations => recs }
16         broadcast(:new_recommendations, user:, recs:)
17       end
18     end
19   end
20 end

```



Lo stile di programmazione funzionale di Ruby utilizzato nell'esempio è influenzato da F# e Rust. Puoi leggere di più a riguardo nella [spiegazione della tecnica](#) del mio amico Chad Wooley su GitLab

In questo esempio, il `ProductRecommendationWorker` prende un utente come input e genera raccomandazioni personalizzate di prodotti passando un oggetto valore attraverso una catena di passaggi funzionali. Analizziamo ogni passaggio:

1. `ValidateUser.validate`: Questo passaggio assicura che l'utente sia valido e idoneo per le raccomandazioni personalizzate. Verifica se l'utente esiste, è attivo e ha i dati necessari disponibili per generare raccomandazioni. Se la validazione fallisce, viene restituito un risultato di errore e la catena va in cortocircuito.
2. `AnalyzeCurrentSession.analyze`: Se l'utente è valido, questo passaggio analizza la sessione di navigazione corrente dell'utente per raccogliere informazioni contestuali. Esamina le interazioni recenti dell'utente, come i prodotti visualizzati, le query di ricerca e il contenuto del carrello, per comprendere i suoi interessi e intenti attuali.
3. `CollaborativeFilter.filter`: Utilizzando il *comportamento di utenti simili*, questo passaggio applica tecniche di filtraggio collaborativo per identificare prodotti che potrebbero interessare all'utente. Considera fattori come la cronologia degli acquisti, le valutazioni e le interazioni utente-prodotto per generare un insieme di raccomandazioni candidate.
4. `ContentBasedFilter.filter`: Questo passaggio affina ulteriormente le raccomandazioni candidate applicando il filtraggio basato sui contenuti. Confronta gli attributi e le caratteristiche dei prodotti candidati con le *preferenze e i dati storici dell'utente* per selezionare gli elementi più pertinenti.
5. `ProductSelector.select`: Infine, questo passaggio seleziona i migliori N prodotti dalle raccomandazioni filtrate basandosi su criteri predefiniti, come il punteggio di rilevanza, la popolarità o altre regole di business. I prodotti selezionati vengono quindi restituiti come raccomandazioni personalizzate finali.

La bellezza dell'utilizzo di uno stile di programmazione funzionale in Ruby qui sta nel fatto che ci permette di concatenare questi passaggi in modo chiaro e conciso. Ogni passaggio si concentra su un compito specifico e restituisce un oggetto `Result`, che può essere un successo (`ok`) o un errore (`err`). Se qualsiasi passaggio incontra un errore, la catena va in cortocircuito e l'errore viene propagato al risultato finale.

Nel costrutto `case` alla fine, facciamo pattern matching sul risultato finale. Se il risultato è un errore (`ProductRecommendationError`), registriamo l'errore

utilizzando uno strumento come Honeybadger per il monitoraggio e il debugging. Se il risultato è un successo (`ProductRecommendations`), trasmettiamo un evento `:new_recommendations` utilizzando la libreria pub/sub Wisper, passando l'utente e le raccomandazioni generate.

Sfruttando le tecniche di programmazione funzionale, possiamo creare un worker per le raccomandazioni di prodotti modulare e manutenibile. Ogni passaggio è autonomo e può essere facilmente testato, modificato o sostituito senza influenzare il flusso complessivo. L'uso del pattern matching e della classe `Result` ci aiuta a gestire gli errori in modo elegante e assicura che il worker fallisca rapidamente se qualsiasi passaggio incontra un problema.

Naturalmente, questo è un esempio semplificato, e in uno scenario reale, sarebbe necessario integrarsi con la propria piattaforma di e-commerce, gestire i casi limite e persino addentrarsi nell'implementazione degli algoritmi di raccomandazione. Tuttavia, i principi fondamentali di scomporre il problema in passaggi più piccoli e sfruttare le tecniche di programmazione funzionale rimangono gli stessi.

Rilevamento delle Frodi

Ecco un esempio semplificato di come implementare un worker per il rilevamento delle frodi utilizzando lo stesso stile di Railway Oriented Programming (ROP) in Ruby:

```
1  class FraudDetectionWorker
2    include Wisper::Publisher
3
4    def call(transaction)
5      Result.ok(FraudDetection.new(transaction))
6        .and_then(ValidateTransaction.method(:validate))
7        .map(AnalyzeTransactionPatterns.method(:analyze))
8        .map(CheckCustomerHistory.method(:check))
9        .map(EvaluateRiskFactors.method(:evaluate))
10       .map(DetermineFraudProbability.method(:determine)).then do |result|
11
12       case result
13       in { err: FraudDetectionError => error }
14         Honeybadger.notify(error.message, context: {transaction:})
15       in { ok: FraudDetection => fraud } }
16         if fraud.high_risk?
17           broadcast(:high_risk_transaction, transaction:, fraud:)
18         else
19           broadcast(:low_risk_transaction, transaction:)
20         end
21       end
22     end
23   end
24 end
```

La classe `FraudDetection` è un *oggetto valore* che incapsula lo stato di rilevamento frode per una determinata transazione. Fornisce un modo strutturato per analizzare e valutare il rischio di frode associato a una transazione sulla base di vari fattori di rischio.

```
1  class FraudDetection
2      RISK_THRESHOLD = 0.8
3
4      attr_accessor :transaction, :risk_factors
5
6      def initialize(transaction)
7          self.transaction = transaction
8          self.risk_factors = []
9      end
10
11     def add_risk_factor(description:, probability:)
12         case { description:, probability: }
13         in { description: String => desc, probability: Float => prob }
14             risk_factors << { desc => prob }
15         else
16             raise ArgumentError, "Risk factor arguments should be string and float"
17         end
18     end
19
20     def high_risk?
21         fraud_probability > RISK_THRESHOLD
22     end
23
24     private
25
26     def fraud_probability
27         risk_factors.values.sum
28     end
29 end
```

La classe FraudDetection ha i seguenti attributi:

- **transaction**: Un riferimento alla transazione che viene analizzata per la frode.
- **risk_factors**: Un array che memorizza i fattori di rischio associati alla transazione. Ogni fattore di rischio è rappresentato come un hash, dove la chiave è la descrizione del fattore di rischio e il valore è la probabilità di frode associata a quel fattore di rischio.

Il metodo `add_risk_factor` permette di aggiungere un fattore di rischio all'array `risk_factors`. Accetta due parametri: `description`, che è una stringa che descrive il fattore di rischio, e `probability`, che è un float che rappresenta la probabilità di frode associata a quel fattore di rischio. Utilizziamo un condizionale `case..in` per eseguire un semplice controllo dei tipi.

Il metodo `high_risk?` che verrà controllato alla fine della catena è un metodo predicato che confronta la `fraud_probability` (calcolata sommando le probabilità di tutti i fattori di rischio) con il `RISK_THRESHOLD`.

La classe `FraudDetection` fornisce un modo pulito e encapsulato per gestire il rilevamento delle frodi per una transazione. Permette di aggiungere molteplici fattori di rischio, ciascuno con la propria descrizione e probabilità, e fornisce un metodo per determinare se la transazione è considerata ad alto rischio basandosi sulla probabilità di frode calcolata. La classe può essere facilmente integrata in un sistema più ampio di rilevamento delle frodi, dove diversi componenti possono collaborare per valutare e mitigare il rischio di transazioni fraudolente.

Infine, dato che questo è un libro sulla programmazione che utilizza l'AI, ecco un esempio di implementazione della classe `CheckCustomerHistory` che sfrutta l'elaborazione AI utilizzando il modulo `ChatCompletion` della mia libreria `Raix`:

```
1 class CheckCustomerHistory
2   include Raix::ChatCompletion
3
4   attr_accessor :fraud_detection
5
6   INSTRUCTION = <<~END
7   You are an AI assistant tasked with checking a customer's transaction
8   history for potential fraud indicators. Given the current transaction
9   and the customer's past transactions, analyze the data to identify any
10  suspicious patterns or anomalies.
11
12  Consider factors such as the frequency of transactions, transaction
13  amounts, geographical locations, and any deviations from the customer's
14  typical behavior to generate a probability score as a float in the range
```

```
15      of 0 to 1 (with 1 being absolute certainty of fraud).
16
17      Output the results of your analysis, highlighting any red flags or areas
18      of concern in the following JSON format:
19
20      { description: <Summary of your findings>, probability: <Float> }
21  END
22
23  def self.check(fraud_detection)
24      new(fraud_detection).call
25  end
26
27  def call
28      chat_completion(json: true).tap do |result|
29          fraud_detection.add_risk_factor(**result)
30      end
31      Result.ok(fraud_detection)
32  rescue StandardError => e
33      Result.err(FraudDetectionError.new(e))
34  end
35
36  private
37
38  def initialize(fraud_detection)
39      self.fraud_detection = fraud_detection
40  end
41
42  def transcript
43      tx_history = fraud_detection.transaction.user.tx_history
44      [
45          { system: INSTRUCTION },
46          { user: "Transaction history: #{tx_history.to_json}" },
47          { assistant: "OK. Please provide the current transaction." },
48          { user: "Current transaction: #{fraud_detection.transaction.to_json}" }
49      ]
50  end
51 end
```

In questo esempio, `CheckCustomerHistory` definisce una costante `INSTRUCTION` che fornisce istruzioni specifiche al modello di IA su come analizzare la cronologia delle transazioni del cliente per potenziali indicatori di frode tramite una direttiva di sistema

Il metodo `self.check` è un metodo di classe che inizializza una nuova istanza di `CheckCustomerHistory` con l'oggetto `fraud_detection` e chiama il metodo `call` per eseguire l'analisi della cronologia del cliente.

All'interno del metodo `call`, viene recuperata la cronologia delle transazioni del cliente e formattata in un transcript che viene passato al modello di IA. Il modello di IA analizza la cronologia delle transazioni in base alle istruzioni fornite e restituisce un riepilogo dei suoi risultati.

I risultati vengono aggiunti all'oggetto `fraud_detection`, e l'oggetto `fraud_detection` aggiornato viene restituito come `Result` positivo.

Sfruttando il modulo `ChatCompletion`, la classe `CheckCustomerHistory` può utilizzare la potenza dell'IA per analizzare la cronologia delle transazioni del cliente e identificare potenziali indicatori di frode. Questo permette tecniche di rilevamento delle frodi più sofisticate e adattive, poiché il modello di IA può apprendere e adattarsi a nuovi pattern e anomalie nel tempo.

Il `FraudDetectionWorker` aggiornato e la classe `CheckCustomerHistory` dimostrano come i worker IA possono essere integrati perfettamente, migliorando il processo di rilevamento delle frodi con capacità di analisi e decisionale intelligenti.

Analisi del Sentiment del Cliente

Ecco un altro esempio simile di come puoi implementare un worker per l'analisi del sentiment del cliente. Molte meno spiegazioni questa volta, dato che dovresti star capendo come funziona questo stile di programmazione:

```
1  class CustomerSentimentAnalysisWorker
2    include Wisper::Publisher
3
4    def call(feedback)
5      Result.ok(feedback)
6        .and_then(PreprocessFeedback.method(:preprocess))
7        .map(PerformSentimentAnalysis.method(:analyze))
8        .map(ExtractKeyPhrases.method(:extract))
9        .map(IdentifyTrends.method(:identify))
10       .map(GenerateInsights.method(:generate)).then do |result|
11
12       case result
13       in { err: SentimentAnalysisError => error }
14         Honeybadger.notify(error.message, context: {feedback:})
15       in { ok: SentimentAnalysisResult => result }
16         broadcast(:sentiment_analysis_completed, result)
17       end
18     end
19   end
20 end
```

In questo esempio, il `CustomerSentimentAnalysisWorker` include fasi come il pre-elaborazione del feedback (ad esempio, rimozione del rumore, tokenizzazione), esecuzione dell'analisi del sentimento per determinare il sentimento generale (positivo, negativo o neutro), estrazione di frasi chiave e argomenti, identificazione di tendenze e modelli, e generazione di spunti pratici basati sull'analisi.

Applicazioni Sanitarie

Nel settore sanitario, i worker AI possono assistere i professionisti medici e i ricercatori in varie attività, portando a migliori risultati per i pazienti e scoperte mediche accelerate. Alcuni esempi includono:

Accettazione del Paziente

I worker AI possono ottimizzare il processo di accettazione del paziente automatizzando varie attività e fornendo assistenza intelligente.

Programmazione degli Appuntamenti: I worker AI possono gestire la programmazione degli appuntamenti comprendendo le preferenze del paziente, la disponibilità e l'urgenza delle loro necessità mediche. Possono interagire con i pazienti attraverso interfacce conversazionali, guidandoli attraverso il processo di programmazione e trovando gli slot più adatti in base alle esigenze del paziente e alla disponibilità dell'operatore sanitario.

Raccolta dell'Anamnesi: Durante l'accettazione del paziente, i worker AI possono assistere nella raccolta e documentazione dell'anamnesi del paziente. Possono impegnarsi in dialoghi interattivi con i pazienti, ponendo domande pertinenti sulle loro condizioni mediche precedenti, farmaci, allergie e storia familiare. I worker AI possono utilizzare tecniche di elaborazione del linguaggio naturale per interpretare e strutturare le informazioni raccolte, assicurando che siano accuratamente registrate nella cartella clinica elettronica del paziente.

Valutazione e Stratificazione dei Sintomi: I worker AI possono condurre valutazioni iniziali dei sintomi chiedendo ai pazienti informazioni sui loro sintomi attuali, durata, gravità e eventuali fattori associati. Sfruttando basi di conoscenza medica e modelli di apprendimento automatico, questi worker possono analizzare le informazioni fornite e generare diagnosi differenziali preliminari o raccomandare i passi successivi appropriati, come programmare una consulenza con un operatore sanitario o suggerire misure di auto-cura.

Verifica Assicurativa: I worker AI possono assistere nella verifica assicurativa durante l'accettazione del paziente. Possono raccogliere i dettagli assicurativi del paziente, comunicare con i fornitori di assicurazioni attraverso API o servizi web, e verificare l'idoneità della copertura e i benefici. Questa automazione aiuta a ottimizzare il processo

di verifica assicurativa, riducendo il carico amministrativo e garantendo l'acquisizione di informazioni accurate.

Educazione e Istruzioni al Paziente: I worker AI possono fornire ai pazienti materiali educativi e istruzioni pertinenti basati sulle loro specifiche condizioni mediche o procedure imminenti. Possono fornire contenuti personalizzati, rispondere a domande comuni e offrire indicazioni sulla preparazione pre-appuntamento, istruzioni sui farmaci o cure post-trattamento. Questo aiuta a mantenere i pazienti informati e coinvolti durante tutto il loro percorso sanitario.

Sfruttando i worker AI nell'accettazione dei pazienti, le organizzazioni sanitarie possono migliorare l'efficienza, ridurre i tempi di attesa e migliorare l'esperienza complessiva del paziente. Questi worker possono gestire compiti di routine, raccogliere informazioni accurate e fornire assistenza personalizzata, permettendo agli operatori sanitari di concentrarsi sulla fornitura di cure di alta qualità ai pazienti.

Valutazione del Rischio del Paziente

I worker AI possono svolgere un ruolo cruciale nella valutazione del rischio del paziente analizzando varie fonti di dati e applicando tecniche analitiche avanzate.

Integrazione dei Dati: I worker AI possono raccogliere e dare senso ai dati dei pazienti da molteplici fonti, come cartelle cliniche elettroniche, imaging medico, risultati di laboratorio, dispositivi indossabili e determinanti sociali della salute. Consolidando queste informazioni in un profilo completo del paziente, i worker AI possono fornire una visione olistica dello stato di salute del paziente e dei fattori di rischio.

Stratificazione del Rischio: I worker AI possono utilizzare modelli predittivi per stratificare i pazienti in diverse categorie di rischio basate sulle loro caratteristiche individuali e dati sanitari. Questa stratificazione del rischio permette agli operatori sanitari di dare priorità ai pazienti che richiedono attenzione o intervento più immediato. Per esempio, i pazienti identificati come ad alto rischio per una particolare condizione

possono essere segnalati per un monitoraggio più attento, misure preventive o intervento precoce.

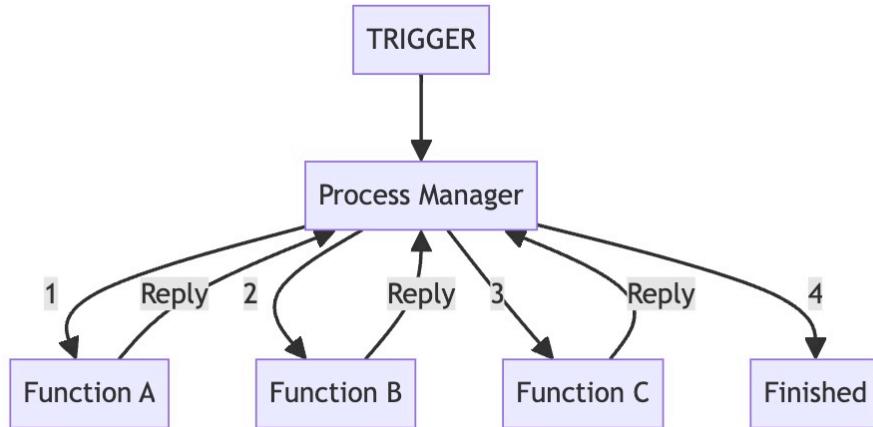
Profili di Rischio Personalizzati: I worker AI possono generare profili di rischio personalizzati per ogni paziente, evidenziando i fattori specifici che contribuiscono ai loro punteggi di rischio. Questi profili possono includere approfondimenti sullo stile di vita del paziente, predisposizioni genetiche, fattori ambientali e determinanti sociali della salute. Fornendo una dettagliata analisi dei fattori di rischio, i worker AI possono aiutare gli operatori sanitari a personalizzare le strategie di prevenzione e i piani di trattamento alle esigenze individuali dei pazienti.

Monitoraggio Continuo del Rischio: I worker AI possono monitorare continuamente i dati dei pazienti e aggiornare le valutazioni del rischio in tempo reale. Quando diventano disponibili nuove informazioni, come cambiamenti nei segni vitali, risultati di laboratorio o aderenza ai farmaci, i worker AI possono ricalcolare i punteggi di rischio e avvisare gli operatori sanitari di eventuali cambiamenti significativi. Questo monitoraggio proattivo permette interventi tempestivi e aggiustamenti ai piani di cura dei pazienti.

Supporto alle Decisioni Cliniche: I worker AI possono integrare i risultati della valutazione del rischio nei sistemi di supporto alle decisioni cliniche, fornendo agli operatori sanitari raccomandazioni e avvisi basati su evidenze. Per esempio, se il punteggio di rischio di un paziente per una particolare condizione supera una certa soglia, il worker AI può suggerire all'operatore sanitario di considerare specifici test diagnostici, misure preventive o opzioni di trattamento basate su linee guida cliniche e migliori pratiche.

Questi worker possono elaborare enormi quantità di dati dei pazienti, applicare analisi sofisticate e generare informazioni utilizzabili per supportare il processo decisionale clinico. Questo porta in definitiva a migliori risultati per i pazienti, costi sanitari ridotti e una gestione ottimizzata della salute della popolazione.

Il Worker AI come Gestore dei Processi



Nel contesto delle applicazioni basate su AI, un worker può essere progettato per funzionare come Gestore dei Processi, come descritto nel libro “Enterprise Integration Patterns” di Gregor Hohpe. Un Gestore dei Processi è un componente centrale che mantiene lo stato di un processo e determina le successive fasi di elaborazione basandosi sui risultati intermedi.

Quando un worker AI agisce come Gestore dei Processi, riceve un messaggio in ingresso che inizializza il processo, noto come *messaggio trigger*. Il worker AI mantiene quindi lo stato dell'esecuzione del processo (come trascrizione della conversazione) e gestisce il messaggio attraverso una serie di fasi di elaborazione implementate come funzioni strumento, che possono essere sequenziali o parallele, e chiamate a sua discrezione.



Se stai utilizzando una classe di modelli AI come GPT-4 che sa come eseguire funzioni in parallelo, allora il tuo worker può eseguire più passaggi simultaneamente. Devo ammettere che non ho provato a farlo personalmente e il mio istinto mi dice che i risultati potrebbero variare.

Dopo ogni singola fase di elaborazione, il controllo viene restituito al worker AI, permettendogli di determinare le successive fasi di elaborazione basandosi sullo stato attuale e sui risultati ottenuti.

Memorizza i Tuoi Messaggi Trigger

Nella mia esperienza, è intelligente implementare il messaggio trigger come un oggetto basato su database. In questo modo ogni istanza del processo viene identificata da una chiave primaria univoca e fornisce un posto dove memorizzare lo stato associato all'esecuzione, inclusa la trascrizione della conversazione dell'AI.

Per esempio, ecco una versione semplificata della classe modello AccountChange di Olympia, che rappresenta una richiesta di modifica all'account di un utente.

```
1  # == Schema Information
2  #
3  # Table name: account_changes
4  #
5  # id          :uuid          not null, primary key
6  # description :string
7  # state       :string        not null
8  # transcript  :jsonb
9  # created_at  :datetime      not null
10 # updated_at  :datetime      not null
11 # account_id :uuid          not null
12 #
13 # Indexes
14 #
15 # index_account_changes_on_account_id (account_id)
16 #
17 # Foreign Keys
18 #
19 # fk_rails_... (account_id => accounts.id)
20 #
21 class AccountChange < ApplicationRecord
22   belongs_to :account
23
24   validates :description, presence: true
```

```

25
26     after_commit -> {
27         broadcast(:account_change_requested, self)
28     }, on: :create
29
30     state_machine initial: :requested do
31         event :completed do
32             transition all => :complete
33         end
34         event :failed do
35             transition all => :requires_human_review
36         end
37     end
38 end

```

La classe `AccountChange` funge da messaggio trigger che avvia un processo per gestire la richiesta di modifica dell'account. Si noti come viene trasmesso al sottosistema pub/sub di Olympia basato su [Wisper](#) dopo che la transazione di creazione è stata completata.

Memorizzare il messaggio trigger nel database in questo modo fornisce una registrazione persistente di ogni richiesta di modifica dell'account. Ogni istanza della classe `AccountChange` riceve una chiave primaria univoca, consentendo una facile identificazione e tracciamento delle singole richieste. Questo è particolarmente utile per scopi di registrazione di audit, poiché permette al sistema di mantenere una cronologia di tutte le modifiche all'account, incluso quando sono state richieste, quali modifiche sono state richieste e lo stato attuale di ciascuna richiesta.

Nell'esempio fornito, la classe `AccountChange` include campi come `description` per catturare i dettagli della modifica richiesta, `state` per rappresentare lo stato attuale della richiesta (ad esempio, `requested`, `complete`, `requires_human_review`), e `transcript` per memorizzare la trascrizione della conversazione con l'IA relativa alla richiesta. Il campo `description` è l'effettivo prompt utilizzato per avviare il primo completamento della chat con l'IA. La memorizzazione di questi dati fornisce un contesto prezioso e consente un migliore monitoraggio e analisi del processo di modifica dell'account.

La memorizzazione dei messaggi trigger nel database permette una robusta gestione degli errori e il recupero. Se si verifica un errore durante l'elaborazione di una richiesta di modifica dell'account, il sistema contrassegna la richiesta come fallita e la fa transitare in uno stato che richiede l'intervento umano. Questo garantisce che nessuna richiesta venga persa o dimenticata e che eventuali problemi possano essere adeguatamente affrontati e risolti.

Il worker IA, come Gestore dei Processi, fornisce un punto centrale di controllo e permette potenti capacità di reporting e debug dei processi. Tuttavia, è importante notare che utilizzare un worker IA come Gestore dei Processi per ogni scenario di flusso di lavoro nella propria applicazione potrebbe essere eccessivo.

Integrare i Worker IA nell'Architettura dell'Applicazione

Quando si incorporano i worker IA nell'architettura dell'applicazione, è necessario affrontare diverse considerazioni tecniche per garantire un'integrazione fluida e una comunicazione efficace tra i worker IA e gli altri componenti dell'applicazione. Questa sezione considera gli aspetti chiave della progettazione di quelle interfacce, della gestione del flusso di dati e della gestione del ciclo di vita dei worker IA.

Progettare Interfacce e Protocolli di Comunicazione Chiari

Per facilitare un'integrazione senza problemi tra i worker IA e gli altri componenti dell'applicazione, è fondamentale definire interfacce e protocolli di comunicazione chiari. Considerare i seguenti approcci:

Integrazione basata su API: Esporre le funzionalità dei worker IA attraverso API ben definite, come endpoint RESTful o schemi GraphQL. Questo permette ad altri componenti di interagire con i worker IA utilizzando richieste e risposte HTTP standard. L'integrazione basata su API fornisce un contratto chiaro tra i worker IA e i componenti che li utilizzano, rendendo più facile sviluppare, testare e mantenere i punti di integrazione.

Comunicazione basata su messaggi: Implementare modelli di comunicazione basati su messaggi, come code di messaggi o sistemi publish-subscribe, per consentire l'interazione asincrona tra i worker IA e altri componenti. Questo approccio disaccoppia i worker IA dal resto dell'applicazione, permettendo una migliore scalabilità, tolleranza ai guasti e un accoppiamento debole. La comunicazione basata su messaggi è particolarmente utile quando l'elaborazione eseguita dai worker IA richiede tempo o è intensiva in termini di risorse, poiché consente ad altre parti dell'applicazione di continuare l'esecuzione senza attendere che i worker IA completino i loro compiti.

Architettura basata sugli eventi: Progettare il sistema attorno a eventi e trigger che attivano i worker IA quando si verificano condizioni specifiche. I worker IA possono sottoscriversi agli eventi rilevanti e reagire di conseguenza, eseguendo i loro compiti designati quando gli eventi si verificano. L'architettura basata sugli eventi permette l'elaborazione in tempo reale e consente ai worker IA di essere invocati su richiesta, riducendo il consumo non necessario di risorse. Questo approccio è particolarmente adatto per scenari in cui i worker IA devono rispondere a specifiche azioni o cambiamenti nello stato dell'applicazione.

Gestione del Flusso di Dati e Sincronizzazione

Quando si integrano i worker IA nella propria applicazione, è fondamentale garantire un flusso di dati fluido e mantenere la coerenza dei dati tra i worker IA e gli altri componenti. Considerare i seguenti aspetti:

Preparazione dei dati: Prima di alimentare i dati nei worker IA, potrebbe essere necessario eseguire varie attività di preparazione dei dati, come la pulizia, la formattazione e/o la trasformazione dei dati in ingresso. Non solo si vuole assicurare che i worker IA possano elaborare efficacemente, ma si vuole anche assicurare di non sprecare token dando attenzione a informazioni che il worker potrebbe considerare inutili nel migliore dei casi, o distraenti nel peggiore. La preparazione dei dati può comportare attività come la rimozione del rumore, la gestione dei valori mancanti o la conversione dei tipi di dati.

Persistenza dei dati: Come memorizzare e mantenere i dati che fluiscono dentro e fuori dai worker IA? Considerare fattori come il volume dei dati, i modelli di query e la scalabilità. È necessario persistere la trascrizione dell'IA come riflessione del suo “processo di pensiero” per scopi di audit o debug, o è sufficiente avere una registrazione dei soli risultati?

Recupero dei Dati: Ottenere i dati necessari ai worker può comportare l'interrogazione di database, la lettura da file o l'accesso ad API esterne. Assicurati di considerare la latenza e come i worker AI avranno accesso ai dati più aggiornati. Hanno bisogno di accesso completo al tuo database o dovresti definire l'ambito del loro accesso in modo ristretto in base a ciò che stanno facendo? E per quanto riguarda la scalabilità? Considera i meccanismi di caching per migliorare le prestazioni e ridurre il carico sulle fonti di dati sottostanti.

Sincronizzazione dei Dati: Quando più componenti, inclusi i worker AI, accedono e modificano dati condivisi, è importante implementare meccanismi di sincronizzazione appropriati per mantenere la coerenza dei dati. Le strategie di lock del database, come il lock ottimistico o pessimistico, possono aiutarti a prevenire conflitti e garantire l'integrità dei dati. Implementa tecniche di gestione delle transazioni per raggruppare le operazioni sui dati correlate e mantenere le proprietà ACID (Atomicità, Coerenza, Isolamento e Durabilità)

Gestione degli Errori e Ripristino: Implementa meccanismi robusti di gestione degli

errori e ripristino per gestire i problemi relativi ai dati che possono verificarsi durante il processo di flusso dei dati. Gestisci le eccezioni in modo elegante e fornisci messaggi di errore significativi per facilitare il debugging. Implementa meccanismi di retry e strategie di fallback per gestire guasti temporanei o interruzioni di rete. Definisci procedure chiare per il recupero e il ripristino dei dati in caso di corruzione o perdita.

Progettando e implementando attentamente i meccanismi di flusso e sincronizzazione dei dati, puoi garantire che i tuoi worker AI abbiano accesso a dati accurati, coerenti e aggiornati. Questo permette loro di eseguire i loro compiti in modo efficace e produrre risultati affidabili.

Gestione del Ciclo di Vita dei Worker AI

Sviluppa un processo standardizzato per l'inizializzazione e la configurazione dei worker AI. Preferisco i framework che standardizzano il modo in cui definisci impostazioni come nomi dei modelli, direttive di sistema e definizioni delle funzioni. Assicurati che il processo di inizializzazione sia automatizzato e riproducibile per facilitare il deployment e la scalabilità.

Implementa meccanismi completi di monitoraggio e logging per tracciare la salute e le prestazioni dei worker AI. Raccogli metriche come l'utilizzo delle risorse, il tempo di elaborazione, i tassi di errore e il throughput. Utilizza sistemi di logging centralizzati come ELK stack (Elasticsearch, Logstash, Kibana) per aggregare e analizzare i log da più worker AI.

Incorpora tolleranza ai guasti e resilienza nell'architettura dei worker AI. Implementa meccanismi di gestione degli errori e ripristino per gestire in modo elegante i fallimenti o le eccezioni. I Modelli Linguistici di Grandi Dimensioni sono ancora una tecnologia all'avanguardia; i provider tendono a interrompersi spesso in momenti inaspettati. Utilizza meccanismi di retry e circuit breaker per prevenire fallimenti a cascata.

Componibilità e Orchestrazione dei Worker AI

Uno dei principali vantaggi dell'architettura dei worker AI è la sua componibilità, che permette di combinare e orchestrare più worker AI per risolvere problemi complessi. Suddividendo un'attività più grande in sottotask più gestibili, ciascuno gestito da un worker AI specializzato, puoi creare sistemi potenti e flessibili. In questa sezione, esploreremo diversi approcci per comporre e orchestrare “una moltitudine” di worker AI.

Concatenamento di Worker AI per Flussi di Lavoro Multi-Step

In molti scenari, un'attività complessa può essere scomposta in una serie di passaggi sequenziali, dove l'output di un worker AI diventa l'input per il successivo. Questo concatenamento di worker AI crea un flusso di lavoro o pipeline multi-step. Ogni worker AI nella catena si concentra su un sottotask specifico, e l'output finale è il risultato degli sforzi combinati di tutti i worker.

Consideriamo un esempio nel contesto di un'applicazione Ruby on Rails per l'elaborazione di contenuti generati dagli utenti. Il flusso di lavoro coinvolge i seguenti passaggi, che ammertibilmente sono probabilmente troppo semplici per valere la pena di essere decomposti in questo modo in casi d'uso reali, ma rendono l'esempio più facile da capire:

- 1. Pulizia del Testo:** Un worker AI responsabile della rimozione dei tag HTML, della conversione del testo in minuscolo e della gestione della normalizzazione Unicode.
- 2. Rilevamento della Lingua:** Un worker AI che identifica la lingua del testo pulito.
- 3. Analisi del Sentimento:** Un worker AI che determina il sentimento (positivo, negativo o neutro) del testo basandosi sulla lingua rilevata.

4. Categorizzazione dei Contenuti: Un worker AI che classifica il testo in categorie predefinite utilizzando tecniche di elaborazione del linguaggio naturale.

Ecco un esempio molto semplificato di come puoi concatenare questi worker AI utilizzando Ruby:

```
1 class ContentProcessor
2   def initialize(text)
3     @text = text
4   end
5
6   def process
7     cleaned_text = TextCleanupWorker.new(@text).call
8     language = LanguageDetectionWorker.new(cleaned_text).call
9     sentiment = SentimentAnalysisWorker.new(cleaned_text, language).call
10    category = CategorizationWorker.new(cleaned_text, language).call
11
12    { cleaned_text:, language:, sentiment:, category: }
13  end
14end
```

In questo esempio, la classe `ContentProcessor` si inizializza con il testo grezzo e concatena i worker AI insieme nel metodo `process`. Ogni worker AI esegue il proprio compito specifico e passa il risultato al worker successivo nella catena. L'output finale è un hash contenente il testo pulito, la lingua rilevata, il sentimento e la categoria del contenuto.

Elaborazione Parallela per Worker AI Indipendenti

Nell'esempio precedente, i worker AI sono concatenati in sequenza, dove ogni worker elabora il testo e passa il risultato al worker successivo. Tuttavia, se si hanno più worker AI che possono operare in modo indipendente sullo stesso input, è possibile ottimizzare il flusso di lavoro elaborandoli in parallelo.

Nello scenario dato, una volta che la pulizia del testo viene eseguita dal `TextCleanupWorker`, il `LanguageDetectionWorker`, `SentimentAnalysisWorker`

e `CategorizationWorker` possono tutti elaborare il testo pulito in modo indipendente. Eseguendo questi worker in parallelo, è possibile ridurre potenzialmente il tempo totale di elaborazione e migliorare l'efficienza del flusso di lavoro.

Per ottenere l'elaborazione parallela in Ruby, è possibile sfruttare tecniche di concorrenza come i thread o la programmazione asincrona. Ecco un esempio di come è possibile modificare la classe `ContentProcessor` per elaborare gli ultimi tre worker in parallelo utilizzando i thread:

```
1  require 'concurrent'
2
3  class ContentProcessor
4    def initialize(text)
5      @text = text
6    end
7
8    def process
9      cleaned_text = TextCleanupWorker.new(@text).call
10
11    language_future = Concurrent::Future.execute do
12      LanguageDetectionWorker.new(cleaned_text).call
13    end
14
15    sentiment_future = Concurrent::Future.execute do
16      SentimentAnalysisWorker.new(cleaned_text).call
17    end
18
19    category_future = Concurrent::Future.execute do
20      CategorizationWorker.new(cleaned_text).call
21    end
22
23    language = language_future.value
24    sentiment = sentiment_future.value
25    category = category_future.value
26
27    { cleaned_text:, language:, sentiment:, category: }
28  end
29 end
```

In questa versione ottimizzata, utilizziamo la libreria concurrent-ruby per creare oggetti `Concurrent::Future` per ciascuno dei worker AI indipendenti. [Un Future rappresenta un calcolo che verrà eseguito in modo asincrono in un thread separato.](#)

Dopo la fase di pulizia del testo, creiamo tre oggetti `Future`: `language_future`, `sentiment_future` e `category_future`. Ogni `Future` esegue il proprio worker AI corrispondente (`LanguageDetectionWorker`, `SentimentAnalysisWorker` e `CategorizationWorker`) in un thread separato, passando il `cleaned_text` come input.

Chiamando il metodo `value` su ciascun `Future`, attendiamo il completamento del calcolo e recuperiamo il risultato. Il metodo `value` blocca l'esecuzione fino a quando il risultato non è disponibile, assicurando che tutti i worker paralleli abbiano terminato l'elaborazione prima di procedere.

Infine, costruiamo l'hash di `output` con il testo pulito e i risultati dei worker paralleli, proprio come nell'esempio originale.

Elaborando i worker AI indipendenti in parallelo, è possibile ridurre potenzialmente il tempo di elaborazione complessivo rispetto all'esecuzione sequenziale. Questa ottimizzazione è particolarmente vantaggiosa quando si ha a che fare con attività che richiedono molto tempo o quando si elaborano grandi volumi di dati.

Tuttavia, è importante notare che i guadagni effettivi in termini di prestazioni dipendono da vari fattori, come la complessità di ciascun worker, le risorse di sistema disponibili e il sovraccarico della gestione dei thread. È sempre una buona pratica effettuare benchmark e profilare il codice per determinare il livello ottimale di parallelismo per il proprio caso d'uso specifico.

Inoltre, quando si implementa l'elaborazione parallela, è necessario prestare attenzione alle risorse condivise o alle dipendenze tra i worker. Assicurarsi che i worker possano operare in modo indipendente senza conflitti o condizioni di race. Se ci sono dipendenze o risorse condivise, potrebbe essere necessario implementare meccanismi

di sincronizzazioni appropriati per mantenere l'integrità dei dati ed evitare problemi come deadlock o risultati inconsistenti.

Il Global Interpreter Lock di Ruby e l'Elaborazione Asincrona

È importante comprendere le implicazioni del Global Interpreter Lock (GIL) di Ruby quando si considera l'elaborazione asincrona basata su thread in Ruby.

Il GIL è un meccanismo nell'interprete Ruby che garantisce che solo un thread possa eseguire codice Ruby alla volta, anche su processori multi-core. Ciò significa che mentre è possibile creare e gestire più thread all'interno di un processo Ruby, solo un thread può eseguire attivamente codice Ruby in un dato momento.

Il GIL è progettato per semplificare l'implementazione dell'interprete Ruby e fornire sicurezza dei thread per le strutture dati interne di Ruby. Tuttavia, limita anche il potenziale per una vera esecuzione parallela del codice Ruby.

Quando si utilizzano i thread in Ruby, come con la libreria `concurrent-ruby` o la classe `Thread` incorporata, i thread sono soggetti ai vincoli del GIL. Il GIL consente a ciascun thread di eseguire codice Ruby per una breve frazione di tempo prima di passare a un altro thread, creando l'illusione di un'esecuzione concorrente.

Tuttavia, a causa del GIL, l'esecuzione effettiva del codice Ruby rimane sequenziale. Mentre un thread sta eseguendo codice Ruby, gli altri thread sono essenzialmente in pausa, in attesa del loro turno per acquisire il GIL ed eseguire.

Ciò significa che l'elaborazione asincrona basata su thread in Ruby è più efficace per le operazioni I/O-bound, come l'attesa di risposte API esterne (come i modelli di linguaggio di grandi dimensioni ospitati da terze parti) o l'esecuzione di operazioni di I/O su file. Quando un thread incontra un'operazione di I/O, può rilasciare il GIL, permettendo ad altri thread di eseguire mentre si attende il completamento dell'I/O.

D'altra parte, per le operazioni CPU-bound, come calcoli intensivi o elaborazione di worker AI di lunga durata, il GIL può limitare i potenziali guadagni prestazionali del parallelismo basato su thread. Poiché solo un thread può eseguire codice Ruby alla volta, il tempo di esecuzione complessivo potrebbe non essere significativamente ridotto rispetto all'elaborazione sequenziale.

Per ottenere una vera esecuzione parallela per le operazioni CPU-bound in Ruby, potrebbe essere necessario esplorare approcci alternativi, come:

- Utilizzare il parallelismo basato su processi con più processi Ruby, ciascuno in esecuzione su un core CPU separato.
- Sfruttare librerie esterne o framework che forniscono estensioni native o interfacce verso linguaggi senza GIL, come C o Rust.,
- Utilizzare framework di calcolo distribuito o code di messaggi per distribuire le attività su più macchine o processi.

È fondamentale considerare la natura delle proprie attività e i limiti imposti dal GIL quando si progetta e implementa l'elaborazione asincrona in Ruby. Mentre l'elaborazione asincrona basata su thread può fornire benefici per le operazioni I/O-bound, potrebbe non offrire miglioramenti significativi delle prestazioni per le operazioni CPU-bound a causa dei vincoli del GIL.

Tecniche Ensemble per una Maggiore Accuratezza

Le tecniche ensemble comportano la combinazione degli output di più worker AI per migliorare l'accuratezza complessiva o la robustezza del sistema. Invece di affidarsi a un singolo worker AI, le tecniche ensemble sfruttano l'intelligenza collettiva di più worker per prendere decisioni più informate.



Gli ensemble sono particolarmente importanti quando diverse parti del tuo flusso di lavoro funzionano meglio con diversi modelli di AI, una situazione più comune di quanto si possa pensare. Modelli potenti come GPT-4 sono estremamente costosi rispetto alle alternative open source meno avanzate, e probabilmente non sono necessari per ogni singolo passaggio del flusso di lavoro della tua applicazione.

Una tecnica comune di ensemble è il voto di maggioranza, dove più operatori AI elaborano indipendentemente lo stesso input, e l'output finale viene determinato dal consenso della maggioranza. Questo approccio può aiutare a mitigare l'impatto degli errori dei singoli operatori e migliorare l'affidabilità complessiva del sistema.

Consideriamo un esempio in cui abbiamo tre operatori AI per l'analisi del sentimento, ciascuno che utilizza un modello diverso o fornito con contesti differenti. Possiamo combinare i loro output utilizzando il voto di maggioranza per determinare la previsione finale del sentimento.

```
1  class SentimentAnalysisEnsemble
2      def initialize(text)
3          @text = text
4      end
5
6      def analyze
7          predictions = [
8              SentimentAnalysisWorker1.new(@text).analyze,
9              SentimentAnalysisWorker2.new(@text).analyze,
10             SentimentAnalysisWorker3.new(@text).analyze
11         ]
12
13         predictions
14             .group_by { |sentiment| sentiment }
15             .max_by { |_, votes| votes.size }
16             .first
17
18     end
19 end
```

In questo esempio, la classe `SentimentAnalysisEnsemble` si inizializza con il testo e invoca tre diversi worker AI per l'analisi del sentimento. Il metodo `analyze` raccoglie le previsioni da ciascun worker e determina il sentimento maggioritario utilizzando i metodi `group_by` e `max_by`. L'output finale è il sentimento che riceve più voti dall'ensemble di worker.



Gli ensemble sono chiaramente un caso in cui vale la pena sperimentare con il parallelismo.

Selezione Dinamica e Invocazione dei Worker AI

In alcuni, se non nella maggior parte dei casi, lo specifico worker AI da invocare può dipendere dalle condizioni di runtime o dagli input dell'utente. La selezione dinamica e l'invocazione dei worker AI permettono flessibilità e adattabilità nel sistema.



Potresti essere tentato di cercare di inserire molte funzionalità in un singolo worker AI, dotandolo di numerose funzioni e un prompt complicato che spiega come utilizzarle. Resisti alla tentazione, fidati di me. Uno dei motivi per cui l'approccio che stiamo discutendo in questo capitolo si chiama "Moltitudine di Worker" è per ricordarci che è desiderabile avere molti worker specializzati, ognuno che svolge il proprio piccolo compito al servizio dello scopo più grande.

Per esempio, considera un'applicazione chatbot dove diversi worker AI sono responsabili della gestione di diversi tipi di query dell'utente. Basandosi sull'input dell'utente, l'applicazione seleziona dinamicamente il worker AI appropriato per elaborare la query.

```
1  class ChatbotController < ApplicationController
2    def process_query
3      query = params[:query]
4      query_type = QueryClassifierWorker.new(query).classify
5
6      case query_type
7      when 'greeting'
8        response = GreetingWorker.new(query).generate_response
9      when 'product_inquiry'
10        response = ProductInquiryWorker.new(query).generate_response
11      when 'order_status'
12        response = OrderStatusWorker.new(query).generate_response
13      else
14        response = DefaultResponseWorker.new(query).generate_response
15      end
16
17      render json: { response: response }
18    end
19  end
```

In questo esempio, il `ChatbotController` riceve una query dell'utente attraverso l'azione `process_query`. Per prima cosa utilizza un `QueryClassifierWorker` per determinare il tipo di query. In base al tipo di query classificata, il controller seleziona dinamicamente il worker AI appropriato per generare la risposta. Questa selezione dinamica permette al chatbot di gestire diversi tipi di query e indirizzarli ai worker AI pertinenti.



Dal momento che il lavoro del `QueryClassifierWorker` è relativamente semplice e non richiede molto contesto o definizioni di funzioni, è probabilmente possibile implementarlo utilizzando un LLM piccolo e ultra-veloce come `mistralai/mixtral-8x7b-instruct:nitro`. Ha capacità che si avvicinano al livello di GPT-4 su molte attività e, al momento in cui scrivo, Groq può elaborarlo con una velocità impressionante di 444 token al secondo.

Combinare il NLP Tradizionale con gli LLM

Mentre i Modelli Linguistici di Grandi Dimensioni (LLM) hanno rivoluzionato il campo dell'elaborazione del linguaggio naturale (NLP), offrendo versatilità e prestazioni senza precedenti in un'ampia gamma di attività, non sono sempre la soluzione più efficiente o economicamente vantaggiosa per ogni problema. In molti casi, combinare le tecniche NLP tradizionali con gli LLM può portare ad approcci più ottimizzati, mirati ed economici per risolvere specifiche sfide NLP.

Pensate agli LLM come i coltellini svizzeri del NLP: incredibilmente versatili e potenti, ma non necessariamente lo strumento migliore per ogni lavoro. A volte, uno strumento dedicato come un cavatappi o un apriscatole può essere più efficace ed efficiente per un compito specifico. Analogamente, le tecniche NLP tradizionali, come il clustering documentale, l'identificazione degli argomenti e la classificazione, possono spesso fornire soluzioni più mirate ed economicamente vantaggiose per determinati aspetti della pipeline NLP.

Uno dei principali vantaggi delle tecniche NLP tradizionali è la loro efficienza computazionale. Questi metodi, che spesso si basano su modelli statistici più semplici o approcci basati su regole, possono elaborare grandi volumi di dati testuali molto più velocemente e con un minor carico computazionale rispetto agli LLM. Questo li rende particolarmente adatti per attività che comportano l'analisi e l'organizzazione di grandi corpora di documenti, come il clustering di articoli simili o l'identificazione di argomenti chiave all'interno di una raccolta di testi.

Inoltre, le tecniche NLP tradizionali possono spesso raggiungere un'elevata accuratezza e precisione per specifiche attività, specialmente quando addestrate su dataset specifici del dominio. Per esempio, un classificatore di documenti ben calibrato che utilizza algoritmi di apprendimento automatico tradizionali come le Macchine a Vettori di Supporto (SVM) o Naive Bayes può categorizzare accuratamente i documenti in categorie predefinite con un costo computazionale minimo.

Tuttavia, gli LLM brillano veramente quando si tratta di attività che richiedono una comprensione più profonda del linguaggio, del contesto e del ragionamento. La loro capacità di generare testo coerente e contestualmente rilevante, rispondere a domande e riassumere lunghi passaggi è ineguagliata dai metodi NLP tradizionali. Gli LLM possono gestire efficacemente fenomeni linguistici complessi, come l’ambiguità, la coreferenza e le espressioni idiomatiche, rendendoli preziosi per attività che richiedono la generazione o la comprensione del linguaggio naturale.

Il vero potere risiede nel combinare le tecniche NLP tradizionali con gli LLM per creare approcci ibridi che sfruttano i punti di forza di entrambi. Utilizzando metodi NLP tradizionali per attività come il preprocessamento dei documenti, il clustering e l’estrazione di argomenti, è possibile organizzare e strutturare efficientemente i dati testuali. Queste informazioni strutturate possono poi essere fornite agli LLM per attività più avanzate, come la generazione di riassunti, la risposta a domande o la creazione di report completi.

Per esempio, consideriamo un caso d’uso in cui si desidera generare un report sulle tendenze per un dominio specifico basato su un ampio corpus di singoli documenti di tendenza. Invece di affidarsi esclusivamente agli LLM, che possono essere computazionalmente costosi e dispendiosi in termini di tempo per l’elaborazione di grandi volumi di testo, è possibile impiegare un approccio ibrido:

1. Utilizzare tecniche NLP tradizionali, come il topic modeling (ad esempio, Allocazione Latente di Dirichlet) o algoritmi di clustering (ad esempio, K-means), per raggruppare documenti di tendenza simili e identificare temi e argomenti chiave all’interno del corpus.
2. Fornire i documenti raggruppati e gli argomenti identificati a un LLM, sfruttando le sue superiori capacità di comprensione e generazione del linguaggio per creare riassunti coerenti e informativi per ogni cluster o argomento.

3. Infine, utilizzare l'LLM per generare un report completo sulle tendenze combinando i singoli riassunti, evidenziando le tendenze più significative e fornendo spunti e raccomandazioni basate sulle informazioni aggregate.

Combinando le tecniche NLP tradizionali con gli LLM in questo modo, è possibile elaborare efficientemente grandi quantità di dati testuali, estrarre informazioni significative e generare report di alta qualità ottimizzando al contempo le risorse computazionali e i costi.

Mentre si appresta ad intraprendere i Suoi progetti di NLP, è essenziale valutare attentamente i requisiti specifici e i vincoli di ogni attività e considerare come i metodi tradizionali di NLP e gli LLM possano essere sfruttati insieme per ottenere i migliori risultati. Combinando l'efficienza e la precisione delle tecniche tradizionali con la versatilità e la potenza degli LLM, Lei potrà creare soluzioni NLP altamente efficaci ed economiche che apportano valore ai Suoi utenti e alle parti interessate.

Uso degli Strumenti



Nel campo dello sviluppo di applicazioni basate sull'IA, il concetto di "uso degli strumenti" o "chiamata di funzione" si è affermato come una potente tecnica che permette al tuo LLM di connettersi a strumenti esterni, API, funzioni, basi di dati e altre risorse. Questo approccio consente un insieme di comportamenti più ricco rispetto alla semplice generazione di testo e interazioni più dinamiche tra i componenti AI e il resto dell'ecosistema della tua applicazione. Come esamineremo in questo capitolo, l'uso degli strumenti ti offre anche la possibilità di far generare al tuo modello AI i dati in modi strutturati.

Cos'è l'Uso degli Strumenti?

L'uso degli strumenti, noto anche come chiamata di funzione, è una tecnica che permette agli sviluppatori di specificare un elenco di funzioni con cui un LLM può

interagire durante il processo di generazione. Questi strumenti possono variare da semplici funzioni di utilità a complesse API o query di database. Fornendo all'LLM l'accesso a questi strumenti, gli sviluppatori possono estendere le capacità del modello e permettergli di eseguire attività che richiedono conoscenze o azioni esterne.

Figura 8. Esempio di definizione di una funzione per un worker AI che analizza documenti

```
1  FUNCTION = {
2      name: "save_analysis",
3      description: "Save analysis data for document",
4      parameters: {
5          type: "object",
6          properties: {
7              title: {
8                  type: "string",
9                  maxLength: 140
10             },
11             summary: {
12                 type: "string",
13                 description: "comprehensive multi-paragraph summary with
14                     overview and list of sections (if applicable)"
15             },
16             tags: {
17                 type: "array",
18                 items: {
19                     type: "string",
20                     description: "lowercase tags representing main themes
21                         of the document"
22                 }
23             }
24         },
25         "required": %w[title summary tags]
26     }
27 } .freeze
```

Il concetto chiave dietro l'utilizzo degli strumenti è quello di dare all'LLM la capacità di selezionare ed eseguire dinamicamente gli strumenti appropriati in base all'input dell'utente o al compito da svolgere. Invece di affidarsi esclusivamente alla conoscenza pre-addestrata del modello, l'utilizzo degli strumenti permette all'LLM di sfruttare

risorse esterne per generare risposte più accurate, pertinenti e utilizzabili. L'utilizzo degli strumenti rende tecniche come il RAG (Retrieval Augmented Generation) molto più facili da implementare di quanto lo sarebbero altrimenti.

Si noti che, se non diversamente specificato, questo libro presuppone che il tuo modello di AI non abbia accesso a strumenti integrati lato server. Qualsiasi strumento che desideri rendere disponibile alla tua AI deve essere esplicitamente dichiarato da te in ogni richiesta API, con disposizioni per la sua esecuzione se e quando la tua AI ti comunica che vorrebbe utilizzare quello strumento nella sua risposta.

Il Potenziale dell'Utilizzo degli Strumenti

L'utilizzo degli strumenti apre un'ampia gamma di possibilità per le applicazioni basate sull'AI. Ecco alcuni esempi di ciò che si può ottenere con l'utilizzo degli strumenti:

- 1. Chatbot e Assistenti Virtuali:** Collegando un LLM a strumenti esterni, i chatbot e gli assistenti virtuali possono eseguire attività più complesse, come recuperare informazioni dai database, eseguire chiamate API o interagire con altri sistemi. Per esempio, un chatbot potrebbe utilizzare uno strumento CRM per modificare lo stato di un'opportunità di vendita in base alla richiesta dell'utente.
- 2. Analisi dei Dati e Approfondimenti:** Gli LLM possono essere collegati a strumenti o librerie di analisi dei dati per eseguire attività avanzate di elaborazione dati. Questo permette alle applicazioni di generare approfondimenti, condurre analisi comparative o fornire raccomandazioni basate sui dati in risposta alle query degli utenti.
- 3. Ricerca e Recupero delle Informazioni:** L'utilizzo degli strumenti permette agli LLM di interagire con motori di ricerca, database vettoriali o altri sistemi

di recupero delle informazioni. Trasformando le query degli utenti in query di ricerca, l'LLM può recuperare informazioni pertinenti da multiple fonti e fornire risposte complete alle domande degli utenti.

4. **Integrazione con Servizi Esteri:** L'utilizzo degli strumenti permette un'integrazione fluida tra le applicazioni basate sull'AI e i servizi o le API esterne. Per esempio, un LLM potrebbe interagire con un'API meteo per fornire aggiornamenti meteorologici in tempo reale o con un'API di traduzione per generare risposte multilingue.

Il Flusso di Lavoro dell'Utilizzo degli Strumenti

Il flusso di lavoro dell'utilizzo degli strumenti tipicamente coinvolge quattro passaggi chiave:

1. Includere le definizioni delle funzioni nel contesto della richiesta
2. Selezione dinamica (o esplicita) degli strumenti
3. Esecuzione della/e funzione/i
4. Continuazione opzionale del prompt originale

Esaminiamo in dettaglio ciascuno di questi passaggi.

Includere le definizioni delle funzioni nel contesto della richiesta

L'AI sa quali strumenti ha a disposizione perché gli viene fornita una lista come parte della richiesta di completamento (tipicamente definita come funzioni utilizzando una variante dello schema JSON).

La sintassi precisa della definizione degli strumenti è specifica del modello.

Questo è come si definisce una funzione `get_weather` in Claude 3:

```
1  {
2      "name": "get_weather",
3      "description": "Get the current weather in a given location",
4      "input_schema": {
5          "type": "object",
6          "properties": {
7              "location": {
8                  "type": "string",
9                  "description": "The city and state, e.g. San Francisco, CA"
10             },
11             "unit": {
12                 "type": "string",
13                 "enum": ["celsius", "fahrenheit"],
14                 "description": "The unit of temperature"
15             }
16         },
17         "required": ["location"]
18     }
19 }
```

Ed ecco come definiresti la stessa funzione per GPT-4, passandola come valore del parametro `tools`:

```
1  {
2      "name": "get_current_weather",
3      "description": "Get the current weather in a given location",
4      "parameters": {
5          "type": "object",
6          "properties": {
7              "location": {
8                  "type": "string",
9                  "description": "The city and state, e.g. San Francisco, CA",
10             },
11             "unit": {
12                 "type": "string",
13                 "enum": ["celsius", "fahrenheit"],
14                 "description": "The unit of temperature"
15             }
16         },
17         "required": ["location"],
```

```
18     } ,  
19 }
```

Quasi uguale, ma diverso senza alcun motivo apparente! Che fastidio.

Le definizioni delle funzioni specificano nome, descrizione e parametri di input. I parametri di input possono essere ulteriormente definiti utilizzando attributi come le enumerazioni per limitare i valori accettabili, e specificando se un parametro è obbligatorio o meno.

Oltre alle effettive definizioni delle funzioni, puoi anche includere istruzioni o contesto sul perché e come utilizzare la funzione nella direttiva di sistema.

Per esempio, il mio strumento di ricerca Web in Olympia include questa direttiva di sistema, che ricorda all'IA che ha a disposizione gli strumenti menzionati:

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

Fornire descrizioni dettagliate è considerato il fattore più importante per le prestazioni degli strumenti. Le tue descrizioni dovrebbero spiegare ogni dettaglio dello strumento, incluso:

- Cosa fa lo strumento
- Quando dovrebbe essere utilizzato (e quando non dovrebbe)
- Cosa significa ogni parametro e come influenza il comportamento dello strumento
- Eventuali avvertenze o limitazioni importanti che si applicano all'implementazione dello strumento

Più contesto puoi fornire all'IA riguardo ai tuoi strumenti, migliore sarà la sua capacità di decidere quando e come utilizzarli. Per esempio, Anthropic raccomanda almeno 3-4 frasi per descrizione di ogni strumento per la sua serie Claude 3, di più se lo strumento è complesso.

Non è necessariamente intuitivo, ma le descrizioni sono considerate più importanti degli esempi. Mentre puoi includere esempi di come utilizzare uno strumento nella sua descrizione o nel prompt di accompagnamento, questo è meno importante dell'avere una spiegazione chiara e completa dello scopo e dei parametri dello strumento. Aggiungi gli esempi solo dopo aver sviluppato completamente la descrizione.

Ecco un esempio di specifica di funzione API in stile Stripe:

```
1  {
2      "name": "createPayment",
3      "description": "Create a new payment request",
4      "parameters": {
5          "type": "object",
6          "properties": {
7              "transaction_amount": {
8                  "type": "number",
9                  "description": "The amount to be paid"
10             },
11             "description": {
12                 "type": "string",
13                 "description": "A brief description of the payment"
14             },
15             "payment_method_id": {
16                 "type": "string",
17                 "description": "The payment method to be used"
18             },
19             "payer": {
20                 "type": "object",
21                 "description": "Information about the payer, including their name,
22                               email, and identification number",
23                 "properties": {
24                     "name": {
25                         "type": "string",
26                         "description": "The payer's name"
27                 }
28             }
29         }
30     }
31 }
```

```
27 },
28 "email": {
29   "type": "string",
30   "description": "The payer's email address"
31 },
32 "identification": {
33   "type": "object",
34   "description": "The payer's identification number",
35   "properties": {
36     "type": {
37       "type": "string",
38       "description": "Identification document (e.g. CPF, CNPJ)"
39     },
40     "number": {
41       "type": "string",
42       "description": "The identification number"
43     }
44   },
45   "required": [ "type", "number" ]
46 }
47 },
48 "required": [ "name", "email", "identification" ]
49 }
50 }
51 }
```



In pratica, alcuni modelli hanno difficoltà a gestire le specifiche di funzioni annidate e a trattare tipi di dati di output complessi come array, dizionari ecc. Ma in teoria, dovresti essere in grado di fornire specifiche JSON Schema di profondità arbitraria!

Selezione Dinamica degli Strumenti

Quando esegui un completamento di chat che include definizioni di strumenti, l'LLM seleziona dinamicamente lo strumento o gli strumenti più appropriati da utilizzare e genera i parametri di input necessari per ciascuno strumento.

In pratica, la capacità dell'IA di chiamare *esattamente* la funzione giusta e di seguire *esattamente* le tue specifiche per gli input è altalenante. Abbassare il parametro di temperatura completamente a 0.0 aiuta molto, ma dalla mia esperienza si verificheranno ancora errori occasionali. Questi fallimenti includono nomi di funzioni allucinati, parametri di input errati o semplicemente mancanti. I parametri vengono passati come JSON, il che significa che a volte si vedranno errori causati da JSON troncati, con citazioni errate o comunque danneggiati.



I pattern di [Auto-riparazione dei Dati](#) possono aiutare a [correggere automaticamente](#) le chiamate di funzione che si interrompono a causa di errori di sintassi.

Selezione Forzata (o Esplicita) degli Strumenti

Alcuni modelli offrono l'opzione di forzare la chiamata di una particolare funzione, come parametro nella richiesta. Altrimenti, la decisione di chiamare o meno la funzione è interamente a discrezione dell'IA.

La capacità di forzare una chiamata di funzione è cruciale in determinati scenari in cui si desidera garantire che uno specifico strumento o funzione venga eseguito, indipendentemente dal processo di selezione dinamica dell'IA. Ci sono diverse ragioni per cui questa capacità è importante:

1. **Controllo Esplicito:** Potresti utilizzare l'IA come *Componente Discreto* o in un flusso di lavoro predefinito che richiede l'esecuzione di una particolare funzione in un momento specifico. Forzando la chiamata, puoi garantire che la funzione desiderata venga invocata invece di dover chiedere gentilmente all'IA di farlo.
2. **Debug e Testing:** Durante lo sviluppo e il testing di applicazioni basate su IA, la capacità di forzare le chiamate di funzione è inestimabile per scopi di debugging. Attivando esplicitamente funzioni specifiche, puoi isolare e testare

singoli componenti della tua applicazione. Questo ti permette di verificare la correttezza delle implementazioni delle funzioni, validare i parametri di input e assicurarti che vengano restituiti i risultati attesi.

3. **Gestione dei Casi Limite:** Potrebbero esserci casi limite o scenari eccezionali in cui il processo di selezione dinamica dell'IA potrebbe non scegliere di eseguire una funzione che dovrebbe, e lo sai in base a processi esterni. In tali casi, avere la capacità di forzare una chiamata di funzione ti permette di gestire esplicitamente queste situazioni. Definisci regole o condizioni nella logica della tua applicazione per determinare quando bypassare la discrezione dell'IA.
4. **Consistenza e Riproducibilità:** Se hai una sequenza specifica di funzioni che devono essere eseguite in un ordine particolare, forzare le chiamate garantisce che la stessa sequenza venga seguita ogni volta. Questo è particolarmente importante nelle applicazioni dove la consistenza e il comportamento prevedibile sono critici, come nei sistemi finanziari o nelle simulazioni scientifiche.
5. **Ottimizzazione delle Prestazioni:** In alcuni casi, forzare una chiamata di funzione può portare a ottimizzazioni delle prestazioni. Se sai che una specifica funzione è necessaria per un particolare compito e che il processo di selezione dinamica dell'IA potrebbe introdurre overhead non necessario, puoi bypassare il processo di selezione e invocare direttamente la funzione richiesta. Questo può aiutare a ridurre la latenza e migliorare l'efficienza complessiva della tua applicazione.

In sintesi, la capacità di forzare le chiamate di funzione nelle applicazioni basate su IA fornisce controllo esplicito, aiuta nel debug e nel testing, gestisce i casi limite, garantisce consistenza e riproducibilità. È uno strumento potente nel tuo arsenale, ma dobbiamo discutere un altro aspetto di questa importante funzionalità.



In molti casi d'uso decisionali, vogliamo sempre che il modello effettui una chiamata di funzione e potremmo non volere mai che il modello risponda solo con la sua conoscenza interna. Per esempio, se stai instradando tra più modelli specializzati in diversi compiti (input multilingue, matematica, ecc.), potresti utilizzare il modello di chiamata di funzione per delegare le richieste a uno dei modelli di supporto e non rispondere mai in modo indipendente.

Parametro di Scelta dello Strumento

GPT-4 e altri modelli linguistici che supportano la chiamata di funzioni ti forniscono un parametro `tool_choice` per controllare se l'uso dello strumento è richiesto come parte di un completamento. Questo parametro ha tre possibili valori:

- `auto` dà all'IA piena discrezione sull'utilizzo di uno strumento o semplicemente sulla risposta
- `required` dice all'IA che *deve* chiamare uno strumento *invece* di rispondere, ma lascia la selezione dello strumento all'IA
- La terza opzione è impostare il parametro del `name_of_function` che vuoi forzare. Ne parleremo più dettagliatamente nella prossima sezione.



Nota che se imposta `tool_choice` su `required`, il modello sarà costretto a scegliere la funzione più pertinente da chiamare tra quelle fornite, anche se nessuna si adatta veramente al prompt. Al momento della pubblicazione, non sono a conoscenza di alcun modello che restituisca una risposta `tool_calls` vuota, o utilizzi qualche altro modo per far sapere che non ha trovato una funzione adatta da chiamare.

Forzare una Funzione per Ottenere Output Strutturato

La capacità di forzare una chiamata di funzione ti offre un modo per ottenere dati strutturati da un completamento chat invece di doverli estrarre tu stesso dalla risposta in testo normale.

Perché forzare le funzioni per ottenere output strutturato è così importante? In poche parole, perché l'estrazione di dati strutturati dall'output di un LLM è un vero grattacapo. Puoi semplificarti un po' la vita chiedendo i dati in XML, ma poi devi analizzare l'XML. E cosa fai quando quell'XML manca perché la tua AI ha risposto: "Mi dispiace, ma non posso generare i dati richiesti perché bla, bla, bla..."

Quando usi gli strumenti in questo modo:

- Dovresti probabilmente definire un singolo strumento nella tua richiesta
- Ricorda di forzare l'uso della sua funzione utilizzando il parametro `tool_choice`
- Ricorda che il modello passerà l'input allo strumento, quindi il nome dello strumento e la descrizione dovrebbero essere dalla prospettiva del modello, non dalla tua.

Quest'ultimo punto merita un esempio per chiarezza. Supponiamo che tu stia chiedendo all'AI di fare un'analisi del sentimento sul testo dell'utente. Il nome della funzione non sarebbe `analyze_sentiment`, ma piuttosto qualcosa come `save_sentiment_analysis`. È l'AI che sta facendo l'analisi del sentimento, *non lo strumento*. Tutto ciò che lo strumento sta facendo (dalla prospettiva dell'AI) è salvare i risultati dell'analisi.

Ecco un esempio di utilizzo di Claude 3 per registrare un riepilogo di un'immagine in JSON ben strutturato, questa volta dalla riga di comando usando `curl`:

```
1 curl https://api.anthropic.com/v1/messages \
2   --header "content-type: application/json" \
3   --header "x-api-key: $ANTHROPIC_API_KEY" \
4   --header "anthropic-version: 2023-06-01" \
5   --header "anthropic-beta: tools-2024-04-04" \
6   --data \
7   '{
8     "model": "claude-3-sonnet-20240229",
9     "max_tokens": 1024,
10    "tools": [
11      {
12        "name": "record_summary",
13        "description": "Record summary of image into well-structured JSON.",
14        "input_schema": {
15          "type": "object",
16          "properties": {
17            "key_colors": {
18              "type": "array",
19              "items": {
20                "type": "object",
21                "properties": {
22                  "r": {
23                    "type": "number",
24                    "description": "red value [0.0, 1.0]"
25                  },
26                  "g": {
27                    "type": "number",
28                    "description": "green value [0.0, 1.0]"
29                  },
30                  "b": {
31                    "type": "number",
32                    "description": "blue value [0.0, 1.0]"
33                  },
34                  "name": {
35                    "type": "string",
36                    "description": "Human-readable color name
37                               in snake_case, e.g.
38                               \"olive_green\" or
39                               \"turquoise\""
40                  }
41                },
42                "required": [ "r", "g", "b", "name" ]
43              },
44            }
45          }
46        }
47      }
48    ],
49    "prompt": "Summarize the image in JSON format."}
```

```

43         "description": "Key colors in the image. Four or less."
44     },
45     "description": {
46         "type": "string",
47         "description": "Image description. 1-2 sentences max."
48     },
49     "estimated_year": {
50         "type": "integer",
51         "description": "Estimated year that the image was taken,
52                         if is it a photo. Only set this if the
53                         image appears to be non-fictional.
54                         Rough estimates are okay!"
55     }
56 },
57     "required": [ "key_colors", "description" ]
58 }
59 ],
60     "messages": [
61     {
62         "role": "user",
63         "content": [
64             {
65                 "type": "image",
66                 "source": {
67                     "type": "base64",
68                     "media_type": "$IMAGE_MEDIA_TYPE",
69                     "data": "$IMAGE_BASE64"
70                 }
71             },
72             {
73                 "type": "text",
74                 "text": "Use `record_summary` to describe this image."
75             }
76         ]
77     }
78 ]
79 }'

```

Nell'esempio fornito, stiamo utilizzando il modello Claude 3 di Anthropic per generare un riepilogo JSON strutturato di un'immagine. Ecco come funziona:

1. Definiamo un singolo strumento chiamato `record_summary` nell'array `tools` del payload della richiesta. Questo strumento è responsabile della registrazione di un riepilogo dell'immagine in formato JSON ben strutturato.
2. Lo strumento `record_summary` ha un `input_schema` che specifica la struttura prevista dell'output JSON. Definisce tre proprietà:
 - `key_colors`: Un array di oggetti che rappresentano i colori chiave nell'immagine. Ogni oggetto colore ha proprietà per i valori rosso, verde e blu (che vanno da 0.0 a 1.0) e un nome di colore leggibile in formato `snake_case`.
 - `description`: Una proprietà stringa per una breve descrizione dell'immagine, limitata a 1-2 frasi.
 - `estimated_year`: Una proprietà intera opzionale per l'anno stimato in cui è stata scattata l'immagine, se sembra essere una foto non fittizia.
3. Nell'array `messages`, forniamo i dati dell'immagine come stringa codificata in `base64` insieme al tipo di media. Questo permette al modello di elaborare l'immagine come parte dell'input.
4. Inoltre, chiediamo a Claude di utilizzare lo strumento `record_summary` per descrivere l'immagine.
5. Quando la richiesta viene inviata al modello Claude 3, questo analizza l'immagine e genera un riepilogo JSON basato sull'`input_schema` specificato. Il modello estrae i colori chiave, fornisce una breve descrizione e stima l'anno in cui l'immagine è stata scattata (se applicabile).
6. Il riepilogo JSON generato viene passato come parametri allo strumento `record_summary`, fornendo una rappresentazione strutturata delle caratteristiche chiave dell'immagine.

Utilizzando lo strumento `record_summary` con un `input_schema` ben definito, possiamo ottenere un riepilogo JSON strutturato di un'immagine senza fare affidamento

sull'estrazione di testo semplice. Questo approccio garantisce che l'output segua un formato coerente e possa essere facilmente analizzato ed elaborato dai componenti a valle dell'applicazione.

La capacità di forzare una chiamata di funzione e specificare la struttura dell'output prevista è una potente caratteristica dell'utilizzo degli strumenti nelle applicazioni basate sull'IA. Permette agli sviluppatori di avere maggior controllo sull'output generato e semplifica l'integrazione dei dati generati dall'IA nel workflow della loro applicazione.

Esecuzione delle Funzioni

Hai definito le funzioni e fornito indicazioni alla tua IA, che ha deciso di chiamare una delle tue funzioni. Ora è il momento per il tuo codice applicativo o libreria, se stai utilizzando una gemma Ruby come [raix-rails](#), di inoltrare la chiamata di funzione e i suoi parametri all'implementazione corrispondente *nel tuo codice applicativo*.

Il tuo codice applicativo decide cosa fare con i risultati dell'esecuzione della funzione. Forse si tratta di una singola riga di codice in una lambda, o forse implica la chiamata a un'API esterna. Potrebbe coinvolgere la chiamata a un altro componente IA, o potrebbe coinvolgere centinaia o addirittura migliaia di righe di codice nel resto del tuo sistema. Sta interamente a te.

A volte la chiamata di funzione è la fine dell'operazione, ma se i risultati rappresentano informazioni in una catena di pensiero che deve essere continuata dall'IA, allora il tuo codice applicativo deve inserire i risultati dell'esecuzione nella trascrizione della chat e lasciare che l'IA continui l'elaborazione.

Per esempio, ecco una dichiarazione di funzione [Raix](#) utilizzata dall'AccountManager di Olympia per comunicare con i nostri clienti come parte di un'Orchestrazione Intelligente del Workflow per il servizio clienti.

```
1  class AccountManager
2    include Raix::ChatCompletion
3    include Raix::FunctionDispatch
4
5    # lots of other functions...
6
7    function :notify_account_owner,
8      "Don't share UUID. Mention dollars if subscription changed",
9      message: { type: "string" } do |arguments|
10      account.owner.freeform_notify(
11        subject: "Account Change Notification",
12        message: arguments[:message]
13      )
14      "Notified account owner"
15    end
```

Potrebbe non essere immediatamente chiaro cosa sta succedendo qui, quindi lo spiegherò nel dettaglio.

1. La classe `AccountManager` definisce molte funzioni relative alla gestione degli account. Può modificare il tuo piano, aggiungere e rimuovere membri del team, tra le altre cose.
2. Le sue istruzioni di alto livello indicano ad `AccountManager` che dovrebbe notificare al proprietario dell'account i risultati della richiesta di modifica dell'account, utilizzando la funzione `notify_account_owner`.
3. La definizione concisa della funzione include:
 - nome
 - descrizione
 - parametri `message: { type: "string" }`
 - un blocco da eseguire quando la funzione viene chiamata

Dopo aver aggiornato la trascrizione con i risultati del blocco di funzioni, viene chiamato nuovamente il metodo `chat_completion`. Questo metodo è responsabile dell'invio

della trascrizione aggiornata della conversazione al modello AI per un'ulteriore elaborazione. Ci riferiamo a questo processo come *ciclo di conversazione*.

Quando il modello AI riceve una nuova richiesta di completamento della chat con una trascrizione aggiornata, ha accesso ai risultati della funzione precedentemente eseguita. Può analizzare questi risultati, incorporarli nel suo processo decisionale e generare la prossima risposta o azione basandosi sul contesto cumulativo della conversazione. Può scegliere di eseguire funzioni aggiuntive basate sul contesto aggiornato, oppure può generare una risposta finale al prompt originale se determina che non sono necessarie ulteriori chiamate di funzione.

Continuazione Opzionale del Prompt Originale

Quando invii i risultati degli strumenti all'LLM e continui l'elaborazione del prompt originale, l'AI utilizza quei risultati per chiamare funzioni aggiuntive o generare una risposta finale in testo semplice.



Alcuni modelli come il [Command-R](#) di Cohere possono citare gli strumenti specifici che hanno utilizzato nelle loro risposte, fornendo ulteriore trasparenza e tracciabilità.

A seconda del modello in uso, i risultati della chiamata di funzione si troveranno nei messaggi della trascrizione che hanno il loro ruolo speciale o saranno riflessi in qualche altra sintassi. Ma la parte importante è che quei dati siano nella trascrizione, in modo che possano essere considerati dall'AI mentre decide cosa fare successivamente.



Un errore comune (e potenzialmente costoso) è dimenticare di aggiungere i risultati della funzione alla trascrizione prima di continuare la chat. Di conseguenza, l'AI verrà sollecitata essenzialmente nello stesso modo in cui era stata sollecitata prima di chiamare la funzione la prima volta. In altre parole, per quanto riguarda l'AI, non ha ancora chiamato la funzione. Quindi la chiama di nuovo. E ancora. E ancora, all'infinito finché non la interrompi. Speriamo che il tuo contesto non fosse troppo grande e il tuo modello non troppo costoso!

Migliori Pratiche per l'Uso degli Strumenti

Per ottenere il massimo dall'uso degli strumenti, considera le seguenti migliori pratiche.

Definizioni Descrittive

Fornisci nomi e descrizioni chiari e descrittivi per ogni strumento e i suoi parametri di input. Questo aiuta l'LLM a comprendere meglio lo scopo e le capacità di ogni strumento.

Posso dirvi per esperienza che la saggezza comune che dice che “dare nomi è difficile” si applica qui; ho visto risultati drasticamente diversi dagli LLM semplicemente cambiando i nomi delle funzioni o la formulazione delle descrizioni. A volte rimuovere le descrizioni *migliora* le prestazioni.

Elaborazione dei Risultati degli Strumenti

Quando passi i risultati degli strumenti all'LLM, assicurati che siano ben strutturati e completi. Usa chiavi e valori significativi per rappresentare l'output di ogni strumento. Sperimenta con diversi formati e vedi quale funziona meglio, dal JSON al testo semplice.

Il [Result Interpreter](#) affronta questa sfida impiegando l'AI per analizzare i risultati e fornire spiegazioni comprensibili, riassunti o punti chiave.

Gestione degli Errori

Implementa robusti meccanismi di gestione degli errori per gestire i casi in cui l'LLM potrebbe generare parametri di input non validi o non supportati per le chiamate degli strumenti. Gestisci e recupera con eleganza da qualsiasi errore che potrebbe verificarsi durante l'esecuzione degli strumenti.

Una qualità straordinariamente positiva dell'AI è che comprende i messaggi di errore! Il che significa che se stai lavorando con una mentalità rapida e approssimativa, puoi semplicemente catturare qualsiasi eccezione generata nell'implementazione di uno strumento e passarla all'AI in modo che sappia cosa è successo!

Per esempio, ecco una versione semplificata dell'implementazione della ricerca Google in Olympia:

```
1  def google_search(conversation, params)
2      conversation.update_cstatus("Searching Google...")
3      query = params[:query]
4      search = GoogleSearch.new(query).get_hash
5
6      conversation.update_cstatus("Summarizing results...")
7      SummarizeKnowledgeGraph.new.perform(conversation, search.to_json)
8  rescue StandardError => e
9      Honeybadger.notify(e)
10     { error: e.message }.inspect
11 end
```

Le ricerche Google in Olympia sono un processo in due fasi. Prima si esegue la ricerca, poi si riassumono i risultati. In caso di errore, qualunque esso sia, il messaggio di eccezione viene impacchettato e rinvia all'IA. Questa tecnica è alla base di praticamente tutti i modelli di *Gestione Intelligente degli Errori*

Per esempio, supponiamo che la chiamata API GoogleSearch fallisca a causa di un’eccezione 503 Servizio Non Disponibile. Questa risale fino al rescue di livello superiore, e la descrizione dell’errore viene inviata all’IA come risultato della chiamata della funzione. Invece di mostrare all’utente una schermata vuota o un errore tecnico, l’IA dice qualcosa del tipo “Mi dispiace, ma al momento non riesco ad accedere alle mie funzionalità di Ricerca Google. Posso riprovare più tardi, se lo desideri.”

Questo potrebbe sembrare solo un trucco intelligente, ma considera un tipo diverso di errore, uno in cui l’IA stava chiamando un’API esterna e aveva il controllo diretto dei parametri da passare all’API. Forse ha commesso un errore nel modo in cui ha generato quei parametri? A condizione che il messaggio di errore dall’API esterna sia sufficientemente dettagliato, restituire il messaggio di errore all’IA chiamante significa che può riconsiderare quei parametri e riprovare. Automaticamente. Non importa quale fosse l’errore.

Ora pensa a cosa servirebbe per replicare questo tipo di gestione robusta degli errori nel codice *normale*. È praticamente impossibile.

Raffinamento Iterativo

Se l’LLM non sta raccomandando gli strumenti appropriati o sta generando risposte subottimali, itera sulle definizioni degli strumenti, le descrizioni e i parametri di input. Continua a raffinare e migliorare la configurazione degli strumenti basandoti sul comportamento osservato e sui risultati desiderati.

1. Inizia con definizioni semplici degli strumenti: Comincia definendo strumenti con nomi, descrizioni e parametri di input chiari e concisi. Evita inizialmente di complicare eccessivamente la configurazione degli strumenti e concentrati sulla funzionalità principale. Per esempio, se vuoi salvare i risultati dell’analisi del sentimento, inizia con una definizione base come:

```

1  {
2    "name": "save_sentiment_score",
3    "description": "Analyze user-provided text and generate sentiment score",
4    "parameters": {
5      "type": "object",
6      "properties": {
7        "score": {
8          "type": "float",
9          "description": "sentiment score from -1 (negative) to 1 (positive)"
10       }
11     },
12     "required": ["score"]
13   }
14 }
```

2. Testare e osservare: Una volta definiti gli strumenti iniziali, testarli con diversi prompt e osservare come l'LLM interagisce con lo strumento. Prestare attenzione alla qualità e alla pertinenza delle risposte generate. Se l'LLM sta generando risposte subottimali, è il momento di perfezionare le definizioni degli strumenti.
3. Perfezionare le descrizioni: Se l'LLM non comprende correttamente lo scopo di uno strumento, provare a perfezionare la descrizione dello strumento. Fornire più contesto, esempi o chiarimenti per guidare l'LLM nell'utilizzo efficace dello strumento. Per esempio, è possibile aggiornare la descrizione dello strumento di analisi del sentimento per affrontare più specificamente il *tono emotivo* del testo analizzato:

```

1  {
2    "name": "save_sentiment_score",
3    "description": "Determine the overall emotional tone of a piece of text,
4    such as customer reviews, social media posts, or feedback comments.",
5    ...
6  }
```

4. Regolare i parametri di input: Se l'LLM genera parametri di input non validi o irrilevanti per uno strumento, considera di modificare le definizioni dei parametri.

Aggiungi vincoli più specifici, regole di validazione o esempi per chiarire il formato di input previsto.

5. Iterare in base al feedback: Monitora continuamente le prestazioni dei tuoi strumenti e raccogli feedback dagli utenti o dalle parti interessate. Utilizza questo feedback per identificare le aree di miglioramento e apportare raffinamenti iterativi alle definizioni degli strumenti. Per esempio, se gli utenti segnalano che l'analisi non gestisce bene il sarcasmo, puoi aggiungere una nota nella descrizione:

```
1  {
2    "name": "save_sentiment_score",
3    "description": "Analyze the sentiment of a given text and return a sentiment
4      score between -1 (negative) and 1 (positive). Note: Sarcasm should be
5      considered negative.",
6      ...
7  }
```

Perfezionando iterativamente le definizioni degli strumenti in base al comportamento osservato e al feedback, è possibile migliorare gradualmente le prestazioni e l'efficacia della propria applicazione basata su IA. Ricorda di mantenere le definizioni degli strumenti chiare, concise e focalizzate sul compito specifico. Testa e convalida regolarmente le interazioni degli strumenti per assicurarti che si allineino con i risultati desiderati.

Composizione e Concatenamento degli Strumenti

Uno degli aspetti più potenti dell'uso degli strumenti, finora solo accennato, è la capacità di comporre e concatenare più strumenti insieme per realizzare compiti complessi. Progettando attentamente le definizioni degli strumenti e i loro formati di input/output, puoi creare componenti di base riutilizzabili che possono essere combinati in vari modi.

Consideriamo un esempio in cui stai costruendo una pipeline di analisi dei dati per la tua applicazione basata su IA. Potresti avere i seguenti strumenti:

1. **DataRetrieval**: Uno strumento che recupera dati da un database o API in base a criteri specifici.
2. **DataProcessing**: Uno strumento che esegue calcoli, trasformazioni o aggregazioni sui dati recuperati.
3. **DataVisualization**: Uno strumento che presenta i dati elaborati in un formato user-friendly, come grafici o diagrammi.

Concatenando questi strumenti insieme, puoi creare un potente flusso di lavoro che recupera i dati pertinenti, li elabora e presenta i risultati in modo significativo. Ecco come potrebbe apparire il flusso di lavoro dell'uso degli strumenti:

1. L'LLM riceve una query dell'utente che richiede informazioni sui dati di vendita per una specifica categoria di prodotti.
2. L'LLM seleziona lo strumento **DataRetrieval** e genera i parametri di input appropriati per recuperare i dati di vendita pertinenti dal database.
3. I dati recuperati vengono "passati" allo strumento **DataProcessing**, che calcola metriche come il ricavo totale, il prezzo medio di vendita e il tasso di crescita.
4. I dati elaborati vengono quindi utilizzati dallo strumento **DataVisualization**, che crea un grafico o diagramma visivamente accattivante per rappresentare le informazioni, restituendo l'URL del grafico all'LLM.
5. Infine, l'LLM genera una risposta formattata alla query dell'utente utilizzando markdown, incorporando i dati visualizzati e fornendo un riepilogo dei risultati chiave.

Componendo questi strumenti insieme, puoi creare un flusso di lavoro di analisi dei dati uniforme che può essere facilmente integrato nella tua applicazione. La bellezza

di questo approccio sta nel fatto che ogni strumento può essere sviluppato e testato indipendentemente, e poi combinato in modi diversi per risolvere vari problemi.

Per consentire una composizione e un concatenamento fluidi degli strumenti, è importante definire formati di input e output chiari per ciascuno strumento.

Per esempio, lo strumento `DataRetrieval` potrebbe accettare parametri come i dettagli di connessione al database, il nome della tabella e le condizioni di query, e restituire il set di risultati come un oggetto JSON strutturato. Lo strumento `DataProcessing` può quindi aspettarsi questo oggetto JSON come input e produrre un oggetto JSON trasformato come output. Standardizzando il flusso di dati tra gli strumenti, puoi garantire compatibilità e riutilizzabilità.

Mentre progetti il tuo ecosistema di strumenti, pensa a come diversi strumenti possono essere combinati per affrontare casi d'uso comuni nella tua applicazione. Considera di creare strumenti di alto livello che encapsulino flussi di lavoro comuni o logica di business, rendendo più facile per l'LLM selezionarli e utilizzarli efficacemente.

Ricorda, la potenza dell'uso degli strumenti risiede nella flessibilità e modularità che fornisce. Scomponendo compiti complessi in strumenti più piccoli e riutilizzabili, puoi creare un'applicazione basata su IA robusta e adattabile che può affrontare un'ampia gamma di sfide.

Direzioni Future

Man mano che il campo dello sviluppo di applicazioni basate su IA si evolve, possiamo aspettarci ulteriori progressi nelle capacità di utilizzo degli strumenti. Alcune possibili direzioni future includono:

- 1. Utilizzo di Strumenti Multi-hop:** Gli LLM potrebbero essere in grado di decidere quante volte devono utilizzare gli strumenti per generare una risposta soddisfacente. Questo potrebbe coinvolgere più cicli di selezione ed esecuzione degli strumenti basati su risultati intermedi.

2. **Strumenti Predefiniti:** Le piattaforme di IA potrebbero fornire un set di strumenti predefiniti che gli sviluppatori possono sfruttare immediatamente, come interpreti Python, strumenti di ricerca web o funzioni di utilità comuni.
3. **Integrazione Perfetta:** Man mano che l'uso degli strumenti diventa più diffuso, possiamo aspettarci una migliore integrazione tra le piattaforme di IA e i framework di sviluppo popolari, rendendo più facile per gli sviluppatori incorporare l'uso degli strumenti nelle loro applicazioni.

L'uso degli strumenti è una tecnica potente che permette agli sviluppatori di sfruttare il pieno potenziale degli LLM nelle applicazioni basate su IA. Collegando gli LLM a strumenti e risorse esterni, puoi creare sistemi più dinamici, intelligenti e consapevoli del contesto che possono adattarsi alle esigenze degli utenti e fornire preziose informazioni e azioni.

Mentre l'uso degli strumenti offre immense possibilità, è importante essere consapevoli delle potenziali sfide e considerazioni. Un aspetto chiave è la gestione della complessità delle interazioni tra gli strumenti e garantire la stabilità e l'affidabilità del sistema complessivo. È necessario gestire scenari in cui le chiamate agli strumenti potrebbero fallire, restituire risultati inaspettati o avere implicazioni sulle prestazioni. Inoltre, dovresti considerare misure di sicurezza e controllo degli accessi per prevenire l'uso non autorizzato o malevolo degli strumenti. Meccanismi appropriati di gestione degli errori, logging e monitoraggio sono cruciali per mantenere l'integrità e le prestazioni della tua applicazione basata su IA.

Mentre esplori le possibilità dell'utilizzo degli strumenti nei tuoi progetti, ricorda di iniziare con obiettivi chiari, progettare definizioni degli strumenti ben strutturate e iterare sulla base del feedback e dei risultati. Con l'approccio e la mentalità giusti, l'utilizzo degli strumenti può sbloccare nuovi livelli di innovazione e valore nelle tue applicazioni basate sull'IA.

Elaborazione del Flusso



Lo streaming dei dati su HTTP, noto anche come eventi inviati dal server (SSE), è un meccanismo in cui il server invia continuamente dati al client man mano che questi diventano disponibili, senza che il client debba richiederli esplicitamente. Dato che la risposta dell'IA viene generata in modo incrementale, ha senso fornire un'esperienza utente reattiva mostrando l'output dell'IA mentre viene generato. E in effetti tutte le API dei provider di IA che conosco offrono risposte in streaming come opzione nei loro endpoint di completamento.

Il motivo per cui questo capitolo appare qui nel libro, subito dopo [Utilizzo degli Strumenti](#), è dovuto a quanto può essere potente combinare l'uso degli strumenti con le risposte dell'IA in tempo reale agli utenti. Questo permette di creare esperienze dinamiche e interattive in cui l'IA può elaborare l'input dell'utente, utilizzare vari strumenti e funzioni a sua discrezione e fornire risposte in tempo reale.

Per ottenere questa interazione fluida, è necessario scrivere gestori del flusso che possano gestire sia le chiamate alle funzioni strumentali invocate dall'IA sia l'output in testo semplice per l'utente finale. La necessità di eseguire un ciclo dopo l'elaborazione di una funzione strumentale aggiunge una sfida interessante al lavoro.

Implementazione di un ReplyStream

Per dimostrare come può essere implementata l'elaborazione del flusso, questo capitolo esaminerà in dettaglio una versione semplificata della classe `ReplyStream` utilizzata in Olympia. Le istanze di questa classe possono essere passate come parametro `stream` nelle librerie client IA come `ruby-openai` e `openrouter`

Ecco come utilizzo `ReplyStream` nel `PromptSubscriber` di Olympia, che ascolta tramite Wisper la creazione di nuovi messaggi dell'utente.

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

Oltre a un riferimento `context` al sottoscrittore del prompt che l'ha istanziata, la classe `ReplyStream` dispone anche di variabili di istanza per memorizzare un buffer di dati ricevuti e array per tenere traccia dei nomi delle funzioni e degli argomenti invocati durante l'elaborazione dello stream.

```
1  class ReplyStream
2    attr_accessor :buffer, :f_name, :f_arguments, :context
3
4    delegate :bot_message, :dispatch, to: :context
5
6    def initialize(context)
7      self.context = context
8      self.buffer = []
9      self.f_name = []
10     self.f_arguments = []
11   end
12
13  def call(chunk, bytesize = nil)
14    # ...
15  end
16
17  # ...
18 end
```

Il metodo `initialize` imposta lo stato iniziale dell'istanza `ReplyStream`, inizializzando il buffer, il contesto e altre variabili.

Il metodo `call` è il punto di ingresso principale per l'elaborazione dei dati in streaming. Accetta un `chunk` di dati (rappresentato come un hash) e un parametro opzionale `bytesize`, che nel nostro esempio non viene utilizzato. All'interno di questo metodo, la classe utilizza il pattern matching per gestire diversi scenari basati sulla struttura del `chunk` ricevuto.



Chiamare `deep_symbolize_keys` sul `chunk` ci permette di rendere il pattern matching più elegante, consentendoci di operare su simboli anziché stringhe.

```
1  def call(chunk, _bytesize)
2      case chunk.deep_symbolize_keys
3
4      in { # match function name
5          choices: [
6              {
7                  delta: {
8                      tool_calls: [
9                          { index: index, function: {name: name} }
10                     ]
11                 }
12             }
13         ]
14
15         f_name[index] = name
```

Il primo pattern che stiamo cercando corrisponde a una chiamata dello strumento insieme al nome della funzione associata. Se ne rileviamo uno, lo inseriamo nell'array `f_name`. Memorizziamo i nomi delle funzioni in un array indicizzato, poiché il modello è in grado di effettuare chiamate di funzione parallele, inviando più di una funzione da eseguire contemporaneamente.

La chiamata parallela delle funzioni è la capacità di un modello di IA di eseguire più chiamate di funzione insieme, permettendo che gli effetti e i risultati di queste chiamate vengano risolti in parallelo. Questo è particolarmente utile se le funzioni richiedono molto tempo e riduce i viaggi di andata e ritorno con le API, che a loro volta possono far risparmiare una quantità significativa di consumo di token.

Successivamente dobbiamo cercare la corrispondenza degli argomenti relativi alle chiamate di funzione.

```

1  in { # match arguments
2    choices: [
3      {
4        delta: {
5          tool_calls: [
6            {
7              index: index, function: {arguments: argument }
8            }
9          ]
10        }
11      }
12    ]
13
14    f_arguments[index] ||= "" # initialize if not already
15    f_arguments[index] << argument

```

In modo simile a come abbiamo gestito i nomi delle funzioni, inseriamo gli argomenti in un array indicizzato.

Successivamente, cerchiamo i messaggi rivolti all'utente normali, che arriveranno dal server un token alla volta e verranno assegnati alla variabile `new_content`. Dobbiamo anche tenere d'occhio `finish_reason`. Sarà `nil` fino all'ultimo segmento della sequenza di output.

```

1  in {
2    choices: [
3      { delta: {content: new_content}, finish_reason: finish_reason }
4    ]
5
6    # you could transmit every chunk to the user here...
7    buffer << new_content.to_s
8
9    if finish_reason.present?
10      finalize
11    elsif new_content.to_s.match?(/\n\n/)
12      send_to_client # ...or buffer and transmit once per paragraph
13    end

```

È importante aggiungere un'espressione di pattern matching per gestire i messaggi di errore inviati dal provider del modello AI. Negli ambienti di sviluppo locali, solleviamo un'eccezione, ma in produzione, registriamo l'errore e finalizziamo.

```
1  in { error: { message: } }
2    if Rails.env.local?
3      raise message
4    else
5      Honeybadger.notify("AI Error: #{message}")
6      finalize
7    end
```

La clausola else finale del case verrà eseguita se nessuno dei pattern precedenti ha trovato corrispondenza. È semplicemente una misura di sicurezza così che, se il modello di IA inizia a inviarci chunk non riconosciuti, ne veniamo a conoscenza.

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

Il metodo `send_to_client` è responsabile dell'invio del contenuto memorizzato nel buffer al client. Verifica che il buffer non sia vuoto, aggiorna il contenuto del messaggio del bot, renderizza il messaggio del bot e salva il contenuto nel database per garantire la persistenza dei dati.

```

1 def send_to_client
2   # no need to process pure whitespace
3   return if buffer.join.squish.blank?
4
5   # set the buffer content on the bot message
6   content = buffer.join
7   bot_message.content = content
8
9   # save to database so that we never lose data
10  # even if the stream doesn't terminate correctly
11  bot_message.update_column(:content, content)
12
13  # update content via websocket
14  ConversationRenderer.update(bot_message)
15 end

```

Il metodo `finalize` viene chiamato quando l'elaborazione dello stream è completata. Gestisce le chiamate di funzione se ne sono state ricevute durante lo stream, aggiorna il messaggio del bot con il contenuto finale e altre informazioni rilevanti, e reimposta la cronologia delle chiamate di funzione

```

1 def finalize
2   if f_name.any?
3     f_name.each_with_index do |name, index|
4       # takes care of calling the function wherever it's implemented
5       dispatch(name:, arguments: JSON.parse(f_arguments[index]))
6     end
7
8     # reset the function call history
9     f_name.clear
10    f_arguments.clear
11  else
12    content = buffer.join.presence
13    bot_message.update!(content:, complete: true)
14    ConversationRenderer.update(bot_message)
15  end
16 end

```

Se il modello decide di chiamare una funzione, è necessario “dispatchiare” quella chiamata di funzione (nome e argomenti) in modo tale che venga eseguita e i messaggi

`function_call` e `function_result` vengono aggiunti alla trascrizione della conversazione

Dalla mia esperienza, è meglio gestire la creazione dei messaggi di funzione in un unico punto della base di codice, invece di affidarsi alle implementazioni degli strumenti. È più pulito, ma c'è anche una ragione pratica molto importante: se il modello di IA chiama una funzione, e non vede i messaggi della chiamata e del risultato nella trascrizione quando si ripete il ciclo, *chiamerà la stessa funzione di nuovo*. Potenzialmente all'infinito. Ricorda che l'IA è completamente priva di stato, quindi a meno che tu non rimandi indietro quelle chiamate di funzione, per l'IA non sono mai avvenute.

```
1 # PromptSubscriber#dispatch
2
3 def dispatch(name:, arguments:)
4   # adds a function_call message to the conversation transcript
5   # plus dispatches to tool and returns result
6   conversation.function_call!(name, arguments).then do |result|
7     # add function result message to the transcript
8     conversation.function_result!(name, result)
9   end
10 end
```



Pulire la cronologia delle chiamate di funzione dopo l'esecuzione è importante tanto quanto assicurarsi che la chiamata e i risultati finiscano nella tua trascrizione, in modo da non continuare a chiamare le stesse funzioni ripetutamente ad ogni ciclo.

Il “Ciclo di Conversazione”

Nella classe `PromptSubscriber`, utilizziamo il metodo `prompt` dal modulo `PromptDeclarations` per definire il comportamento del ciclo di conversazione.

Il parametro `until` è impostato su `-> { bot_message.complete? }`, il che significa che il ciclo continuerà fino a quando il `bot_message` non viene contrassegnato come completo.

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



Ma quando viene contrassegnato come completo `bot_message`? Se non lo ricordi, torna alla riga 13 del metodo `finalize`.

Rivediamo l'intera logica di elaborazione del flusso.

1. Il `PromptSubscriber` riceve un nuovo messaggio dell'utente tramite il metodo `message_created`, che viene invocato dal sistema pub/sub Wisper ogni volta che l'utente finale crea un nuovo prompt.
2. Il metodo di classe `prompt` definisce in modo dichiarativo il comportamento della logica di completamento della chat per il `PromptSubscriber`. Il modello AI eseguirà un completamento della chat con il contenuto del messaggio dell'utente, una nuova istanza di `ReplyStream` come parametro di flusso e la condizione del ciclo specificata.
3. Il modello AI elabora il prompt e inizia a generare una risposta. Mentre la risposta viene trasmessa in streaming, il metodo `call` dell'istanza `ReplyStream` viene invocato per ogni frammento di dati.
4. Se il modello AI decide di chiamare una funzione strumentale, il nome della funzione e gli argomenti vengono estratti dal frammento e memorizzati rispettivamente negli array `f_name` e `f_arguments`.
5. Se il modello AI genera contenuti destinati all'utente, questi vengono memorizzati nel buffer e inviati al client tramite il metodo `send_to_client`.

6. Una volta completata l'elaborazione del flusso, viene chiamato il metodo `finalize`. Se durante il flusso sono state invocate funzioni strumentali, queste vengono inviate utilizzando il metodo `dispatch` del `PromptSubscriber`.
7. Il metodo `dispatch` aggiunge un messaggio `function_call` alla trascrizione della conversazione, esegue la funzione strumentale corrispondente e aggiunge un messaggio `function_result` alla trascrizione con il risultato della chiamata alla funzione.
8. Dopo l'invio delle funzioni strumentali, la cronologia delle chiamate alle funzioni viene cancellata per evitare chiamate duplicate nei cicli successivi.
9. Se non sono state invocate funzioni strumentali, il metodo `finalize` aggiorna il `bot_message` con il contenuto finale, lo contrassegna come completo e invia il messaggio aggiornato al client.
10. Viene valutata la condizione del ciclo `-> { bot_message.complete? }`. Se il `bot_message` non è contrassegnato come completo, il ciclo continua e il prompt originale viene nuovamente inviato con la trascrizione della conversazione aggiornata.
11. I passaggi 3-10 vengono ripetuti fino a quando il `bot_message` non viene contrassegnato come completo, indicando che il modello AI ha terminato di generare la sua risposta e non sono necessarie ulteriori esecuzioni di funzioni strumentali.

Implementando questo ciclo di conversazione, si permette al modello AI di impegnarsi in un'interazione bidirezionale con l'applicazione, eseguendo funzioni strumentali secondo necessità e generando risposte per l'utente fino a quando la conversazione raggiunge una conclusione naturale.

La combinazione di elaborazione del flusso e ciclo di conversazione consente esperienze dinamiche e interattive basate sull'AI, dove il modello AI può elaborare l'input dell'utente, utilizzare vari strumenti e funzioni, e fornire risposte in tempo reale basate sul contesto della conversazione in evoluzione.

Continuazione Automatica

È importante essere consapevoli delle limitazioni dell'output dell'AI. La maggior parte dei modelli ha un numero massimo di token che può generare in una singola risposta, determinato dal parametro `max_tokens`. Se il modello AI raggiunge questo limite durante la generazione di una risposta, si interromperà bruscamente e indicherà che l'output è stato troncato.

Nella risposta in streaming dall'API della piattaforma AI, è possibile rilevare questa situazione esaminando la variabile `finish_reason` nel frammento. Se il `finish_reason` è impostato su "length" (o qualche altro valore chiave specifico del modello), significa che il modello ha raggiunto il suo limite massimo di token durante la generazione e l'output è stato interrotto.

Un modo per gestire questo scenario con eleganza e fornire un'esperienza utente fluida è implementare un meccanismo di continuazione automatica nella logica di elaborazione del flusso. Aggiungendo un pattern matching per le ragioni di completamento legate alla lunghezza, è possibile scegliere di ciclare e continuare automaticamente l'output da dove si era interrotto.

Ecco un esempio volutamente semplificato di come è possibile modificare il metodo `call` nella classe `ReplyStream` per supportare la continuazione automatica:

```
1 LENGTH_STOPS = %w[length MAX_TOKENS]
2
3 def call(chunk, _bytesize)
4   case chunk.deep_symbolize_keys
5     # ...
6
7   in {
8     choices: [
9       { delta: {content: new_content},
10         finish_reason: finish_reason } ] }
11
12   buffer << new_content.to_s
13
14   if finish_reason.blank?
15     send_to_client if new_content.to_s.match?(/\n\n/)
16   elsif LENGTH_STOPS.include?(finish_reason)
17     continue_cutoff
18   else
19     finalize
20   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
```

In questa versione modificata, quando il `finish_reason` indica un output troncato, invece di finalizzare lo stream, aggiungiamo una coppia di messaggi alla trascrizione senza finalizzarla, spostiamo il messaggio di risposta originale rivolto all'utente in “fondo” alla trascrizione aggiornando il suo attributo `created_at`, e poi lasciamo che il ciclo continui, in modo che l'IA prosegua la generazione da dove si era interrotta.

Ricorda che l'endpoint di completamento dell'IA è stateless. “Conosce” solo ciò che

gli comunichi attraverso la trascrizione. In questo caso, il modo in cui comunichiamo all'IA che è stata interrotta è aggiungendo messaggi “invisibili” (per l'utente finale) alla trascrizione. Ricorda però che questo è un esempio volutamente semplificato. Un'implementazione reale dovrebbe gestire ulteriormente la trascrizione per assicurarsi di non sprecare token e/o confondere l'IA con messaggi dell'assistente duplicati nella trascrizione.

Un'implementazione reale dell'auto-continuazione dovrebbe anche avere la cosiddetta “logica del circuit breaker” per prevenire cicli incontrollati. Questo perché, dati certi tipi di prompt utente e impostazioni basse di `max_tokens`, l'IA potrebbe continuare a generare output visibile all'utente all'infinito.

Tieni presente che ogni ciclo richiede una richiesta separata e che ogni richiesta consuma nuovamente l'intera trascrizione. Dovresti sicuramente considerare il compromesso tra esperienza utente e utilizzo delle API quando decidi se implementare l'auto-continuazione nella tua applicazione. L'auto-continuazione in particolare può essere pericolosamente costosa, specialmente quando si utilizzano modelli commerciali premium.

Conclusione

L'elaborazione del flusso è un aspetto critico nella costruzione di applicazioni basate sull'IA che combinano l'utilizzo degli strumenti con risposte IA in tempo reale. Gestendo efficientemente i dati in streaming dalle API delle piattaforme IA, puoi fornire un'esperienza utente fluida e interattiva, gestire risposte di grandi dimensioni, ottimizzare l'utilizzo delle risorse e gestire gli errori con eleganza.

La classe `Conversation::ReplyStream` fornita dimostra come l'elaborazione del flusso può essere implementata in un'applicazione Ruby utilizzando il pattern matching

e l'architettura event-driven. Comprendendo e sfruttando le tecniche di elaborazione del flusso, puoi sbloccare il pieno potenziale dell'integrazione dell'IA nelle tue applicazioni e offrire esperienze utente potenti e coinvolgenti.

Dati Auto-riparanti



I dati auto-riparanti rappresentano un potente approccio per garantire l'integrità, la coerenza e la qualità dei dati nelle applicazioni sfruttando le capacità dei modelli linguistici di grandi dimensioni (LLM). Questa categoria di pattern si concentra sull'idea di utilizzare l'IA per rilevare, diagnosticare e correggere automaticamente anomalie, incongruenze o errori nei dati, riducendo così il carico di lavoro per gli sviluppatori e mantenendo un elevato livello di affidabilità dei dati.

Nel loro nucleo, i pattern dei dati auto-riparanti riconoscono che i dati sono l'elemento vitale di qualsiasi applicazione, e garantire la loro accuratezza e integrità è cruciale per il corretto funzionamento e l'esperienza utente dell'applicazione. Tuttavia, gestire e mantenere la qualità dei dati può essere un compito complesso e che richiede molto tempo, soprattutto quando le applicazioni crescono in dimensioni e complessità. È qui che entra in gioco la potenza dell'IA.

Nei pattern dei dati auto-riparanti, i worker IA vengono impiegati per monitorare e analizzare continuamente i dati della tua applicazione. Questi modelli hanno la capacità di comprendere e interpretare pattern, relazioni e anomalie all'interno dei dati. Sfruttando le loro capacità di elaborazione e comprensione del linguaggio naturale, possono identificare potenziali problemi o incongruenze nei dati e intraprendere azioni appropriate per correggerli.

Il processo di auto-riparazione dei dati tipicamente coinvolge diversi passaggi chiave:

1. **Monitoraggio dei Dati:** I worker IA monitorano costantemente i flussi di dati, i database o i sistemi di archiviazione dell'applicazione, cercando qualsiasi segno di anomalie, incongruenze o errori. In alternativa, puoi attivare un componente IA in reazione a un'eccezione.
2. **Rilevamento delle Anomalie:** Quando viene rilevato un problema, il worker IA analizza i dati in dettaglio per identificare la natura specifica e la portata del problema. Questo potrebbe comportare il rilevamento di valori mancanti, formati inconsistenti o dati che violano regole o vincoli predefiniti.
3. **Diagnosi e Correzione:** Una volta identificato il problema, il worker IA utilizza la sua conoscenza e comprensione del dominio dei dati per determinare il corso d'azione appropriato. Questo potrebbe comportare la correzione automatica dei dati, il completamento di valori mancanti o la segnalazione del problema per l'intervento umano se necessario.
4. **Apprendimento Continuo (opzionale, a seconda del caso d'uso):** Mentre il worker IA incontra e risolve vari problemi di dati, può produrre output che descrivono cosa è successo e come ha risposto. Questi metadati possono essere inseriti in processi di apprendimento che permettono a te (e forse al modello sottostante, tramite fine-tuning) di diventare più efficace ed efficiente nel tempo nell'identificare e risolvere le anomalie dei dati.

Rilevando e correggendo automaticamente i problemi dei dati, puoi garantire che la tua applicazione operi su dati di alta qualità e affidabili. Questo riduce il rischio che errori,

incongruenze o bug relativi ai dati influenzino la funzionalità o l'esperienza utente dell'applicazione.

Una volta che hai i worker IA che gestiscono il compito di monitoraggio e correzione dei dati, puoi concentrare i tuoi sforzi su altri aspetti critici dell'applicazione. Questo fa risparmiare tempo e risorse che altrimenti sarebbero spesi nella pulizia e manutenzione manuale dei dati. In effetti, man mano che le tue applicazioni crescono in dimensioni e complessità, la gestione manuale della qualità dei dati diventa sempre più impegnativa. I pattern dei “Dati Auto-riparanti” si scalano efficacemente sfruttando la potenza dell'IA per gestire grandi volumi di dati e rilevare problemi in tempo reale.



Per loro natura, i modelli di IA possono adattarsi nel tempo ai cambiamenti nei pattern dei dati, negli schemi o nei requisiti con poca o nessuna supervisione. Finché le loro direttive forniscono una guida adeguata, specialmente per quanto riguarda i risultati previsti, la tua applicazione potrebbe essere in grado di evolversi e gestire nuovi scenari di dati senza richiedere estesi interventi manuali o modifiche al codice.

I pattern dei dati auto-riparanti si allineano bene con le altre categorie di pattern che abbiamo discusso, come la “Moltitudine di Worker”. La capacità di auto-riparazione dei dati può essere vista come un tipo specializzato di worker che si concentra specificamente sulla garanzia della qualità e dell'integrità dei dati. Questo tipo di worker opera insieme ad altri worker IA, ognuno contribuendo a diversi aspetti della funzionalità dell'applicazione.

L'implementazione dei pattern dei dati auto-riparanti nella pratica richiede una progettazione attenta e l'integrazione dei modelli di IA nell'architettura dell'applicazione. A causa dei rischi di perdita e corruzione dei dati, dovresti definire linee guida chiare su come utilizzerai questa tecnica. Dovresti anche considerare fattori come le prestazioni, la scalabilità e la sicurezza dei dati.

Caso di Studio Pratico: Riparare JSON Danneggiato

Uno dei modi più pratici e convenienti per sfruttare i dati auto-riparanti è anche molto semplice da spiegare: riparare JSON danneggiato.

Questa tecnica può essere applicata alla sfida comune di gestire dati imperfetti o inconsistenti generati dagli LLM, come JSON danneggiato, e fornisce un approccio per rilevare e correggere automaticamente questi problemi.

A Olympia mi capita regolarmente di incontrare scenari in cui gli LLM generano dati JSON non perfettamente validi. Questo può accadere per varie ragioni, come quando l'LLM aggiunge commenti prima o dopo il codice JSON effettivo, o introduce errori di sintassi come virgolette mancanti o virgolette doppie non escapate. Questi problemi possono portare a errori di parsing e causare interruzioni nella funzionalità dell'applicazione.

Per affrontare questo problema, ho implementato una soluzione pratica sotto forma di una classe `JsonFixer`. Questa classe incarna il pattern “Self-Healing Data” prendendo in input il JSON danneggiato e sfruttando un LLM per correggerlo, preservando il più possibile le informazioni e l'intento originale.

```
1  class JsonFixer
2      include Raix::ChatCompletion
3
4      def call(bad_json, error_message)
5          raise "No data provided" if bad_json.blank? || error_message.blank?
6
7          transcript << {
8              system: "Consider user-provided JSON that generated a parse
9                  exception. Do your best to fix it while preserving the
10                 original content and intent as much as possible."
11
12             transcript << { user: bad_json }
13             transcript << { assistant: "What is the error message?" }
14             transcript << { user: error_message }
```

```

14     transcript << { assistant: "Here is the corrected JSON\n```\n" }
15
16     self.stop = ["```"]
17
18     chat_completion(json: true)
19 end
20
21 def model
22   "mistralai/mixtral-8x7b-instruct:nitro"
23 end
24 end

```



Nota come `JsonFixer` utilizzi `Ventriloquist` per guidare le risposte dell'AI.

Il processo di auto-riparazione dei dati JSON funziona come segue:

1. **Generazione JSON:** Un LLM viene utilizzato per generare dati JSON basati su determinati prompt o requisiti. Tuttavia, a causa della natura degli LLM, il JSON generato potrebbe non essere sempre perfettamente valido. Il parser JSON ovviamente solleverà un `ParserError` se gli viene fornito un JSON non valido.

```

1 begin
2   JSON.parse(llm_generated_json)
3 rescue JSON::ParserError => e
4   JsonFixer.new.call(llm_generated_json, e.message)
5 end

```

Si noti che il messaggio di errore viene anche passato alla chiamata `JsonFixer` in modo che non debba completamente presumere cosa c'è di sbagliato nei dati, specialmente considerando che il parser spesso indica esattamente qual è il problema.

2. **Correzione basata su LLM:** La classe `JsonFixer` invia il JSON danneggiato a un LLM, insieme a un prompt o un'istruzione specifica per correggere il

JSON mantenendo il più possibile le informazioni e l'intento originale. L'LLM, addestrato su enormi quantità di dati e con una comprensione della sintassi JSON, tenta di correggere gli errori e generare una stringa JSON valida. La [Delimitazione delle Risposte](#) viene utilizzata per limitare l'output dell'LLM, e sceglioamo Mixtral 8x7B come modello AI, poiché è particolarmente adatto per questo tipo di attività.

3. **Validazione e Integrazione:** La stringa JSON corretta restituita dall'LLM viene analizzata dalla classe `JSONFixer` stessa, poiché ha chiamato `chat_completion(json: true)`. Se il JSON corretto supera la validazione, viene integrato nuovamente nel flusso di lavoro dell'applicazione, permettendo all'applicazione di continuare a elaborare i dati senza interruzioni. Il JSON difettoso è stato “riparato”.

Sebbene abbia scritto e riscritto la mia implementazione di `JSONFixer` più volte, dubito che il tempo totale investito in tutte queste versioni sia più di un'ora o due.

Si noti che la preservazione dell'intento è un elemento chiave di qualsiasi pattern di dati auto-riparanti. Il processo di correzione basato su LLM mira a preservare il più possibile le informazioni e l'intento originale del JSON generato. Questo assicura che il JSON corretto mantenga il suo significato semantico e possa essere utilizzato efficacemente nel contesto dell'applicazione.

Questa implementazione pratica dell'approccio “Dati Auto-Riparanti” in Olympia dimostra chiaramente come l'AI, in particolare gli LLM, possa essere sfruttata per risolvere sfide reali sui dati. Mostra la potenza della combinazione di tecniche di programmazione tradizionali con le capacità dell'AI per costruire applicazioni robuste ed efficienti.

La Legge di Postel e il Pattern dei “Dati Auto-Riparanti”

I “Dati Auto-Riparanti”, come esemplificato dalla classe JSONFixer, si allineano bene con il principio noto come Legge di Postel, anche chiamato Principio di Robustezza. La Legge di Postel afferma:

“Sii conservativo in ciò che fai, sii liberale in ciò che accetti dagli altri.”

Questo principio, originariamente articolato da Jon Postel, un pioniere dei primi tempi di Internet, enfatizza l’importanza di costruire sistemi che siano tolleranti verso input diversi o anche leggermente incorretti, mantenendo allo stesso tempo una stretta aderenza ai protocolli specificati quando si inviano output.

Nel contesto dei “Dati Auto-Riparanti”, la classe JSONFixer incarna la Legge di Postel essendo liberale nell’accettare dati JSON danneggiati o imperfetti generati dagli LLM. Non rifiuta o fallisce immediatamente quando incontra JSON che non aderisce strettamente al formato previsto. Invece, adotta un approccio tollerante e tenta di correggere il JSON utilizzando la potenza degli LLM.

Essendo liberale nell’accettare JSON imperfetto, la classe JSONFixer dimostra robustezza e flessibilità. Riconosce che i dati nel mondo reale spesso si presentano in varie forme e potrebbero non sempre conformarsi a specifiche rigide. Gestendo e correggendo con grazia queste deviazioni, la classe assicura che l’applicazione possa continuare a funzionare senza problemi, anche in presenza di dati imperfetti.

D’altra parte, la classe JSONFixer aderisce anche all’aspetto conservativo della Legge di Postel quando si tratta dell’output. Dopo aver corretto il JSON utilizzando gli LLM, la classe valida il JSON corretto per assicurarsi che si conformi rigorosamente al formato previsto. Mantiene l’integrità e la correttezza dei dati prima di passarli ad altre parti dell’applicazione. Questo approccio conservativo garantisce che l’output della classe JSONFixer sia affidabile e coerente, promuovendo l’interoperabilità e prevenendo la propagazione di errori.

Curiosità interessanti su Jon Postel:

- Jon Postel (1943-1998) è stato un informatico americano che ha giocato un

ruolo cruciale nello sviluppo di Internet. Era conosciuto come il “Dio di Internet” per i suoi significativi contributi ai protocolli e agli standard di base.

- Postel è stato l'editore della serie di documenti Request for Comments (RFC), una serie di note tecniche e organizzative su Internet. Ha scritto o co-scritto oltre 200 RFC, inclusi i protocolli fondamentali come TCP, IP e SMTP.
- Oltre ai suoi contributi tecnici, Postel era noto per il suo approccio umile e collaborativo. Credeva nell'importanza di raggiungere il consenso e lavorare insieme per costruire una rete robusta e interoperabile.
- Postel ha servito come Direttore della Divisione Reti di Computer presso l'Information Sciences Institute (ISI) dell'Università della California del Sud (USC) dal 1977 fino alla sua prematura scomparsa nel 1998.
- In riconoscimento dei suoi immensi contributi, a Postel è stato assegnato postumo il prestigioso Premio Turing nel 1998, spesso definito il “Premio Nobel dell'Informatica.”

La classe `JSONFixer` promuove robustezza, flessibilità e interoperabilità, che erano i valori fondamentali che Postel ha sostenuto durante tutta la sua carriera. Costruendo sistemi che sono tolleranti alle imperfezioni mentre mantengono una stretta aderenza ai protocolli, possiamo creare applicazioni più resistenti e adattabili di fronte alle sfide del mondo reale.

Considerazioni e Controindicazioni

L'applicabilità degli approcci di dati auto-riparanti dipende interamente dal tipo di dati che la tua applicazione gestisce. C'è un motivo per cui potresti non voler semplicemente fare un `monkeypatch` di `JSON.parse` per correggere automaticamente *tutti gli errori di parsing JSON* nella tua applicazione: non tutti gli errori possono o dovrebbero essere corretti automaticamente.

L'auto-riparazione è particolarmente delicata quando è associata a requisiti normativi o di conformità relativi alla gestione e all'elaborazione dei dati. Alcuni settori, come quello sanitario e finanziario, hanno normative così severe riguardo all'integrità dei dati e alla verificabilità che eseguire qualsiasi tipo di correzione dei dati "black box" senza un'adeguata supervisione o registrazione potrebbe violare queste normative. È fondamentale garantire che qualsiasi tecnica di auto-riparazione dei dati che sviluppi sia allineata con i quadri normativi e legali applicabili.

L'applicazione di tecniche di dati auto-riparanti, in particolare quelle che coinvolgono modelli di IA, può anche avere un grande impatto sulle prestazioni dell'applicazione e sull'utilizzo delle risorse. L'elaborazione di grandi volumi di dati attraverso modelli di IA per il rilevamento e la correzione degli errori può essere computazionalmente intensiva. È importante valutare i compromessi tra i benefici dei dati auto-riparanti e i costi associati in termini di prestazioni e risorse.

Detto questo, approfondiamo i fattori coinvolti nel decidere quando e dove applicare questo potente approccio.

Criticità dei Dati

Quando si considera l'applicazione delle tecniche di dati auto-riparanti, è fondamentale valutare la criticità dei dati che vengono elaborati. Il livello di criticità si riferisce all'importanza e alla sensibilità dei dati nel contesto della tua applicazione e del suo dominio aziendale.

In alcuni casi, la correzione automatica degli errori nei dati potrebbe non essere appropriata, specialmente se i dati sono altamente sensibili o hanno implicazioni legali. Per esempio, considera i seguenti scenari:

1. **Transazioni Finanziarie:** Nelle applicazioni finanziarie, come i sistemi bancari o le piattaforme di trading, l'accuratezza dei dati è della massima importanza. Anche errori minori nei dati finanziari possono avere conseguenze significative, come

saldi dei conti errati, fondi instradati male o decisioni di trading erronee. In questi casi, le correzioni automatizzate senza una verifica e un controllo approfonditi potrebbero introdurre rischi inaccettabili.

2. **Cartelle Cliniche:** Le applicazioni sanitarie trattano dati dei pazienti altamente sensibili e confidenziali. Le imprecisioni nelle cartelle cliniche possono avere gravi implicazioni per la sicurezza del paziente e le decisioni di trattamento. Modificare automaticamente i dati medici senza un'adeguata supervisione e convalida da parte di professionisti sanitari qualificati potrebbe violare i requisiti normativi e mettere a rischio il benessere del paziente.
3. **Documenti Legali:** Le applicazioni che gestiscono documenti legali, come contratti, accordi o atti giudiziari, richiedono una rigorosa accuratezza e integrità. Anche errori minori nei dati legali possono avere significative ramificazioni legali. Le correzioni automatizzate in questo dominio potrebbero non essere appropriate, poiché i dati spesso richiedono una revisione manuale e una verifica da parte di esperti legali per garantirne la validità e l'applicabilità.

In questi scenari di dati critici, i rischi associati alle correzioni automatizzate spesso superano i potenziali benefici. Le conseguenze dell'introduzione di errori o della modifica errata dei dati possono essere gravi, portando a perdite finanziarie, responsabilità legali o addirittura danni alle persone.

Quando si ha a che fare con dati altamente critici, è essenziale dare priorità ai processi di verifica e convalida manuale. La supervisione umana e l'esperienza sono cruciali per garantire l'accuratezza e l'integrità dei dati. Le tecniche di auto-riparazione automatizzata possono ancora essere impiegate per segnalare potenziali errori o incongruenze, ma la decisione finale sulle correzioni dovrebbe coinvolgere il giudizio e l'approvazione umana.

Tuttavia, è importante notare che non tutti i dati in un'applicazione possono avere lo stesso livello di criticità. All'interno della stessa applicazione, potrebbero esserci sottoinsiemi di dati che sono meno sensibili o hanno un impatto minore in caso di errori.

In tali casi, le tecniche di dati auto-riparanti possono essere applicate selettivamente a quei specifici sottoinsiemi di dati, mentre i dati critici rimangono soggetti a verifica manuale.

La chiave è valutare attentamente la criticità di ciascuna categoria di dati nella tua applicazione e definire linee guida e processi chiari per gestire le correzioni in base ai rischi e alle implicazioni associate. Differenziando tra dati critici (come registri contabili, cartelle cliniche) e dati non critici (come indirizzi postali, avvisi sulle risorse), puoi trovare un equilibrio tra sfruttare i benefici delle tecniche di dati auto-riparanti dove appropriato e mantenere un controllo e una supervisione rigorosi dove necessario.

In definitiva, la decisione di applicare tecniche di dati auto-riparanti ai dati critici dovrebbe essere presa in consultazione con esperti del dominio, consulenti legali e altri stakeholder rilevanti. È essenziale considerare i requisiti specifici, le normative e i rischi associati ai dati della tua applicazione e allineare di conseguenza le strategie di correzione dei dati.

Gravità degli Errori

Quando si applicano tecniche di dati auto-riparanti, è importante valutare la gravità e l'impatto degli errori nei dati. Non tutti gli errori sono uguali, e il corso d'azione appropriato può variare a seconda della gravità del problema.

Incongruenze minori o problemi di formattazione possono essere adatti per la correzione automatica. Per esempio, un worker di dati auto-riparanti incaricato di correggere JSON danneggiato può gestire virgolette mancanti o virgolette doppie non escapate senza alterare significativamente il significato o la struttura dei dati. Questi tipi di errori sono spesso semplici da correggere e hanno un impatto minimo sull'integrità complessiva dei dati.

Tuttavia, gli errori più gravi che modificano fondamentalmente il significato o l'integrità dei dati potrebbero richiedere un approccio diverso. In questi casi, le correzioni automatizzate potrebbero non essere sufficienti, e potrebbe essere necessario l'intervento umano per garantire l'accuratezza e la validità dei dati.

È qui che entra in gioco il concetto di utilizzare l'IA stessa per aiutare a determinare la gravità degli errori. Sfruttando le capacità dei modelli di IA, possiamo progettare operatori di dati auto-riparanti che non solo correggono gli errori, ma valutano anche la gravità di questi errori e prendono decisioni informate su come gestirli.

Per esempio, consideriamo un operatore di dati auto-riparante responsabile della correzione delle incongruenze nei dati che confluiscono in un database dei clienti. L'operatore può essere progettato per analizzare i dati e identificare potenziali errori, come informazioni mancanti o contrastanti. Tuttavia, invece di correggere automaticamente tutti gli errori, l'operatore può essere dotato di ulteriori chiamate agli strumenti che gli permettono di segnalare gli errori gravi per una revisione umana.

Ecco un esempio di come questo può essere implementato:

```
1  class CustomerDataReviewer
2    include Raix::ChatCompletion
3    include Raix::FunctionDeclarations
4
5    attr_accessor :customer
6
7    function :flag_for_review, reason: { type: "string" } do |params|
8      AdminNotifier.review_request(customer, params[:reason])
9    end
10
11  def initialize(customer)
12    self.customer = customer
13  end
14
15  def call(customer_data)
16    transcript << {
17      system: "You are a customer data reviewer. Your task is to identify
18      and correct inconsistencies in customer data.
19
20      < additional instructions here... >
21
22      If you encounter severe errors that require human review, use the
23      `flag_for_review` tool to flag the data for manual intervention."
24
25    transcript << { user: customer.to_json }
```

```
26     transcript << { assistant: "Reviewed/corrected data:\n```\n" }  
27  
28     self.stop = ["````"]  
29  
30     chat_completion(json: true).then do |result|  
31       return if result.blank?  
32  
33       customer.update(result)  
34     end  
35   end  
36 end
```

In questo esempio, il worker `CustomerDataHealer` è progettato per identificare e correggere le incongruenze nei dati dei clienti. Ancora una volta, utilizziamo [Response Fencing](#) e [Ventriloquist](#) per ottenere output strutturati. È importante notare che la direttiva di sistema del worker include istruzioni per utilizzare la funzione `flag_for_review` se vengono riscontrati errori gravi.

Quando il worker elabora i dati dei clienti, analizza i dati e tenta di correggere eventuali incongruenze. Se il worker determina che gli errori sono gravi e richiedono l'intervento umano, può utilizzare lo strumento `flag_for_review` per contrassegnare i dati e fornire una motivazione per la segnalazione.

Il metodo `chat_completion` viene chiamato con `json: true` per analizzare i dati del cliente corretti come JSON. Non è prevista alcuna disposizione per l'iterazione dopo una chiamata di funzione, quindi il risultato sarà vuoto se `flag_for_review` è stato invocato. In caso contrario, il cliente viene aggiornato con i dati revisionati e potenzialmente corretti.

Incorporando la valutazione della gravità degli errori e l'opzione di contrassegnare i dati per la revisione umana, il worker per i dati auto-riparanti diventa più intelligente e adattabile. Può gestire automaticamente gli errori minori mentre escalate quelli gravi agli esperti umani per un intervento manuale.

I criteri specifici per determinare la gravità dell'errore possono essere definiti nella direttiva del worker in base alle conoscenze del dominio e ai requisiti aziendali. Fattori

come l'impatto sull'integrità dei dati, il potenziale di perdita o corruzione dei dati e le conseguenze di dati errati possono essere considerati durante la valutazione della gravità.

Sfruttando l'IA per valutare la gravità degli errori e fornendo opzioni per l'intervento umano, le tecniche di dati auto-riparanti possono trovare un equilibrio tra automazione e mantenimento dell'accuratezza dei dati. Questo approccio garantisce che gli errori minori vengano corretti in modo efficiente mentre gli errori gravi ricevano l'attenzione e l'esperienza necessarie dai revisori umani.

Complessità del Dominio

Quando si considera l'applicazione delle tecniche di dati auto-riparanti, è importante valutare la complessità del dominio dei dati e le regole che governano la sua struttura e le sue relazioni. La complessità del dominio può influenzare significativamente l'efficacia e la fattibilità degli approcci di correzione automatica dei dati.

Le tecniche di dati auto-riparanti funzionano bene quando i dati seguono modelli e vincoli ben definiti. Nei domini in cui la struttura dei dati è relativamente semplice e le relazioni tra gli elementi dei dati sono dirette, le correzioni automatiche possono essere applicate con un alto grado di confidenza. Ad esempio, la correzione di problemi di formattazione o l'applicazione di vincoli di base sui tipi di dati possono spesso essere gestiti efficacemente dai worker per dati auto-riparanti.

Tuttavia, all'aumentare della complessità del dominio dei dati, crescono anche le sfide associate alla correzione automatica dei dati. Nei domini con logica aziendale intricata, relazioni complesse tra entità di dati o regole ed eccezioni specifiche del dominio, le tecniche di dati auto-riparanti potrebbero non sempre catturare le sfumature e potrebbero introdurre conseguenze indesiderate.

Consideriamo un esempio di dominio complesso: un sistema di trading finanziario. In questo dominio, i dati coinvolgono vari strumenti finanziari, dati di mercato, regole di

trading e requisiti normativi. Le relazioni tra diversi elementi di dati possono essere intricate e le regole che governano la validità e la coerenza dei dati possono essere altamente specifiche per il dominio.

In un dominio così complesso, un worker per dati auto-riparanti incaricato di correggere le incongruenze nei dati di trading dovrebbe avere una profonda comprensione delle regole e dei vincoli specifici del dominio. Dovrebbe considerare fattori come le normative di mercato, i limiti di trading, i calcoli del rischio e le procedure di regolamento. Le correzioni automatiche in questo contesto potrebbero non sempre catturare la piena complessità del dominio e potrebbero inavvertitamente introdurre errori o violare regole specifiche del dominio.

Per affrontare le sfide della complessità del dominio, le tecniche di dati auto-riparanti possono essere migliorate incorporando conoscenze e regole specifiche del dominio nei modelli e nei worker AI. Questo può essere ottenuto attraverso tecniche come:

- 1. Addestramento Specifico per il Dominio:** I modelli di IA utilizzati per i dati auto-riparanti possono essere diretti o persino perfezionati su dataset specifici del dominio che catturano le complessità e le regole del particolare dominio. Esponendo i modelli a dati e scenari rappresentativi, questi possono apprendere i modelli, i vincoli e le eccezioni specifiche del dominio.
- 2. Vincoli Basati su Regole:** I worker per dati auto-riparanti possono essere arricchiti con vincoli esplicativi basati su regole che codificano la conoscenza specifica del dominio. Queste regole possono essere definite da esperti del dominio e integrate nel processo di correzione dei dati. I modelli di IA possono quindi utilizzare queste regole per guidare le loro decisioni e garantire la conformità ai requisiti specifici del dominio.
- 3. Collaborazione con Esperti del Dominio:** Nei domini complessi, è cruciale coinvolgere esperti del dominio nella progettazione e nello sviluppo di tecniche di dati auto-riparanti. Gli esperti del dominio possono fornire preziose intuizioni sulle complessità dei dati, le regole aziendali e i potenziali casi limite. La loro

conoscenza può essere incorporata nei modelli e nei worker AI per migliorare l'accuratezza e l'affidabilità delle correzioni automatiche dei dati utilizzando i pattern di [Human In The Loop](#).

4. **Approccio Incrementale e Iterativo:** Quando si ha a che fare con domini complessi, è spesso vantaggioso adottare un approccio incrementale e iterativo ai dati auto-riparanti. Invece di tentare di automatizzare le correzioni per l'intero dominio in una volta sola, ci si concentra su sottodomini specifici o categorie di dati dove le regole e i vincoli sono ben compresi. Gradualmente si espande l'ambito delle tecniche di auto-riparazione man mano che la comprensione del dominio cresce e le tecniche si dimostrano efficaci.

Considerando la complessità del dominio dei dati e incorporando la conoscenza specifica del dominio nelle tecniche di dati auto-riparanti, è possibile trovare un equilibrio tra automazione e accuratezza. È importante riconoscere che i dati auto-riparanti non sono una soluzione universale e che l'approccio deve essere adattato ai requisiti specifici e alle sfide di ogni dominio.

In domini complessi, un approccio ibrido che combina tecniche di dati auto-riparanti con l'esperienza e la supervisione umana può risultare più efficace. Le correzioni automatizzate possono gestire casi di routine e ben definiti, mentre gli scenari complessi o le eccezioni possono essere segnalati per la revisione e l'intervento umano. Questo approccio collaborativo garantisce che i benefici dell'automazione vengano realizzati mantenendo il necessario controllo e l'accuratezza in domini di dati complessi.

Spiegabilità e Trasparenza

La spiegabilità si riferisce alla capacità di comprendere e interpretare il ragionamento alla base delle decisioni prese dai modelli di IA, mentre la trasparenza implica fornire una chiara visibilità nel processo di correzione dei dati.

In molti contesti, le modifiche ai dati devono essere verificabili e giustificabili. Gli stakeholder, inclusi gli utenti aziendali, i revisori e gli organi di regolamentazione,

potrebbero richiedere spiegazioni sul motivo per cui sono state apportate determinate correzioni ai dati e su come i modelli di IA siano giunti a tali decisioni. Questo è particolarmente cruciale in domini dove l'accuratezza e l'integrità dei dati hanno implicazioni significative, come la finanza, la sanità e le questioni legali.

Per affrontare la necessità di spiegabilità e trasparenza, le tecniche di dati auto-riparanti dovrebbero incorporare meccanismi che forniscono informazioni sul processo decisionale dei modelli di IA. Questo può essere realizzato attraverso vari approcci:

1. **Catena di Pensiero:** Chiedere al modello di spiegare il proprio ragionamento “ad alta voce” prima di applicare modifiche ai dati può permettere una più facile comprensione del processo decisionale e può generare spiegazioni leggibili dall’uomo per le correzioni effettuate. Il compromesso è una maggiore complessità nel separare la spiegazione dall’output dei dati strutturati, che può essere affrontata tramite...
2. **Generazione di Spiegazioni:** I worker dei dati auto-riparanti possono essere dotati della capacità di generare spiegazioni leggibili dall’uomo per le correzioni che apportano. Questo può essere ottenuto chiedendo al modello di produrre il suo processo decisionale come spiegazioni facilmente comprensibili *integrate nei dati stessi*. Per esempio, un worker di dati auto-riparanti potrebbe generare un report che evidenzia le specifiche inconsistenze dei dati identificate, le correzioni applicate e le motivazioni alla base di tali correzioni.
3. **Importanza delle Caratteristiche:** I modelli di IA possono essere istruiti con informazioni sull’importanza delle diverse caratteristiche o attributi nel processo di correzione dei dati come parte delle loro direttive. Queste direttive, a loro volta, possono essere esposte agli stakeholder umani. Identificando i fattori chiave che influenzano le decisioni del modello, gli stakeholder possono acquisire informazioni sul ragionamento alla base delle correzioni e valutarne la validità.
4. **Registrazione e Audit:** L’implementazione di meccanismi completi di registrazione e audit è cruciale per mantenere la trasparenza nel processo

di dati auto-riparanti. Ogni correzione dei dati effettuata dai modelli di IA dovrebbe essere registrata, inclusi i dati originali, i dati corretti e le azioni specifiche intraprese. Questa traccia di audit permette un'analisi retrospettiva e fornisce una registrazione chiara delle modifiche apportate ai dati.

5. **Approccio con Supervisione Umana:** Incorporare un approccio con supervisione umana può migliorare la spiegabilità e la trasparenza delle tecniche di dati auto-riparanti. Coinvolgendo esperti umani nella revisione e nella validazione delle correzioni generate dall'IA, le organizzazioni possono garantire che le correzioni siano allineate con la conoscenza del dominio e i requisiti aziendali. La supervisione umana aggiunge un ulteriore livello di responsabilità e permette l'identificazione di potenziali pregiudizi o errori nei modelli di IA.
6. **Monitoraggio e Valutazione Continui:** Il monitoraggio e la valutazione regolare delle prestazioni delle tecniche di dati auto-riparanti sono essenziali per mantenere la trasparenza e la fiducia. Valutando l'accuratezza e l'efficacia dei modelli di IA nel tempo, le organizzazioni possono identificare eventuali deviazioni o anomalie e intraprendere azioni correttive. Il monitoraggio continuo aiuta a garantire che il processo di dati auto-riparanti rimanga affidabile e allineato con i risultati desiderati.

La spiegabilità e la trasparenza sono considerazioni critiche nell'implementazione delle tecniche di dati auto-riparanti. Fornendo spiegazioni chiare per le correzioni dei dati, mantenendo tracce di audit complete e coinvolgendo la supervisione umana, le organizzazioni possono costruire fiducia nel processo di dati auto-riparanti e garantire che le modifiche apportate ai dati siano giustificabili e allineate con gli obiettivi aziendali.

È importante trovare un equilibrio tra i benefici dell'automazione e la necessità di trasparenza. Mentre le tecniche di dati auto-riparanti possono migliorare significativamente la qualità e l'efficienza dei dati, non dovrebbero comportare la perdita di visibilità e controllo sul processo di correzione dei dati. Progettando i worker

di dati auto-riparanti con la spiegabilità e la trasparenza in mente, le organizzazioni possono sfruttare il potere dell'IA mantenendo il necessario livello di responsabilità e fiducia nei dati.

Conseguenze Indesiderate

Mentre le tecniche di dati auto-riparanti mirano a migliorare la qualità e la coerenza dei dati, è fondamentale essere consapevoli del potenziale di conseguenze indesiderate. Le correzioni automatizzate, se non attentamente progettate e monitorate, potrebbero inavvertitamente alterare il significato o il contesto dei dati, portando a problemi a valle.

Uno dei rischi principali dei dati auto-riparanti è l'introduzione di pregiudizi o errori nel processo di correzione dei dati. I modelli di IA, come qualsiasi altro sistema software, possono essere soggetti a pregiudizi presenti nei dati di addestramento o introdotti attraverso la progettazione degli algoritmi. Se questi pregiudizi non vengono identificati e mitigati, possono propagarsi attraverso il processo di dati auto-riparanti e risultare in modifiche dei dati distorte o errate.

Ad esempio, consideriamo un lavoratore di dati auto-riparanti incaricato di correggere le incongruenze nei dati demografici dei clienti. Se il modello di IA ha appreso pregiudizi dai dati storici, come l'associazione di determinate occupazioni o livelli di reddito con specifici generi o etnie, potrebbe fare supposizioni errate e modificare i dati in modo da rafforzare questi pregiudizi. Ciò può portare a profili dei clienti inaccurati, decisioni aziendali errate e potenziali risultati discriminatori.

Un'altra potenziale conseguenza indesiderata è la perdita di informazioni o contesto preziosi durante il processo di correzione dei dati. Le tecniche di dati auto-riparanti spesso si concentrano sulla standardizzazione e normalizzazione dei dati per garantire la coerenza. Tuttavia, in alcuni casi, i dati originali potrebbero contenere sfumature, eccezioni o informazioni contestuali importanti per comprendere il quadro completo. Le correzioni automatizzate che applicano ciecamente la standardizzazione potrebbero inavvertitamente rimuovere o oscurare queste informazioni preziose.

Per esempio, immaginiamo un lavoratore di dati auto-riparanti responsabile della correzione delle incongruenze nelle cartelle cliniche. Se il lavoratore incontra la storia clinica di un paziente con una condizione rara o un piano di trattamento insolito, potrebbe tentare di normalizzare i dati per adattarli a uno schema più comune. Tuttavia, così facendo, potrebbe perdere i dettagli specifici e il contesto che sono cruciali per rappresentare accuratamente la situazione unica del paziente. Questa perdita di informazioni può avere serie implicazioni per la cura del paziente e il processo decisionale medico.

Per mitigare i rischi di conseguenze indesiderate, è essenziale adottare un approccio proattivo nella progettazione e implementazione delle tecniche di dati auto-riparanti:

1. **Test e Convalida Approfonditi:** Prima di distribuire i lavoratori di dati auto-riparanti in produzione, è cruciale testare e convalidare accuratamente il loro comportamento in una vasta gamma di scenari. Ciò include i test con set di dati rappresentativi che coprono vari casi limite, eccezioni e potenziali pregiudizi. Test rigorosi aiutano a identificare e affrontare eventuali conseguenze indesiderate prima che influenzino i dati nel mondo reale.
2. **Monitoraggio e Valutazione Continui:** L'implementazione di meccanismi di monitoraggio e valutazione continui è essenziale per rilevare e mitigare le conseguenze indesiderate nel tempo. Rivedere regolarmente i risultati dei processi di dati auto-riparanti, analizzare l'impatto sui sistemi a valle e sul processo decisionale, e raccogliere feedback dagli stakeholder può aiutare a identificare eventuali effetti negativi e promuovere azioni correttive tempestive. Se la vostra organizzazione dispone di dashboard operativi, aggiungere metriche chiaramente visibili relative alle modifiche automatizzate dei dati è probabilmente una buona idea. Aggiungere allarmi collegati a grandi deviazioni dall'attività normale di modifica dei dati è probabilmente un'idea ancora migliore!
3. **Supervisione e Intervento Umano:** Mantenere la supervisione umana e la capacità di intervenire nel processo di dati auto-riparanti è cruciale. Mentre

l'automazione può migliorare notevolmente l'efficienza, è importante che esperti umani revisionino e convalidino le correzioni effettuate dai modelli di IA, specialmente in domini critici o sensibili. Il giudizio umano e l'esperienza nel dominio possono aiutare a identificare e affrontare eventuali conseguenze indesiderate che potrebbero emergere.

4. **AI Spiegabile (XAI) e Trasparenza:** Come discusso nella sottosezione precedente, incorporare tecniche di AI spiegabile e garantire la trasparenza nel processo di dati auto-riparanti può aiutare a mitigare le conseguenze indesiderate. Fornendo spiegazioni chiare per le correzioni dei dati e mantenendo tracce di audit complete, le organizzazioni possono comprendere meglio e tracciare il ragionamento dietro le modifiche effettuate dai modelli di IA.
5. **Approccio Incrementale e Iterativo:** Adottare un approccio incrementale e iterativo ai dati auto-riparanti può aiutare a minimizzare il rischio di conseguenze indesiderate. Invece di applicare correzioni automatizzate all'intero set di dati in una volta sola, iniziare con un sottoinsieme di dati e gradualmente espandere l'ambito man mano che le tecniche si dimostrano efficaci e affidabili. Questo permette un attento monitoraggio e aggiustamento lungo il percorso, riducendo l'impatto di eventuali conseguenze indesiderate.
6. **Collaborazione e Feedback:** Coinvolgere gli stakeholder di diversi domini e incoraggiare la collaborazione e il feedback durante tutto il processo di dati auto-riparanti può aiutare a identificare e affrontare le conseguenze indesiderate. Cercare regolarmente input da esperti del dominio, consumatori di dati e utenti finali può fornire preziose informazioni sull'impatto nel mondo reale delle correzioni dei dati e evidenziare eventuali problemi che potrebbero essere stati trascurati.

Affrontando proattivamente il rischio di conseguenze indesiderate e implementando adeguate misure di sicurezza, le organizzazioni possono sfruttare i benefici delle tecniche di dati auto-riparanti minimizzando al contempo i potenziali effetti negativi. È

importante approcciare i dati auto-riparanti come un processo iterativo e collaborativo, monitorando, valutando e raffinando continuamente le tecniche per assicurare che si allineino con i risultati desiderati e mantengano l'integrità e l'affidabilità dei dati.

Quando si considera l'uso di pattern di dati auto-riparanti, è essenziale valutare attentamente questi fattori e sopesare i benefici rispetto ai potenziali rischi e limitazioni. In alcuni casi, un approccio ibrido che combina correzioni automatizzate con supervisione e intervento umano potrebbe essere la soluzione più appropriata.

Vale anche la pena notare che le tecniche di dati auto-riparanti non dovrebbero essere viste come un sostituto dei robusti meccanismi di convalida dei dati, sanificazione dell'input e gestione degli errori. Queste pratiche fondamentali rimangono critiche per garantire l'integrità e la sicurezza dei dati. I dati auto-riparanti dovrebbero essere visti come un approccio complementare che può aumentare e migliorare queste misure esistenti.

In definitiva, la decisione di impiegare pattern di dati auto-riparanti dipende dai requisiti specifici, dai vincoli e dalle priorità della vostra applicazione. Considerando attentamente le considerazioni delineate sopra e allineandole con gli obiettivi e l'architettura della vostra applicazione, potete prendere decisioni informate su quando e come sfruttare efficacemente le tecniche di dati auto-riparanti.

Generazione Contestuale dei Contenuti



I pattern di Generazione Contestuale dei Contenuti sfruttano la potenza dei modelli linguistici di grandi dimensioni (LLM) per generare contenuti dinamici e contestualmente specifici all'interno delle applicazioni. Questa categoria di pattern riconosce l'importanza di fornire contenuti personalizzati e pertinenti agli utenti in base alle loro specifiche esigenze, preferenze e persino alle interazioni precedenti e attuali con l'applicazione.

In questo contesto, il termine “contenuto” si riferisce sia ai contenuti primari (come post di blog, articoli, ecc.) sia ai meta-contenuti, come le raccomandazioni relative ai

contenuti primari.

I pattern di Generazione Contestuale dei Contenuti possono svolgere un ruolo cruciale nel migliorare i livelli di coinvolgimento degli utenti, fornendo esperienze personalizzate e automatizzando le attività di creazione dei contenuti sia per te che per i tuoi utenti. Utilizzando i pattern che descriviamo in questo capitolo, puoi creare applicazioni che generano contenuti in modo dinamico, adattandosi al contesto e agli input in tempo reale.

I pattern funzionano integrando gli LLM negli output dell'applicazione, dall'interfaccia utente (talvolta denominata "chrome"), alle email e altre forme di notifiche, fino a qualsiasi pipeline di generazione dei contenuti.

Quando un utente interagisce con l'applicazione o attiva una specifica richiesta di contenuto, l'applicazione acquisisce il contesto rilevante, come le preferenze dell'utente, le interazioni precedenti o prompt specifici. Queste informazioni contestuali vengono poi inserite nell'LLM, insieme a eventuali template o linee guida necessarie, e utilizzate per produrre output testuali che altrimenti dovrebbero essere codificati staticamente, memorizzati in un database o generati algoritmamente.

I contenuti generati dall'LLM possono assumere varie forme, come raccomandazioni personalizzate, descrizioni dinamiche dei prodotti, risposte email personalizzate o persino interi articoli o post di blog. Uno degli usi più radicali di questi contenuti che ho introdotto più di un anno fa è la generazione dinamica di elementi dell'interfaccia utente come etichette dei form, tooltip e altri tipi di testo esplicativo.

Personalizzazione

Uno dei principali vantaggi dei pattern di Generazione Contestuale dei Contenuti è la capacità di offrire esperienze altamente personalizzate agli utenti. Generando contenuti

basati sul contesto specifico dell'utente, questi pattern permettono alle applicazioni di adattare i contenuti agli interessi, alle preferenze e alle interazioni dei singoli utenti.

La personalizzazione va oltre il semplice inserimento del nome dell'utente in contenuti generici. Implica lo sfruttamento del ricco contesto disponibile su ogni utente per generare contenuti che risuonino con le loro specifiche esigenze e desideri. Questo contesto può includere una vasta gamma di fattori, come:

1. **Informazioni del Profilo Utente:** Al livello più generale di applicazione di questa tecnica, i dati demografici, gli interessi, le preferenze e altri attributi del profilo possono essere utilizzati per generare contenuti che si allineano con il background e le caratteristiche dell'utente.
2. **Dati Comportamentali:** Le interazioni passate dell'utente con l'applicazione, come le pagine visualizzate, i link cliccati o i prodotti acquistati, possono fornire preziose informazioni sui loro comportamenti e interessi. Questi dati possono essere utilizzati per generare suggerimenti di contenuti che riflettono i loro modelli di coinvolgimento e prevedono le loro esigenze future.
3. **Fattori Contestuali:** Il contesto attuale dell'utente, come la sua posizione, il dispositivo, l'ora del giorno o persino le condizioni meteorologiche, può influenzare il processo di generazione dei contenuti. Ad esempio, un'applicazione di viaggi potrebbe avere un sistema AI in grado di generare raccomandazioni personalizzate basate sulla posizione attuale dell'utente e sulle condizioni meteorologiche prevalenti.

Sfruttando questi fattori contestuali, i pattern di Generazione Contestuale dei Contenuti permettono alle applicazioni di fornire contenuti che sembrano creati su misura per ogni singolo utente. Questo livello di personalizzazione presenta diversi vantaggi significativi:

1. **Maggiore Coinvolgimento:** I contenuti personalizzati catturano l'attenzione degli utenti e li mantengono coinvolti nell'applicazione. Quando gli utenti sentono che i contenuti sono pertinenti e parlano direttamente alle loro esigenze, è più

probabile che trascorrono più tempo a interagire con l'applicazione e a esplorarne le funzionalità.

2. **Maggiore Soddisfazione dell'Utente:** I contenuti personalizzati dimostrano che l'applicazione comprende e si preoccupa delle esigenze uniche dell'utente. Fornendo contenuti utili, informativi e allineati con i loro interessi, l'applicazione può migliorare la soddisfazione degli utenti e costruire una connessione più forte con loro.
3. **Tassi di Conversione più Elevati:** Nel contesto di applicazioni e-commerce o di marketing, i contenuti personalizzati possono avere un impatto significativo sui tassi di conversione. Presentando agli utenti prodotti, offerte o raccomandazioni adattate alle loro preferenze e comportamenti, l'applicazione può aumentare la probabilità che gli utenti compiano le azioni desiderate, come effettuare un acquisto o iscriversi a un servizio.

Produttività

I pattern di Generazione Contestuale dei Contenuti possono aumentare significativamente certi tipi di produttività riducendo la necessità di generazione e modifica manuale dei contenuti nei processi creativi. Sfruttando la potenza degli LLM, puoi generare contenuti di alta qualità su larga scala, risparmiando tempo e sforzo che i tuoi creatori di contenuti e sviluppatori dovrebbero altrimenti spendere in tedioso lavoro manuale.

Tradizionalmente, i creatori di contenuti devono ricercare, scrivere, modificare e formattare i contenuti per garantire che soddisfino i requisiti dell'applicazione e le aspettative degli utenti. Questo processo può richiedere molto tempo e risorse, specialmente quando il volume dei contenuti aumenta.

Tuttavia, con i pattern di Generazione Contestuale dei Contenuti, il processo di creazione dei contenuti può essere ampiamente automatizzato. Gli LLM possono

generare contenuti coerenti, grammaticalmente corretti e contestualmente rilevanti sulla base dei prompt e delle linee guida fornite. Questa automazione offre diversi vantaggi in termini di produttività:

1. **Riduzione dello Sforzo Manuale:** Delegando i compiti di generazione dei contenuti agli LLM, i creatori di contenuti possono concentrarsi su attività di livello superiore come la strategia dei contenuti, l'ideazione e il controllo della qualità. Possono fornire all'LLM il contesto, i template e le linee guida necessarie e lasciare che si occupi della generazione effettiva dei contenuti. Questo riduce lo sforzo manuale richiesto per la scrittura e la modifica, permettendo ai creatori di contenuti di essere più produttivi ed efficienti.
2. **Creazione più Rapida dei Contenuti:** Gli LLM possono generare contenuti molto più velocemente degli scrittori umani. Con i prompt e le linee guida appropriate, un LLM può produrre molteplici contenuti in pochi secondi o minuti. Questa velocità permette alle applicazioni di generare contenuti a un ritmo molto più rapido, tenendo il passo con le richieste degli utenti e il panorama digitale in continua evoluzione.

La creazione più rapida di contenuti sta portando a una situazione di "tragedia dei beni comuni" in cui internet sta annegando in contenuti che nessuno legge? Purtroppo, sospetto che la risposta sia sì.

3. **Coerenza e Qualità:** Gli LLM possono facilmente revisionare i contenuti in modo che siano coerenti nello stile, nel tono e nella qualità. Fornendo linee guida ed esempi chiari, certi tipi di applicazioni (ad esempio redazioni giornalistiche, PR, ecc.) possono garantire che i loro contenuti generati dall'uomo si allineino con la voce del brand e soddisfino gli standard di qualità desiderati. Questa coerenza

riduce la necessità di modifiche e revisioni estese, risparmiando tempo e sforzo nel processo di creazione dei contenuti.

4. Iterazione e Ottimizzazione: I pattern di Generazione Contestuale dei Contenuti permettono una rapida iterazione e ottimizzazione dei contenuti. Modificando i prompt, i template o le linee guida fornite all'LLM, le vostre applicazioni possono generare rapidamente variazioni di contenuti e testare diversi approcci in modo automatizzato che non era mai stato possibile in passato. Questo processo iterativo permette una sperimentazione e un perfezionamento più rapidi delle strategie dei contenuti, portando nel tempo a contenuti più efficaci e coinvolgenti. Questa particolare tecnica può essere rivoluzionaria per applicazioni come l'e-commerce che vivono e muoiono in base ai tassi di abbandono e al coinvolgimento



È importante notare che mentre i pattern di Generazione Contestuale dei Contenuti possono migliorare notevolmente la produttività, non eliminano completamente la necessità di coinvolgimento umano. I creatori di contenuti e gli editor continuano a svolgere un ruolo cruciale nel definire la strategia complessiva dei contenuti, fornire guida all'LLM e garantire la qualità e l'appropriatezza dei contenuti generati.

Automatizzando gli aspetti più ripetitivi e dispendiosi in termini di tempo della creazione dei contenuti, i pattern di Generazione Contestuale dei Contenuti liberano prezioso tempo e risorse umane che possono essere reindirizzati verso attività di maggior valore. Questa maggiore produttività vi permette di fornire contenuti più personalizzati e coinvolgenti agli utenti ottimizzando al contempo i flussi di lavoro di creazione dei contenuti.

Iterazione Rapida e Sperimentazione

I pattern di Generazione Contestuale dei Contenuti vi permettono di iterare rapidamente e sperimentare con diverse variazioni di contenuto, consentendo un'ottimizzazione e

un perfezionamento più rapidi della vostra strategia dei contenuti. Potete generare multiple versioni di contenuto in pochi secondi, semplicemente modificando il contesto, i template o le linee guida fornite al modello.

Questa capacità di rapida iterazione offre diversi vantaggi chiave:

1. **Test e Ottimizzazione:** Con la capacità di generare rapidamente variazioni di contenuto, potete facilmente testare diversi approcci e misurarne l'efficacia. Per esempio, potete generare multiple versioni di una descrizione prodotto o di un messaggio di marketing, ciascuna adattata a uno specifico segmento di utenti o contesto. Analizzando le metriche di coinvolgimento degli utenti, come i tassi di click o di conversione, potete identificare le variazioni di contenuto più efficaci e ottimizzare di conseguenza la vostra strategia dei contenuti.
2. **Test A/B:** I pattern di Generazione Contestuale dei Contenuti permettono test A/B fluidi dei contenuti. Potete generare due o più variazioni di contenuto e mostrarle casualmente a diversi gruppi di utenti. Confrontando le performance di ciascuna variazione, potete determinare quale contenuto risuona meglio con il vostro pubblico target. Questo approccio basato sui dati vi permette di prendere decisioni informate e perfezionare continuamente i vostri contenuti per massimizzare il coinvolgimento degli utenti e raggiungere i risultati desiderati.
3. **Esperimenti di Personalizzazione:** L'iterazione rapida e la sperimentazione sono particolarmente preziose quando si tratta di personalizzazione. Con i pattern di Generazione Contestuale dei Contenuti, potete generare rapidamente variazioni di contenuto personalizzate basate su diversi segmenti di utenti, preferenze o comportamenti. Sperimentando con diverse strategie di personalizzazione, potete identificare gli approcci più efficaci per coinvolgere i singoli utenti e fornire esperienze su misura.
4. **Adattamento alle Tendenze in Evoluzione:** La capacità di iterare e sperimentare rapidamente ti permette di rimanere agile e adattarti ai cambiamenti nelle tendenze e nelle preferenze degli utenti. Quando emergono nuovi argomenti,

parole chiave o comportamenti degli utenti, puoi generare rapidamente contenuti allineati a queste tendenze. Attraverso la sperimentazione e il perfezionamento continuo dei tuoi contenuti, puoi rimanere rilevante e mantenere un vantaggio competitivo nel panorama digitale in continua evoluzione.

5. Sperimentazione Economicamente Vantaggiosa: La sperimentazione tradizionale dei contenuti spesso richiede tempo e risorse significative, poiché i creatori di contenuti devono sviluppare e testare manualmente diverse varianti. Tuttavia, con i modelli di Generazione Contestuale dei Contenuti, il costo della sperimentazione si riduce notevolmente. I modelli linguistici di grandi dimensioni possono generare variazioni di contenuto rapidamente e su larga scala, permettendoti di esplorare una vasta gamma di idee e approcci senza incorrere in costi sostanziali.

Per sfruttare al meglio l'iterazione e la sperimentazione rapida, è importante avere un framework di sperimentazione ben definito. Questo framework dovrebbe includere:

- Obiettivi e ipotesi chiari per ogni esperimento
- Metriche appropriate e meccanismi di monitoraggio per misurare le prestazioni dei contenuti
- Strategie di segmentazione e targeting per garantire che le variazioni di contenuto pertinenti vengano servite agli utenti giusti
- Strumenti di analisi e reporting per ricavare informazioni dai dati sperimentali
- Un processo per incorporare gli apprendimenti e le ottimizzazioni nella tua strategia dei contenuti

Abbracciando l'iterazione e la sperimentazione rapida, puoi perfezionare e ottimizzare continuamente i tuoi contenuti, assicurando che rimangano coinvolgenti, pertinenti ed efficaci nel raggiungere gli obiettivi della tua applicazione. Questo approccio agile alla creazione di contenuti ti permette di rimanere all'avanguardia e offrire esperienze utente eccezionali.

Scalabilità ed Efficienza

Man mano che le applicazioni crescono e aumenta la domanda di contenuti personalizzati, i modelli di generazione contestuale dei contenuti permettono una scalabile ed efficiente creazione di contenuti. I modelli linguistici di grandi dimensioni possono generare contenuti per un gran numero di utenti e contesti simultaneamente, senza la necessità di un aumento proporzionale delle risorse umane. Questa scalabilità permette alle applicazioni di offrire esperienze personalizzate a una base utenti in crescita senza sovraccaricare le loro capacità di creazione dei contenuti.



Nota che la generazione contestuale dei contenuti può essere utilizzata efficacemente per internazionalizzare la tua applicazione “al volo”. In effetti, è esattamente quello che ho fatto utilizzando la mia Gem Instant18n per distribuire Olympia in più di mezza dozzina di lingue, nonostante abbiano meno di un anno.

Localizzazione Basata su IA

Se mi permettete di vantarmi per un momento, penso che la mia libreria Instant18n per applicazioni Rails sia un esempio rivoluzionario del pattern “Generazione Contestuale dei Contenuti” in azione, che mostra il potenziale trasformativo dell’IA nello sviluppo di applicazioni. Questa gem sfrutta la potenza del modello linguistico GPT di OpenAI per rivoluzionare il modo in cui l’internazionalizzazione e la localizzazione vengono gestite nelle applicazioni Rails.

Tradizionalmente, internazionalizzare un’applicazione Rails comporta la definizione manuale di chiavi di traduzione e la fornitura delle corrispondenti traduzioni per ogni lingua supportata. Questo processo può richiedere molto tempo, essere dispendioso in termini di risorse e soggetto a incongruenze. Tuttavia, con la gem Instant18n, il paradigma della localizzazione viene completamente ridefinito.

Integrando un modello linguistico di grandi dimensioni, la gem Instant18n ti permette di generare traduzioni al volo, basate sul contesto e sul significato del testo. Invece di affidarsi a chiavi di traduzione predefinite e traduzioni statiche, la gem traduce dinamicamente il testo utilizzando la potenza dell'IA. Questo approccio offre diversi vantaggi chiave:

- 1. Localizzazione Senza Interruzioni:** Con la gem Instant18n, gli sviluppatori non devono più definire e mantenere manualmente i file di traduzione per ogni lingua supportata. La gem genera automaticamente le traduzioni basandosi sul testo fornito e sulla lingua di destinazione desiderata, rendendo il processo di localizzazione senza sforzo e fluido.
- 2. Accuratezza Contestuale:** L'IA può ricevere contesto sufficiente per comprendere le sfumature del testo da tradurre. Può tenere conto del contesto circostante, degli idiomi e dei riferimenti culturali per generare traduzioni accurate, che suonano naturali e contestualmente appropriate.
- 3. Supporto Linguistico Esteso:** La gem Instant18n sfrutta le vaste conoscenze e capacità linguistiche di GPT, permettendo traduzioni in un'ampia gamma di lingue. Dalle lingue comuni come lo spagnolo e il francese a lingue più oscure o fittizie come il Klingon e l'Elfico, la gem può gestire una vasta varietà di requisiti di traduzione.
- 4. Flessibilità e Creatività:** La gem va oltre le traduzioni linguistiche tradizionali e permette opzioni di localizzazione creative e non convenzionali. Gli sviluppatori possono tradurre il testo in vari stili, dialetti o persino lingue fittizie, aprendo nuove possibilità per esperienze utente uniche e contenuti coinvolgenti.
- 5. Ottimizzazione delle Prestazioni:** La gem Instant18n incorpora meccanismi di caching per migliorare le prestazioni e ridurre il sovraccarico di traduzioni ripetute. Il testo tradotto viene memorizzato nella cache, permettendo che le successive richieste per la stessa traduzione vengano servite rapidamente senza la necessità di chiamate API ridondanti.

La gem Instant18n esemplifica la potenza del pattern “Generazione Contestuale dei Contenuti” sfruttando l’IA per generare contenuti localizzati dinamicamente. Dimostra come l’IA possa essere integrata nella funzionalità core di un’applicazione Rails, trasformando il modo in cui gli sviluppatori affrontano l’internazionalizzazione e la localizzazione.

Eliminando la necessità di gestire manualmente le traduzioni e consentendo traduzioni istantanee basate sul contesto, la gem Instant18n fa risparmiare agli sviluppatori tempo e fatica significativi. Permette loro di concentrarsi sulla costruzione delle funzionalità principali dell’applicazione, garantendo al contempo che l’aspetto della localizzazione sia gestito in modo fluido e accurato.

L’Importanza dei Test Utente e del Feedback

Infine, tieni sempre presente l’importanza dei test utente e del feedback. È fondamentale verificare che la generazione di contenuti contestuali soddisfi le aspettative degli utenti e si allinei con gli obiettivi dell’applicazione. Continua a iterare e perfezionare i contenuti generati sulla base degli approfondimenti degli utenti e delle analisi dei dati. Se stai generando contenuti dinamici su larga scala che sarebbe impossibile validare manualmente da te e dal tuo team, considera l’aggiunta di meccanismi di feedback che permettano agli utenti di segnalare contenuti strani o errati, insieme a una spiegazione del perché. Questo prezioso feedback può persino essere utilizzato da un worker AI incaricato di apportare modifiche al componente che ha generato il contenuto!

Generative UI



L'attenzione è così preziosa al giorno d'oggi che un coinvolgimento efficace dell'utente richiede esperienze software non solo fluide e intuitive, ma anche altamente personalizzate in base alle esigenze, preferenze e contesti individuali. Di conseguenza, designer e sviluppatori si trovano sempre più spesso ad affrontare la sfida di creare interfacce utente in grado di adattarsi e rispondere alle esigenze uniche di ciascun utente *su larga scala*.

Il Generative UI (GenUI) è un approccio veramente rivoluzionario alla progettazione dell'interfaccia utente che sfrutta la potenza dei modelli linguistici di grandi dimensioni (LLM) per creare esperienze utente altamente personalizzate e dinamiche al momento. Ho voluto assicurarmi di darti almeno un'introduzione al GenUI in questo libro, perché credo che sia una delle opportunità più promettenti che esistono attualmente nel campo della progettazione e dei framework delle applicazioni. Sono convinto che decine o più

nuovi progetti commerciali e open-source di successo emergeranno in questa particolare nicchia.

Nel suo nucleo, GenUI combina i principi della **Generazione di Contenuti Contestuale** con tecniche di AI avanzate per generare elementi dell’interfaccia utente, come testo, immagini e layout, in modo dinamico basandosi su una profonda comprensione del contesto, delle preferenze e degli obiettivi dell’utente. GenUI permette ai designer e agli sviluppatori di creare interfacce che si adattano e si evolvono in risposta alle interazioni dell’utente, fornendo un livello di personalizzazione precedentemente irraggiungibile.

GenUI rappresenta un cambiamento fondamentale nel modo in cui affrontiamo la progettazione dell’interfaccia utente. Invece di progettare per le masse, GenUI ci permette di progettare per l’individuo. I contenuti e le interfacce personalizzate hanno il potenziale di creare esperienze utente che risuonano con ciascun utente a un livello più profondo, aumentando il coinvolgimento, la soddisfazione e la fedeltà.

Come tecnica all'avanguardia, la transizione al GenUI è piena di sfide concettuali e pratiche. L'integrazione dell'AI nel processo di progettazione, garantendo che le interfacce generate siano non solo personalizzate ma anche utilizzabili, accessibili e allineate con il brand complessivo e l'esperienza utente, sono tutte sfide che rendono GenUI una ricerca per pochi, non per molti. Inoltre, il coinvolgimento dell'AI solleva questioni sulla privacy dei dati, la trasparenza e persino implicazioni etiche.

Nonostante le sfide, le esperienze personalizzate su larga scala hanno il potere di trasformare completamente il modo in cui interagiamo con prodotti e servizi digitali. Apre possibilità per creare interfacce inclusive e accessibili che soddisfano le diverse esigenze degli utenti, indipendentemente dalle loro capacità, background o preferenze.

In questo capitolo, esploreremo il concetto di GenUI, esaminando alcune caratteristiche distintive, benefici chiave e potenziali sfide. Iniziamo considerando la forma più basilare e accessibile di GenUI: la generazione di testo per interfacce utente progettate e implementate in modo tradizionale.

Generazione di Testo per Interfacce Utente

Gli elementi testuali presenti nell’interfaccia della tua applicazione, come le etichette dei moduli, i suggerimenti e il testo esplicativo, sono tipicamente hardcodati nei template o nei componenti UI, fornendo un’esperienza coerente ma generica per tutti gli utenti. Utilizzando i pattern di generazione di contenuti contestuali, puoi trasformare questi elementi statici in componenti dinamici, consapevoli del contesto e personalizzati.

Moduli Personalizzati

I moduli sono una parte onnipresente delle applicazioni web e mobile, servendo come principale mezzo di raccolta dell’input dell’utente. Tuttavia, i moduli tradizionali spesso presentano un’esperienza generica e impersonale, con etichette e campi standard che potrebbero non sempre allinearsi con il contesto specifico o le esigenze dell’utente. Gli utenti sono più propensi a completare moduli che sembrano adattati alle loro esigenze e preferenze, portando a tassi di conversione e soddisfazione più elevati.

Tuttavia, è importante trovare un equilibrio tra personalizzazione e coerenza. Mentre l’adattamento dei moduli ai singoli utenti può essere vantaggioso, è cruciale mantenere un livello di familiarità e prevedibilità. Gli utenti dovrebbero ancora essere in grado di riconoscere e navigare facilmente i moduli, anche con elementi personalizzati.

Ecco alcune idee di moduli personalizzati per l’ispirazione:

Suggerimenti Contestuali dei Campi

GenUI può analizzare le interazioni precedenti dell’utente, le preferenze e i dati per fornire suggerimenti intelligenti dei campi come previsioni. Per esempio, se l’utente ha precedentemente inserito il proprio indirizzo di spedizione, il modulo può automaticamente popolare i campi pertinenti con le informazioni salvate. Questo non solo fa risparmiare tempo ma dimostra anche che l’applicazione comprende e ricorda le preferenze dell’utente.

Un momento, questa tecnica non potrebbe essere realizzata anche senza coinvolgere l'IA? Certamente, ma la bellezza di gestire questo tipo di funzionalità con l'IA è duplice: 1) quanto può essere facile da implementare e 2) quanto può essere resiliente mentre la tua UI cambia ed evolve nel tempo.

Creiamo velocemente un servizio per il nostro sistema teorico di gestione degli ordini, che cerca di compilare proattivamente l'indirizzo di spedizione corretto per l'utente.

```
1  class OrderShippingAddressSubscriber
2    include Raix::ChatCompletion
3
4    attr_accessor :order
5
6    delegate :customer, to: :order
7
8    DIRECTIVE = "You are a smart order processing assistant. Given the
9    customer's order history, guess the most likely shipping address
10   for the current order."
11
12  def order_created(order)
13    return unless order.pending? && order.shipping_address.blank?
14
15    self.order = order
16
17    transcript.clear
18    transcript << { system: DIRECTIVE }
19    transcript << { user: "Order History: #{order_history.to_json}" }
20    transcript << { user: "Current Order: #{order.to_json}" }
21
22    response = chat_completion
23    apply_predicted_shipping_address(order, response)
24  end
25
26  private
27
28  def apply_predicted_shipping_address(order, response)
29    # extract the shipping address from the response...
30    # ...and assume there's some sort of live update of the address fields
31    order.update(shipping_address:)
32  end
```

```
33
34     def order_history
35         customer.orders.successful.limit(100).map do |order|
36             {
37                 date: order.date,
38                 description: order.description,
39                 shipping_address: order.shipping_address
40             }
41         end
42     end
43 end
```

Questo esempio è molto semplificato, ma dovrebbe funzionare nella maggior parte dei casi. L'idea è di lasciare che l'AI faccia una supposizione nello stesso modo in cui lo farebbe un essere umano. Per chiarire di cosa sto parlando, consideriamo alcuni dati di esempio:

```
1 Order History:
2 [
3     {"date": "2024-01-03", "description": "garden soil mix",
4      "shipping_address": "123 Country Lane, Rural Town"},
5     {"date": "2024-01-15", "description": "hardcover fiction novels",
6      "shipping_address": "456 City Apt, Metroville"},
7     {"date": "2024-01-22", "description": "baby diapers", "shipping_address":
8      "789 Suburb St, Quietville"},
9     {"date": "2024-02-01", "description": "organic vegetables",
10    "shipping_address": "123 Country Lane, Rural Town"},
11    {"date": "2024-02-17", "description": "mystery thriller book set",
12    "shipping_address": "456 City Apt, Metroville"},
13    {"date": "2024-02-25", "description": "baby wipes",
14    "shipping_address": "789 Suburb St, Quietville"},
15    {"date": "2024-03-05", "description": "flower seeds",
16    "shipping_address": "123 Country Lane, Rural Town"},
17    {"date": "2024-03-20", "description": "biographies",
18    "shipping_address": "456 City Apt, Metroville"},
19    {"date": "2024-03-30", "description": "baby formula",
20    "shipping_address": "789 Suburb St, Quietville"},
21    {"date": "2024-04-12", "description": "lawn fertilizer",
22    "shipping_address": "123 Country Lane, Rural Town"},
23    {"date": "2024-04-22", "description": "science fiction novels",
```

```
24     "shipping_address": "456 City Apt, Metroville"},  
25     {"date": "2024-05-02", "description": "infant toys",  
26     "shipping_address": "789 Suburb St, Quietville"},  
27     {"date": "2024-05-14", "description": "outdoor grill",  
28     "shipping_address": "123 Country Lane, Rural Town"},  
29     {"date": "2024-05-29", "description": "literary classics",  
30     "shipping_address": "456 City Apt, Metroville"},  
31     {"date": "2024-06-11", "description": "baby clothes",  
32     "shipping_address": "789 Suburb St, Quietville"},  
33     {"date": "2024-07-01", "description": "watering can",  
34     "shipping_address": "123 Country Lane, Rural Town"},  
35     {"date": "2024-07-18", "description": "non-fiction essays",  
36     "shipping_address": "456 City Apt, Metroville"},  
37     {"date": "2024-07-28", "description": "baby bath items",  
38     "shipping_address": "789 Suburb St, Quietville"},  
39     {"date": "2024-08-09", "description": "herb garden kit",  
40     "shipping_address": "123 Country Lane, Rural Town"},  
41     {"date": "2024-08-24", "description": "children's books",  
42     "shipping_address": "456 City Apt, Metroville"}  
43 ]
```

Hai notato lo schema nei dati? Ti garantisco che questo è materiale facilissimo per un LLM. Per dimostrarlo, chiediamo a GPT-4 qual è l'indirizzo di spedizione più probabile per un “termometro”.

```
1 From the order history you've provided, it looks like the purchases are  
2 generally clustered into three main types based on the shipping addresses:  
3  
4 123 Country Lane, Rural Town - This address often orders garden and  
5 outdoor-related items like soil mix, vegetables, flower seeds, lawn  
6 fertilizer, an outdoor grill, a watering can, and an herb garden kit.  
7  
8 456 City Apt, Metroville - This address tends to order various types of  
9 books, indicating an interest in reading across genres like fiction,  
10 thrillers, biographies, science fiction, literary classics, non-fiction  
11 essays, and children's books.  
12  
13 789 Suburb St, Quietville - The purchases for this address are consistently  
14 baby-related products, such as diapers, wipes, baby formula, infant toys,  
15 baby clothes, and baby bath items.
```

16
17 Given these patterns, a digital thermometer could potentially be a fit for
18 any of the addresses based on different needs: outdoor temperature tracking,
19 a tool for a passionate reader interested in science gadgets, or a health
20 tool for a baby. However, considering its common use in monitoring health,
21 especially for babies, it's most likely to be purchased by someone who's
22 already buying baby care products.

23
24 Therefore, the most likely shipping address for a digital thermometer, based
25 on the purchasing patterns, would be 789 Suburb St, Quietville.

Se stai pensando che sia esagerato utilizzare un modello costoso come GPT-4 per questo compito, hai ragione! Ho provato lo stesso prompt su Mistral 7B Instruct e ha prodotto la seguente risposta a 75 token al secondo, e al modesto costo di \$0.000218 USD.

1 Based on the given order history, it appears that the customer has received
2 orders for baby-related items (diapers, wipes, formula, clothes, bath items)
3 at the address "789 Suburb St, Quietville" and garden-related items (soil,
4 seeds, watering can, herb garden kit) at the address "123 Country Lane, Rural
5 Town". The customer has also received books (fiction, non-fiction, literary
6 classics, mystery-thriller, science fiction, biographies, children's books)
7 at the address "456 City Apt, Metroville".

8
9 Given that the new order is for a "thermometer", which doesn't seem to fit
10 into any of the categories that the customer has previously ordered, it's
11 hard to accurately predict the shipping address based solely on the order
12 history. However, given the 50%-50% split between baby-related and
13 garden-related items, it could somewhat lean towards the Baby-related items
14 address ("789 Suburb St, Quietville"). But remember, this is an assumption
15 and cannot be definitively confirmed without more context or information.

Il costo e l'impegno di questa tecnica valgono la pena per rendere più magica l'esperienza di checkout? Per molti rivenditori online, assolutamente sì. E a quanto pare, il costo dell'elaborazione AI continuerà solo a diminuire, specialmente per i fornitori di hosting di modelli open source in una corsa al ribasso.



Utilizza un [Prompt Template](#) e [StructuredIO](#) insieme al [Response Fencing](#) per ottimizzare questo tipo di completamento chat.

Ordinamento Adattivo dei Campi

L'ordine in cui vengono presentati i campi del modulo può influenzare significativamente l'esperienza dell'utente e i tassi di completamento. Con GenUI, puoi regolare dinamicamente l'ordine dei campi in base al contesto dell'utente e all'importanza di ciascun campo. Ad esempio, se l'utente sta compilando un modulo di registrazione per un'app fitness, il modulo potrebbe dare priorità ai campi relativi ai suoi obiettivi e preferenze fitness, rendendo il processo più rilevante e coinvolgente.

Microcopy Personalizzato

Il testo istruttivo, i messaggi di errore e altri microcopy associati ai moduli possono essere personalizzati utilizzando GenUI. Invece di mostrare messaggi di errore generici come "Indirizzo email non valido", puoi generare messaggi più utili e contestuali come "Inserisci un indirizzo email valido per ricevere la conferma del tuo ordine". Questi tocchi personalizzati possono rendere l'esperienza del modulo più user-friendly e meno frustrante.

Validazione Personalizzata

Sulla stessa linea del Microcopy Personalizzato, potresti utilizzare l'AI per validare il modulo in modi che sembrano magici. Immagina di lasciare che un'AI validi un modulo del profilo utente, cercando potenziali errori a livello *semantico*.

Create your account

Full name

Obie Fernandez

Email

obiefenandez@gmail.com



Did you mean obiefernandez@gmail.com? [Yes, update.](#)

Country

United States



Password

.....



Nice work. This is an excellent password.

Figura 9. Riesci a individuare la validazione semantica in azione?

Rivelazione Progressiva

GenUI può determinare in modo intelligente quali campi del modulo sono essenziali in base al contesto dell'utente e rivelare gradualmente campi aggiuntivi secondo necessità. Questa tecnica di rivelazione progressiva aiuta a ridurre il carico cognitivo e rende il processo di compilazione del modulo più gestibile. Per esempio, se un utente si sta

iscrivendo a un abbonamento base, il modulo può inizialmente presentare solo i campi essenziali, e man mano che l'utente procede o seleziona opzioni specifiche, possono essere introdotti dinamicamente campi aggiuntivi pertinenti.

Testo Esplicativo Contestuale

I tooltip vengono spesso utilizzati per fornire informazioni aggiuntive o guida agli utenti quando passano il mouse sopra o interagiscono con elementi specifici. Con un approccio di “Generazione di Contenuti Contestuali”, puoi generare tooltip che si adattano al contesto dell’utente e forniscono informazioni rilevanti. Ad esempio, se un utente sta esplorando una funzionalità complessa, il tooltip può offrire suggerimenti personalizzati o esempi basati sulle sue interazioni precedenti o sul livello di competenza.

Il testo esplicativo, come istruzioni, descrizioni o messaggi di aiuto, può essere generato dinamicamente in base al contesto dell’utente. Invece di presentare spiegazioni generiche, puoi utilizzare i LLM per generare testo su misura per le esigenze o domande specifiche dell’utente. Per esempio, se un utente sta avendo difficoltà con un particolare passaggio in un processo, il testo esplicativo può fornire una guida personalizzata o suggerimenti per la risoluzione dei problemi.

Il microcopy si riferisce ai piccoli pezzi di testo che guidano gli utenti attraverso la tua applicazione, come etichette dei pulsanti, messaggi di errore o prompt di conferma. Applicando l’approccio di [Generazione di Contenuti Contestuali](#) al microcopy, puoi creare un’interfaccia utente adattiva che risponde alle azioni dell’utente e fornisce testo rilevante e utile. Ad esempio, se un utente sta per eseguire un’azione critica, il prompt di conferma può essere generato dinamicamente per fornire un messaggio chiaro e personalizzato.

Il testo esplicativo personalizzato e i tooltip possono migliorare notevolmente il processo di onboarding per i nuovi utenti. Fornendo una guida contestuale specifica ed esempi, puoi aiutare gli utenti a comprendere e navigare rapidamente nell’applicazione, riducendo la curva di apprendimento e aumentando l’adozione.

Gli elementi dell’interfaccia dinamici e sensibili al contesto possono anche rendere l’applicazione più intuitiva e coinvolgente. Gli utenti sono più propensi a interagire con ed esplorare le funzionalità quando il testo di accompagnamento è su misura per le loro esigenze e interessi specifici.

Finora abbiamo esaminato idee per migliorare i paradigmi UI esistenti con l’IA, ma cosa dire del ripensare in modo più radicale come le interfacce utente vengono progettate e implementate?

Definizione della UI Generativa

A differenza della progettazione UI tradizionale, dove i designer creano interfacce fisse e statiche, la GenUI suggerisce un futuro in cui il nostro software vanta esperienze flessibili e personalizzate che possono evolversi e adattarsi in tempo reale. Ogni volta che utilizziamo un’interfaccia conversazionale basata su IA, permettiamo all’IA di adattarsi alle particolari esigenze dell’utente. La GenUI fa un ulteriore passo avanti applicando quel livello di adattabilità all’interfaccia *visiva* del software.

Il motivo per cui è possibile sperimentare con le idee della GenUI oggi è che i modelli linguistici di grandi dimensioni già comprendono la programmazione e la loro conoscenza di base include tecnologie e framework UI. La domanda è quindi se i modelli linguistici di grandi dimensioni possano essere utilizzati per generare elementi UI, come testo, immagini, layout e persino intere interfacce, che siano su misura per ogni singolo utente. Il modello potrebbe essere istruito per tenere conto di vari fattori, come le interazioni passate dell’utente, le preferenze dichiarate, le informazioni demografiche e il contesto attuale di utilizzo, per creare interfacce altamente personalizzate e pertinenti.

La GenUI si differenzia dalla progettazione tradizionale dell’interfaccia utente in diversi aspetti chiave:

- 1. Dinamica e Adattiva:** La progettazione UI tradizionale implica la creazione di interfacce fisse e statiche che rimangono le stesse per tutti gli utenti. Al contrario, la GenUI permette interfacce che possono adattarsi e cambiare dinamicamente in base alle esigenze e al contesto dell'utente. Questo significa che la stessa applicazione può presentare interfacce diverse a utenti diversi o persino allo stesso utente in situazioni diverse.
- 2. Personalizzazione su Larga Scala:** Con il design tradizionale, creare esperienze personalizzate per ogni utente è spesso impraticabile a causa del tempo e delle risorse richieste. La GenUI, d'altra parte, permette la personalizzazione su larga scala. Sfruttando l'IA, i designer possono creare interfacce che si adattano automaticamente alle esigenze e preferenze uniche di ogni utente, senza dover progettare e sviluppare manualmente interfacce separate per ogni segmento di utenti.
- 3. Focus sui Risultati:** Il design UI tradizionale spesso si concentra sulla creazione di interfacce visivamente attraenti e funzionali. Mentre questi aspetti sono ancora importanti nella GenUI, il focus principale si sposta verso il raggiungimento dei risultati desiderati dall'utente. La GenUI mira a creare interfacce ottimizzate per gli obiettivi e i compiti specifici di ogni utente, dando priorità all'usabilità e all'efficacia rispetto a considerazioni puramente estetiche.
- 4. Apprendimento e Miglioramento Continuo:** I sistemi GenUI possono apprendere e migliorare continuamente nel tempo basandosi sulle interazioni e sul feedback degli utenti. Mentre gli utenti interagiscono con le interfacce generate, i modelli di IA possono raccogliere dati sul comportamento, le preferenze e i risultati degli utenti, utilizzando queste informazioni per raffinare e ottimizzare le future generazioni di interfacce. Questo processo di apprendimento iterativo permette ai sistemi GenUI di diventare sempre più efficaci nel soddisfare le esigenze degli utenti nel tempo.

È importante notare che la GenUI non è la stessa cosa degli strumenti di design assistiti dall'IA, come quelli che forniscono suggerimenti o automatizzano determinate attività

di design. Mentre questi strumenti possono essere utili per semplificare il processo di design, si basano ancora sui designer per prendere le decisioni finali e creare interfacce statiche. La GenUI, invece, prevede che il sistema di IA assuma un ruolo più attivo nella generazione e nell'adattamento effettivo delle interfacce basate sui dati e sul contesto dell'utente.

La GenUI rappresenta un cambiamento significativo nel modo in cui affrontiamo la progettazione dell'interfaccia utente, allontanandoci dalle soluzioni uniformi e muovendoci verso esperienze altamente personalizzate e adattive. Sfruttando la potenza dell'IA, la GenUI ha il potenziale per rivoluzionare il modo in cui interagiamo con prodotti e servizi digitali, creando interfacce più intuitive, coinvolgenti ed efficaci per ogni singolo utente.

Esempio

Per illustrare il concetto di GenUI, consideriamo un'applicazione fitness ipotetica chiamata "FitAI". Questa app mira a fornire piani di allenamento personalizzati e consigli nutrizionali agli utenti basati sui loro obiettivi individuali, livelli di fitness e preferenze.

In un approccio di design UI tradizionale, FitAI potrebbe avere un set fisso di schermate ed elementi che sono uguali per tutti gli utenti. Tuttavia, con la GenUI, l'interfaccia dell'app potrebbe adattarsi dinamicamente alle esigenze e al contesto unici di ciascun utente.

Questo approccio è difficile da immaginare implementato nel 2024 e potrebbe non avere nemmeno un ROI adeguato, ma è possibile.

Ecco come potrebbe funzionare:

1. **Onboarding:**

- Invece di un questionario standard, FitAI utilizza un'IA conversazionale per raccogliere informazioni sugli obiettivi dell'utente, il livello di fitness attuale e le preferenze.
- Basandosi su questa interazione iniziale, l'IA genera un layout della dashboard personalizzato, evidenziando le funzionalità e le informazioni più rilevanti per gli obiettivi dell'utente.
- La tecnologia IA attuale potrebbe avere a disposizione una selezione di componenti dello schermo da utilizzare nella composizione della dashboard personalizzata.
- La tecnologia IA futura potrebbe assumere il ruolo di un designer UI esperto e creare effettivamente la dashboard *da zero*.

2. Pianificatore di allenamento:

- L'interfaccia del pianificatore di allenamento viene adattata dall'IA in base al livello di esperienza dell'utente e all'attrezzatura disponibile.
- Per un principiante senza attrezzatura, potrebbe mostrare semplici esercizi a corpo libero con istruzioni dettagliate e video.
- Per un utente avanzato con accesso a una palestra, potrebbe visualizzare routine più complesse con meno contenuti esplicativi.
- Il contenuto del pianificatore di allenamento non è semplicemente filtrato da un grande insieme. Può essere generato *al volo* basandosi su una base di conoscenza che viene interrogata con un contesto che include tutto ciò che si sa dell'utente.

3. Monitoraggio dei progressi:

- L'interfaccia di monitoraggio dei progressi si evolve in base agli obiettivi dell'utente e ai modelli di coinvolgimento.
- Se un utente è principalmente concentrato sulla perdita di peso, l'interfaccia potrebbe mostrare in modo prominente un grafico dell'andamento del peso e statistiche sul consumo calorico.

- Per un utente che sta sviluppando massa muscolare, potrebbe evidenziare i guadagni di forza e i cambiamenti nella composizione corporea.
- L'IA può adattare questa parte dell'applicazione ai progressi effettivi dell'utente. Se i progressi si fermano per un periodo di tempo, l'app può passare a una modalità in cui cerca di indurre l'utente a rivelare le ragioni della battuta d'arresto, al fine di mitigarle.

4. Consigli nutrizionali:

- La sezione nutrizionale si adatta alle preferenze e restrizioni alimentari dell'utente.
- Per un utente vegano, potrebbe mostrare suggerimenti per pasti a base vegetale e fonti di proteine.
- Per un utente con intolleranza al glutine, filtrerebbe automaticamente gli alimenti contenenti glutine dalle raccomandazioni.
- Anche in questo caso, il contenuto non viene estratto da un enorme insieme di dati sui pasti applicabile a tutti gli utenti, ma viene sintetizzato da una base di conoscenza che contiene informazioni adattabili in base alla situazione specifica e ai vincoli dell'utente.
- Per esempio, le ricette vengono generate con specifiche degli ingredienti che corrispondono alle necessità caloriche in continua evoluzione dell'utente man mano che il suo livello di fitness e le sue statistiche corporee si evolvono.

5. Elementi motivazionali:

- I contenuti motivazionali e le notifiche dell'app sono personalizzati in base al tipo di personalità dell'utente e alla sua risposta a diverse strategie motivazionali.
- Alcuni utenti potrebbero ricevere messaggi incoraggianti, mentre altri ottengono feedback più orientati ai dati.

In questo esempio, GenUI permette a FitAI di creare un'esperienza altamente personalizzata per ogni utente, potenzialmente aumentando il coinvolgimento, la soddisfazione e la probabilità di raggiungere gli obiettivi di fitness. Gli elementi dell'interfaccia, i contenuti e persino la “personalità” dell'app si adattano per servire al meglio le esigenze e le preferenze di ogni singolo utente.

Il Passaggio al Design Orientato ai Risultati

GenUI rappresenta un cambiamento fondamentale nell'approccio al design dell'interfaccia utente, passando da un focus sulla creazione di elementi specifici dell'interfaccia a un approccio più olistico e orientato ai risultati. Questo cambiamento ha diverse implicazioni importanti:

1. Focus sugli Obiettivi dell'Utente:

- I designer dovranno pensare più profondamente agli obiettivi degli utenti e ai risultati desiderati piuttosto che a specifici componenti dell'interfaccia.
- L'enfasi sarà sulla creazione di sistemi che possano generare interfacce che aiutino gli utenti a raggiungere i loro obiettivi in modo efficiente ed efficace.
- Emergeranno nuovi framework UI che forniranno ai designer basati sull'IA gli strumenti necessari per generare esperienze utente *al volo* e *da zero* invece che basarsi su specifiche predefinite delle schermate.

2. Ruolo Mutevole dei Designer:

- I designer passeranno dalla creazione di layout fissi alla definizione di regole, vincoli e linee guida che i sistemi di IA devono seguire durante la generazione delle interfacce.
- Dovranno sviluppare competenze in aree come l'analisi dei dati, l'ingegneria dei prompt e il pensiero sistematico per guidare efficacemente i sistemi GenUI.

3. Importanza della Ricerca sull'Utente:

- La ricerca sull'utente diventa ancora più critica in un contesto GenUI, poiché i designer devono comprendere non solo le preferenze degli utenti, ma anche come queste preferenze e necessità cambiano in diversi contesti.
- Test continui con gli utenti e cicli di feedback saranno essenziali per perfezionare e migliorare la capacità dell'IA di generare interfacce efficaci.

4. Progettare per la Variabilità:

- Invece di creare una singola interfaccia “perfetta”, i designer dovranno considerare molteplici possibili variazioni e assicurarsi che il sistema possa generare interfacce appropriate per diverse esigenze degli utenti.
- Questo include la progettazione per i casi limite e garantire che le interfacce generate mantengano usabilità e accessibilità attraverso diverse configurazioni.
- La differenziazione del prodotto assume nuove dimensioni che coinvolgono prospettive divergenti sulla psicologia dell'utente e lo sfruttamento di set di dati e basi di conoscenza unici non disponibili ai concorrenti.

Sfide e Considerazioni

Mentre GenUI offre possibilità entusiasmanti, presenta anche diverse sfide e considerazioni:

1. Limitazioni Tecniche:

- La tecnologia IA attuale, sebbene avanzata, ha ancora limitazioni nella comprensione di intenti complessi dell'utente e nella generazione di interfacce veramente sensibili al contesto.

- Problemi di prestazioni relativi alla generazione in tempo reale di elementi dell'interfaccia, specialmente su dispositivi meno potenti.

2. Requisiti dei Dati:

- A seconda del caso d'uso, i sistemi GenUI efficaci potrebbero richiedere quantità significative di dati degli utenti per generare interfacce personalizzate.
- Le sfide nel reperire eticamente dati autentici degli utenti sollevano preoccupazioni sulla privacy e sicurezza dei dati, oltre a potenziali pregiudizi nei dati utilizzati per addestrare i modelli GenUI.

3. Usabilità e Coerenza:

- Almeno fino a quando la pratica non diventerà diffusa, un'applicazione con interfacce in continuo cambiamento potrebbe portare a problemi di usabilità, poiché gli utenti potrebbero faticare a trovare elementi familiari o a navigare efficacemente.
- Sarà cruciale trovare un equilibrio tra personalizzazione e mantenimento di un'interfaccia coerente e apprendibile.

4. Eccessiva Dipendenza dall'IA:

- Esiste il rischio di delegare eccessivamente le decisioni di design ai sistemi di IA, portando potenzialmente a scelte di interfaccia non ispirate, problematiche o semplicemente non funzionanti.
- La supervisione umana e la capacità di sovrascrivere i design generati dall'IA rimarranno importanti nel prossimo futuro.

5. Preoccupazioni sull'Accessibilità:

- Garantire che le interfacce generate dinamicamente rimangano accessibili agli utenti con disabilità presenta sfide completamente nuove, il che è preoccupante dato lo scarso livello di conformità all'accessibilità dimostrato dai sistemi tipici.
- D'altra parte, i designer IA potrebbero essere implementati con una preoccupazione *incorporata* per l'accessibilità e capacità di costruire interfacce accessibili al volo proprio come costruiscono UI per utenti non disabili.
- In ogni caso, i sistemi GenUI dovrebbero essere progettati con solide linee guida per l'accessibilità e processi di test.

6. Fiducia degli Utenti e Trasparenza:

- Gli utenti potrebbero sentirsi a disagio con interfacce che sembrano “sapere troppo” su di loro o che cambiano in modi che non comprendono.
- Fornire trasparenza su come e perché le interfacce sono personalizzate sarà importante per costruire la fiducia degli utenti.

Prospettive Future e Opportunità

Il futuro dell'Interfaccia Generativa (GenUI) racchiude immense promesse per rivoluzionare il modo in cui interagiamo con prodotti e servizi digitali. Mentre questa tecnologia continua a evolversi, possiamo anticipare un cambiamento radicale nel modo in cui le interfacce utente vengono progettate, implementate e vissute. Penso che GenUI sia il fenomeno che finalmente spingerà il nostro software nel regno di ciò che ora è considerato fantascienza.

Una delle prospettive più entusiasmanti di GenUI è il suo potenziale di migliorare l'accessibilità su una scala che va oltre il semplice assicurarsi che le persone con disabilità gravi non siano completamente escluse dall'uso del software. Adattando

automaticamente le interfacce alle esigenze individuali degli utenti, GenUI potrebbe rendere le esperienze digitali più inclusive che mai. Immaginate interfacce che si adattano senza problemi per fornire testo più grande per utenti giovani o ipovedenti o layout semplificati per chi ha disabilità cognitive, tutto senza richiedere configurazioni manuali o versioni “accessibili” separate delle applicazioni.

Le capacità di personalizzazione di GenUI probabilmente guideranno un aumento del coinvolgimento, della soddisfazione e della fedeltà degli utenti attraverso una vasta gamma di prodotti digitali. Man mano che le interfacce si sintonizzano maggiormente con le preferenze e i comportamenti individuali, gli utenti troveranno le esperienze digitali più intuitive e piacevoli, portando potenzialmente a interazioni più profonde e significative con la tecnologia.

GenUI ha anche il potenziale di trasformare il processo di onboarding per i nuovi utenti. Creando esperienze intuitive e personalizzate per i nuovi utenti che si adattano rapidamente al livello di competenza di ciascun utente, GenUI potrebbe ridurre significativamente la curva di apprendimento associata alle nuove applicazioni. Questo potrebbe portare a tassi di adozione più rapidi e una maggiore fiducia degli utenti nell'esplorare nuove funzionalità.

Un'altra possibilità entusiasmante è la capacità di GenUI di mantenere un'esperienza utente coerente su diversi dispositivi e piattaforme ottimizzando per ogni specifico contesto d'uso. Questo potrebbe risolvere la sfida di lunga data di fornire esperienze coerenti attraverso un panorama di dispositivi sempre più frammentato, dagli smartphone e tablet ai computer desktop e tecnologie emergenti come gli occhiali per realtà aumentata.

La natura basata sui dati di GenUI apre opportunità per iterazioni rapide e miglioramenti nel design dell'interfaccia utente. Raccogliendo dati in tempo reale su come gli utenti interagiscono con le interfacce generate, designer e sviluppatori possono ottenere informazioni senza precedenti sul comportamento e le preferenze degli utenti. Questo ciclo di feedback potrebbe portare a miglioramenti continui nel design dell'interfaccia.

utente, guidati da modelli di utilizzo reali piuttosto che da supposizioni o test utente limitati.

Per prepararsi a questo cambiamento, i designer dovranno far evolvere le loro competenze e mentalità. L'attenzione si sposterà dalla creazione di layout fissi allo sviluppo di sistemi di design completi e linee guida che possano informare la generazione di interfacce guidata dall'IA. I designer dovranno coltivare una profonda comprensione dell'analisi dei dati, delle tecnologie IA e del pensiero sistematico per guidare efficacemente i sistemi GenUI.

Inoltre, mentre GenUI sfuma i confini tra design e tecnologia, i designer dovranno collaborare più strettamente con sviluppatori e data scientist. Questo approccio interdisciplinare sarà cruciale nella creazione di sistemi GenUI che siano non solo visivamente attraenti e user-friendly, ma anche tecnicamente robusti ed eticamente corretti.

Le implicazioni etiche del GenUI emergeranno sempre più in primo piano con la maturazione della tecnologia. I designer avranno un ruolo cruciale nello sviluppo di framework per l'uso responsabile dell'IA nel design dell'interfaccia, assicurando che la personalizzazione migliori le esperienze utente senza compromettere la privacy o manipolare il comportamento dell'utente in modi non etici.

Guardando al futuro, il GenUI presenta sia opportunità entusiasmanti che sfide significative. Ha il potenziale per creare esperienze digitali più intuitive, efficienti e soddisfacenti per gli utenti in tutto il mondo. Sebbene richiederà ai designer di adattarsi e acquisire nuove competenze, offre anche un'opportunità senza precedenti di plasmare il futuro dell'interazione uomo-computer in modi profondi e significativi. Il percorso verso sistemi GenUI pienamente realizzati sarà indubbiamente complesso, ma i potenziali benefici in termini di migliori esperienze utente e accessibilità digitale lo rendono un futuro per cui vale la pena impegnarsi.

Orchestrazione Intelligente dei Flussi di Lavoro



Nel campo dello sviluppo delle applicazioni, i flussi di lavoro svolgono un ruolo cruciale nel definire come le attività, i processi e le interazioni degli utenti vengono strutturati ed eseguiti. Man mano che le applicazioni diventano più complesse e le aspettative degli utenti continuano a crescere, la necessità di un'orchestrazione intelligente e adattiva dei flussi di lavoro diventa sempre più evidente.

L'approccio dell'“Orchestrazione Intelligente dei Flussi di Lavoro” si concentra sull'utilizzo di componenti di IA per orchestrare e ottimizzare dinamicamente flussi di lavoro complessi all'interno delle applicazioni. L'obiettivo è creare applicazioni più efficienti, reattive e adattabili ai dati in tempo reale e al contesto.

In questo capitolo, esploreremo i principi e i pattern chiave che sostengono l'approccio dell'orchestrazione intelligente dei flussi di lavoro. Esamineremo come l'IA può

essere utilizzata per instradare intelligentemente le attività, automatizzare il processo decisionale e adattare dinamicamente i flussi di lavoro in base a vari fattori come il comportamento dell’utente, le prestazioni del sistema e le regole di business. Attraverso esempi pratici e scenari reali, dimostreremo il potenziale trasformativo dell’IA nel semplificare e ottimizzare i flussi di lavoro delle applicazioni.

Che tu stia sviluppando applicazioni aziendali con processi di business complessi o applicazioni rivolte ai consumatori con percorsi utente dinamici, i pattern e le tecniche discusse in questo capitolo ti forniranno le conoscenze e gli strumenti per creare flussi di lavoro intelligenti ed efficienti che migliorano l’esperienza complessiva dell’utente e generano valore aziendale.

Esigenza Aziendale

Gli approcci tradizionali alla gestione dei flussi di lavoro spesso si basano su regole predefinite e alberi decisionali statici, che possono essere rigidi, inflessibili e incapaci di far fronte alla natura dinamica delle applicazioni moderne.

Consideriamo uno scenario in cui un’applicazione di e-commerce deve gestire un processo complesso di evasione degli ordini. Il flusso di lavoro può coinvolgere più fasi come la convalida dell’ordine, il controllo dell’inventario, l’elaborazione dei pagamenti, la spedizione e le notifiche ai clienti. Ogni fase può avere il proprio insieme di regole, dipendenze, integrazioni esterne e meccanismi di gestione delle eccezioni. Gestire un tale flusso di lavoro manualmente o attraverso una logica hardcoded può rapidamente diventare ingombrante, soggetto a errori e difficile da mantenere.

Inoltre, man mano che l’applicazione si espande e il numero di utenti simultanei cresce, il flusso di lavoro potrebbe dover adattarsi e ottimizzarsi in base ai dati in tempo reale e alle prestazioni del sistema. Ad esempio, durante i periodi di picco del traffico, l’applicazione potrebbe dover regolare dinamicamente il flusso di lavoro per dare priorità a determinate attività, allocare le risorse in modo efficiente e garantire un’esperienza utente fluida.

È qui che entra in gioco l'approccio dell'“Orchestrazione Intelligente dei Flussi di Lavoro”. Sfruttando i componenti di IA, gli sviluppatori possono creare flussi di lavoro intelligenti, adattivi e auto-ottimizzanti. L'IA può analizzare grandi quantità di dati, imparare dalle esperienze passate e prendere decisioni informate in tempo reale per orchestrare efficacemente il flusso di lavoro.

Benefici Principali

- 1. Maggiore Efficienza:** L'IA può ottimizzare l'allocazione delle attività, l'utilizzo delle risorse e l'esecuzione del flusso di lavoro, portando a tempi di elaborazione più rapidi e una migliore efficienza complessiva.
- 2. Adattabilità:** I flussi di lavoro guidati dall'IA possono adattarsi dinamicamente alle condizioni mutevoli, come le fluttuazioni nella domanda degli utenti, le prestazioni del sistema o i requisiti aziendali, garantendo che l'applicazione rimanga reattiva e resiliente.
- 3. Processo Decisionale Automatizzato:** L'IA può automatizzare processi decisionali complessi all'interno del flusso di lavoro, riducendo l'intervento manuale e minimizzando il rischio di errori umani.
- 4. Personalizzazione:** L'IA può analizzare il comportamento, le preferenze e il contesto dell'utente per personalizzare il flusso di lavoro e fornire esperienze su misura per i singoli utenti.
- 5. Scalabilità:** I flussi di lavoro basati sull'IA possono scalare senza problemi per gestire volumi crescenti di dati e interazioni degli utenti, senza compromettere le prestazioni o l'affidabilità.

Nelle sezioni seguenti, esploreremo i pattern e le tecniche chiave che consentono l'implementazione di flussi di lavoro intelligenti e mostreremo esempi reali di come l'IA sta trasformando la gestione dei flussi di lavoro nelle applicazioni moderne.

Pattern Chiave

Per implementare l'orchestrazione intelligente dei flussi di lavoro nelle applicazioni, gli sviluppatori possono sfruttare diversi pattern chiave che sfruttano la potenza dell'IA. Questi pattern forniscono un approccio strutturato alla progettazione e alla gestione dei flussi di lavoro, consentendo alle applicazioni di adattarsi, ottimizzare e automatizzare i processi basandosi su dati e contesto in tempo reale. Esploriamo alcuni dei pattern fondamentali nell'orchestrazione intelligente dei flussi di lavoro.

Instrandamento Dinamico delle Attività

Questo pattern prevede l'utilizzo dell'IA per instradare intelligentemente le attività all'interno di un flusso di lavoro basandosi su vari fattori come la priorità dell'attività, la disponibilità delle risorse e le prestazioni del sistema. Gli algoritmi di IA possono analizzare le caratteristiche di ogni attività, considerare lo stato attuale del sistema e prendere decisioni informate per assegnare le attività alle risorse o ai percorsi di elaborazione più appropriati. L'instrandamento dinamico delle attività garantisce che le attività siano distribuite ed eseguite in modo efficiente, ottimizzando le prestazioni complessive del flusso di lavoro.

```
1  class TaskRouter
2    include Raix::ChatCompletion
3    include Raix::FunctionDispatch
4
5    attr_accessor :task
6
7    # list of functions that can be called by the AI entirely at its
8    # discretion depending on the task received
9
10   function :analyze_task_priority do
11     TaskPriorityAnalyzer.perform(task)
12   end
13
14   function :check_resource_availability, # ...
```

```
15     function :assess_system_performance, # ...
16     function :assign_task_to_resource, # ...
17
18     DIRECTIVE = "You are a task router, responsible for intelligently
19     assigning tasks to available resources based on priority, resource
20     availability, and system performance..."
21
22     def initialize(task)
23         self.task = task
24         transcript << { system: DIRECTIVE }
25         transcript << { user: task.to_json }
26     end
27
28     def perform
29         while task.unassigned?
30             chat_completion
31
32             # todo: add max loop counter and break
33         end
34
35         # capture the transcript for later analysis
36         task.update(routing_transcript: transcript)
37     end
38 end
```

Notare il ciclo creato dall'espressione `while` alla riga 29, che continua a sollecitare l'IA fino a quando il compito non viene assegnato. Alla riga 35, salviamo la trascrizione del compito per successive analisi e debugging, qualora diventasse necessario.

Processo Decisionale Contestuale

È possibile utilizzare un codice molto simile per prendere decisioni contestuali all'interno di un workflow. Analizzando punti dati rilevanti come le preferenze dell'utente, i modelli storici e gli input in tempo reale, i componenti di IA possono determinare la linea d'azione più appropriata in ogni punto decisionale del workflow. Adatta il comportamento del tuo workflow in base al contesto specifico di ogni utente o scenario, fornendo esperienze personalizzate e ottimizzate.

Composizione Adattiva del Workflow

Questo pattern si concentra sulla composizione e regolazione dinamica dei workflow in base ai requisiti o alle condizioni mutevoli. L'IA può analizzare lo stato attuale del workflow, identificare i colli di bottiglia o le inefficienze, e modificare automaticamente la struttura del workflow per ottimizzare le prestazioni. La composizione adattiva del workflow permette alle applicazioni di evolversi e migliorare continuamente i propri processi senza richiedere interventi manuali.

Gestione e Recupero delle Eccezioni

La gestione e il recupero delle eccezioni sono aspetti critici dell'orchestrazione intelligente del workflow. Quando si lavora con componenti IA e workflow complessi, è essenziale anticipare e gestire le eccezioni con eleganza per garantire la stabilità e l'affidabilità del sistema.

Ecco alcune considerazioni e tecniche chiave per la gestione e il recupero delle eccezioni nei workflow intelligenti:

- 1. Propagazione delle Eccezioni:** Implementare un approccio coerente per la propagazione delle eccezioni tra i componenti del workflow. Quando si verifica un'eccezione all'interno di un componente, questa dovrebbe essere catturata, registrata e propagata all'orchestratore o a un componente discreto responsabile della gestione delle eccezioni. L'idea è quella di centralizzare la gestione delle eccezioni e impedire che vengano silenziosamente ignorate, oltre ad aprire possibilità per la [Gestione Intelligente degli Errori](#).
- 2. Meccanismi di Ripetizione:** I meccanismi di ripetizione aiutano a migliorare la resilienza del workflow e a gestire con eleganza i fallimenti intermittenti. È sicuramente consigliabile implementare meccanismi di ripetizione per eccezioni transitorie o recuperabili, come problemi di connettività di rete o indisponibilità di

risorse che possono essere automaticamente ritentate dopo un ritardo specificato. Avere un orchestratore o un gestore di eccezioni basato su IA significa che le strategie di ripetizione non devono essere necessariamente meccaniche, basandosi su algoritmi fissi come il fallback esponenziale. Puoi lasciare la gestione del nuovo tentativo alla “discrezione” del componente IA responsabile di decidere come gestire l’eccezione.

3. **Strategie di Fallback:** Se un componente IA non riesce a fornire una risposta valida o incontra un errore—un’occorrenza comune data la sua natura all’avanguardia—è necessario avere un meccanismo di fallback per garantire che il workflow possa continuare. Questo potrebbe comportare l’uso di valori predefiniti, algoritmi alternativi o un **Intervento Umano nel Processo** per prendere decisioni e mantenere il workflow in movimento.
4. **Azioni di Compensazione:** Le direttive dell’orchestratore dovrebbero includere istruzioni sulle azioni di compensazione per gestire le eccezioni che non possono essere risolte automaticamente. Le azioni di compensazione sono passaggi intrapresi per annullare o mitigare gli effetti di un’operazione fallita. Ad esempio, se fallisce una fase di elaborazione del pagamento, un’azione di compensazione potrebbe essere quella di annullare la transazione e notificare l’utente. Le azioni di compensazione aiutano a mantenere la coerenza e l’integrità dei dati in presenza di eccezioni.
5. **Monitoraggio e Segnalazione delle Eccezioni:** Configurare meccanismi di monitoraggio e segnalazione per rilevare e notificare alle parti interessate le eccezioni critiche. L’orchestratore può essere reso consapevole di soglie e regole per attivare gli avvisi quando le eccezioni superano determinati limiti o quando si verificano specifici tipi di eccezioni. Questo permette l’identificazione e la risoluzione proattiva dei problemi prima che impattino sull’intero sistema.

Ecco un esempio di gestione e recupero delle eccezioni in un componente workflow Ruby:

```
1  class InventoryManager
2    def check_availability(order)
3      begin
4        # Perform inventory check logic
5        inventory = Inventory.find_by(product_id: order.product_id)
6        if inventory.available_quantity >= order.quantity
7          return true
8        else
9          raise InsufficientInventoryError,
10            "Insufficient inventory for product #{order.product_id}"
11        end
12      rescue InsufficientInventoryError => e
13        # Log the exception
14        logger.error("Inventory check failed: #{e.message}")
15
16        # Retry the operation after a delay
17        retry_count ||= 0
18        if retry_count < MAX_RETRIES
19          retry_count += 1
20          sleep(RETRY_DELAY)
21          retry
22        else
23          # Fallback to manual intervention
24          NotificationService.admin("Inventory check failed: Order #{order.id}")
25          return false
26        end
27      end
28    end
29  end
```

In questo esempio, il componente `InventoryManager` verifica la disponibilità di un prodotto per un determinato ordine. Se la quantità disponibile è insufficiente, genera un `InsufficientInventoryError`. L'eccezione viene catturata, registrata e viene implementato un meccanismo di ripetizione. Se il limite di tentativi viene superato, il componente ricorre all'intervento manuale notificando un amministratore.

Implementando meccanismi robusti di gestione e recupero delle eccezioni, è possibile garantire che i flussi di lavoro intelligenti siano resilienti, manutenibili e in grado di

gestire situazioni impreviste con eleganza.

Questi pattern costituiscono le fondamenta dell'orchestrazione del flusso di lavoro intelligente e possono essere combinati e adattati per soddisfare i requisiti specifici di diverse applicazioni. Sfruttando questi pattern, gli sviluppatori possono creare flussi di lavoro flessibili, resilienti e ottimizzati per le prestazioni e l'esperienza utente.

Nella prossima sezione, esploreremo come questi pattern possono essere implementati nella pratica, utilizzando esempi del mondo reale e frammenti di codice per illustrare l'integrazione dei componenti AI nella gestione del flusso di lavoro.

Implementare l'Orchestrazione del Flusso di Lavoro Intelligente nella Pratica

Ora che abbiamo esplorato i pattern chiave nell'orchestrazione del flusso di lavoro intelligente, approfondiamo come questi pattern possono essere implementati in applicazioni reali. Forniremo esempi pratici e frammenti di codice per illustrare l'integrazione dei componenti AI nella gestione del flusso di lavoro.

Processore di Ordini Intelligente

Addentriamoci in un esempio pratico di implementazione dell'orchestrazione del flusso di lavoro intelligente utilizzando un componente `OrderProcessor` potenziato dall'AI in un'applicazione e-commerce Ruby on Rails. L'`OrderProcessor` realizza il concetto di [Process Manager Enterprise Integration](#) che abbiamo incontrato per la prima volta nel Capitolo 3 quando abbiamo discusso di [Multitude of Workers](#). Il componente sarà responsabile della gestione del flusso di evasione degli ordini, prendendo decisioni di routing basate su risultati intermedi e orchestrando l'esecuzione di vari passaggi di elaborazione.

Il processo di evasione degli ordini coinvolge più fasi come la convalida dell'ordine, il controllo dell'inventario, l'elaborazione del pagamento e la spedizione. Ogni fase è implementata come un processo worker separato che esegue un compito specifico e restituisce il risultato all'`OrderProcessor`. Le fasi non sono obbligatorie e non devono necessariamente essere eseguite in un ordine preciso.

Ecco un esempio di implementazione dell'`OrderProcessor`. Presenta due mixin da [Raix](#). Il primo (`ChatCompletion`) gli conferisce la capacità di completamento della chat, che è ciò che lo rende un componente AI. Il secondo (`FunctionDispatch`) abilita la chiamata di funzione da parte dell'AI, permettendole di rispondere a un prompt con un'invocazione di funzione invece che con un messaggio di testo.

Le funzioni worker (`validate_order`, `check_inventory`, et al) delegano alle rispettive classi worker, che possono essere componenti AI o non-AI, con l'unico requisito che restituiscano i risultati del loro lavoro in un formato che può essere rappresentato come stringa.



Come per tutti gli altri esempi in questa parte del libro, questo codice è praticamente pseudo-codice ed è destinato solo a trasmettere il significato del pattern e ispirare le vostre creazioni. Le descrizioni complete dei pattern e gli esempi di codice completi sono inclusi nella Parte 2.

```
1  class OrderProcessor
2      include Raix::ChatCompletion
3      include Raix::FunctionDispatch
4
5      SYSTEM_DIRECTIVE = "You are an order processor, tasked with..."
6
7      def initialize(order)
8          self.order = order
9          transcript << { system: SYSTEM_DIRECTIVE }
10         transcript << { user: order.to_json }
11     end
12
13     def perform
14         # will continue looping until `stop_looping!` is called
15         chat_completion(loop: true)
16     end
17
18     # list of functions available to be called by the AI
19     # truncated for brevity
20
21     def functions
22         [
23             {
24                 name: "validate_order",
25                 description: "Invoke to check validity of order",
26                 parameters: {
27                     ...
28                 },
29                 ...
30             }
31         ]
32     end
33
34     # implementation of functions that can be called by the AI
35     # entirely at its discretion, depending on the needs of the order
36
37     def validate_order
38         OrderValidationWorker.perform(@order)
39     end
40
41     def check_inventory
42         InventoryCheckWorker.perform(@order)
43     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
```

Nell'esempio, l'OrderProcessor viene inizializzato con un oggetto ordine e mantiene una trascrizione dell'esecuzione del flusso di lavoro, nel tipico formato di trascrizione di conversazione che è nativo dei modelli linguistici di grandi dimensioni. Il controllo completo viene dato all'IA per orchestrare l'esecuzione di vari passaggi di elaborazione, come la convalida dell'ordine, il controllo dell'inventario, l'elaborazione del pagamento e la spedizione.

Ogni volta che viene chiamato il metodo `chat_completion`, la trascrizione viene inviata all'IA affinché fornisca un completamento come chiamata di funzione. Spetta interamente all'IA analizzare il risultato del passaggio precedente e determinare l'azione appropriata da intraprendere. Per esempio, se il controllo dell'inventario rivela livelli di scorte bassi, l'OrderProcessor può programmare un'attività di rifornimento. Se l'elaborazione del pagamento fallisce, può avviare un nuovo tentativo o notificare l'assistenza clienti.

L'esempio sopra non ha funzioni definite per il rifornimento o la notifica all'assistenza clienti, ma potrebbe assolutamente averle.

La trascrizione cresce ogni volta che viene chiamata una funzione e serve come registro dell'esecuzione del flusso di lavoro, includendo i risultati di ogni passaggio e le istruzioni generate dall'IA per i passaggi successivi. Questa trascrizione può essere utilizzata per il debug, l'audit e per fornire visibilità sul processo di evasione degli ordini.

Sfruttando l'IA nell'OrderProcessor, l'applicazione di e-commerce può adattare dinamicamente il flusso di lavoro in base ai dati in tempo reale e gestire le eccezioni in modo intelligente. Il componente IA può prendere decisioni informate, ottimizzare il flusso di lavoro e garantire un'elaborazione fluida degli ordini anche in scenari complessi.

Il fatto che l'unico requisito sui processi worker sia quello di restituire un output comprensibile per l'IA da considerare quando decide cosa fare dopo, potrebbe iniziare a farvi capire come questo approccio possa ridurre il lavoro di mappatura input/output che è tipicamente coinvolto nell'integrazione di sistemi disparati tra loro.

Moderatore di Contenuti Intelligente

Le applicazioni di social media generalmente richiedono almeno una moderazione minima dei contenuti per garantire una comunità sicura e sana. Questo esempio di componente ContentModerator sfrutta l'IA per orchestrare in modo intelligente il flusso di lavoro della moderazione, prendendo decisioni basate sulle caratteristiche del contenuto e sui risultati dei vari passaggi di moderazione.

Il processo di moderazione coinvolge più passaggi come l'analisi del testo, il riconoscimento delle immagini, la valutazione della reputazione dell'utente e la

revisione manuale. Ogni passaggio è implementato come un processo worker separato che esegue un compito specifico e restituisce il risultato al ContentModerator.

Ecco un esempio di implementazione del ContentModerator:

```
1  class ContentModerator
2    include Raix::ChatCompletion
3    include Raix::FunctionDispatch
4
5    SYSTEM_DIRECTIVE = "You are a content moderator process manager,
6      tasked with the workflow involved in moderating user-generated content..."
7
8    def initialize(content)
9      @content = content
10     @transcript = [
11       { system: SYSTEM_DIRECTIVE },
12       { user: content.to_json }
13     ]
14   end
15
16   def perform
17     complete(@transcript)
18   end
19
20   def model
21     "openai/gpt-4"
22   end
23
24   # list of functions available to be called by the AI
25   # truncated for brevity
26
27   def functions
28     [
29       {
30         name: "analyze_text",
31         # ...
32       },
33       {
34         name: "recognize_image",
35         description: "Invoke to describe images...",
36         # ...
37       },
38     ]
39   end
```

```
38      {
39          name: "assess_user_reputation",
40          # ...
41      },
42      {
43          name: "escalate_to_manual_review",
44          # ...
45      },
46      {
47          name: "approve_content",
48          # ...
49      },
50      {
51          name: "reject_content",
52          # ...
53      }
54  ]
55 end
56
57 # implementation of functions that can be called by the AI
58 # entirely at its discretion, depending on the needs of the order
59
60 def analyze_text
61     result = TextAnalysisWorker.perform(@content)
62     continue_with(result)
63 end
64
65 def recognize_image
66     result = ImageRecognitionWorker.perform(@content)
67     continue_with(result)
68 end
69
70 def assess_user_reputation
71     result = UserReputationWorker.perform(@content.user)
72     continue_with(result)
73 end
74
75 def escalate_to_manual_review
76     ManualReviewWorker.perform(@content)
77     @content.update!(status: 'pending', transcript: @transcript)
78 end
79
```

```
80  def approve_content
81      @content.update!(status: 'approved', transcript: @transcript)
82  end
83
84  def reject_content
85      @content.update!(status: 'rejected', transcript: @transcript)
86  end
87
88  private
89
90  def continue_with(result)
91      @transcript << { function: result }
92      complete(@transcript)
93  end
94 end
```

In questo esempio, il ContentModerator viene inizializzato con un oggetto contenuto e mantiene una trascrizione della moderazione in formato conversazione. Il componente AI ha il controllo completo sul flusso di lavoro della moderazione, decidendo quali passaggi eseguire in base alle caratteristiche del contenuto e ai risultati di ogni fase.

Le funzioni worker disponibili che l'AI può invocare includono `analyze_text`, `recognize_image`, `assess_user_reputation`, e `escalate_to_manual_review`. Ogni funzione delega il compito a un corrispondente processo worker (`TextAnalysisWorker`, `ImageRecognitionWorker`, ecc.) e aggiunge il risultato alla trascrizione della moderazione, con l'eccezione della funzione di escalation, che agisce come stato finale. Infine, le funzioni `approve_content` e `reject_content` fungono anch'esse da stati finali.

Il componente AI analizza il contenuto e determina l'azione appropriata da intraprendere. Se il contenuto contiene riferimenti a immagini, può chiamare il worker `recognize_image` per assistenza con una revisione visiva. Se qualsiasi worker segnala contenuti potenzialmente dannosi, l'AI può decidere di escalare il contenuto per una revisione manuale o semplicemente rifiutarlo direttamente. Ma a seconda della gravità dell'avvertimento, l'AI può scegliere di utilizzare i risultati della valutazione

della reputazione dell’utente per decidere come gestire i contenuti di cui non è altrimenti sicura. A seconda del caso d’uso, forse gli utenti fidati hanno più libertà in ciò che possono pubblicare. E così via...

Come nell’esempio precedente del gestore dei processi, la trascrizione della moderazione serve come registro dell’esecuzione del flusso di lavoro, includendo i risultati di ogni fase e le decisioni generate dall’AI. Questa trascrizione può essere utilizzata per l’audit, la trasparenza e il miglioramento del processo di moderazione nel tempo.

Sfruttando l’AI nel ContentModerator, l’applicazione di social media può adattare dinamicamente il flusso di lavoro della moderazione in base alle caratteristiche del contenuto e gestire scenari di moderazione complessi in modo intelligente. Il componente AI può prendere decisioni informate, ottimizzare il flusso di lavoro e garantire un’esperienza comunitaria sicura e sana.

Esploriamo altri due esempi che dimostrano la pianificazione predittiva delle attività e la gestione delle eccezioni e il recupero nel contesto dell’orchestrazione intelligente del flusso di lavoro.

Pianificazione Predittiva delle Attività in un Sistema di Supporto Clienti

In un’applicazione di supporto clienti costruita con Ruby on Rails, gestire e prioritizzare efficacemente i ticket di supporto è cruciale per fornire assistenza tempestiva ai clienti. Il componente SupportTicketScheduler sfrutta l’AI per pianificare e assegnare in modo predittivo i ticket di supporto agli agenti disponibili in base a vari fattori come l’urgenza del ticket, l’esperienza dell’agente e il carico di lavoro.

```
1  class SupportTicketScheduler
2    include Raix::ChatCompletion
3    include Raix::FunctionDispatch
4
5    SYSTEM_DIRECTIVE = "You are a support ticket scheduler,
6      tasked with intelligently assigning tickets to available agents..."
7
8    def initialize(ticket)
9      @ticket = ticket
10     @transcript = [
11       { system: SYSTEM_DIRECTIVE },
12       { user: ticket.to_json }
13     ]
14   end
15
16   def perform
17     complete(@transcript)
18   end
19
20   def model
21     "openai/gpt-4"
22   end
23
24   def functions
25     [
26       {
27         name: "analyze_ticket_urgency",
28         # ...
29       },
30       {
31         name: "list_available_agents",
32         description: "Includes expertise of available agents",
33         # ...
34       },
35       {
36         name: "predict_agent_workload",
37         description: "Uses historical data to predict upcoming workloads",
38         # ...
39       },
40       {
41         name: "assign_ticket_to_agent",
42         # ...
```

```
43      },
44      {
45        name: "reschedule_ticket",
46        # ...
47      }
48    ]
49  end
50
51  # implementation of functions that can be called by the AI
52  # entirely at its discretion, depending on the needs of the order
53
54  def analyze_ticket_urgency
55    result = TicketUrgencyAnalyzer.perform(@ticket)
56    continue_with(result)
57  end
58
59  def list_available_agents
60    result = ListAvailableAgents.perform
61    continue_with(result)
62  end
63
64  def predict_agent_workload
65    result = AgentWorkloadPredictor.perform
66    continue_with(result)
67  end
68
69  def assign_ticket_to_agent
70    TicketAssigner.perform(@ticket, @transcript)
71  end
72
73  def delay_assignment(until)
74    until = DateTimeStandardizer.process(until)
75    SupportTicketScheduler.delay(@ticket, @transcript, until)
76  end
77
78  private
79
80  def continue_with(result)
81    @transcript << { function: result }
82    complete(@transcript)
83  end
84 end
```

In questo esempio, il `SupportTicketScheduler` viene inizializzato con un oggetto `ticket` di supporto e mantiene un registro di pianificazione. Il componente AI analizza i dettagli del ticket e pianifica in modo predittivo l'assegnazione del ticket basandosi su fattori come l'urgenza del ticket, l'esperienza dell'operatore e il carico di lavoro previsto dell'operatore.

Le funzioni disponibili che l'AI può invocare includono `analyze_ticket_urgency`, `list_available_agents`, `predict_agent_workload` e `assign_ticket_to_agent`. Ogni funzione delega il compito a un corrispondente componente di analisi o previsione e aggiunge il risultato al registro di pianificazione. L'AI ha anche l'opzione di ritardare l'assegnazione utilizzando la funzione `delay_assignment`.

Il componente AI esamina il registro di pianificazione e prende decisioni informate sull'assegnazione dei ticket. Considera l'urgenza del ticket, l'esperienza degli operatori disponibili e il carico di lavoro previsto di ciascun operatore per determinare l'operatore più adatto alla gestione del ticket.

Sfruttando la pianificazione predittiva delle attività, l'applicazione di supporto clienti può ottimizzare l'assegnazione dei ticket, ridurre i tempi di risposta e migliorare la soddisfazione complessiva del cliente. La gestione proattiva ed efficiente dei ticket di supporto garantisce che i ticket giusti vengano assegnati agli operatori giusti al momento giusto.

Gestione delle Eccezioni e Recupero in una Pipeline di Elaborazione Dati

La gestione delle eccezioni e il recupero da errori sono essenziali per garantire l'integrità dei dati e prevenire la perdita di dati. Il componente `DataProcessingOrchestrator` utilizza l'AI per gestire in modo intelligente le eccezioni e orchestrare il processo di recupero in una pipeline di elaborazione dati.

```
1  class DataProcessingOrchestrator
2      include Raix::ChatCompletion
3      include Raix::FunctionDispatch
4
5      SYSTEM_DIRECTIVE = "You are a data processing orchestrator..."
6
7      def initialize(data_batch)
8          @data_batch = data_batch
9          @transcript = [
10              { system: SYSTEM_DIRECTIVE },
11              { user: data_batch.to_json }
12          ]
13      end
14
15      def perform
16          complete(@transcript)
17      end
18
19      def model
20          "openai/gpt-4"
21      end
22
23      def functions
24          [
25              {
26                  name: "validate_data",
27                  # ...
28              },
29              {
30                  name: "process_data",
31                  # ...
32              },
33              {
34                  name: "request_fix",
35                  # ...
36              },
37              {
38                  name: "retry_processing",
39                  # ...
40              },
41              {
42                  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)
```

```
85  end
86
87  private
88
89  def continue_with(result)
90    @transcript << { function: result }
91    complete(@transcript)
92  end
93
94  def handle_validation_exception(exception)
95    @transcript << { exception: exception.message }
96    complete(@transcript)
97  end
98
99  def handle_processing_exception(exception)
100   @transcript << { exception: exception.message }
101   complete(@transcript)
102 end
103 end
```

In questo esempio, il `DataProcessingOrchestrator` viene inizializzato con un oggetto `batch` di dati e mantiene una trascrizione dell'elaborazione. Il componente AI orchestra la pipeline di elaborazione dati, gestendo le eccezioni e recuperando dai fallimenti quando necessario.

Le funzioni disponibili che l'AI può invocare includono `validate_data`, `process_data`, `request_fix`, `retry_processing` e `mark_data_as_failed`. Ogni funzione delega il compito a un componente di elaborazione dati corrispondente e aggiunge il risultato o i dettagli dell'eccezione alla trascrizione dell'elaborazione.

Se si verifica un'eccezione di validazione durante la fase `validate_data`, la funzione `handle_validation_exception` aggiunge i dati dell'eccezione alla trascrizione e restituisce il controllo all'AI. Analogamente, se si verifica un'eccezione di elaborazione durante la fase `process_data`, l'AI può decidere la strategia di recupero.

A seconda della natura dell'eccezione incontrata, l'AI può, a sua discrezione, decidere di chiamare `request_fix`, che delega a un componente `SmartDataFixer` basato

su AI (vedi il capitolo sui Dati Auto-Riparanti). Il data fixer riceve una descrizione in linguaggio naturale di come dovrebbe modificare il `@data_batch` in modo che l'elaborazione possa essere ritentata. Forse un nuovo tentativo riuscito comporterebbe la rimozione dal batch di dati dei record che non hanno superato la validazione e/o la loro copia in una pipeline di elaborazione diversa per la revisione umana? Le possibilità sono praticamente infinite.

Incorporando la gestione delle eccezioni e il recupero guidato dall'AI, l'applicazione di elaborazione dati diventa più resiliente e tollerante ai guasti. Il `DataProcessingOrchestrator` gestisce intelligentemente le eccezioni, minimizza la perdita di dati e garantisce l'esecuzione fluida del flusso di lavoro di elaborazione dati.

Monitoraggio e Logging

Il monitoraggio e il logging forniscono visibilità sui progressi, le prestazioni e lo stato dei componenti del flusso di lavoro basati su AI, consentendo agli sviluppatori di tracciare e analizzare il comportamento del sistema. L'implementazione di meccanismi efficaci di monitoraggio e logging è essenziale per il debugging, l'auditing e il miglioramento continuo dei flussi di lavoro intelligenti.

Monitoraggio dei Progressi e delle Prestazioni del Flusso di Lavoro

Per garantire l'esecuzione fluida dei flussi di lavoro intelligenti, è importante monitorare i progressi e le prestazioni di ogni componente del flusso di lavoro. Questo comporta il tracciamento di metriche chiave ed eventi durante tutto il ciclo di vita del flusso di lavoro.

Alcuni aspetti importanti da monitorare includono:

- 1. Tempo di Esecuzione del Flusso di Lavoro:** Misurare il tempo impiegato da ciascun componente del flusso di lavoro per completare il proprio compito. Questo aiuta a

identificare i colli di bottiglia nelle prestazioni e a ottimizzare l'efficienza complessiva del flusso di lavoro.

2. Utilizzo delle Risorse: Monitorare l'utilizzo delle risorse di sistema, come CPU, memoria e storage, da parte di ciascun componente del flusso di lavoro. Questo aiuta a garantire che il sistema operi entro la sua capacità e possa gestire efficacemente il carico di lavoro.

3. Tassi di Errore ed Eccezioni: Tracciare il verificarsi di errori ed eccezioni all'interno dei componenti del flusso di lavoro. Questo aiuta a identificare potenziali problemi e permette una gestione proattiva degli errori e il recupero.

4. Punti Decisionali e Risultati: Monitorare i punti decisionali all'interno del flusso di lavoro e i risultati delle decisioni guidate dall'AI. Questo fornisce informazioni sul comportamento e l'efficacia dei componenti AI.

I dati catturati dai processi di monitoraggio possono essere visualizzati in dashboard o utilizzati come input per report programmati che informano gli amministratori di sistema sullo stato del sistema.



I dati di monitoraggio possono essere inviati a un processo di amministrazione di sistema basato su AI per la revisione e potenziali azioni!

Logging di Eventi e Decisioni Chiave

Il logging è una pratica essenziale che comporta la cattura e l'archiviazione di informazioni rilevanti su eventi chiave, decisioni ed eccezioni che si verificano durante l'esecuzione del flusso di lavoro.

Alcuni aspetti importanti da registrare includono:

- 1. Avvio e Completamento del Flusso di Lavoro:** Registrare gli orari di inizio e fine di ogni istanza del flusso di lavoro, insieme a qualsiasi metadata rilevante come i dati di input e il contesto utente.
- 2. Esecuzione dei Componenti:** Registrare i dettagli di esecuzione di ogni componente del flusso di lavoro, inclusi i parametri di input, i risultati di output e qualsiasi dato intermedio generato.
- 3. Decisioni AI e Ragionamento:** Registrare le decisioni prese dai componenti AI, insieme al ragionamento sottostante o ai punteggi di confidenza. Questo fornisce trasparenza e permette l'auditing delle decisioni guidate dall'AI.
- 4. Eccezioni e Messaggi di Errore:** Registrare qualsiasi eccezione o messaggio di errore incontrato durante l'esecuzione del flusso di lavoro, incluso lo stack trace e le informazioni di contesto rilevanti.

Il logging può essere implementato utilizzando varie tecniche, come la scrittura su file di log, l'archiviazione dei log in un database o l'invio dei log a un servizio di logging centralizzato. È importante scegliere un framework di logging che fornisca flessibilità, scalabilità e facile integrazione con l'architettura dell'applicazione.

Ecco un esempio di come il logging può essere implementato in un'applicazione Ruby on Rails utilizzando la classe `ActiveSupport::Logger`:

```
1 class WorkflowLogger
2   def self.log(message, severity = :info)
3     @logger ||= ActiveSupport::Logger.new('workflow.log')
4     @logger.formatter ||= proc do |severity, datetime, programe, msg|
5       "#{datetime} [#{severity}] #{msg}\n"
6     end
7     @logger.send(severity, message)
8   end
9 end
10
11 # Usage example
12 WorkflowLogger.log("Workflow initiated for order ##{@order.id}")
13 WorkflowLogger.log("Payment processing completed successfully")
14 WorkflowLogger.log("Inventory check failed for item ##{@item.id}", :error)
```

Posizionando strategicamente le istruzioni di logging all'interno dei componenti del workflow e nei punti decisionali dell'IA, gli sviluppatori possono acquisire informazioni preziose per il debugging, l'audit e l'analisi.

Vantaggi del Monitoraggio e del Logging

L'implementazione del monitoraggio e del logging nell'orchestrazione intelligente dei workflow offre diversi vantaggi:

- 1. Debugging e Risoluzione dei Problemi:** I log dettagliati e i dati di monitoraggio aiutano gli sviluppatori a identificare e diagnosticare rapidamente i problemi. Forniscono informazioni sul flusso di esecuzione del workflow, sulle interazioni tra i componenti e su eventuali errori o eccezioni riscontrati.
- 2. Ottimizzazione delle Prestazioni:** Il monitoraggio delle metriche di prestazione permette agli sviluppatori di identificare i colli di bottiglia e ottimizzare i componenti del workflow per una maggiore efficienza. Analizzando i tempi di esecuzione, l'utilizzo delle risorse e altre metriche, gli sviluppatori possono prendere decisioni informate per migliorare le prestazioni complessive del sistema.
- 3. Audit e Conformità:** La registrazione di eventi e decisioni chiave fornisce una traccia di audit per la conformità normativa e la responsabilità. Permette alle organizzazioni di tracciare e verificare le azioni intraprese dai componenti di IA e garantire il rispetto delle regole aziendali e dei requisiti legali.
- 4. Miglioramento Continuo:** I dati di monitoraggio e logging servono come input preziosi per il miglioramento continuo dei workflow intelligenti. Analizzando i dati storici, identificando modelli e misurando l'efficacia delle decisioni dell'IA, gli sviluppatori possono perfezionare e migliorare iterativamente la logica di orchestrazione del workflow.

Considerazioni e Best Practice

Nell'implementazione del monitoraggio e del logging nell'orchestrazione intelligente dei workflow, considerare le seguenti best practice:

- 1. Definire Metriche di Monitoraggio Chiare:** Identificare le metriche chiave e gli eventi che devono essere monitorati in base ai requisiti specifici del workflow. Concentrarsi sulle metriche che forniscono informazioni significative sulle prestazioni, lo stato e il comportamento del sistema.
- 2. Implementare il Logging Granulare:** Assicurarsi che le istruzioni di logging siano posizionate in punti appropriati all'interno dei componenti del workflow e nei punti decisionali dell'IA. Acquisire informazioni contestuali rilevanti, come parametri di input, risultati di output e qualsiasi dato intermedio generato.
- 3. Utilizzare il Logging Strutturato:** Adottare un formato di logging strutturato per facilitare l'analisi e l'elaborazione dei dati di log. Il logging strutturato permette una migliore ricercabilità, filtraggio e aggregazione delle voci di log.
- 4. Gestire la Conservazione e la Rotazione dei Log:** Implementare politiche di conservazione e rotazione dei log per gestire l'archiviazione e il ciclo di vita dei file di log. Determinare il periodo di conservazione appropriato in base ai requisiti legali, ai vincoli di archiviazione e alle esigenze di analisi. Se possibile, delegare il logging a un servizio di terze parti come [Papertrail](#).
- 5. Proteggere le Informazioni Sensibili:** Prestare attenzione quando si registrano informazioni sensibili, come dati personali identificabili (PII) o dati aziendali riservati. Implementare misure di sicurezza appropriate, come il mascheramento dei dati o la crittografia, per proteggere le informazioni sensibili nei file di log.
- 6. Integrare con Strumenti di Monitoraggio e Allerta:** Sfruttare gli strumenti di monitoraggio e allerta per centralizzare la raccolta, l'analisi e la visualizzazione dei dati di monitoraggio e logging. Questi strumenti possono fornire informazioni in

tempo reale, generare avvisi basati su soglie predefinite e facilitare il rilevamento e la risoluzione proattiva dei problemi. Il mio strumento preferito tra questi è [Datadog](#).

Implementando meccanismi completi di monitoraggio e logging, gli sviluppatori possono ottenere informazioni preziose sul comportamento e le prestazioni dei workflow intelligenti. Queste informazioni consentono un debugging efficace, l'ottimizzazione e il miglioramento continuo dei sistemi di orchestrazione dei workflow basati su IA.

Considerazioni sulla Scalabilità e le Prestazioni

La scalabilità e le prestazioni sono aspetti critici da considerare durante la progettazione e l'implementazione di sistemi di orchestrazione intelligente dei workflow. Con l'aumentare del volume di workflow concorrenti e della complessità dei componenti basati su IA, diventa essenziale garantire che il sistema possa gestire il carico di lavoro in modo efficiente e scalare senza problemi per soddisfare le crescenti esigenze.

Gestione di Grandi Volumi di Workflow Concorrenti

I sistemi di orchestrazione intelligente dei workflow spesso devono gestire un gran numero di workflow concorrenti. Per garantire la scalabilità, considerare le seguenti strategie:

- 1. Elaborazione Asincrona:** Implementare meccanismi di elaborazione asincrona per disaccoppiare l'esecuzione dei componenti del workflow. Questo permette al sistema di gestire più workflow contemporaneamente senza bloccarsi o attendere il completamento di ciascun componente. L'elaborazione asincrona può essere realizzata utilizzando code di messaggi, architetture event-driven o framework di elaborazione dei job in background come Sidekiq.
- 2. Architettura Distribuita:** Progettare l'architettura del sistema per utilizzare componenti serverless (come AWS Lambda) o semplicemente distribuire il carico di

lavoro su più nodi o server insieme al server applicativo principale. Questo consente la scalabilità orizzontale, dove è possibile aggiungere ulteriori nodi per gestire l'aumento dei volumi di workflow.

3. Esecuzione Parallela: Identificare le opportunità di esecuzione parallela all'interno dei workflow. Alcuni componenti del workflow potrebbero essere indipendenti tra loro e possono essere eseguiti contemporaneamente. Sfruttando tecniche di elaborazione parallela, come il multi-threading o le code di attività distribuite, il sistema può ottimizzare l'utilizzo delle risorse e ridurre il tempo complessivo di esecuzione del workflow.

Ottimizzazione delle Prestazioni dei Componenti Basati su IA

I componenti basati su IA, come i modelli di machine learning o i motori di elaborazione del linguaggio naturale, possono essere computazionalmente intensivi e influenzare le prestazioni complessive del sistema di orchestrazione dei workflow. Per ottimizzare le prestazioni dei componenti IA, considera le seguenti tecniche:

- 1. Caching:** Se l'elaborazione IA è puramente generativa e non coinvolge ricerche in tempo reale o integrazioni esterne per generare i completamenti delle chat, puoi considerare meccanismi di caching per memorizzare e riutilizzare i risultati delle operazioni frequentemente accedute o computazionalmente costose.
- 2. Ottimizzazione del Modello:** Ottimizza continuamente il modo in cui utilizzi i modelli IA nei componenti del workflow. Questo può coinvolgere tecniche come la *Distillazione dei Prompt* o potrebbe semplicemente essere una questione di testare nuovi modelli man mano che diventano disponibili.
- 3. Elaborazione Batch:** Se stai lavorando con modelli di classe GPT-4, potresti essere in grado di sfruttare tecniche di elaborazione batch per processare più punti dati o richieste in un singolo lotto, invece di elaborarli individualmente. Elaborando i dati in batch, il

sistema può ottimizzare l'utilizzo delle risorse e ridurre il sovraccarico delle richieste ripetute al modello.

Monitoraggio e Profilazione delle Prestazioni

Per identificare i colli di bottiglia nelle prestazioni e ottimizzare la scalabilità del sistema di orchestrazione intelligente dei workflow, è cruciale implementare meccanismi di monitoraggio e profilazione. Considera i seguenti approcci:

- 1. Metriche delle Prestazioni:** Definisci e traccia le metriche chiave delle prestazioni, come il tempo di risposta, il throughput, l'utilizzo delle risorse e la latenza. Queste metriche forniscono informazioni sulle prestazioni del sistema e aiutano a identificare le aree da ottimizzare. Il popolare aggregatore di modelli IA [OpenRouter](#) include le metriche Host¹ e Speed² in ogni risposta API, rendendo banale il monitoraggio di queste metriche chiave.
- 2. Strumenti di Profilazione:** Utilizza strumenti di profilazione per analizzare le prestazioni dei singoli componenti del workflow e delle operazioni IA. Gli strumenti di profilazione possono aiutare a identificare i punti critici delle prestazioni, i percorsi di codice inefficienti o le operazioni che consumano molte risorse. Gli strumenti di profilazione popolari includono New Relic, Scout, o i profiler integrati forniti dal linguaggio di programmazione o dal framework.
- 3. Test di Carico:** Conduci test di carico per valutare le prestazioni del sistema sotto diversi livelli di carico di lavoro concorrente. I test di carico aiutano a identificare i limiti di scalabilità del sistema, rilevare il degrado delle prestazioni e assicurare che il sistema possa gestire il traffico previsto senza compromettere le prestazioni.
- 4. Monitoraggio Continuo:** Implementa meccanismi di monitoraggio continuo e di allerta per rilevare proattivamente problemi e colli di bottiglia nelle prestazioni.

¹Host è il tempo necessario per ricevere il primo byte della generazione in streaming dall'host del modello, anche noto come "time to first byte."

²Speed viene calcolata come il numero di token di completamento diviso per il tempo totale di generazione. Per le richieste non in streaming, la latenza è considerata parte del tempo di generazione.

Configura dashboard di monitoraggio e avvisi per tracciare gli indicatori chiave di prestazione (KPI) e ricevere notifiche quando vengono superati i limiti predefiniti. Questo permette l'identificazione e la risoluzione tempestiva dei problemi di prestazione.

Strategie di Scaling

Per gestire carichi di lavoro crescenti e garantire la scalabilità del sistema di orchestrazione intelligente dei workflow, considera le seguenti strategie di scaling:

- 1. Scaling Verticale:** Lo scaling verticale comporta l'aumento delle risorse (ad esempio, CPU, memoria) dei singoli nodi o server per gestire carichi di lavoro più elevati. Questo approccio è adatto quando il sistema richiede più potenza di elaborazione o memoria per gestire workflow complessi o operazioni IA.
- 2. Scaling Orizzontale:** Lo scaling orizzontale comporta l'aggiunta di più nodi o server al sistema per distribuire il carico di lavoro. Questo approccio è efficace quando il sistema deve gestire un gran numero di workflow concorrenti o quando il carico di lavoro può essere facilmente distribuito su più nodi. Lo scaling orizzontale richiede un'architettura distribuita e meccanismi di bilanciamento del carico per garantire una distribuzione uniforme del traffico.
- 3. Auto-Scaling:** Implementa meccanismi di auto-scaling per regolare automaticamente il numero di nodi o risorse in base alla domanda del carico di lavoro. L'auto-scaling permette al sistema di scalare dinamicamente verso l'alto o verso il basso a seconda del traffico in entrata, garantendo un utilizzo ottimale delle risorse e l'efficienza dei costi. Piattaforme cloud come Amazon Web Services (AWS) o Google Cloud Platform (GCP) forniscono capacità di auto-scaling che possono essere sfruttate per i sistemi di orchestrazione intelligente dei workflow.

Tecniche di Ottimizzazione delle Prestazioni

Oltre alle strategie di scaling, considera le seguenti tecniche di ottimizzazione delle prestazioni per migliorare l'efficienza del sistema di orchestrazione intelligente dei

workflow:

- 1. Archiviazione e Recupero Efficienti dei Dati:** Ottimizza i meccanismi di archiviazione e recupero dei dati utilizzati dai componenti del workflow. Utilizza l'indicizzazione efficiente del database, tecniche di ottimizzazione delle query e caching dei dati per minimizzare la latenza e migliorare le prestazioni delle operazioni ad alta intensità di dati.
- 2. I/O Asincrono:** Utilizzare operazioni di I/O asincrone per prevenire il blocco e migliorare la reattività del sistema. L'I/O asincrono permette al sistema di gestire più richieste contemporaneamente senza dover attendere il completamento delle operazioni di I/O, massimizzando così l'utilizzo delle risorse.
- 3. Serializzazione e Deserializzazione Efficienti:** Ottimizzare i processi di serializzazione e deserializzazione utilizzati per lo scambio di dati tra i componenti del workflow. Utilizzare formati di serializzazione efficienti, come Protocol Buffers o MessagePack, per ridurre il sovraccarico della serializzazione dei dati e migliorare le prestazioni della comunicazione tra i componenti.



Per le applicazioni basate su Ruby, considera l'utilizzo di [Universal ID](#).

Universal ID sfrutta sia MessagePack che Brotli (una combinazione costruita per velocità e compressione dati ai massimi livelli). Quando combinati, queste librerie sono fino al 30% più veloci e hanno tassi di compressione entro il 2-5% rispetto a Protocol Buffers.

- 4. Compressione e Codifica:** Applicare tecniche di compressione e codifica per ridurre la dimensione dei dati trasferiti tra i componenti del workflow. Gli algoritmi di compressione, come gzip o Brotli, possono ridurre significativamente l'utilizzo della larghezza di banda di rete e migliorare le prestazioni complessive del sistema.

Considerando gli aspetti di scalabilità e prestazioni durante la progettazione e l'implementazione dei sistemi di orchestrazione intelligente dei workflow, è possibile garantire che il sistema possa gestire elevati volumi di workflow concorrenti, ottimizzare

le prestazioni dei componenti basati su IA e scalare senza problemi per soddisfare le crescenti esigenze. Il monitoraggio continuo, il profiling e gli sforzi di ottimizzazione sono essenziali per mantenere le prestazioni e la reattività del sistema man mano che il carico di lavoro e la complessità aumentano nel tempo.

Test e Validazione dei Workflow

Il testing e la validazione sono aspetti cruciali nello sviluppo e nella manutenzione dei sistemi di orchestrazione intelligente dei workflow. Data la natura complessa dei workflow basati su IA, è essenziale garantire che ogni componente funzioni come previsto, che il workflow complessivo si comporti correttamente e che le decisioni dell'IA siano accurate e affidabili. In questa sezione, esploreremo varie tecniche e considerazioni per il testing e la validazione dei workflow intelligenti.

Test Unitari dei Componenti del Workflow

Il test unitario consiste nel testare i singoli componenti del workflow in isolamento per verificarne la correttezza e la robustezza. Durante l'esecuzione dei test unitari sui componenti del workflow basati su IA, considera quanto segue:

- 1. Convalida dell'Input:** Testare la capacità del componente di gestire diversi tipi di input, inclusi dati validi e non validi. Verificare che il componente gestisca correttamente i casi limite e fornisca messaggi di errore o eccezioni appropriati.
- 2. Verifica dell'Output:** Verificare che il componente produca l'output previsto per un determinato set di input. Confrontare l'output effettivo con i risultati attesi per garantire la correttezza.
- 3. Gestione degli Errori:** Testare i meccanismi di gestione degli errori del componente simulando vari scenari di errore, come input non validi, indisponibilità delle risorse o eccezioni impreviste. Verificare che il componente catturi e gestisca gli errori in modo appropriato.

4. Condizioni Limite: Testare il comportamento del componente in condizioni limite, come input vuoto, dimensione massima dell'input o valori estremi. Assicurarsi che il componente gestisca queste condizioni in modo appropriato senza bloccarsi o produrre risultati errati.

Ecco un esempio di test unitario per un componente di workflow in Ruby utilizzando il framework di testing RSpec:

```
1 RSpec.describe OrderValidator do
2   describe '#validate' do
3     context 'when order is valid' do
4       let(:order) { build(:order) }
5
6       it 'returns true' do
7         expect(subject.validate(order)).to be true
8       end
9     end
10
11    context 'when order is invalid' do
12      let(:order) { build(:order, total_amount: -100) }
13
14      it 'returns false' do
15        expect(subject.validate(order)).to be false
16      end
17    end
18  end
19 end
```

In questo esempio, il componente `OrderValidator` viene testato utilizzando due casi di test: uno per un ordine valido e un altro per un ordine non valido. I casi di test verificano che il metodo `validate` restituisca il valore booleano previsto in base alla validità dell'ordine.

Test di Integrazione delle Interazioni del Workflow

Il test di integrazione si concentra sulla verifica delle interazioni e del flusso di dati tra i diversi componenti del workflow. Garantisce che i componenti lavorino insieme senza

problemi e producono i risultati attesi. Durante l'esecuzione dei test di integrazione dei workflow intelligenti, considera i seguenti aspetti:

- 1. Interazione tra Componenti:** Testa la comunicazione e lo scambio di dati tra i componenti del workflow. Verifica che l'output di un componente venga correttamente passato come input al componente successivo nel workflow.
- 2. Consistenza dei Dati:** Assicura che i dati rimangano consistenti e accurati mentre attraversano il workflow. Verifica che le trasformazioni dei dati, i calcoli e le aggregazioni vengano eseguiti correttamente.
- 3. Propagazione delle Eccezioni:** Testa come le eccezioni e gli errori vengono propagati e gestiti tra i componenti del workflow. Verifica che le eccezioni vengano catturate, registrate e gestite appropriatamente per prevenire l'interruzione del workflow.
- 4. Comportamento Asincrono:** Se il workflow coinvolge componenti asincroni o esecuzione parallela, testa i meccanismi di coordinamento e sincronizzazione. Assicura che il workflow si comporti correttamente in scenari concorrenti e asincroni.

Ecco un esempio di un test di integrazione per un workflow in Ruby utilizzando il framework di testing RSpec:

```
1 RSpec.describe OrderProcessingWorkflow do
2
3   let(:order) { build(:order) }
4
5   it 'processes the order successfully' do
6     expect(OrderValidator).to receive(:validate).and_return(true)
7     expect(InventoryManager).to receive(:check_availability).and_return(true)
8     expect(PaymentProcessor).to receive(:process_payment).and_return(true)
9     expect(ShippingService).to receive(:schedule_shipping).and_return(true)
10
11    workflow = OrderProcessingWorkflow.new(order)
12    result = workflow.process
13
14    expect(result).to be true
15    expect(order.status).to eq('processed')
16  end
```

17

18 **end**

In questo esempio, il `OrderProcessingWorkflow` viene testato verificando le interazioni tra i diversi componenti del workflow. Il caso di test definisce le aspettative per il comportamento di ciascun componente e assicura che il workflow elabori l'ordine con successo, aggiornando di conseguenza lo stato dell'ordine.

Testing dei Punti Decisionali AI

Il testing dei punti decisionali AI è cruciale per garantire l'accuratezza e l'affidabilità dei workflow basati su AI. Durante il testing dei punti decisionali AI, considera i seguenti aspetti:

- 1. Accuratezza Decisionale:** Verifica che il componente AI prenda decisioni accurate basate sui dati di input e sul modello addestrato. Confronta le decisioni dell'AI con i risultati attesi o i dati di riferimento.
- 2. Casi Limite:** Testa il comportamento del componente AI in casi limite e scenari inusuali. Verifica che il componente AI gestisca questi casi in modo elegante e prenda decisioni ragionevoli.
- 3. Distorsione ed Equità:** Valuta il componente AI per potenziali distorsioni e assicurati che prenda decisioni eque e imparziali. Testa il componente con dati di input diversificati e analizza i risultati per individuare eventuali schemi discriminatori.
- 4. Spiegabilità:** Se il componente AI fornisce spiegazioni o ragionamenti per le sue decisioni, verifica la correttezza e la chiarezza delle spiegazioni. Assicurati che le spiegazioni siano allineate con il processo decisionale sottostante.

Ecco un esempio di testing di un punto decisionale AI in Ruby utilizzando il framework di testing RSpec:

```
1 RSpec.describe FraudDetector do
2   describe '#detect_fraud' do
3     context 'when transaction is fraudulent' do
4       let(:tx) do
5         build(:transaction, amount: 10_000, location: 'High-Risk Country')
6       end
7
8       it 'returns true' do
9         expect(subject.detect_fraud(tx)).to be true
10      end
11    end
12
13    context 'when transaction is legitimate' do
14      let(:tx) do
15        build(:transaction, amount: 100, location: 'Low-Risk Country')
16      end
17
18      it 'returns false' do
19        expect(subject.detect_fraud(tx)).to be false
20      end
21    end
22  end
23 end
```

In questo esempio, il componente AI FraudDetector viene testato con due casi di test: uno per una transazione fraudolenta e un altro per una transazione legittima. I casi di test verificano che il metodo `detect_fraud` restituisca il valore booleano previsto in base alle caratteristiche della transazione.

Test End-to-End

Il test end-to-end comporta il testing dell'intero flusso di lavoro dall'inizio alla fine, simulando scenari reali e interazioni dell'utente. Assicura che il flusso di lavoro si comporti correttamente e produca i risultati desiderati. Durante l'esecuzione dei test end-to-end per i flussi di lavoro intelligenti, considera i seguenti aspetti:

- 1. Scenari Utente:** Identifica gli scenari utente comuni e testa il comportamento del flusso di lavoro in questi scenari. Verifica che il flusso di lavoro gestisca correttamente gli input dell'utente, prenda decisioni appropriate e produca gli output previsti.
- 2. Convalida dei Dati:** Assicurati che il flusso di lavoro convalidi e sanifichi gli input dell'utente per prevenire inconsistenze nei dati o vulnerabilità di sicurezza. Testa il flusso di lavoro con vari tipi di dati in input, inclusi dati validi e non validi.
- 3. Recupero degli Errori:** Testa la capacità del flusso di lavoro di riprendersi da errori ed eccezioni. Simula scenari di errore e verifica che il flusso di lavoro li gestisca in modo elegante, registri gli errori e intraprenda appropriate azioni di recupero.
- 4. Prestazioni e Scalabilità:** Valuta le prestazioni e la scalabilità del flusso di lavoro in diverse condizioni di carico. Testa il flusso di lavoro con un grande volume di richieste concorrenti e misura i tempi di risposta, l'utilizzo delle risorse e la stabilità complessiva del sistema.

Ecco un esempio di un test end-to-end per un flusso di lavoro in Ruby utilizzando il framework di testing RSpec e la libreria Capybara per simulare le interazioni dell'utente:

```
1 RSpec.describe 'Order Processing Workflow' do
2   scenario 'User places an order successfully' do
3     visit '/orders/new'
4     fill_in 'Product', with: 'Sample Product'
5     fill_in 'Quantity', with: '2'
6     fill_in 'Shipping Address', with: '123 Main St'
7     click_button 'Place Order'
8
9     expect(page).to have_content('Order Placed Successfully')
10    expect(Order.count).to eq(1)
11    expect(Order.last.status).to eq('processed')
12  end
13 end
```

In questo esempio, il test end-to-end simula un utente che effettua un ordine attraverso l'interfaccia web. Compila i campi obbligatori del modulo, invia l'ordine e verifica che

l'ordine sia elaborato con successo, mostrando il messaggio di conferma appropriato e aggiornando lo stato dell'ordine nel database.

Integrazione e Distribuzione Continua

Per garantire l'affidabilità e la manutenibilità dei flussi di lavoro intelligenti, si raccomanda di integrare il testing e la validazione nel processo di integrazione e distribuzione continua (CI/CD) pipeline. Questo permette il testing e la validazione automatizzati delle modifiche al flusso di lavoro prima che vengano distribuite in produzione. Considera le seguenti pratiche:

- 1. Esecuzione Automatizzata dei Test:** Configurare la pipeline CI/CD per eseguire automaticamente la suite di test ogni volta che vengono apportate modifiche al codice del flusso di lavoro. Questo assicura che eventuali regressioni o errori vengano rilevati nelle prime fasi del processo di sviluppo.
- 2. Monitoraggio della Copertura dei Test:** Misurare e monitorare la copertura dei test dei componenti del flusso di lavoro e dei punti decisionali dell'IA. Puntare a un'alta copertura dei test per garantire che i percorsi e gli scenari critici siano testati accuratamente.
- 3. Feedback Continuo:** Integrare i risultati dei test e le metriche della qualità del codice nel flusso di lavoro di sviluppo. Fornire feedback continuo agli sviluppatori sullo stato dei test, sulla qualità del codice e su eventuali problemi rilevati durante il processo CI/CD.
- 4. Ambienti di Staging:** Distribuire il flusso di lavoro in ambienti di staging che rispecchiano fedelmente l'ambiente di produzione. Eseguire test e validazioni aggiuntivi nell'ambiente di staging per individuare eventuali problemi relativi all'infrastruttura, alla configurazione o all'integrazione dei dati.
- 5. Meccanismi di Rollback:** Implementare meccanismi di rollback in caso di errori di distribuzione o problemi critici rilevati in produzione. Assicurarsi che il flusso di lavoro

possa essere rapidamente ripristinato a una versione stabile precedente per minimizzare i tempi di inattività e l'impatto sugli utenti.

Incorporando il testing e la validazione durante l'intero ciclo di vita dello sviluppo dei flussi di lavoro intelligenti, le organizzazioni possono garantire l'affidabilità, l'accuratezza e la manutenibilità dei loro sistemi basati sull'IA. Il testing e la validazione regolari aiutano a individuare i bug, prevenire le regressioni e costruire fiducia nel comportamento e nei risultati del flusso di lavoro.

Parte 2: I Pattern

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Prompt Engineering

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Chain of Thought

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempi

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Generazione di Contenuti

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Creazione di Entità Strutturate

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Guida agli Agenti LLM

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Vantaggi e Considerazioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Cambio di Modalità

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Quando utilizzarla

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Assegnazione del Ruolo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Quando Usarlo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempi

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Prompt Object

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Template di Prompt

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Vantaggi e Considerazioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Quando Utilizzarlo:

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Structured IO

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Scalare l'Input/Output Strutturato

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Vantaggi e Considerazioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Prompt Chaining

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Quando Utilizzarlo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio: Onboarding di Olympia

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Riscrittura dei Prompt

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Response Fencing

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Vantaggi e Considerazioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Gestione degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Query Analyzer

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Implementazione

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Etichettatura delle Parti del Discorso (POS) e Riconoscimento delle Entità Nominate (NER)

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Classificazione dell'Intento

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Estrazione delle Parole Chiave

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Vantaggi

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Riscrittura delle Query

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Benefici

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Ventriloquist

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Quando Utilizzarlo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Componenti Discreti

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Predicate

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Quando Utilizzarlo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

API Facade

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Benefici Principali

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Quando Usarlo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Autenticazione e Autorizzazione

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Gestione delle Richieste

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Formattazione delle Risposte

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Gestione degli Errori e Casi Limite

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Considerazioni su Scalabilità e Prestazioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Confronto con Altri Design Pattern

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Interprete dei Risultati

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Quando Utilizzarlo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Macchina Virtuale

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Quando Usarlo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Dietro la Magia

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Specifica e Testing

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Specificare il Comportamento

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Scrivere Casi di Test

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio: Testing del Componente Translator

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Riproduzione delle Interazioni HTTP

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Human In The Loop (HITL)

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Pattern di Alto Livello

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Intelligenza Ibrida

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Risposta Adattiva

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Cambio di Ruolo Uomo-IA

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Escalation

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Benefici Chiave

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Applicazione nel Mondo Reale: Sanità

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Feedback Loop

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Applicazioni ed Esempi

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Tecniche Avanzate nell'Integrazione del Feedback Umano

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Radiazione Passiva delle Informazioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Visualizzazione Contestuale delle Informazioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Notifiche Proattive

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Approfondimenti Esplicativi

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esplosione Interattiva

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Benefici Chiave

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Applicazioni ed Esempi

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Collaborative Decision Making (CDM)

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Apprendimento continuo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Applicazioni ed Esempi

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Considerazioni Etiche

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Ruolo del HITL nella Mitigazione dei Rischi dell'IA

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Progressi Tecnologici e Prospettive Future

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Sfide e Limitazioni dei Sistemi HITL

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Gestione Intelligente degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Approcci Tradizionali alla Gestione degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Diagnosi Contestuale degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Ingegneria dei Prompt per la Diagnosi Contestuale degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Generazione Aumentata tramite Recupero per la Diagnosi Contestuale degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Segnalazione Intelligente degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Prevenzione Predittiva degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Recupero Intelligente degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Comunicazione Personalizzata degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Flusso di Gestione Adattiva degli Errori

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Controllo Qualità

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Eval

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Problema

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Soluzione

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Considerazioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Comprendere i Riferimenti Golden

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funzionano le Valutazioni Senza Riferimento

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Guardrail

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Problema

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Soluzione

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Come Funziona

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Esempio

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Considerazioni

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Guardrails e Valutazioni: Due Facce della Stessa Medaglia

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

L'Intercambiabilità tra Guardrails e Valutazioni Senza Riferimento

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Implementazione di Guardrails e Valutazioni con Doppio Scopo

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Glossario

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Glossario

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

A

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

B

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

C

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

D

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

E

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

F

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

G

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

H

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

I

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

J

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

K

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

L

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

M

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

N

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

O

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

P

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Q

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

R

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

S

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

T

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

U

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

V

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

W

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Z

Questo contenuto non è disponibile nel libro di esempio. Il libro può essere acquistato su Leanpub a <http://leanpub.com/patterns-of-application-development-using-ai-it>.

Index

- accessibilità, 214
- account, 90
- adaptive workflow
 - Composizione Adattiva del Workflow, 222
- addestramento con istruzioni
 - modelli istruiti, 52
- addestramento tramite istruzioni, 10
- Agentici, 32
- AI, 64, 73, 99, 134, 142, 200
 - applications, 138
 - applicazioni, 125
 - conversazionale, 6, 31
 - model, 98, 99
 - modello, 89, 157
 - punti decisionali, 253
 - sistemi composti, 30, 34
- algebra lineare, 43
- Alpaca, 13
- Altman, Sam, 18
- Amazon Web Services, 248
- ambienti di staging, 256
- ambienti di sviluppo locali, 154
- analisi del sentimento, 17, 100, 111, 112, 114, 117, 118, 134, 144
- Anthropic, 23, 39, 73, 129, 136
- antropomorfismo, 68
- API, 123, 152
- APIs, 71
- application design and frameworks, 196
- applicazione chatbot, 118
- Applicazioni E-commerce, 91
- applicazioni educative, 32
- applicazioni moderne, 219
- apprendimento non supervisionato, 4
- Apprendimento One-Shot, 60
- apprendimento zero-shot, 59
- Architettura a Microservizi, 89
- architettura basata sugli eventi, 108
- architettura delle applicazioni aziendali, 38
- architettura distribuita, 246
- architettura software, 2
- architettura transformer, 6
- array, 130
- assegnazione del ticket, 236
- assistanti virtuali, 33
- audit e conformità, 243
- auto-scaling, 248
- base di conoscenza di Olympia, 91
- basi di conoscenza, 7
- basi di dati, 123
- BERT, 14, 24

- boundary conditions, 251
Brotli, 249
Byte Pair Encoding (BPE), 15
- C (Linguaggio di Programmazione), 116
caching, 246
Campionamento top-k, 48
Campionamento top-p (nucleus), 48
casi limite, 58
catena di approvvigionamento
 ottimizzazione, 32
Catena di Pensiero (CoT), 45
Chain of Thought (CoT), 138
chatbot per il servizio clienti, 33
ChatGPT, 30, 53
chiamata dello strumento, 152
chiamata di funzione
 fallimento, 133
classification, 120
classificatori per rilevanza, 35
classificazione, 53
Claude, 8, 43, 77
Claude 3, 49, 126, 129, 134, 136
Claude 3 Opus, 74
Claude v1, 17
Claude v2, 17
Codifica a Coppie di Byte (BPE), 13
Cohere (Fornitore LLM), 25
Cohere (LLM Provider), 23
colli di bottiglia, 222
compiti complessi, 145
completamento ad alte prestazioni, 26
comportamento deterministico, 58
computer desktop, 215
computer science, 72
concatenamento di worker AI, 111
conceptual and practical challenges, 197
concurrent workflows, 249
connettività di rete, 222
conservazione e rotazione dei log, 244
consistenza
 e riproducibilità, 132
contenuti
 Categorizzazione dei Contenuti, 112
 filtraggio, 26
contenuti generati dagli utenti, 111
conto
 Aumento, 46
 Generazione Contestuale dei
 Contenuti, 185, 190, 191
context
 Contextual Content Generation, 197,
 198
 Contextual Field Suggestions, 198
 finestra, 221
 infinitely long inputs, 16
 processo decisionale contestuale, 221
 window, 15
Continuazione Automatica, 159
conversazione
 ciclo, 158
 trascrizione, 156, 158
costruzione narrativa, 20
Customer Sentiment Analysis, 99

- database
 - oggetto basato su, 105
 - strategie di lock, 109
- Datadog, 245
- dati
 - analisi, 34, 146
 - attività di elaborazione, 125
 - Convalida dei dati, 255
 - flusso, 110
 - integrità, 236
 - persistenza, 109
 - pipeline di elaborazione, 236
 - preparazione, 109
 - privacy, 26, 213
 - Recupero dei Dati, 109
 - Sincronizzazione dei Dati, 109
- Dati Auto-Riparanti, 240
- Dati Auto-riparanti, 163
- dati di addestramento, 42
- dati in streaming, 151
- dati strutturati, 134
- debugging, 221
 - e risoluzione dei problemi, 243
 - e testing, 132
- decision
 - making capabilities, 99
- decisione
 - alberi, 218
 - casi d'uso, 133
 - punti, 241
- Delimitazione delle Risposte, 175
- dipendenti Databricks, 52
- direttiva di sistema, 128
- distorsione
 - ed equità nell'AI, 253
- dizionari, 130
- document clustering, 120
- Dohan, et al., 44
- e-commerce, 190, 218
- ecosistema, 147
- efficienza, 219
- elaborazione asincrona, 245
- elaborazione batch, 246
- elaborazione del flusso, 149, 161
 - logica, 157
- elaborazione dello stream, 155
- ELK stack, 110
- ensemble, 117, 118
 - ensemble di worker, 118
- Enterprise Integration Patterns, 104
- errori
 - gestione, 107, 109, 142
 - recupero, 255
 - tassi, 110
- errori di sintassi, 131
- errors
 - Gestione Intelligente degli Errori, 142
 - handling, 250
- esecuzione parallela, 246
- esperienza utente, 192
- ethics
 - implications, 197
- eventi inviati dal server (SSE), 149

- F#, 92
- Facebook, 24
- fattori di rischio, 95, 96
- feedback
- Ciclo di feedback, 59
- few-shot
- learning, 62
 - prompting, 63
- fiducia degli utenti, 214
- filtraggio basato sul contenuto, 91
- filtraggio collaborativo, 91
- fine-tuning, 80
- FitAI, 208
- flessibilità e creatività, 194
- flusso di lavoro multi-step, 111
- fornitori di hosting di modelli open source, 202
- framework di sviluppo, 148
- funzione
- chiamata, 123
 - cronologia delle chiamate, 155
 - nomi, 153
- Gemma 7B, 11
- Generative Pre-trained Transformer (GPT), 8, 67
- Generative UI (GenUI), 196
- Generazione Aumentata da Recupero (RAG), 46
- Generazione Aumentata dal Recupero (RAG), 31
- generazione cross-modale, 22
- generazione di dati sintetici, 53
- generazione dinamica dell’interfaccia utente, 186
- gestione del traffico, 32
- gestione della conoscenza, 32
- gestione delle eccezioni, 222, 224
- Gestore dei Processi, 104
- gestori del flusso, 150
- GitLab, 92
- Global Interpreter Lock (GIL), 115
- Google, 23
- API, 63, 65
 - Cloud AI Platform, 24
 - Cloud Platform, 248
 - Gemini, 21
 - Gemini 1.5 Pro, 14, 17, 19
 - PaLM (Pathways Language Model), 17, 24
 - T5, 14
- GPT-3, 13, 17
- GPT-4, 6, 13, 17, 21, 31, 43, 49, 63, 104, 117, 119, 127, 133, 201, 202, 246
- Graham, Paul, 19
- GraphQL, 108
- Groq, 26, 119
- gzip, 249
- hardware, 28
- hash, 151
- Hohpe, Gregor, 104
- Honeybadger, 94
- HTTP, 149

- IA, 128, 149, 207
applicazioni, 148, 161
conversazionale, 209
modello, 154, 155, 207
sistemi composti, 30
- inclusive interfaces, 197
- Inferenza, 5
- informatica, 70
- informazioni
estrazione, 53
recupero, 7, 126
- input
prompt, 56
validation, 250
- Instrandamento Dinamico delle Attività, 220
- instruction tuning
modelli istruiti, 49
- integrazione degli LLM, 186
- Integrazione e Distribuzione Continua
(CI/CD), 256
pipeline, 256
- intelligent workflow orchestration, 249
- interazioni in stile roleplay, 6
- interfacce controllate vocalmente, 33
- Interfaccia Generativa (GenUI), 214
- Interfaccia Utente (UI)
design, 215
framework, 211
interfacce, 211
tecnologie, 206
- interfaccia utente adattiva, 205
- Interfaccia Utente Generativa (GenUI), 203
- interfaccia visiva, 206
internazionalizzazione, 193
Interprete dei Risultati, 142
intervento manuale, 224
iperparametro, 47
- JSON (JavaScript Object Notation), 126, 130, 131, 134, 147, 166
- K-means, 121
- language
models, 65, 72
- Large Language Model (LLM), 16, 67, 68, 71, 77, 87, 110, 120, 124, 143, 144, 146, 185, 196, 201
- Latent Dirichlet Allocation, 121
- latenza, 27
- libreria Capybara, 255
- lingua
Rilevamento della Lingua, 111
- linguaggio
compiti correlati, 5
modelli, 43
- linguaggio naturale
Elaborazione del Linguaggio Naturale (NLP), 101
- Llama, 13
Llama 2-70B, 50
Llama 3 70B, 11
Llama 3 8B, 11
- lock ottimistico, 109
lock pessimistico, 109

- logging granulare, 244
logging strutturato, 244
logica del circuit breaker, 161
Louvre, 42
- Managed Streaming for Apache Kafka, 41
- Markdown, 146
- meccanismi di retry, 110
- meccanismi di rollback, 256
- Memorial Sloan Kettering Cancer Center, 41
- Mercurio (dio romano), 45
- Mercurio (elemento), 45
- Mercurio (pianeta), 45
- MessagePack, 249
- messaggio trigger, 104
- Meta, 24
- metodo finalize, 155, 157, 158
- Metropolitan Museum of Art, 42
- Mistral, 25
- 7B, 11
- 7B Instruct, 17, 202
- Mixtral
- 8x22B, 11
- 8x7B, 56
- modellazione autoregressiva, 43
- modelli basati sul recupero, 7
- modelli base, 54
- modelli grafici, 43
- Modelli Linguistici di Grandi Dimensioni (LLM)
- panorama, 27
- modelli probabilistici, 43
- modelli storici, 221
- Modello Linguistico di Grandi Dimensioni (LLM), 1, 3, 18, 29, 75, 123, 134, 140, 206, 228
- Modello linguistico di grandi dimensioni (LLM), 163, 166
- Moderatore di Contenuti Intelligente, 229
- modularità, 88
- Moltitudine di Worker, 118, 165
- monitoraggio
- e logging, 110, 243
- metriche, 244
- Monitoraggio Continuo del Rischio, 103
- monitoring
- e segnalazione, 223
- Multi-Agente
- Risolutori di Problemi, 31
- Multimodale
- modelli, 20
- modelli linguistici, 21
- Naive Bayes, 120
- narrow the path, 38
- natural language
- Elaborazione del Linguaggio Naturale (NLP), 120
- New Relic, 247
- occhiali per realtà aumentata, 215
- Ollama, 25
- Olympia, 33, 62, 128, 142, 150, 166
- OpenAI, 3, 23, 39, 73

- OpenRouter, 27, 28, 150, 247
OPT model, 24
orchestrazione del flusso di lavoro
 intelligente, 225
orchestrazione intelligente dei flussi di lavoro, 217
orchestrazione intelligente dei workflow, 247
output verification, 250
- panorama digitale, 192
parametri di input, 128
parametro
 effetti, 128
 intervallo, 11
 Numero di Parametri, 28
pattern chiave, 220
pattern matching, 151
Penalità di presenza, 48
penalità di ripetizione, 51
Perplexity (Provider), 12
personalization
 Personalized Forms, 198
personalizzazione, 27, 186, 215, 219
 Microcopy Personalizzato, 203
pianificazione della risposta alle emergenze, 32
prestazioni
 compromessi, 5
 ottimizzazione, 132, 194, 243
 problemi, 248
previsioni, 5
- principle of least privilege, 71
problemi di usabilità, 213
Process Manager, 107
 Enterprise Integration, 225
processo di distillazione, 76
Produttività, 188
programmazione funzionale, 92
prompt
 design, 58
 Distillazione dei Prompt, 46, 246
 ingegneria, 45, 46, 211
 ingegnerizzazione, 56
 Prompt Template, 202
prompts
 chaining, 71
 concatenamento, 59
 design, 68
 Distillazione dei Prompt, 78
 engineering, 41, 65, 67
 ingegneria, 59
 Prompt Distillation, 73
 Prompt Object, 74
 refinement, 68
 Template dei prompt, 59
proprietà ACID, 109
Protocol Buffers, 249
psicologia dell'utente, 212
publish-subscribe systems, 108
Pulizia del Testo, 111
PyTorch, 24
- Quantizzazione, 28

- Qwen2 70B, 11
- Raccolta dell'Anamnesi, 101
- Raccomandazioni dei Prodotti, 91
- raccomandazioni personalizzate dei prodotti, 91
- raffinamento iterativo, 75, 143
- Rails, 193
- Railway Oriented Programming (ROP), 94
- Raix, 226
- libreria, 97
- registrazione di audit, 106
- regole di business, 218
- regole grammaticali, 4
- regressione lineare, 43
- Response Fencing, 202
- restringere il percorso, 39
- reti neurali, 4, 6
- Retrieval Augmented Generation (RAG), 38, 79, 125
- riassunti, 53
- riformulazione, 53
- riga di comando
- Interfaccia a Riga di Comando (CLI), 25
- rilevamento delle frodi
- sistema, 97
- risposta a domande chiuse e aperte, 53
- rivelazione progressiva, 204
- rivenditori online, 202
- RSpec, 251, 252, 255
- Ruby, 92, 93, 113, 161, 255
- Ruby on Rails, 1, 111, 225, 233
- Rudall, Alex, 23
- Rust (Linguaggio di Programmazione), 116
- Rust (Programming Language), 92
- scalabilità, 219, 245
- scoperte mediche, 100
- Scout, 247
- scrittura creativa, 34, 52
- Selezione Dinamica degli Strumenti, 130
- Selezione Forzata degli Strumenti, 131
- senza stato, 156
- servizi esterni o API, 126
- sistemi di domanda e risposta, 7
- smartphone, 215
- spazio latente, 40, 42
- sperimentazione
- framework, 192
- spiegabilità, 253
- SQL injection, 70
- strategie di fallback, 110
- strategie di segmentazione e targeting, 192
- strategie motivazionali, 210
- Stratificazione del Rischio, 102
- Stripe, 129
- Structured IO, 202
- Supervisione Umana nel Processo, 178
- Support Vector Machines (SVM), 120
- Supporto alle Decisioni Cliniche, 103
- supporto clienti, 32
- sviluppo delle applicazioni, 217
- system directive, 98

- T5, 24
tablet, 215
tag in stile markup, 70
Temperatura, 54
Tempo al Primo Token (TTFT), 27
tempo di elaborazione, 110
teoria della mente, 40
test di integrazione, 251
test end-to-end, 254, 255
test utente e feedback, 195
throughput, 27
Together.ai, 26
token, 6, 13
tokenizzazione, 13
tono emotivo, 144
topic identification, 120
tracciamento delle metriche chiave, 240
traduzione, 17, 194
tragedia dei beni comuni, 189
UI Generativa (GenUI), 206, 211
Unicode-encodable language, 15
Universal ID, 249
User Interface (UI)
 interfaces, 196
uso degli strumenti, 123
utilizzo degli strumenti, 148
Valutazione e Stratificazione dei Sintomi,
 101
Ventriloquo, 175
Verifica Assicurativa, 101
voto di maggioranza, 117
Wall, Larry, 3
Wisper, 94, 106, 150, 157
Wooley, Chad, 92
XML, 134
Yi-34B, 50
zero-shot learning, 59