Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Strumenti basati su LLMs: Sviluppo di una applicazione di Document Intelligence

Relatore:
Chiar.mo Prof.
Paolo Ciancarini

Correlatore:
Giovanni Fidanza

Presentata da:
Samuele Busi

II sessione - secondo appello
Anno Accademico 2022/2023

Abstract

Il presente studio rivela il potenziale trasformativo dei Large Language Models (LLMs) nell'automatizzare l'analisi di dati complessi, un dominio un tempo esclusivo di architetture specializzate e team di sviluppo. Attraverso l'uso di modelli avanzati come GPT-4, si superano i limiti tradizionali, consentendo non solo la ricerca informativa in vasti database ma anche la generazione di codice eseguibile per l'analisi dati. Focalizzando su un caso di studio in ambito Industria 4.0, per la mia azienda Digibelt, questo lavoro illustra la costruzione di uno strumento basato su Large Language Models, esponendo le peculiarità di questa tecnologia e gli aspetti da tenere in considerazione per compiere scelte informate riguardo il modello. Sono delineate le tecniche per migliorare la comunicazione con i LLM e per affinare le risposte ricevute, sottolineando come un design di prompt incisivo possa elevare l'efficacia di tali sistemi. In particolare vedremo da dove derivano le abilità più utili dei modelli e apprenderemo come integrarle nella creazione di strumenti intelligenti. Riconoscendo la rapidità dell'evoluzione tecnologica, lo studio enfatizza l'importanza di progettare sviluppi che incorporino pro attivamente le imminenti innovazioni, assicurando che gli strumenti rimangano all'avanguardia e pronti per i cambiamenti futuri.

Dedica

Dedico questo lavoro ai miei colleghi di Digibelt che mi hanno insegnato tantissimo in questi anni facendomi crescere professionalmente e personalmente.

Indice dei Contenuti

1. Introduzione

1.1. Necessità di Strumenti Intelligenti

Nel campo dell'intelligenza artificiale, l'avanzamento dei Large Language Models ha portato a un significativo incremento nella capacità di svolgere compiti complessi in modo autonomo. Questi modelli avanzati non si limitano più a semplici interazioni basate su testo, ma possono ora essere personalizzati e adattati per risolvere specifici problemi in contesti diversi. La flessibilità dei LLM permette di configurarli per rispondere a richieste uniche, adattandosi dinamicamente al contesto presentato.

La vera rivoluzione avviene con l'introduzione di strumenti intelligenti alimentati da questi modelli. Questi strumenti hanno la capacità di interfacciarsi non solo con informazioni testuali, ma anche con strutture dati complesse, API, applicazioni esistenti e persino elementi visivi. La loro comunicazione, che si avvale del linguaggio naturale o della voce, rappresenta un salto qualitativo nella facilità d'uso e nell'accessibilità.

I vantaggi di questi strumenti intelligenti sono molteplici. In primo luogo, essi dimostrano una notevole resilienza ai problemi o alle eccezioni, grazie alla capacità innata dei LLM di tentare soluzioni alternative e di adattarsi per superare gli ostacoli incontrati. In secondo luogo, questi strumenti sono in grado di comprendere e soddisfare richieste che non sono state esplicitamente programmate durante la fase di progettazione, mostrando così un'eccezionale flessibilità. Infine, la loro natura modulare permette una facile espansione e modificabilità, in quanto sono spesso costituiti da catene di modelli specializzati e funzionalità accessibili tramite API.

Un tipico strumento intelligente si configura come una catena di modelli, ognuno specializzato in un compito specifico, integrato con un insieme di API che possono essere utilizzate direttamente o indirettamente dai modelli. Questi strumenti sono progettati per facilitare interazioni sofisticate con dati e applicazioni, eseguire ragionamenti complessi, svolgere calcoli, recuperare informazioni rilevanti e, infine, interagire con gli utenti in linguaggio naturale. Questa architettura modulare non solo migliora l'efficienza e l'efficacia delle operazioni, ma fornisce anche una base robusta per l'innovazione e l'adattamento a nuove sfide e requisiti futuri.

1.1.1. Document Intelligence

La Document Intelligence, nota anche come Information Retrieval, è un'area interdisciplinare focalizzata sulla rappresentazione, memorizzazione, organizzazione e accesso a informazioni contenute in vari oggetti, come documenti, pagine web, cataloghi online e oggetti multimediali. Il termine fu coniato da Calvin Mooers alla fine degli anni quaranta del Novecento e oggi è prevalentemente utilizzato in ambito informatico [Calvin N. Mooers, Charlotte Davis, Charles Babbage Institute, 1993].

Le tecnologie basate sull'intelligenza artificiale hanno trasformato il campo della Document Intelligence, consentendo l'estrazione e l'elaborazione di grandi quantità di dati, anche non strutturati, all'interno di documentazione di vario genere. Queste soluzioni riducono al minimo l'intervento umano e permettono di gestire con maggiore efficienza i Big Data. L'intelligenza artificiale, in particolare, permette di elaborare enormi quantità di dati e di gestirne ogni aspetto in tempi ridotti, migliorando così la produttività, la precisione e l'accuratezza delle operazioni su informazioni.

La Document Intelligence, o Intelligent Document Processing (IDP), combina l'intelligenza artificiale con tecnologie di elaborazione del linguaggio naturale (NLP) per comprendere, analizzare e gestire documenti digitali in modo automatico. Questo sistema consente di acquisire dati da documenti di vario genere e di trasformarli in dati utili e immediatamente interpretabili, spesso attraverso l'ausilio di ulteriori sistemi di intelligenza artificiale come NLP, computer vision, deep learning e machine learning.

Un sistema di Document Intelligence tipicamente include fasi come l'acquisizione di documenti, il preprocessing, il riconoscimento ottico dei caratteri (OCR), l'estrazione delle informazioni, l'indicizzazione e la categorizzazione, l'archiviazione e la gestione dei documenti, la ricerca e il recupero, nonché l'analisi e il fornire insights.

L'avvento dei Large Language Models ha ulteriormente potenziato le capacità degli strumenti di Document Intelligence. i LLMs sono in grado di scrivere codice per analizzare matematicamente e senza errori le informazioni, grazie alla loro capacità di gestire grandi volumi di dati e compiere analisi complesse, che possono essere ulteriormente rifinite e perfezionate per essere presentate all'utente finale.

1.2. Industria 4.0 e Digibelt

L'Industria 4.0, con la sua rivoluzione nell'ambito della manifattura, si fonda su quattro pilastri chiave: l'Internet delle cose (IoT), i Big Data e l'analisi dei dati, l'intelligenza artificiale e l'apprendimento automatico, e la robotica avanzata e l'automazione. Questi pilastri permettono di connettere macchine e dispositivi, raccogliere dati in tempo reale, analizzare i processi produttivi, prevedere necessità di manutenzione, gestire scorte e ottimizzare l'efficienza operativa. L'Industria 4.0 sposa perfettamente i principi fondamentali della Lean Economy, eliminando gli sprechi, ottimizzando i flussi di valore, seguendo il modello di produzione pull e promuovendo il perfezionamento continuo.

Un esempio eccellente di questa integrazione è rappresentato da Digibelt, un'azienda all'avanguardia nell'Industria 4.0 e nella Lean Economy. Grazie alla loro piattaforma versatile, Digibelt offre la raccolta dati in tempo reale e supporto all'operatore, riducendo sprechi e consentendo decisioni basate sui dati. Collaborando con CORVINA, Exor International e Bonfiglioli Consulting, Digibelt ha sviluppato una piattaforma Lean Industry 4.0 per mappare completamente e ottimizzare i processi manifatturieri, integrando processi produttivi e dati in tempo reale per il miglioramento continuo e la riduzione degli sprechi.

Gli sviluppi nell'ambito dei Large Language Models stanno influenzando profondamente l'Industria 4.0, offrendo strumenti avanzati per l'analisi, la gestione dei dati in tempo reale e l'integrazione dell'intelligenza artificiale nei processi produttivi. Questi modelli possono essere applicati per analizzare i dati di produzione, ottimizzare i flussi di lavoro e prevedere la manutenzione delle macchine. La loro capacità di elaborare grandi volumi di dati e fornire intuizioni significative rende i processi industriali più efficienti e adattabili, potenziando la produzione e supportando decisioni strategiche più informate.

Nel caso di Digibelt, i LLM possono essere sfruttati per analizzare i dati raccolti in tempo reale dalle loro piattaforme e migliorare ulteriormente i processi manifatturieri. Ad esempio, oltre a sistemi per l'assistenza automatica possono essere utilizzati per sviluppare modelli predittivi con conoscenza contestuale del processo, che anticipano i guasti delle macchine o per ottimizzare la pianificazione della produzione in base a molteplici fattori misurabili.

Ho intrapreso un progetto ambizioso per sviluppare uno strumento di Document Intelligence basato su Large Language Models per Digibelt, la mia azienda. Questa iniziativa sfrutta le capacità avanzate di analisi del linguaggio naturale e di comprensione dei testi dei LLM per ottimizzare i processi aziendali. L'applicazione creata integra di fatto tutti i dati gestiti dalla piattaforma Lean con tutte le abilità

di un Large Language Model; utilizzando le capacità e le tecniche che vedremo nei prossimi capitoli, portiamo il modello a comprendere le informazioni, analizzarle e interpretarle secondo il contesto per presentare soluzioni efficaci e personalizzate alle necessità aziendali.

2. Capacità dei Modelli

In questa sezione, ci immergeremo nel mondo dei Large Language Models per esplorare e comprendere a fondo ciò che li definisce e li distingue. Partendo dalle fondamenta, delineeremo le caratteristiche chiave che identificano e costituiscono un modello come GPT o LLAMA. Analizzeremo le abilità intrinseche di questi modelli, esaminando come il loro training su vasti set di dati influisca sulle loro capacità di elaborazione e generazione del linguaggio.

Approfondiremo la questione del fine tuning, investigando come questa pratica permetta di personalizzare ulteriormente i modelli per adattarli a specifici ambiti di utilizzo. Un focus particolare sarà dedicato alla 'context window', ovvero la quantità di informazione che un modello può gestire e ricordare durante una singola interazione, un aspetto fondamentale che discrimina le abilità dei vari modelli.

Infine, esploreremo le modalità di inferenza, ovvero come i modelli vengono effettivamente eseguiti per essere utilizzati, sia tramite API commerciali sia attraverso soluzioni cloud gestite o di self-hosting. Questo percorso ci permetterà di apprezzare pienamente la versatilità, la potenza e le potenziali applicazioni dei LLM nel contesto tecnologico attuale.

2.1. Abilità

Le abilità dei LLM sono notevolmente avanzate e multiformi. Il loro punto di forza principale è l'elaborazione del linguaggio naturale (NLP), che permette loro di comprendere, interpretare e generare il linguaggio umano in modo efficace e versatile. Questi modelli sono capaci di rispondere a domande complesse, scrivere testi coerenti e creativi, tradurre lingue e creare contenuti originali. Un'altra area di competenza riguarda la generazione di codice, dove i LLM possono aiutare nello sviluppo di software, scrivendo codice completo, offrendo suggerimenti di programmazione e facilitando il debugging. Sono anche in grado di interagire con API esterne, integrando e manipolando dati da diverse fonti, il che amplia ulteriormente il loro campo di applicazione.

2.2. Training e Fine Tuning

Il training dei Large Language Models è quanto li rende intelligenti, in grado di capire e comunicare; rappresenta un imponente sforzo ingegneristico e di ricerca nel

campo dell'intelligenza artificiale e del machine learning. Questi modelli utilizzano architetture neurali avanzate per apprendere l'interpretazione e la generazione del linguaggio umano. Il processo di training si avvale di tecniche di apprendimento automatico sia supervisionate che non supervisionate e si basa sull'uso di ampi dataset. Un esempio è il training di GPT-3, che ha richiesto un investimento significativo sia in termini di risorse computazionali, utilizzando 256 GPU e raggiungendo 20 petaflops di potenza, sia in termini economici, con un costo stimato di circa $12 milioni [Wiggers, 2023]. Questa impresa richiede settimane o mesi di lavoro distribuito e dimostra la necessità di risorse ingenti per addestrare LLMs di questa scala.

Nella scelta dei dataset per il training, si includono fonti come Common Crawl, WebText, testi di libri e Wikipedia, ognuno contribuendo in modo unico alle competenze del modello. Il costo energetico ed economico di questo processo lo rende accessibile principalmente a grandi entità o istituzioni di ricerca.

La composizione del dataset influenza direttamente le capacità e le caratteristiche del modello, con dataset specifici che possono portare allo sviluppo di modelli specializzati in determinati settori o stili comunicativi. Mentre i modelli più ampi offrono prestazioni superiori, quelli più piccoli richiedono meno risorse ma possono avere limitazioni nelle loro capacità.

In conclusione, il training di LLMs è un processo complesso e costoso. Per molte applicazioni, il fine-tuning di modelli pre-esistenti rappresenta una soluzione efficiente e accessibile, offrendo un equilibrio tra la potenza computazionale e l'accessibilità senza la necessità di affrontare direttamente il costo e la complessità del training da zero.

2.2.1. Fine Tuning

Il fine-tuning è un processo chiave nel raffinamento dei Large Language Models. Attraverso questa fase, un modello pre-addestrato su un vasto dataset generico viene specializzato per adattarsi a specifici contesti o esigenze, incrementando notevolmente la sua precisione e applicabilità.

Il fine-tuning consiste nell'adattare un modello di machine learning già addestrato su un nuovo insieme di dati specifici, che lo specializza in un particolare dominio o compito. Questo processo è cruciale per ottimizzare l'efficacia di un modello in specifici ambiti applicativi, rendendolo più utile e pertinente rispetto al modello originale.

Il processo di Fine-Tuning inizia con la selezione di un dataset mirato che

rappresenti il dominio o compito specifico. La qualità e la diversità dei dati sono essenziali per un fine-tuning efficace. Successivamente, il modello viene ulteriormente allenato su questo dataset, affinando i pesi e i parametri per migliorare le sue prestazioni nel contesto desiderato. Dopo il fine-tuning, il modello viene valutato per assicurarsi che soddisfi gli standard richiesti, ripetendo il processo con aggiustamenti nel dataset o nella strategia di training se necessario.

Tra i benefici principali ci sono un incremento significativo nella precisione delle risposte del modello e una riduzione dei costi e del tempo rispetto al training completo di un nuovo modello. Il fine-tuning consente anche una notevole versatilità applicativa, adattando il modello a vari domini e compiti mantenendo un'unica architettura di base.

La selezione di un dataset appropriato è fondamentale per il successo del fine-tuning. È altresì importante prevenire l'overfitting, ovvero l'apprendimento eccessivo da dati specifici, mantenendo la capacità del modello di generalizzare. La valutazione continua delle prestazioni è essenziale per garantire che il modello rimanga efficace e pertinente.

Il fine-tuning è un passaggio vitale nel ciclo di vita di un LLM, offrendo un modo efficace per personalizzare modelli avanzati per specifici compiti o domini. Rappresenta un equilibrio ideale tra prestazioni ottimali e uso efficiente delle risorse, aumentando significativamente la flessibilità e l'utilità dei LLMs in una vasta gamma di applicazioni pratiche.

2.3. Context Window

Prima di approfondire le context window, è essenziale considerare perché la comprensione (NLU) e la generazione del linguaggio naturale (NLG) siano sfide notevoli. Una ragione primaria è l'ambiguità intrinseca del linguaggio, dove parole o frasi possono avere significati multipli a seconda del contesto. Ad esempio, la frase "We need to run this program today" può avere significati diversi a seconda del contesto più ampio fornito, risolvendo l'ambiguità solo con informazioni contestuali aggiuntive.

La context window in un Large Language Model rappresenta il numero massimo di token che il modello può considerare contemporaneamente durante l'elaborazione del linguaggio. Un token può essere una parola, un frammento di essa o un simbolo di punteggiatura. La dimensione di questa finestra è cruciale per determinare quanto ampiamente e coerentemente un modello può comprendere e rispondere alle richieste basandosi su un contesto fornito.

Uno dei primi modelli avanzati BERT, aveva una context window di soli 512 token, ad oggi GPT-4 di OpenAI rappresenta un salto significativo in termini di capacità contestuale, con una context window estesa a 128k token. Questo ampliamento permette al modello di mantenere una coerenza notevolmente migliorata nelle conversazioni lunghe e nella gestione di testi complessi, garantendo una comprensione contestuale più profonda rispetto ai modelli con finestre più piccole. In confronto, il predecessore di GPT-4, GPT-3.5, aveva una context window più limitata, di 16k token [OpenAI, 2023][Ye et al., 2023].

D'altra parte, Mistral 7B, un potente modello open source, presenta una context window teorica di dimensioni simili, 128k token, sfruttando l'innovativa tecnologia Flash Attention-2[Jiang, Sablayrolles, Mensch, et al., 2023][Dao, 2023]. Questo meccanismo consente al modello di processare un ampio contesto con maggiore efficienza e velocità. Tuttavia, la dimensione effettiva della context window in Mistral 7B può variare a seconda di specifiche implementazioni e utilizzi del modello.

La grandezza della context window ha un impatto diretto sulla qualità della generazione del linguaggio dei LLM. Modelli con una context window più ampia sono in grado di riferirsi a elementi contestuali discussi in precedenza, mantenendo una narrazione coerente e contestualmente ricca. Tuttavia, ciò implica anche sfide tecniche, come aumenti nei requisiti computazionali e di memoria [Liu et al., 2023].

In conclusione, la context window nei LLM è fondamentale per determinare le capacità di elaborazione linguistica di questi modelli. Modelli come GPT-4 e Mistral 7B, con context window estese, offrono prestazioni superiori in termini di coerenza e comprensione contestuale, sebbene con un costo computazionale maggiore. La dimensione della context window e le relative tecniche di ottimizzazione rimangono aspetti cruciali nello sviluppo di LLM efficaci ed efficienti.

2.4. Inferenza: l'Esecuzione di un Modello

Nell'esecuzione di un Large Language Model, il focus centrale è il processo di inferenza, che mette in pratica le capacità del modello addestrato. Questa fase inizia quando il modello affronta nuovi input di testo, che vengono prima tokenizzati e poi processati attraverso la rete neurale. È in questa fase di inferenza che l'LLM dimostra le sue abilità apprese, come l'analisi del contesto e il riconoscimento di pattern linguistici. L'output generato, che può essere una

risposta a una domanda, la continuazione di un testo o qualsiasi altra forma di elaborazione linguistica, è il risultato di questo sofisticato processo di inferenza

Uno dei metodi prevede l'uso di piattaforme cloud come AWS SageMaker, che offre un ambiente gestito per il deploy di modelli con risorse scalabili e accesso facilitato a servizi di calcolo, adatto a soddisfare anche necessità di dimensioni considerevoli. In questa configurazione il modello viene caricato su SageMaker e configurato con parametri specifici per il tuning delle prestazioni. Ad esempio per eseguire un modello con 7B di parametri come Mistral 7B efficacemente, è necessario utilizzare una macchina virtuale con almeno 24GB di vRAM per un buon throughput e l'uso di pesi in formato float16 [Adithya S K, 2023].

Hugging Face Inference Endpoints consente di deployare facilmente Mistral 7B su un'infrastruttura completamente gestita a un costo di circa $1.30 per ora di utilizzo. Il modello viene eseguito su una singola GPU NVIDIA A10G, garantendo tempi di risposta rapidi per l'inferenza di generazione di testo [Hugging Face , 2023].

SkyPilot è un framework versatile e personalizzabile per l'esecuzione di LLM, AI e lavori batch su qualsiasi cloud, offrendo risparmi sui costi e la massima disponibilità di GPU. Questo metodo di deploy è particolarmente vantaggioso per il suo approccio flessibile completamente configurabile che permette di ottenere un risparmio economico [SkyPilot Team, 2023].

Il deploy on-premise o bare metal implica l'installazione e l'esecuzione del modello direttamente su hardware fisico. SkyPilot facilita questo processo, consentendo di sfruttare al meglio le capacità hardware disponibili e offrendo un controllo totale sull'ambiente di esecuzione grazie al supporto per Kubernetes. Offre un controllo totale sull'ambiente hardware e software, consentendo una personalizzazione senza precedenti. In un ambiente bare metal, la gestione dei dati e la privacy sono completamente controllati dall'utente.

Il deploy indipendente di modelli open source offre vantaggi significativi rispetto all'utilizzo di API commerciali, inclusa una personalizzazione più ampia, e un maggiore controllo sui dati privati. Inoltre, permette di ridurre i costi e la latenza e di creare soluzioni su misura.

Il deploy di un LLM offre un'ampia gamma di opzioni, ciascuna con i propri vantaggi in termini di costi, personalizzazione e controllo dei dati. La scelta del metodo di deploy più adatto dipende dalle esigenze specifiche e dalle risorse disponibili. Con una pianificazione adeguata e un'attenta valutazione delle opzioni, è possibile sfruttare al meglio le capacità del modello o dei modelli che si desidera utilizzare.

3. Scelta di un Modello

La selezione di un Large Language Model adeguato è un passo cruciale nella realizzazione di uno strumento intelligente. Questo capitolo introduce la decisione tra modelli proprietari, come GPT-4, e soluzioni open source, rappresentate da alternative emergenti come Mistral 7B. Esploreremo come ciascuna opzione si adatti a esigenze diverse, bilanciando potenza e personalizzazione. Inoltre, accenneremo ai progressi nella distillazione dei modelli per massimizzare efficienza e specificità mantenendo dimensioni contenute e abbassando il costo dell'inferenza del modello.

3.1. Modelli Proprietari: GPT-4 di OpenAI

GPT-4, sviluppato da OpenAI, è un modello di linguaggio multimodale avanzato che accetta input sia testuali che visivi e produce output testuali [OpenAI, 2023]. Si distingue per prestazioni di livello umano su vari benchmark professionali e accademici, come il posizionamento nel top 10% su esami simulati del bar exam; Ha inoltre ottenuto risultati eccellenti in altre prove difficili e disegnate per essere impervie ai LLM come HellaSwag. Al momento della scrittura GPT-4 sorpassa ogni modello open source su tutti i principali benchmark tecnici per LLM e ML. Una delle sue versioni più avanzate, "GPT-4 Turbo", migliora nel seguire le istruzioni, offre una modalità JSON, output riproducibili e chiamate di funzione parallele, con una finestra di contesto di 128K (oltre 300 pagine di testo) [Open AI, 2023].

Sebbene GPT-4 supporti attualmente solo input testuali, sono previste capacità di input visivi e vocali nel futuro. Rispetto a GPT-3.5, GPT-4 è più affidabile, creativo e in grado di gestire istruzioni più sfumate per compiti complessi, migliorando le prestazioni in diverse lingue. Un'innovazione significativa è la capacità di "steer" (guidare) il modello per ottenere risposte in uno stile e tono specifici, utile per personalizzare applicazioni e ottenere risultati più precisi. Le sue capacità di generazione di testo sono vastissime, consentendo applicazioni come la redazione di documenti, la scrittura di codice, la risposta a domande su basi di conoscenza, l'analisi di testi, la traduzione di lingue e la simulazione di personaggi per giochi.

Una funzionalità chiave è il "function calling", che permette agli utenti di descrivere funzioni e far sì che il modello produca oggetti JSON per chiamare una o più funzioni o API, essenziale per ottenere dati strutturati in modo affidabile;

questa funzionalità è corredata da altre varie abilità come il web browsing e l'esecuzione di codice. Inoltre, GPT-4 integra una funzionalità di retrieval di informazioni nota come RAG (Retrieval-Augmented Generation), che migliora la qualità e la pertinenza delle informazioni fornite dal modello utilizzando i dati forniti.

Nonostante le sue capacità avanzate, GPT-4 ha delle limitazioni, tra cui la tendenza comune a tutti i LLM ad "allucinare", e non supporta il fine-tuning, limitando la personalizzazione e l'addestramento su specifici set di dati o per specifici casi d'uso dopo il rilascio iniziale.

3.2. Modelli Open Source: Mistral 7B di Mistral AI

Il team Mistral AI introduce Mistral 7B un modello di linguaggio da 7 miliardi di parametri, progettato per ottenere prestazioni superiori e maggiore efficienza. Mistral 7B si distingue per aver superato i modelli open source da 13 miliardi di parametri LLAMA 2 in tutti i benchmark valutati e il modello da 34 miliardi di parametri (LLAMA 1) in ambiti quali ragionamento, matematica e generazione di codice [Jiang, Sablayrolles, Mensch, et al., 2023].

Mistral 7B utilizza l'attenzione raggruppata per query (GQA) e l'attenzione a finestra scorrevole (SWA). La GQA accelera significativamente la velocità di inferenza e riduce i requisiti di memoria durante la decodifica. La SWA, d'altra parte, gestisce sequenze più lunghe in modo più efficace e a un costo computazionale ridotto. Queste tecnologie di attenzione contribuiscono alla performance e all'efficienza inferenziale di Mistral 7B, consentendo al modello di fornire output di alta qualità con usi minori di risorse computazionali.

Mistral 7B è accompagnato da un'implementazione di riferimento che facilita il deployment in ambienti locali o su piattaforme cloud come AWS, GCP o Azure. L'integrazione con Hugging Face è stata semplificata per una maggiore facilità di integrazione.

Il modello utilizza un buffer cache a dimensione fissa che limita la dimensione della cache, riducendo l'uso della memoria di cache fino a 8 volte senza influenzare la qualità del modello. Questo approccio contribuisce ulteriormente all'efficienza del modello.

Mistral 7B suggerisce che i modelli di linguaggio possono comprimere la conoscenza più di quanto si pensasse in precedenza. Questa ricerca apre prospettive interessanti sulla tridimensionalità del problema (capacità del modello, costo di

addestramento, costo di esecuzione), indicando che c'è ancora molto da esplorare per ottenere la migliore performance con il modello più piccolo possibile.

In conclusione, Mistral 7B rappresenta un importante passo avanti nella creazione di modelli di linguaggio efficienti e ad alte prestazioni. La combinazione di innovazioni architettoniche e tecniche avanzate di attenzione lo rende un modello eccellente per un'ampia gamma di applicazioni reali, offrendo un equilibrio ottimale tra qualità dei risultati ed efficienza computazionale.

3.2.1. Distillazione di LLM

La distillazione dei modelli è una tecnica avanzata nel campo dell'intelligenza artificiale, essenziale per ottimizzare al massimo i Large Language Models basati sull'architettura Transformer [Gu, 2023]. Il processo coinvolge l'addestramento di un modello più piccolo ed efficiente, lo "studente", tramite un modello più grande "insegnante". Il risultato è un modello agile con prestazioni elevate e minori requisiti computazionali.

I modelli distillati, meno onerosi in termini di potenza di calcolo, sono più accessibili per varie applicazioni, specialmente in contesti con risorse limitate. Riducono i costi operativi, essendo meno dispendiosi da eseguire frequentemente o su grandi volumi di dati.

Nonostante le dimensioni ridotte, i modelli distillati mantengono una buona parte delle capacità del modello "insegnante", grazie al trasferimento efficiente delle conoscenze. Sono inoltre adattabili a casi d'uso specifici, fornendo soluzioni personalizzate.

Distillare un modello è complesso e richiede risorse, ma i benefici in termini di efficienza e adattabilità lo rendono un approccio valido. Molti gruppi di ricerca e aziende stanno sviluppando modelli distillati per scopi specifici, ideali per strumenti intelligenti che necessitano funzionalità mirate. Questa pratica migliora l'accessibilità e l'applicabilità dei LLM, ampliando le possibilità di impiego nell'intelligenza artificiale.

3.2.2. Da Mistral-7B a ZEPHYR-7B

Un esempio notevole di applicazione di questa pratica è il passaggio da Mistral-7B, sviluppato da Jiang et al. (2023) , a ZEPHYR-7B, un progetto guidato dal team H4 (2023) [Tunstall, Beeching, et al., 2023]. ZEPHYR-7B rappresenta un'evoluzione significativa, dove un modello LLM viene raffinato in un formato più

gestibile e mirato a precisi casi d'uso, mantenendo al contempo un'alta qualità di performance.

In questo specifico caso, ZEPHYR-7B è stato sviluppato utilizzando tecniche di distillazione per creare un modello che non solo è efficiente dal punto di vista computazionale, ma che soddisfa anche le esigenze specifiche in termini di allineamento intenzionale e risposta alle preferenze umane. La distillazione è stata effettuata in tre passaggi successivi, GPT-4 è stato utilizzato come insegnante, sia per supervisionare il fine tuning, sia per allineare gli output di Zephyr dando riscontri basati su dataset di interazioni con LLM come UltraFeedback.

Questo è un salto qualitativo nell'ambito dei LLM, poiché combina la potenza generale di prodotti come Mistral 7B con la necessità di modelli più snelli e specifici per determinati contesti. Facilmente eseguibile anche su hardware affittabile in cloud a basso costo(meno di 1$ per ora) rappresenta una valida soluzione per integrare funzionalità di comunicazione in linguaggio naturale.

3.3. Valutazione e Scelta

La scelta di un modello di linguaggio di grandi dimensioni tra le soluzioni commerciali private a pagamento, come GPT-4 di OpenAI o Cohere AI, e i vari modelli open source disponibili, come LLAMA di Meta o Mistral 7B di Mistral AI, rappresenta un'importante decisione strategica. Al momento della scrittura, i modelli commerciali tendono a primeggiare nelle prestazioni, come dimostrato dalla maggior parte dei benchmark, offrendo funzionalità avanzate come la Retrieval Augmented Generation o la ricerca su internet integrata "out of the box". Tuttavia, i modelli open source, nonostante necessitino inizialmente di maggiore personalizzazione, possono essere equipaggiati con abilità simili, come vedremo in capitoli successivi.

I modelli commerciali, pur garantendo prestazioni elevate e immediate, implicano una dipendenza da un servizio esterno a pagamento, con potenziali problemi di sicurezza dati e costi fissi. Inoltre, l'uso di tali modelli limita il controllo sull'architettura e sull'elaborazione dei dati. Al contrario, i modelli open source, una volta implementati in soluzioni personalizzate, offrono costi ridotti e maggiore flessibilità. La personalizzazione permette di eseguire solo quanto necessario, mantenendo i dati privati. Tuttavia, questa scelta comporta una maggiore complessità e sforzo gestionale.

Stiamo assistendo a una crescente disponibilità di modelli open source ottimizzati o distillati per eseguire task specifici con minore impiego di risorse, permettendo così scelte più mirate. Questa evoluzione offre un'alternativa valida ai modelli

commerciali, specialmente per quelle organizzazioni che cercano un maggiore controllo e flessibilità, pur mantenendo un'elevata qualità delle prestazioni.

Esistono numerosi benchmark e strumenti di valutazione delle capacità dei modelli, questi coprono compiti e test che spaziano da contesti generali ad applicazioni incredibilmente specifiche [Hugging Face, 2023]. La validità di tali comparazioni rimane contestuale e dovrebbe essere verificata prima di procedere con una scelta definitiva, tenendo conto delle specifiche esigenze e risorse disponibili; data la rapidissima evoluzione sia dei modelli, ma soprattutto delle metodologie di test è cruciale consultare questi dati nella loro versione più aggiornata.

4. Interazione e Ottimizzazione Modello

In questo capitolo, esploreremo le dinamiche dell'interazione con i Large Language Models e le modalità per ottenere output efficaci. Discuteremo di come i modelli comunicano con i dati, strumenti esterni e, infine, con l'utente finale.

Un focus particolare sarà dedicato al prompt engineering, una nuova scienza che definisce le strategie ottimali per interagire con un LLM, garantendo risultati in linea con le aspettative. Per ottenere maggiore precisione e allineamento agli obiettivi in un Large Language Model, è essenziale guidare adeguatamente il modello, fornendogli un contesto sufficientemente informativo. Tale obiettivo è realizzabile tramite il prompt engineering. Tuttavia, quando i limiti della context window lo rendono insufficiente, si può integrare conoscenza esterna attraverso la Retrieval Augmented Generation, che discuteremo in un capitolo successivo. In casi dove si richiede una precisione superiore, il fine tuning diventa la pratica più adatta, come evidenziato nell'introduzione ai modelli. La selezione delle tecniche da adottare si basa prevalentemente sulla sperimentazione diretta e sulla valutazione dei risultati ottenuti. L'impiego combinato di tutte queste tecniche costituisce il grado massimo di personalizzazione attualmente realizzabile senza ricorrere a metodi più onerosi, come la distillazione del modello o l'addestramento ex novo [Colin Jarvis, John Allard, Open AI, 2023].

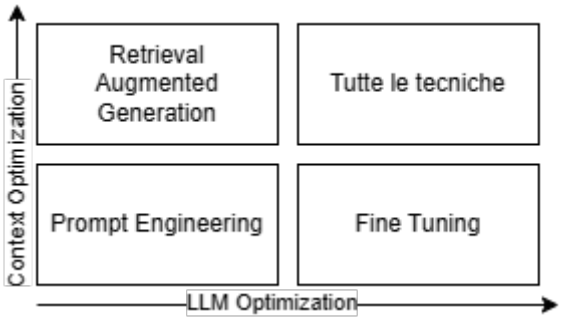

Figure 1. Ottimizzazione di un modello

L'immagine mostra come la scelta di una o più tecniche possa spostare la rifinitura più verso un'ottimizzazione e specializzazione dell'intero modello oppure portare all'utilizzo ottimale del contesto disponibile. L'applicazione di tutte le pratiche rappresenta il massimo della personalizzazione ottenibile.

Affronteremo anche il tema dell'espansione delle capacità dei modelli, esaminando come possono interagire con basi di dati o API, e persino eseguire codice su un interprete o un computer. Un aspetto fondamentale di questa discussione sarà l'introduzione della tecnica del model chaining. Questo approccio consente di

costruire sistemi basati su LLM con capacità specifiche, adatte agli scopi richiesti.

4.1. Prompt Engineering

Il Prompt Engineering è una disciplina fondamentale per massimizzare l'efficacia dei Large Language Models (LLMs). In questo ambito, la capacità di formulare prompt adeguati può determinare il successo o il fallimento nell'ottenere risposte utili dai modelli di linguaggio. La ricerca e lo sviluppo nel campo del Prompt Engineering hanno portato a tecniche avanzate e strategie innovative per affrontare sfide comuni e per applicare efficacemente i LLMs in vari contesti.

4.1.1. Chain of Thought Prompting

Una tecnica significativa nel campo del Prompt Engineering è il Chain of Thought (CoT) Prompting. Questo approccio consiste nell'incorporare nel prompt una serie di passaggi logici che guidano il modello verso una conclusione. Il CoT si rivela particolarmente efficace in situazioni dove è necessario un ragionamento passo-passo, come nella soluzione di problemi matematici o nella formulazione di argomentazioni logiche.

Il principio alla base del CoT Prompting è che, fornendo un percorso di ragionamento esplicito, il modello può seguire una sequenza logica che lo aiuta a raggiungere la risposta corretta. Questa tecnica è stata dimostrata essere particolarmente utile per migliorare la capacità dei modelli di gestire compiti che richiedono un'analisi approfondita o una risoluzione di problemi complessa.

4.1.2. Zero-Shot e Few-Shot Prompting

Nel Zero-Shot Prompting, un LLM affronta richieste senza un addestramento specifico, dimostrando la sua capacità di utilizzare conoscenze generali per generare risposte pertinenti. Questo approccio è utile quando non si dispone di dati specifici per l'addestramento, sfruttando l'ampiezza della comprensione generale del modello. Nonostante zero-shot funzioni per molte richieste, i LLM possono essere meno precisi in compiti complessi.

Il Few-Shot Prompting migliora l'apprendimento in contesto, fornendo dimostrazioni all'interno del prompt per guidare il modello. Queste dimostrazioni condizionano il modello per esempi successivi, aumentando l'accuratezza delle risposte.

Questo metodo sfrutta esempi limitati per ottimizzare le prestazioni, migliorando l'adattabilità del modello a richieste specifiche. Le dimostrazioni servono come riferimento, insegnando al modello come applicare il contesto fornito a nuove situazioni.

4.1.3. Self-Consistency Prompting

Il Self-Consistency Prompting è un metodo avanzato che coinvolge l'uso di più prompt per lo stesso problema, al fine di generare diverse soluzioni. La risposta finale viene selezionata sulla base della coerenza tra queste soluzioni multiple. Questa pratica è stata identificata come un modo efficace per aumentare l'accuratezza e la affidabilità delle risposte fornite dai LLMs, specialmente in situazioni in cui una singola iterazione potrebbe non portare alla soluzione corretta.

4.1.4. Instructed Prompting

L'Instructed Prompting implica l'uso di istruzioni specifiche all'interno del prompt per guidare il modello su come affrontare un determinato problema o compito. Questa pratica è particolarmente utile quando si desidera che il modello segua un certo approccio o metodologia nella risposta. L'Instructed Prompting può essere utilizzato per incoraggiare il modello a concentrarsi su aspetti specifici di un problema o per guidarlo a ignorare determinate informazioni irrilevanti.

4.1.5. Tree of Thoughts

Il framework Tree of Thoughts (ToT), sviluppato da Shunyu Yao e Jieyi Long [Yao et al., 2023][Long, 2023], rappresenta un'evoluzione nel campo del prompt engineering per modelli linguistici di grandi dimensioni. Questa tecnica struttura la risoluzione dei problemi attraverso un "albero di pensieri", dove ogni "pensiero" è un passo intermedio verso la soluzione. Il modello non solo genera questi pensieri ma li autovaluta anche, integrando algoritmi di ricerca per un'esplorazione metodica delle diverse possibilità. Hulbert ha poi semplificato questo approccio nel Tree-of-Thought Prompting, dove il modello valuta direttamente i pensieri intermedi in un unico prompt. Questo metodo si dimostra particolarmente utile per problemi complessi che necessitano di una sequenza di ragionamenti logici.

Per esemplificare, un prompt ToT potrebbe essere formulato così:

```
"Immagina che tre esperti differenti rispondano a questa
domanda.
```

```
Ogni esperto annota un passo del proprio ragionamento, poi lo
condivide con il gruppo.
Dopo, tutti gli esperti procedono al passo successivo.
Se un esperto si rende conto di un errore, si ritira.
La domanda è..."
```
Questo esempio mostra come ToT incoraggi il modello a esplorare diverse soluzioni attraverso un processo iterativo e collaborativo.

4.1.6. ReAct Prompting

La tecnica di "ReAct Prompting" rappresenta un'avanguardia nel campo dell'intelligenza artificiale, specificamente nella progettazione di prompt per Large Language Models. Introdotto da Shunyu Yao nel 2022 [Yao et al., Revised in 2023], questo framework permette ai LLM di generare tracce di ragionamento e azioni specifiche per il compito in modo intercalato. Le tracce di ragionamento consentono al modello di indurre, tracciare e aggiornare piani d'azione, gestendo anche eccezioni. Durante la fase di ragionamento, il modello elabora piani d'azione, tenendo traccia dei cambiamenti e adattandosi a possibili eccezioni. Successivamente, nella fase di azione, l'LLM interagisce con strumenti esterni per raccogliere informazioni, integrando così la sua conoscenza interna con dati aggiuntivi. Questo approccio migliora notevolmente l'affidabilità e la precisione delle risposte fornite dall'LLM. In un contesto più ampio, questa complessa tecnica di prompt engineering può essere applicata automaticamente attraverso il model chaining, un concetto che verrà approfondito nei capitoli successivi.

4.1.7. System Message

I "System Message" nel Prompt Engineering sono fondamentali per influenzare le risposte dei Large Language Models come GPT. Questi messaggi, posizionati prima del prompt principale, stabiliscono il contesto e il tono e forniscono istruzioni su come il modello dovrebbe agire. Questo approccio è cruciale per compiti che richiedono all'LLM di seguire direttive specifiche, mantenere uno stile coerente o aderire a procedure dettagliate.

L'utilizzo dei System Message consente di impartire comandi chiari al modello senza necessità di una rieducazione o di un fine-tuning estensivo. Questi messaggi funzionano come parametri che orientano le risposte dell'LLM, risultando utili in contesti che richiedono un output uniforme, come nei servizi di assistenza clienti automatizzati o nella generazione di contenuti.

La lunghezza dei System Message deve essere bilanciata con la dimensione della

finestra di contesto del modello. Nei modelli con finestre di contesto limitate, i System Message lunghi possono ridurre la capacità del modello di includere il contesto originale del prompt. Al contrario, modelli con finestre di contesto più ampie possono gestire System Message più dettagliati.

4.1.8. Conclusione

Le tecniche di Prompt Engineering rappresentano uno strumento cruciale per sfruttare appieno le capacità dei LLMs. Attraverso l'uso strategico di prompt ben progettati, è possibile guidare questi modelli verso risposte più precise, contestualmente rilevanti e logicamente coerenti. Man mano che la comprensione e l'applicazione del Prompt Engineering continua a evolversi , si aprono nuove opportunità per l'impiego efficace dei LLMs in una vasta gamma di settori e applicazioni.

4.2. Retrieval Augmented Generation

I Large Language Models, vengono di solito addestrati offline, il che rende il modello agnostico rispetto ai dati creati dopo l'addestramento del modello. Inoltre, i dataset di addestramento sono spesso prevalentemente corpora di dominio molto generale, rendendoli meno efficaci per compiti specifici di un determinato dominio. La possibilità di aggiungere conoscenza nel prompt non è sempre applicabile, per motivi banali come la dimensione della context window, e risulta spesso inefficace se non accompagnata da una strutturazione studiata del prompt. La Retrieval Augmented Generation consente di superare queste sfide integrando un meccanismo di recupero dati esterno, che permette ai LLM di accedere e utilizzare informazioni aggiornate e specifiche del dominio.

Con RAG, i dati esterni utilizzati per arricchire i prompt possono provenire da diverse fonti di dati, come repository di documenti, database o API. Il primo passo è la conversione dei documenti e delle query degli utenti in un formato compatibile per eseguire una ricerca di rilevanza. A tal fine, una raccolta di documenti, detta anche libreria di conoscenza, e le query degli utenti vengono convertite in rappresentazioni numeriche mediante modelli di embedding linguistico. L'embedding è il processo mediante il quale il testo viene rappresentato numericamente in uno spazio vettoriale. Le architetture dei modelli RAG confrontano gli embedding delle query degli utenti all'interno del vettore della libreria di conoscenza. Nella pratica questo meccanismo è implementato con l'impiego di uno o più modelli detti Retriever. Successivamente, il prompt originale dell'utente viene ampliato con il contesto rilevante proveniente da documenti simili all'interno della libreria di conoscenza [Lewis, Revised in 2023]. Questo prompt arricchito viene quindi inviato al modello di base. È possibile aggiornare le librerie di conoscenza e i relativi embedding in modo asincrono.

La Retrieval Augmented Generation rappresenta un passo avanti nella capacità di gestire dati aggiornati e specifici del dominio da parte dei modelli di base, migliorando notevolmente la loro capacità di generare risposte contestualmente appropriate e informativamente ricche.

4.3. Utilizzo di Strumenti

L'integrazione di strumenti esterni utilizzabili Large Language Models, come dimostrato dal framework Automatic Reasoning and Tool-use (ART), rappresenta un notevole progresso nella capacità di questi modelli di affrontare compiti complessi, permettendo ad esempio l'utilizzo di un calcolatore da parte del modello

per rispondere a quesiti matematici [Paranjape et al., 2023]. Questo approccio combina il ragionamento a catena di pensiero (Chain of Thought) con l'uso di risorse esterne come calcolatrici, browser web e funzioni di programmazione, consentendo ai LLM di superare i limiti delle loro capacità native. In ART, durante il processo di generazione delle risposte, l'LLM può fermarsi temporaneamente per richiamare uno strumento esterno, come una funzione di ricerca web o una chiamata di funzione in un linguaggio di programmazione, e poi riprendere la generazione della risposta integrando i risultati ottenuti. Questa funzionalità resa inizialmente disponibile da OpenAI per l'API di GPT-3 è ora fortemente integrata in tutti i nuovi modelli creati dall'azienda, ma per l'utilizzo con modelli open source che non la possiedono nativamente può essere aggiunta mediante chaining di modelli.

4.4. Code Execution

La rivoluzione portata dai LLM nel campo della programmazione risiede nella loro capacità di scrivere codice senza errori, perfettamente eseguibile, che utilizza in modo efficace e corretto librerie e API. La vera svolta, però, è stata aggiungere la possibilità di fare eseguire il codice e valutarne il risultato direttamente dal modello; sbloccando così l'intero potenziale del codice generato dai LLM, fornendo loro accesso a interpreti, terminali e in generale l'intera macchina fisica per l'esecuzione diretta del codice.

I LLM hanno dimostrato di poter generare codice che non solo risolve problemi complessi, ma che lo fa con precisione ed efficienza. Questo include il corretto utilizzo di librerie esterne e l'interazione con tutti i tipi di API, persino quelle di un sistema operativo.

Con la Code Execution, i LLM non sono più limitati alla sola generazione di codice. Possono ora eseguirlo autonomamente, analizzando in tempo reale l'output per utilizzarlo alla scopo dato o per correggere automaticamente il codice in caso di errore o risultati non attesi. Questo apre un mondo di possibilità in cui i LLM non solo suggeriscono soluzioni, ma le mettono anche in pratica.

4.4.1. Strumenti di Code Execution

ChatGPT Code Interpreter

Il Code Interpreter di ChatGPT, sviluppato da OpenAI, utilizza GPT-4 per

eseguire codice generato in un ambiente sandboxed. Offre un modo sicuro e controllato per testare e utilizzare il codice, ideale per l'apprendimento e la sperimentazione in un contesto sicuro, ma con alcune limitazioni come l'assenza di accesso a Internet.

Open Interpreter

Open Interpreter è un progetto open source che estende le possibilità di esecuzione del codice agli ambienti locali. Questo strumento permette agli utenti di eseguire codice direttamente sui loro dispositivi, fornendo maggiore flessibilità e controllo, e può essere integrato in applicazioni personalizzate grazie alla sua libreria Python. Questo strumento è comprensivo dell'integrazione con quasi tutti i LLM disponibili, eseguiti esternamente o localmente grazie a Lite LLM [Berri AI Team , 2023].

LangChain CodeBoxAPI

CodeBoxAPI di LangChain offre un ambiente di esecuzione del codice innovativo per i LLMs, consentendo loro di interagire con un interprete Python. Questo approccio permette agli Agent di LangChain di eseguire codice, come la visualizzazione di dati o l'elaborazione di immagini, direttamente su un dispositivo dell'utente, creando un blocco Jupyter personale per un'interazione avanzata e versatile.

4.5. Rifinitura Risultati

4.5.1. Model Chaining

Il Model Chaining nei LLM (Large Language Models) sta emergendo come una tecnica fondamentale per estendere e potenziare le capacità degli attuali modelli di intelligenza artificiale. Questo approccio, che implica l'integrazione sequenziale di diversi modelli o componenti software, dove output finali o intermedi di un modello diventano input per altri modelli, si rivela particolarmente efficace nell'aggiungere funzionalità avanzate a modelli che non le supportano nativamente. Un esempio rilevante è l'implementazione della tecnica Retrieval Augmented Generation in modelli che non la includono di default, permettendo loro di accedere e integrare informazioni da una vasta gamma di fonti esterne.

Inoltre, il Model Chaining apre nuove possibilità nel campo del prompt engineering

avanzato. Attraverso l'utilizzo di catene di modelli, è possibile rifinire e ottimizzare i prompt, utilizzando altri modelli o algoritmi per migliorare l'efficacia dei prompt iniziali. Questo processo consente di creare input più precisi e su misura per specifiche esigenze o contesti, aumentando significativamente la qualità e la rilevanza delle risposte generate.

Un altro vantaggio significativo del Model Chaining è la capacità di integrare strumenti esterni, come dimostrato dal framework ART (Adaptive Retrieval Tools). Questo approccio consente di sfruttare modelli specializzati nell'elaborazione di strategie di Chain of Thought che prevedono l'impiego di risorse esterne. Attraverso il Model Chaining, i modelli possono delineare piani di azione che includono la chiamata a strumenti esterni, estendendo notevolmente il loro ambito di applicazione e la loro utilità pratica.

Queste tecniche avanzate di Model Chaining non solo aumentano la potenza e la versatilità dei LLM, ma offrono anche ai ricercatori e agli sviluppatori la possibilità di costruire soluzioni su misura per sfide complesse e specifiche.Sebbene sia possibile realizzare Model Chaining programmando direttamente in codice, l'impiego di framework specifici come LangChain, uno tra i diversi disponibili, fornisce vantaggi significativi in termini di flessibilità, modularità e facilità di integrazione. Prendendo LangChain come esempio, esploreremo le funzionalità che possono arricchire e rendere più efficace il Model Chaining, mostrando come questo strumento possa trasformare l'uso dei LLM in soluzioni personalizzate e complesse.

Vantaggi di LangChain

1. **Flessibilità e Modularità**: LangChain si distingue per la sua enfasi sulla flessibilità e modularità. Suddividendo la pipeline di elaborazione del linguaggio naturale in componenti separati, permette agli sviluppatori di personalizzare i flussi di lavoro in base alle loro esigenze.
2. **Componenti e Catene**: In LangChain, i componenti sono moduli che eseguono funzioni specifiche nella pipeline di elaborazione linguistica. Questi possono essere collegati in "catene" per flussi di lavoro su misura, ad esempio per costruire un chatbot di assistenza clienti con moduli di analisi del sentimento, riconoscimento delle intenzioni e generazione di risposte.
3. **Template di Prompt**: I template di prompt sono prompt predefiniti riutilizzabili in diverse catene. Questi template possono diventare dinamici e adattabili inserendo "valori" specifici, rendendoli utili per generare prompt basati su risorse dinamiche.
4. **Memorizzazione e Ricerca di Informazioni**: LangChain utilizza "Vector Stores" per memorizzare e cercare informazioni tramite embedding,

analizzando rappresentazioni numeriche dei significati dei documenti. Gli indici e i retriever agiscono come database per immagazzinare dettagli e metadati sui dati provvisti, migliorando le risposte del modello fornendo contesto e informazioni correlate.

5. **Parser di Output**: Questi moduli gestiscono e affinano le risposte generate dal modello, eliminando contenuti indesiderati, adattando il formato dell'output o integrando dati aggiuntivi.
6. **Selettori di Esempi e Agenti**: I selettori di esempi identificano istanze appropriate dai dati di addestramento del modello, migliorando la precisione e la pertinenza delle risposte generate. Gli agenti sono istanze uniche di LangChain, ognuna con prompt, memoria e catena specifici per un caso d'uso particolare, e possono essere distribuiti su varie piattaforme.

L'uso di un framework come LangChain nel Model Chaining nei LLM offre una serie di vantaggi, tra cui la possibilità di costruire applicazioni complesse e personalizzate con maggiore facilità e flessibilità. Questo approccio apre nuove possibilità nell'uso dei LLM, permettendo lo sviluppo di soluzioni innovative in vari scenari e settori.

5. Delta AI

5.1. Introduzione

Vediamo ora un esempio pratico di uno strumento basato sulle capacità di un Large Language Model, evidenziando come adattare e interagire con un modello per ottenere soluzioni applicabili direttamente ai fini aziendali. Delta AI, integrata con la Lean Digital Platform di Digibelt, si presenta come una soluzione di Document Intelligence per l'Industria 4.0. È progettato per elaborare dati produttivi di aziende manifatturiere, creando funzioni di analisi avanzata e visualizzazione, controllo e suggerimento mediante il linguaggio naturale. Quest'approccio consente un utilizzo intuitivo e versatile. La progressiva evoluzione degli LLM, incorporati in Delta AI, ne amplifica le capacità, rendendo possibile la creazione di funzionalità sempre più elaborate. L'interfaccia web di Delta AI facilita la programmazione e la schedulazione di queste funzioni, rendendole accessibili e semplici da gestire.

5.1.1. Scopo e Obiettivo di Delta AI

L'obiettivo primario dell'applicazione Delta AI è di creare funzionalità attraverso la descrizione di un obiettivo da realizzare a partire da dati estratti. Disponibile al personale di Digibelt, Delta AI deve permettere di specificare quali dati estrarre, definire la procedura desiderata di analisi dei dati e determinare come rielaborare o creare suggerimenti basati sull'analisi. Questo è realizzato con l'intento di fornire informazioni utili agli operatori che lavorano nelle aziende manifatturiere. Inoltre, Delta AI deve consentire la schedulazione dell'esecuzione delle funzionalità, garantendo così che gli operatori ricevano suggerimenti sempre aggiornati. Infine, Delta AI deve essere progettata per l'uso tramite REST API, permettendo l'integrazione con altri applicativi per eseguire funzionalità in risposta a specifici eventi. La manipolazione dei dati di produzione dei clienti solleva anche importanti requisiti di sicurezza, in particolare la necessità di non inviare quanto raccolto dalla Lean Digital Platform al di fuori dei server on premise, di chi sceglie di non utilizzare processing in cloud.

5.1.2. Contesto di Utilizzo nell'Industria 4.0

Nel contesto della Lean Industry, un caso d'uso di Delta AI mira a ottimizzare la gestione delle risorse minimizzando le inefficienze. In particolare, nelle operazioni di produzione, si possono verificare interruzioni, note come fermi, causate da guasti alle macchinari. Per affrontare questa problematica, si propone l'implementazione di una funzionalità su Delta AI. Questa consiste nell'estrazione e nell'analisi dei dati relativi alla produzione attuale e storica, nonché dei fermi e altre problematiche. L'analisi comparativa tra dati storici e correnti consente di individuare tendenze dominanti e potenziali discrepanze nella produzione in corso. Successivamente, si prevede l'elaborazione di un report sintetico che metta in evidenza eventuali problemi riscontrati e proponga soluzioni appropriate, assegnando a ciascuna problematica un livello di gravità. Questo sistema permette agli operatori di ricevere notifiche tempestive, ad esempio nel caso di rallentamenti produttivi rispetto ai valori medi storici, facilitando interventi mirati e tempestivi per mantenere l'efficienza operativa.

5.2. Architettura

Nello sviluppo dell'architettura di Delta AI, l'obiettivo era mappare efficacemente le funzionalità richieste con i Large Language Models, e costruire attorno a questi un sistema che facilitasse l'integrazione e l'interazione con applicazioni esterne. L'interfaccia di gestione e configurazione è progettata per consentire un'interazione agevole e intuitiva, rendendo Delta AI un ponte tra le avanzate capacità degli LLM e le necessità pratiche degli utenti finali in ambienti industriali.

5.2.1. Processo di Progettazione

La sfida di rendere possibile il recupero e l'analisi di dati attraverso procedure e codice implementato da un LLM era considerevole solo pochi mesi prima della redazione di questa ricerca. In particolare, i dati derivanti dalla Lean Digital Platform presentano strutture complesse e non immediatamente interpretabili. Ad esempio, una richiesta banale come "Conta gli ordini completati sulla linea ASSEMBLAGGIO 3" richiede che l'LLM abbia una comprensione approfondita del datamodel.

Per superare questa sfida, è stata introdotta la creazione di un glossario per collegare efficacemente i dati di Digibelt alle query in linguaggio naturale. Tuttavia, con modelli LLM che supportano una Context Window di 8k Token, il caricamento del glossario in un prompt diventa problematico, essendo sia rischioso che potenzialmente insufficiente. Di conseguenza, l'adozione di metodi alternativi per integrare conoscenza aggiuntiva nel modello è diventata una necessità.

Le opzioni disponibili, come il fine tuning e il Model Chaining per aggiungere la Retrieval Augmented Generation, sono state esplorate. Sebbene LangChain faciliti notevolmente la creazione di un'architettura LLM multi-componente, l'introduzione di modelli con Context Window dieci volte più ampie ha spostato l'attenzione dal model chaining al prompt engineering.

5.2.2. Scelte Progettuali e Integrazione dei Componenti

Un importante aspetto nella progettazione dell'architettura di Delta AI era determinare come rendere accessibili i dati della Lean Digital Platform al Large Language Model senza trasferirli direttamente. L'obiettivo primario del modelo, focalizzato sulla scrittura di codice per l'analisi dei dati, non richiede la visualizzazione diretta dei dati stessi. È sufficiente che l'LLM sia informato sulla

loro ubicazione e struttura affinchè la inserisca nel codice scritto. Pertanto, il compito di recuperare i dati sensibili è stato affidato a un componente non intelligente, il quale carica i dati sul filesystem e comunica al modello dove trovarli. L'LLM, a sua volta, può interagire con alcuni metodi di questo modulo, ma esclusivamente attraverso quelle funzioni che sono state esplicitamente esposte per l'uso.

Per permettere al modello di eseguire codice e ricevere l'output dell'esecuzione, è stata integrata la soluzione Open Interpreter. Questo strumento, già discusso, si è rivelato particolarmente robusto e immediatamente compatibile con i principali modelli commerciali e open source, richiedendo solo piccoli aggiustamenti del system message per invogliare il modello all'esecuzione del codice. Essendo completamente configurabile, adattare l'esecuzione del codice alle esigenze di Delta AI è stato relativamente semplice.

Il maggiore impegno nello sviluppo di Delta AI, dopo l'integrazione con i modelli di nuova generazione, è stato rivolto al perfezionamento delle pratiche di prompt engineering. Questo era necessario per rendere la descrizione di una procedura accessibile e chiara per il personale non tecnico. Prima dell'adozione dei nuovi modelli, il model chaining per aggiungere una fase di raffinamento del prompt sembrava la pratica più promettente. Tuttavia, con l'espansione della context window, l'approccio Chain-of-Thought nel prompt engineering si è rivelato estremamente efficace. Poiché i compiti richiesti dall'LLM sono simili e variabili solo nell'obiettivo per cui il codice deve essere scritto, è sufficiente strutturare i prompt delineando passo dopo passo i compiti e incorporando la conoscenza, come il glossario, e le informazioni specifiche della funzione in punti stabiliti del prompt.

In sostanza, una funzionalità intelligente viene creata compilando in linguaggio naturale i punti fissi: l'obiettivo principale, quali dati e risorse utilizzare, come collegarli e come presentare il dato. Il prompt viene costruito inserendo in ordine i system message, l'obiettivo principale e seguendo la struttura della procedura generale che include il glossario, arricchito in punti prestabiliti dalle informazioni fornite dall'utente. Questo metodo ha dimostrato un'elevata efficienza, guidando sempre il modello a scrivere codice capace di analizzare i dati e a tentare approcci validi in caso di errore.

Un aspetto importante nel funzionamento di Delta AI è la schedulazione e la riesecuzione delle sue funzionalità intelligenti. Una pratica comune potrebbe essere salvare il codice della procedura e rieseguirla deterministicamente, lasciando all'LLM l'interpretazione del solo risultato della computazione. Tuttavia, questo approccio potrebbe limitare notevolmente la flessibilità del sistema, soprattutto in situazioni con errori nei dati o quando sono necessarie ulteriori analisi.

Per superare questo ostacolo, Delta AI sfrutta il prompt engineering, in particolare l'aggiunta di un system message [Isayah Culbertson, 2023]; questa soluzione consente di chiedere al modello di salvare il codice generato, commentandolo con dettagli sulla procedura eseguita. Il risultato è la creazione di uno strumento riutilizzabile, conservato in una specifica cartella. Viene inoltre introdotta la prassi di verificare se esiste già uno strumento scritto per la procedura richiesta all'interno di questa cartella, utilizzando quello esistente se disponibile. Questa metodologia ha dimostrato di funzionare efficacemente sia con GPT-4 che con Mistral 7B, garantendo così una gestione efficiente e flessibile delle funzionalità intelligenti di Delta AI.

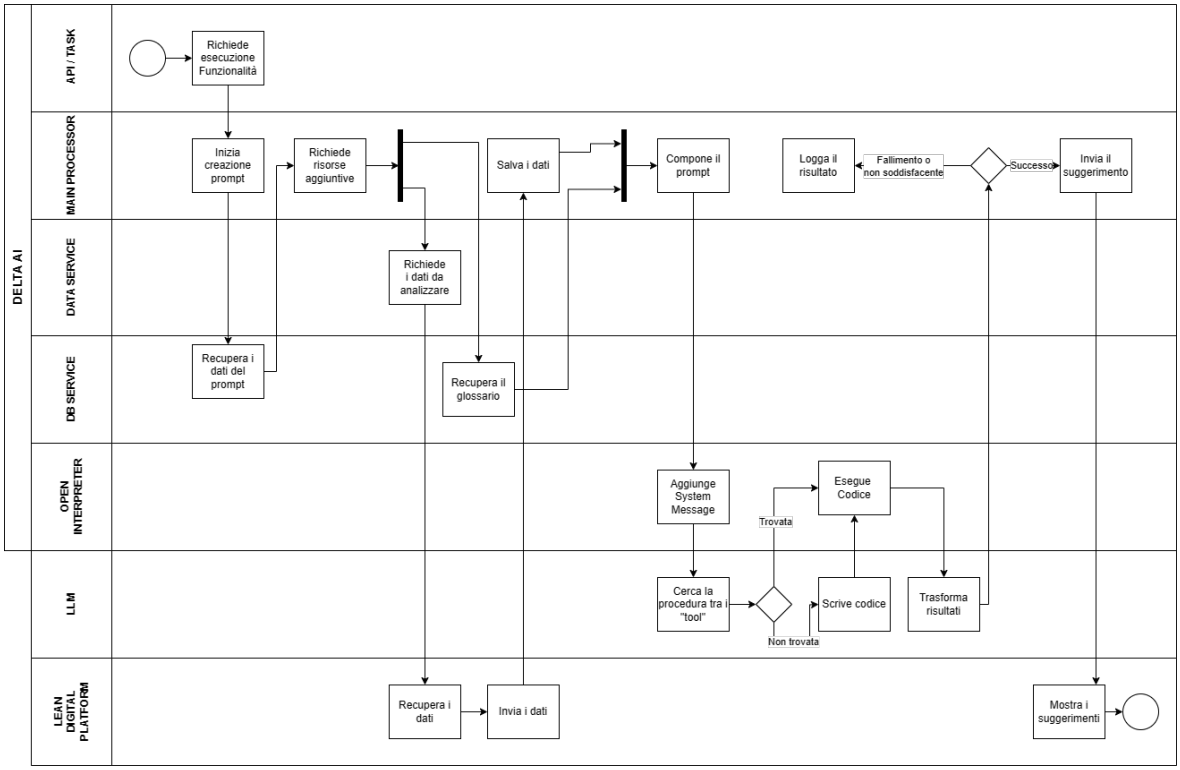

Figure 2. Funzionamento di Delta AI.

Il diagramma di attività mostra il flusso di esecuzione di una funzionalità intelligente, a partire dal trigger dell'esecuzione schedulata o una richiesta via API fino alla presentazione dell'ouput sulla piattaforma Lean. Sono inquadrati i ruoli e le azioni di ogni componente.

Un'ultima scelta fondamentale nel processo di sviluppo di Delta AI riguarda la selezione del Large Language Model da utilizzare. L'intento principale era rendere la piattaforma indipendente dal modello specifico scelto. Fortunatamente, molti LLM aderiscono a pattern di interazione standard, tanto che la libreria di OpenAI è compatibile anche con modelli open source. In Delta AI, l'interazione con il modello è gestita da Open Interpreter, un componente che vanta un'estesa compatibilità, supportando persino modelli eseguiti localmente.

Nelle fasi iniziali di test, Delta AI è stata valutata con GPT-3 e Code LLaMa 7B. Tuttavia, con l'avvento di modelli con context window più estesa, i principali contendenti sono diventati GPT-4 e Mistral 7B. Una distinzione fondamentale tra questi modelli è la possibilità di utilizzarli su server privati. Le politiche di privacy e sicurezza del cliente finale influenzano la scelta, portando talvolta all'esclusione di GPT-4 a causa della condivisione di dati al di fuori dei loro server privati.

Comparare le performance dei modelli non è semplice; nonostante Mistral sia meno potente di GPT-4, è capace di scrivere ed eseguire codice seguendo i prompt per ottenere analisi. Tuttavia, il modo in cui Mistral presenta i dati e scrive suggerimenti è qualitativamente inferiore rispetto a GPT-4. Inoltre, l'esecuzione di Mistral senza ricorrere a modelli quantizzati richiede hardware potente per raggiungere velocità comparabili a quelle di GPT-4. La rapida evoluzione dei modelli fa presagire che le limitazioni attuali dei modelli open source saranno superate in futuro.

5.3. Deploy

Delta AI, completamente integrata nel sistema CI/CD di GitLab aziendale, supporta un deploy automatico e flessibile per rispondere alle esigenze di Digibelt sia on-premise che in cloud. La configurazione tramite Docker Compose assicura un deploy rapido e gestibile per i clienti on-premise, offrendo un ambiente sicuro e facilmente controllabile all'interno dell'infrastruttura del cliente. Per i clienti che optano per Digibelt in cloud, Delta AI è deployabile su infrastruttura Kubernetes, che garantisce scalabilità, resilienza e gestione efficiente del carico in ambienti cloud.

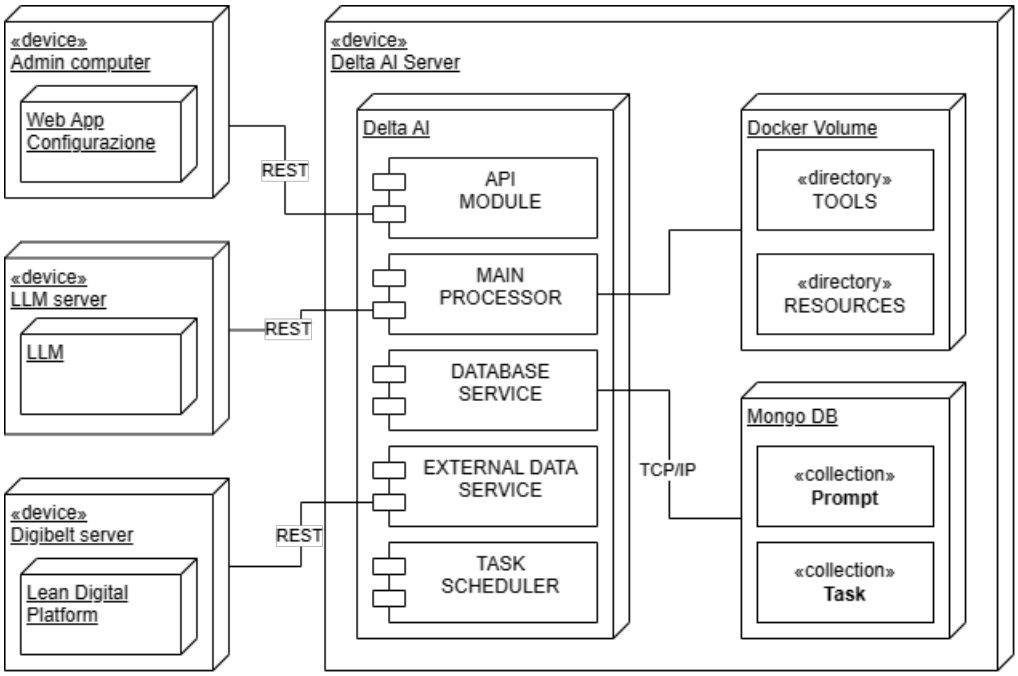

Figure 3. Delta AI, deploy on premise su ambiente DockerCompose

5.4. Moduli dell'Applicazione

5.4.1. Modulo Processore Principale

Il modulo Processore Principale è il cuore dell'applicazione Delta AI; la sua funzione principale è orchestrare il flusso di lavoro complessivo dell'applicazione, interfacciandosi con gli altri moduli per elaborare dati e richieste. Questo modulo forma i prompt basati sui dati raccolti, inviandoli all'LLM per la generazione del codice. È responsabile anche per la gestione dei dati esterni e per fornire prompt di sistema all'LLM, necessari per salvare procedure specifiche nella cartella 'tool_kit' e per chiamare una funzione che registri l'errore in caso di risultati incerti o problemi nella risposta. La sua responsabilità include la supervisione dell'esecuzione del codice generato e la gestione dei risultati, garantendo che l'analisi prodotta sia coerente e affidabile; gestisce eventuali errori e problemi grazie al sistema di logging.

5.4.2. Modulo API

Il Modulo API rappresenta il fulcro della comunicazione esterna di Delta AI. Questo modulo espone una serie di API CRUD (Create, Read, Update, Delete) per la gestione dei task e dei prompt, rendendo possibile la manipolazione di questi

elementi attraverso richieste esterne. Inoltre, il modulo API mette a disposizione un'interfaccia per l'esecuzione delle funzionalità intelligenti, consentendo agli utenti e ad altri sistemi di interagire con Delta AI in modo dinamico e flessibile. La sua importanza risiede nella capacità di facilitare un'interazione fluida e sicura con l'applicazione, permettendo l'integrazione di Delta AI con una varietà di altri sistemi e piattaforme esterne.

5.4.3. Modulo Servizio Dati

Il modulo Servizio Dati è incaricato dell'acquisizione, elaborazione e fornitura dei dati necessari per le analisi svolte dall'applicazione. Questo modulo estrae i dati dalla Lean Digital Platform e da altre fonti esterne, li trasforma in un formato utilizzabile e li fornisce al Processore Principale. Una componente chiave di questo modulo è la sua capacità di interpretare le richieste di dati e di fornire un set di dati pertinente e aggiornato. Inoltre, dispone di funzioni, esposte anche all'LLM, per presentare i risultati delle analisi sulle piattaforme di Digibelt.

5.4.4. Modulo Task Scheduler

Il modulo Task Scheduler è responsabile per la programmazione e l'esecuzione automatizzata di task all'interno dell'applicazione. Questo modulo consente agli utenti di impostare e gestire task ricorrenti, come l'analisi periodica dei dati o la generazione di report. La sua funzionalità chiave include l'attivazione e la disattivazione di task, la definizione di intervalli di esecuzione e la gestione delle priorità di esecuzione. Interagisce strettamente con il Processore Principale per garantire che i task programmati vengano eseguiti in maniera efficiente e tempestiva.

5.4.5. Modulo Database

Il Modulo Database in Delta AI è incaricato di archiviare e organizzare le funzionalità intelligenti ed i task. Questo modulo si occupa della persistenza e recupero dei dati relativi a ciascuna funzionalità, assicurando un accesso efficiente e sicuro alle informazioni necessarie per le operazioni dell'applicazione.

5.5. Interfaccia Utente

Nel creare l'interfaccia utente di Delta AI, l'obiettivo era fornire una piattaforma che consentisse agli utenti di configurare e gestire le funzionalità intelligenti con facilità. Il design è stato pensato per essere diretto e funzionale, con una chiara separazione tra le diverse aree di interazione.

5.5.1. Layout e Design

Il layout presenta un'interfaccia ordinata e sistematica, con sezioni dedicate che includono la gestione dei prompts, i test e la schedulazione dei task. Ogni sezione è progettata per guidare gli utenti attraverso i processi necessari, senza complicazioni o passaggi eccessivi.

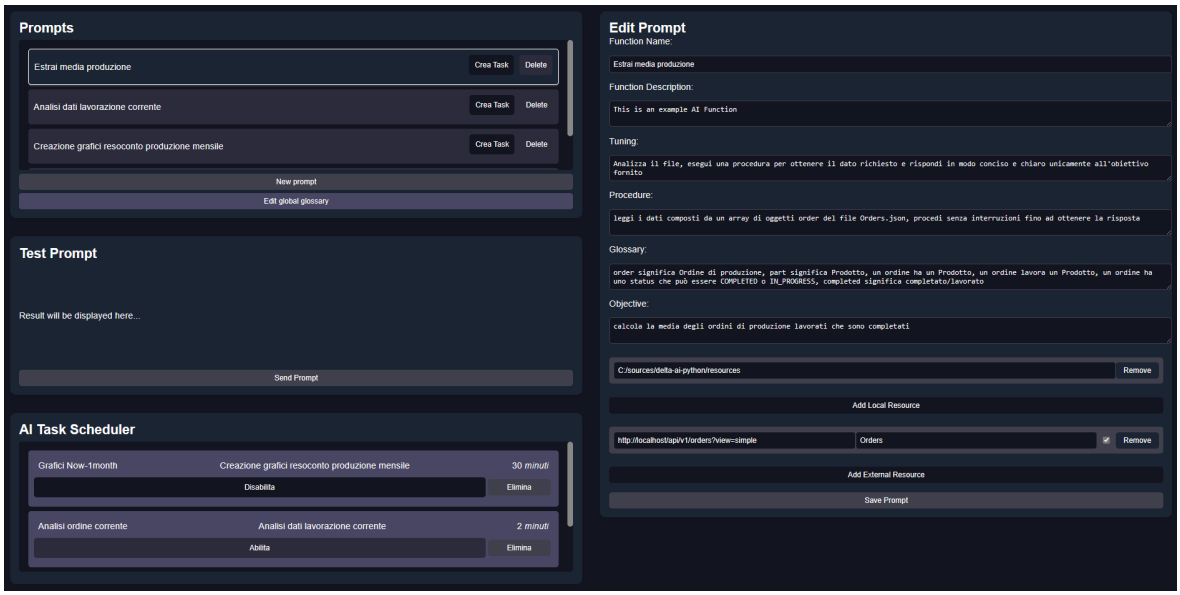

Figure 4. Il Frontend di Delta AI:

In alto a sinistra il componente di scelta e gestione delle funzionalità, insieme al bottone di modifica glossario. Sotto la card per testare i prompt, seguita da quella di scheduling dei task. A destra la visualizzazione/modifica del prompt e delle risorse che costituiscono la funzionalità.

5.5.2. Funzionalità del Front End

- **Gestione dei Prompts**: Questa sezione fornisce una lista navigabile dei prompts esistenti, con funzioni di modifica e cancellazione. Un pulsante prominente invita gli utenti a creare nuovi prompts, enfatizzando la semplicità di aggiungere nuove funzionalità.

※ **Test di Prompts**: Questa card contiene una funzione di test che consente agli utenti di eseguire i prompts e visualizzare i risultati immediatamente, offrendo un modo diretto per valutare e affinare le funzionalità create.

※ **Schedulazione dei Task**: È possibile impostare l'esecuzione automatica dei task, con controlli chiari per la definizione di frequenza e tempistica, facilitando la pianificazione delle attività senza sforzo.

5.5.3. Interazione e Usabilità

Elementi come formulari e pulsanti sono stati sviluppati seguendo principi di design consolidati per offrire un'esperienza utente coerente e riconoscibile. I feedback visivi rassicurano l'utente in ogni fase, dalle conferme di salvataggio ai segnali di completamento dei task.

5.5.4. Integrazione e Funzionalità

L'integrazione con il backend è gestita tramite API RESTful, permettendo un flusso di dati sicuro e controllato. L'interfaccia utente funge da ponte per la potente logica di Delta AI, facilitando la gestione delle funzionalità intelligenti in un ambiente strutturato.

5.6. Casi d'Uso e Scenari di Applicazione

Nell'esempio seguente, illustreremo una funzionalità intelligente implementata tramite Delta AI. È importante sottolineare che il valore principale di Delta AI non risiede unicamente nella potenza delle singole funzionalità, ma nella capacità di creare soluzioni personalizzate rapidamente e senza la necessità di scrivere o mantenere codice complesso permettendo così di assecondare le molteplici richieste dei clienti. Questo aspetto è fondamentale per capire l'impatto e l'efficacia dell'applicazione in contesti operativi reali.

Delta AI, al momento, ha come principale mezzo di presentazione dei risultati delle funzionalità intelligenti delle notifiche pop-up che vengono visualizzate sul pannello degli operatori. Queste notifiche non solo forniscono informazioni tempestive ma sono anche salvate e consultabili in una sezione dedicata, permettendo un'analisi retrospettiva e un monitoraggio continuo.

```
"Scrivi un breve rapporto strutturato che compari la
```

produzione odierna del reparto con lo storico delle ultime due settimane, evidenzia in percentuali gli scarti di velocità di produzione, durata dei fermi e quantità di pezzi da rilavorare.
Aggiungi un confronto tra il numero di Unità di Trasporto occupate in questo momento e la media settimanale.
Infine dai tre suggerimenti utili nel caso l'andamento generale sia peggiore o in caso contrario congratulati per il buon lavoro e indica il campo in cui si sta eccellendo"

Questa richiesta deve essere corredata dagli indirizzi delle API di Lean Digital Platform dalle quali recuperare i dati da analizzare. Il prompt sarà corredato da Delta AI con i system messages, il glossario e strutturato correttamente; in questo caso, nella casella di 'tuning' è stata aggiunta la richiesta di formattare il risultato finale in HTML.

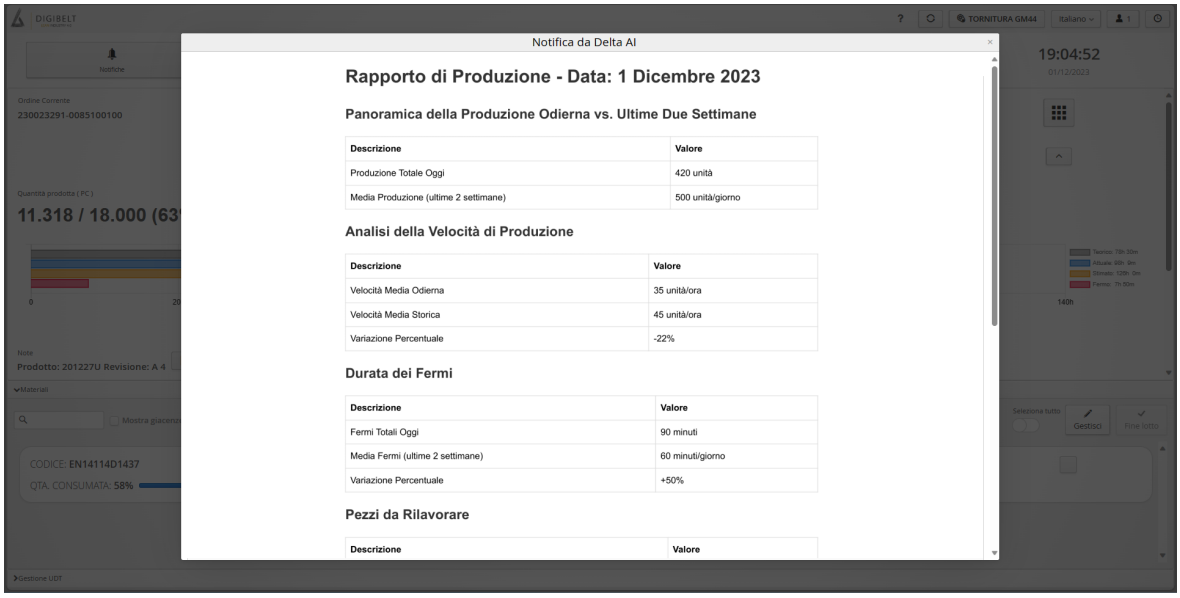

Figure 5. Risultato finale presentato su Lean Digital Platform come notifica.

Rapporto di Produzione - Data: 1 Dicembre 2023

Panoramica della Produzione Odierna vs. Ultime Due Settimane

Descrizione	Valore
Produzione Totale Oggi	420 unità
Media Produzione (ultime 2 settimane)	500 unità/giorno

Analisi della Velocità di Produzione

Descrizione	Valore
Velocità Media Odierna	35 unità/ora
Velocità Media Storica	45 unità/ora
Variazione Percentuale	-22%

Durata dei Fermi

Descrizione	Valore
Fermi Totali Oggi	90 minuti
Media Fermi (ultime 2 settimane)	60 minuti/giorno
Variazione Percentuale	+50%

Pezzi da Rilavorare

Descrizione	Valore
Quantità Oggi	50 pezzi
Media Quantità (ultime 2 settimane)	30 pezzi/giorno
Variazione Percentuale	+66.7%

Occupazione delle Unità di Trasporto

Descrizione	Valore
Unità di Trasporto Occupate Oggi	15
Media Settimanale Occupazione	10
Variazione Percentuale	+50%

Analisi e Suggerimenti

Andamento Generale: L'andamento generale mostra una diminuzione della produzione e della velocità, unita a un aumento nei fermi e nei pezzi da rilavorare. Sebbene l'aumento nell'efficienza delle Unità di Trasporto sia positivo, gli altri indicatori suggeriscono un andamento complessivamente peggiore.

Suggerimenti:

1. **Ottimizzazione della Velocità di Produzione**: Rivedere le linee di produzione per identificare colli di bottiglia o malfunzionamenti. Valutare la necessità di aggiornamenti tecnici o adeguamenti procedurali.
2. **Riduzione dei Fermi**: Analizzare le cause dei fermi prolungati e migliorare le procedure di manutenzione preventiva. Considerare la formazione aggiuntiva per il personale su come gestire e prevenire i fermi.
3. **Controllo Qualità Migliorato**: Implementare controlli qualitativi più rigorosi per ridurre il numero di pezzi da rilavorare. Analizzare i processi di produzione per individuare le cause alla base della scarsa qualità.

Nota: Questo rapporto è generato automaticamente basato su dati immaginari. Per decisioni operative, si raccomanda un'analisi dettagliata basata su dati reali.

Figure 6. Report completo.

5.7. Costi di Sviluppo e Operativi

5.7.1. Tempi di Sviluppo

Lo sviluppo di Delta AI, in particolare della sua parte applicativa, è stato tracciato attraverso la piattaforma per la gestione dello sviluppo Agile, Jira. Data la natura di questo strumento, tuttavia, la parte più considerevole dell'implementazione è stata lo studio degli LLM, durato diversi mesi, i cui risultati sono esposti in questo

testo. Il progetto è stato creato come Proof Of Concept ed ha recentemente iniziato il percorso per diventare un prodotto, per questo motivo il suoi sviluppi sono stati realizzati interamente da me e non sottoposti a review durante l'implementazione. Una volta acquisita la conoscenza necessaria e creati i primi prototipi non ufficiali, lo sviluppo di Delta AI ha richiesto i seguenti tempi divisi per attività:

Task svolto	Tempo stimato	Tempo effettivo
Riunioni interne per progettazione e presentazione	8h	8h
Test scrittura funzionalità intelligenti su ChatGPT	8h	8h
Analisi framework e strumenti per implementazione	4h	8h
Sviluppo infrastruttura backend	4h	4h
Integrazione con Open Interpreter	4h	2h
Creazione database e servizio	2h	1h
Sviluppo API e servizio di comunicazione	2h	1h
Implementazione Task	2h	2h
Creazione frontend completo	8h	4h
Test frontend integrato	1h	1h
Logica prompt e funzionalità	6h	8h
Deploy e CI/CD	4h	4h
Test Delta AI	2h	2h
Totali	**55h**	**53h**

5.7.2. Costi Operativi

Nella seguente sezione, presentiamo un'analisi dettagliata dei costi associati all'esecuzione di una funzionalità di Delta AI, con un focus particolare sulle API commerciali di OpenAI. È importante sottolineare che i prezzi indicati sono suscettibili a variazioni, specialmente con l'introduzione di nuovi modelli. In questa analisi, confrontiamo due modelli distinti: GPT-4-1106-preview, distinto per le migliori performance, e GPT-3 .5-turbo-1106, che ha prestazioni decenti ed un costo che ha avuto modo di stabilizzarsi.

Il nostro obiettivo è valutare il costo complessivo dell'esecuzione di una funzione intelligente di Delta AI, presupponendo che il modello abbia già elaborato e salvato una procedura che verrà riutilizzata (Task recall). Questo processo include sia il prompt iniziale, che è completato da messaggi di sistema e il glossario, sia i messaggi intermedi utilizzati per valutare l'output della computazione. L'output comprende la prima risposta generata dal modello e la formattazione del risultato

finale in formato HTML.

I costi presentati sono stati calcolati utilizzando Open Interpreter e successivamente confrontati con i dati di spesa effettiva forniti da OpenAI. Questa comparazione mira a fornire una stima accurata e aggiornata del costo operativo per l'impiego di tali tecnologie avanzate in contesti commerciali e di ricerca.

Modello	Costo per Input (2K token)	Costo per Output (500 token)	Costo Totale esecuzione
GPT-4-1106-preview	$0.0200	$0.0150	$0.0350
GPT-3.5-turbo-1106	$0.0020	$0.0010	$0.0030

Alla luce dell'analisi dei costi associati all'utilizzo delle API commerciali di OpenAI per il funzionamento di Delta AI, emerge una prospettiva interessante quando confrontiamo questi costi con quelli derivanti dal deploy privato di un modello open source, come Mistral 7B, su Inference Endpoint di Hugging Face.

Mistral 7B, con un costo di esecuzione di 1,30 dollari all'ora per il modello non compresso, permette di eseguire potenzialmente migliaia di prompt ogni ora. In contrasto, per spendere 1,30 dollari all'ora utilizzando le API commerciali di OpenAI, si potrebbero eseguire circa 37 prompt con il modello GPT-4-1106-preview e 433 prompt con il GPT-3.5-turbo-1106. Questo dimostra che, nonostante la capacità potenzialmente più elevata di esecuzione di prompt del modello open source, le API commerciali rimangono estremamente competitive.

Per un'applicazione come Delta AI, che si concentra su suggerimenti e report periodici o in risposta a eventi specifici in un singolo stabilimento, l'utilizzo delle API commerciali si rivela un'opzione economica e adeguata. Tuttavia, l'adozione su larga scala, come nel caso di più stabilimenti, potrebbe rendere più conveniente il deploy di un modello dedicato come Mistral 7B. Questa soluzione offrirebbe la flessibilità di espandere l'utilizzo delle funzionalità AI senza subire un incremento proporzionale dei costi, garantendo al contempo un controllo diretto e personalizzato sulle operazioni intelligenti.

5.8. Sviluppi Futuri e Potenziale di Espansione

Il potenziale di Delta AI è ancora per buona parte inesplorato, il concetto di dare ad un LLM la possibilità di interagire con una piattaforma ovvero i suoi dati e le sue funzionalità implica un'incredibile libertà di creazione di nuovi strumenti o di impieghi per effettuare controlli e verifiche, testing automatizzati ed intelligenti dell'intera piattaforma o ancora, sviluppare dispendiose integrazioni con altri

software esterni in modo rapido. Una delle prospettive più promettenti è lo sviluppo di funzionalità con integrazioni visive avanzate, che si riveleranno fondamentali per incorporare nuovi processi direttamente negli stabilimenti produttivi, eliminando i problemi legati alle informazioni su carta non digitalizzabili.

I programmi per il futuro di Delta AI prevedono di aggiungere l'accesso ad altre funzionalità direttamente al modello, come la possibilità di richiedere di chiamare direttamente le API della Lean Digital Platform, accedere ed eseguire query in sola lettura ai database interni ed utilizzare in modo autonomo un database riservato per salvare qualsiasi dato ritenga necessario. Inoltre, è in programma l'aggiunta di altri mezzi di comunicazione per Delta AI, come una integrazione via mail o Microsoft Teams per inviare alert, per consentire a Delta AI di effettuare monitoring intelligente in modo simile a strumenti come Grafana.

Con l'evoluzione continua dei modelli di apprendimento automatico, Delta AI è destinato a diventare un agente intelligente con una conoscenza approfondita di tutti i dati della Lean Digital Platform. Sarà in grado di eseguire un'ampia gamma di compiti su richiesta, utilizzando il linguaggio naturale. Questo rappresenta un salto qualitativo nella gestione dei dati e nell'automazione dei processi, trasformando Delta AI in uno strumento essenziale e versatile, capace di adattarsi e rispondere in modo efficace alle esigenze dinamiche del settore industriale e le necessità degli sviluppatori.

6. Conclusioni

La presente ricerca ha esplorato in modo approfondito le modalità di sviluppo e le conoscenze necessarie per creare strumenti avanzati basati sui Large Language Models. Attraverso un'analisi dettagliata, abbiamo evidenziato le pratiche fondamentali e le competenze tecniche che sono state cruciali nella realizzazione di strumenti innovativi di document intelligence.

Questo studio non solo sottolinea l'importanza degli LLM nel panorama tecnologico attuale, ma fornisce anche una base solida di conoscenze per gli sviluppatori che si apprestano a entrare in questo campo rivoluzionario. L'enfasi posta sullo sviluppo di strumenti di document intelligence basati su LLM pone in evidenza il potenziale di questa tecnologia nel trasformare il modo in cui interagiamo ed elaboriamo le informazioni.

Riconoscendo le sfide e le opportunità che gli LLM presentano, questo lavoro si propone come una guida e uno spunto per i futuri sviluppatori, incoraggiandoli a sfruttare al meglio questa tecnologia. Le applicazioni degli LLM, come dimostrato dalla nostra ricerca, vanno ben oltre il semplice elaborazione del linguaggio, aprendo nuove strade nel campo dell'intelligenza artificiale.

In conclusione, il nostro studio non solo getta le basi per ulteriori ricerche in questo settore in rapida evoluzione, ma offre anche una visione ispiratrice per i futuri innovatori. Spero che il mio lavoro serva da catalizzatore per lo sviluppo di nuovi e rivoluzionari strumenti basati su Large Language Models, segnando un passo significativo verso un futuro in cui la tecnologia e l'innovazione si fondono per creare soluzioni più intelligenti ed efficaci.

Bibliografia

- [Calvin N. Mooers, Charlotte Davis, Charles Babbage Institute, 1993] Calvin N. Mooers, Charlotte Davis (1993) *Oral history interview with Calvin N. Mooers and Charlotte D. Mooers*
  https://conservancy.umn.edu/handle/11299/107510
- [OpenAI, 2023] OpenAI. (2023). *GPT-4 Technical Report.*
  https://arxiv.org/abs/2303.08774
- [Prompting Guide, 2023] *GPT-4*
  https://www.promptingguide.ai/models/gpt-4
- [OpenAI Documentation,2023] *Open AI Docs*
  https://platform.openai.com/docs
- [j , 2023] j *Efficient Processing of Multiple Complex Prompts with GPT-4.*
  https://community.openai.com/t/efficient-processing-of-multiple-complex-prompts-with-gpt-4/463976/2
- [Jiang, Sablayrolles, Mensch, et al., 2023] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, William El Sayed. (2023). *Mistral 7B.*
  https://arxiv.org/abs/2310.06825
- [Touvron, Lavril, Izacard, et al., 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Guillaume Lample. (2023). *LLaMA: Open and Efficient Foundation Language Models.*
  https://research.facebook.com/publications/llama-open-and-efficient-foundation-language-models/
- [Ye et al., 2023] Ye, J., Chen, X., Xu, N., Zu, C., Shao, Z., Liu, S., Cui, Y., Zhou, Z., Gong, C., Shen, Y., Zhou, J., Chen, S., Gui, T., Zhang, Q., Huang, X. (2023). *A Comprehensive Capability Analysis of GPT-3 and GPT-3.5 Series Models.*
  https://arxiv.org/abs/2303.10420
- [Vaswani, et al., 2023] Ashish Vaswani, et al. (2023). *Attention Is All You Need.*
  https://arxiv.org/abs/1706.03762

- [Lehman, Gordon, Jain, et al., 2023] Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, Kenneth O. Stanley. (2023). *Evolution through Large Models.* https://arxiv.org/abs/2206.08896
- [Wan, Z. et al., 2023] Zhen Wan, et al. (2023). *Reformulating Domain Adaptation of Large Language Models as Adapt-Retrieve-Revise.* https://arxiv.org/abs/2310.03328
- [Perot, Boppana, Wang, et al., 2023] Vincent Perot, Ramya Sree Boppana, Zilong Wang, Jiaqi Mu, Hao Zhang, Nan Hua. (2023). *LMDX: Language Model-based Document Information Extraction And Localization.* https://arxiv.org/abs/2309.10952
- [Zhao, Zhou, Li, et al., 2023] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, Ji-Rong Wen. (2023). *A Survey of Large Language Models.* https://arxiv.org/abs/2303.18223
- [Qingyao Ai, Ting Bai, Zhao Cao, 2023]Qingyao Ai, Ting Bai, Zhao Cao, Yi Chang, Jiawei Chen, Zhumin Chen, Zhiyong Cheng, Shoubin Dong, Zhicheng Dou, Fuli Feng, Shen Gao, Jiafeng Guo, Xiangnan He, Yanyan Lan, Chenliang Li, Yiqun Liu, Ziyu Lyu, Weizhi Ma, Jun Ma, Zhaochun Ren, Pengjie Ren, Zhiqiang Wang, Mingwen Wang, Ji-Rong Wen, Le Wu, Xin Xin, Jun Xu, Dawei Yin, Peng Zhang, Fan Zhang, Weinan Zhang, Min Zhang, Xiaofei Zhu (2023) *Information Retrieval meets Large Language Models: A strategic report from Chinese IR community* https://doi.org/10.1016/j.aiopen.2023.08.001
- [Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, Yanai Elazar, 2023] Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, Yanai Elazar, (2023) *Few-shot Fine-tuning vs. In-context Learning: A Fair Comparison and Evaluation* https://arxiv.org/abs/2305.16938
- [Dao, 2023] Tri Dao. (2023). *FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning.* https://arxiv.org/abs/2307.08691
- [Gu, 2023] Gu, Y. (2023). *Knowledge Distillation of Large Language Models.* https://arxiv.org/abs/2306.08543

- [Tunstall, Beeching, et al., 2023] Lewis Tunstall, Edward Beeching, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. (2023). *Zephyr: Direct Distillation of LM Alignment.*
  https://arxiv.org/abs/2310.16944
- [Colin Jarvis, John Allard, Open AI, 2023] Colin Jarvis, John Allard, Open AI.(2023) *A Survey of Techniques for Maximizing LLM Performance*
  https://www.youtube.com/watch?v=ahnGLM-RC1Y
- [Yao et al., 2023] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., Narasimhan, K. (2023). *Tree of Thoughts: Deliberate Problem Solving with Large Language Models.*
  https://arxiv.org/abs/2305.10601
- [Yao et al., Revised in 2023] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y. (v3 2023). *Synergizing Reasoning and Acting in Language Models.*
  https://arxiv.org/abs/2210.03629v3
- [Long, 2023] Long, J. (2023). *Large Language Model Guided Tree-of-Thought.*
  https://arxiv.org/abs/2305.08291
- [Lewis et al., 2020] Lewis, P., et al. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.*
  https://arxiv.org/abs/2005.11401
- [Lewis, Revised in 2023] Lewis, P. (2023). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.*
  https://arxiv.org/abs/2005.11401v4
- [Paranjape et al., 2023] Paranjape, B., Lundberg, S., Singh, S., Hajishirzi, H., Zettlemoyer, L., Ribeiro, M. T. (2023). *ART: Automatic multi-step reasoning and tool-use for large language models.*
  https://arxiv.org/abs/2303.09014
- [Ali, 2023] Ali, M. (2023). *How to Build LLM Applications with LangChain.*
  https://www.datacamp.com/tutorial/how-to-build-llm-applications-with-langchain
- [Open Interpreter, 2023] *Open Interpreter Documentation.*
  https://docs.openinterpreter.com/introduction
- [Accenture, 2023] *7 Architecture Considerations for Generative AI.*
  https://www.accenture.com/us-en/blogs/cloud-computing/7-generative-ai-architecture-considerations
- [Hugging Face, 2023] H4 Team and community. *Open LLM Leaderboard.*
  https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

- [SkyPilot Team, 2023] SkyPilot Team *SkyPilot Docs* https://skypilot.readthedocs.io/en/latest/
- [LangChain, 2023] LangChain *LangChain Docs* https://python.langchain.com/docs/get_started/introduction
- [Adithya S K, 2023] Adithya S K *Deploy Mistral/Llama 7b on AWS in 10 mins* https://adithyask.medium.com/deploy-mistral-llama-7b-on-aws-in-10-mins-cc80e88d13f2
- [Hugging Face , 2023] *Pricing* https://huggingface.co/pricing
- [Liu et al., 2023] Liu, N. F., Bevilacqua, M., Petroni, F., Liang, P. (2023). *Lost in the Middle: How Language Models Use Long Contexts.* https://arxiv.org/abs/2307.03172v3
- [Wiggers, 2023] Wiggers, K. (2023). *OpenAI's massive GPT-3 model is impressive, but size isn't everything.* https://venturebeat.com/ai/ai-machine-learning-openai-gpt-3-size
- [Snowflake, 2023] *The Role of Large Language Models in Machine Learning.* https://www.snowflake.com/guides/large-language-models-llms-machine-learning
- [Open AI, 2023] Open AI *New Models and Developer Products* https://openai.com/blog/new-models-and-developer-products-announced-at-devday
- [Isayah Culbertson, 2023] Isayah Culbertson *Open Interpreter Hackathon* https://github.com/isayahc/open-interpreter-hackathon/tree/main
- [Berri AI Team , 2023] Berri AI Team *Lite LLM* https://litellm.ai/