UNIVERSITÀ DEGLI STUDI DI UDINE
HIC SUNT FUTURA

DIPARTIMENTO DI SCIENZE MATEMATICHE, INFORMATICHE E FISICHE

TESI DI LAUREA IN
INTERNET OF THINGS, BIG DATA E MACHINE LEARNING

SqlTalk: Sviluppo di un'Applicazione Multitenancy di Analisi Dati Utilizzando l'IA Generativa

CANDIDATO
Mancardi Devin

RELATORE
Prof. Gian Luca Foresti

TUTOR AZIENDALE
Mauro Carniel

Anno accademico 2023-2024

Contatti dell'Istituto
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
https://www.dmif.uniud.it/

A chi c'è sempre stato.
A chi se nè andato.
A chi ha creduto in me fin dall'inizio.
E fino alla fine.
A te che più di chiunque altro avresti voluto esserci.

A mio nonno Vittorio,
spero di averti reso orgoglioso di me

Ringraziamenti

Desidero esprimere la mia profonda gratitudine a tutte le persone che mi hanno accompagnato e sostenuto in questi anni di studio, rendendo possibile il raggiungimento di questo importante traguardo. In primis vorrei ringraziare il mio tutor universitario, Gian Luca Foresti, per avermi seguito con dedizione e professionalità durante tutto il mio processo di stesura della tesi, e il mio tutor aziendale, Mauro Carniel.

Un ringraziamento speciale va alla mia famiglia: a mia madre Juli, mio padre Diego, mia nonna Marisa e mio zio Ervin. Papà, grazie per avermi insegnato il valore del sacrificio; mamma, grazie per avermi trasmesso la perseveranza. Mi avete spronato a dare sempre il massimo, credendo in me spesso più di quanto io stesso facessi. Grazie soprattutto per avermi insegnato ad essere la persona che sono oggi.

Grazie Zio per avermi trasmesso la passione a questo mondo affascinante dell'informatica e per avermi insegnato, insieme al papà, che l'impegno paga sempre.

Vorrei inoltre ringraziare il mio migliore amico da una vita, Erik, per avermi ascoltato nei miei infiniti sfoghi e per i tanti momenti di leggerezza condivisi, e Luca, per i divertenti momenti trascorsi all'università a ideare le strategie più improbabili per superare gli esami.

Infine, il mio più sentito ringraziamento va alla mia fidanzata Paola. Senza di te, tutto questo non sarebbe stato possibile. Grazie per esserci sempre stata, nei momenti difficili come in quelli gioiosi, e per aver reso questo percorso molto più bello. Questa laurea è tanto mia quanto tua.

Sommario

L'evoluzione delle tecnologie di Natural Language Processing (NLP) e dei modelli di intelligenza artificiale generativa ha aperto nuove frontiere nell'interazione con i dati aziendali. Questa tesi affronta questa nuova era dell'analisi dei dati grazie al progetto SqlTalk (nel quale si è andati anche ad approfondire la nuova tecnologia della RAG) sviluppato durante il mio tirocinio aziendale grazie al LLM Google Gemini. SqlTalk mira a sfruttare le potenzialità dell'Ai Generativa per semplificare l'interrogazione dei database in real-time da parte dei clienti aziendali, al fine di dargli un prodotto che possa soddisfare tutte le attività quotidiane di data-analysis.

Grazie all'integrazione con modelli avanzati di LLM come Google Gemini, l'applicativo consente agli utenti di ottenere insights e scoprire nuovi trend utilizzando semplicemente il linguaggio naturale, senza la necessità di conoscere il linguaggio SQL, andando ad analizzare database in modo tempestivo e veloce.

Questo strumento non solo democratizza l'accesso alle informazioni nei database, ma contribuisce anche a migliorare le strategie di vendita, permettendo una comprensione più immediata e approfondita dei dati.

In questa tesi, affrontando un problema in ambito aziendale, si è approfondito il anche il confronto tra i costi di utilizzo dei modelli generativi, prendendo in esame GPT e GEMINI, analizzando anche le performance in termini di velocità di risposta.

Indice

1

Introduzione

Negli ultimi anni, l'introduzione, diffusione ed evoluzione delle tecnologie di Intelligenza Artificiale (o semplicemente AI) e del Machine Learning (ML) ha rivoluzionato completamente il mondo del lavoro, andando a cambiare radicalmente come interagire con i dati e come affrontare problemi complessi diminuendone drasticamente il tempo di risoluzione.

Un area di particolare interesse per le aziende odierne è l'Ai Generativa, che rappresenta un potente strumento per sviluppare soluzione innovative aprendo nuove prospettive nell'automazione e nell'analisi dei dati.

1.1 AI, ML e Deep learning

Spesso i termini IA, Machine Learning e Deep Learning vengono utilizzati come sinonimi, ma in realtà svolgono funzioni diverse tra di loro. l'IA è un termine usato per indicare in modo generico la creazione di sistemi intelligenti che sono in grado di simulare il pensiero umano. In generale, quindi ci riferiamo al (vastissimo) campo dell'IA come tutti i software che emulano in qualche modo una capacità umana. Il Machine Learning è una sottocategoria dell'IA che si concentra sulla capacità dei sistemi di apprendere dai dati e migliorare le loro prestazioni nel tempo senza essere esplicitamente programmati per ogni singolo compito. Il Deep Learning a sua volta, è una sottocategoria del machine learning, che utilizza reti neurali artificiali per analizzare grandi moli di dati complessi e comprendere in profondità i pattern presenti in quest'ultimi.

Il Deep Learning è alla base di tutte le rivoluzioni IA a cui stiamo assistendo. I modelli statistici addestrati su dataset per sviluppare capacità di classificazione o previsione imparano esclusivamente dagli esempi forniti, senza essere esplicitamente programmati per ogni singolo compito. I casi d'uso del Deep Learning sono molteplici e spesso si classificano con il tipo di dato trattato dalla rete.

La creazione del progetto SqlTalk non sarebbe mai stata possibile senza l'ausilio di uno specifico caso d'uso di deep learning ovvero il NLP (Natural Language Processing).

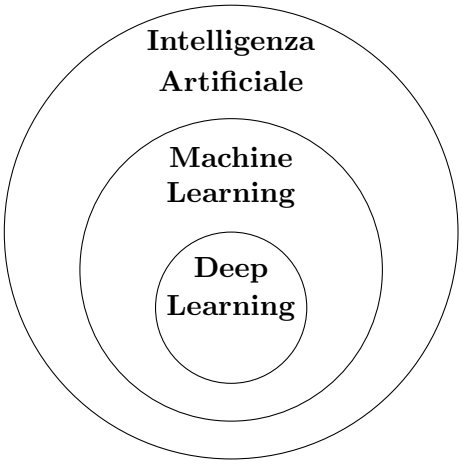


Figura 1.1: Diagramma di Venn dei componenti dell'intelligenza artificiale

1.1.1 Qual è la differenza tra l'IA «storica» e l'IA «moderna»

Le «IA Tradizionali» si basavano su metodi deterministici, dove a un dato input corrispondeva sempre lo stesso output. Questi sistemi si fondavano su regole statiche del tipo "se → allora".

Le «IA Contemporanee», invece, impiegano modelli statistici, che possono produrre output differenti a partire dallo stesso input. Questi modelli rappresentano la tecnologia più diffusa oggi nel settore industriale. I sistemi di apprendimento automatico non "consultano un database" per prendere decisioni, ma fanno affidamento sulla conoscenza incorporata nei parametri del modello statistico, acquisita durante il processo di apprendimento dai dati.

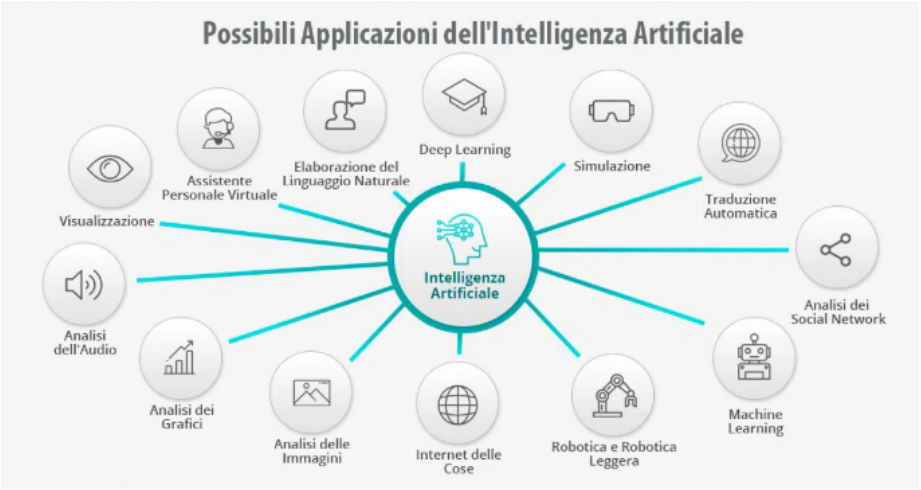


Figura 1.2: Applicazioni IA nel mondo del lavoro [3]

1.1.2 L'IA per trasformare i dati in informazioni utili

Le applicazioni dell'Intelligenza Artificiale offrono notevoli vantaggi, soprattutto nella gestione e nell'analisi dei dati [3]. Grazie a software capaci di trasformare i dati grezzi in informazioni utili, i leader e i responsabili di area possono prendere decisioni in modo più rapido e con maggiore efficacia. Oggi, non solo le grandi aziende necessitano di supporto in questo ambito, ma anche le piccole e medie imprese si

trovano sempre più spesso in difficoltà durante le fasi strategiche. Questo accade perché i dati provenienti dai diversi reparti richiedono tempo e risorse per essere rielaborati. È proprio qui che l'Intelligenza Artificiale interviene, aiutando a sintetizzare, elaborare e aggregare automaticamente grandi quantità di dati. In questo modo, diventa più facile interpretare le informazioni e sviluppare strategie previsionali.

Inoltre, l'avvento dell'IA Generativa e del Natural Language Processing (NLP) ha messo a disposizione delle aziende strumenti in grado di interrogare enormi database semplicemente utilizzando il linguaggio naturale. Questo rende tali prodotti accessibili a un pubblico sempre più vasto.

1.2 Scopo del progetto

Il Progetto SqlTalk nasce con l'obiettivo di innovare e migliorare l'interazione tra gli utenti e i database aziendali contenenti le informazioni di quest'ultimi, sfruttando le avanzate tecnologie di intelligenza artificiale (IA) generativa, Natural language processing e i LLM (Large Language Models). L'obiettivo principale del progetto è rendere l'accesso ai dati aziendali più intuitivo e veloce, eliminando la necessità di conoscere linguaggi di interrogazione complessi come SQL. Il progetto rappresenta anche un evoluzione rispetto al sistema precedente che consentiva l'interrogazione del database solo grazie ad un istantanea creata nel momento in cui l'utente andava ad interagire con il software in questione. SqlTalk supera queste limitazioni andando ad interrogare il database in Real-Time, offrendo agli utenti risposte immediate e aggiornate, direttamente correlate alle loro necessità. Questa tesi fornisce una panoramica sullo sviluppo dell'applicativo e sulle tecniche utilizzate, esplorando l'integrazione delle tecnologie di intelligenza artificiale generativa, NLP e Large Language Models (LLM) per migliorare l'interrogazione dei database in tempo reale.

1.2.1 Contesto Aziendale

Questa sottosezione introduce Sinesy SRL, la software house che ha reso possibile la stesura di questa tesi ospitandomi per il mio percorso di tirocinio formtativo in un ambiente aziendale altamente professionale e congruo al mio percorso di studi triennale. Quest'esperienza non solo ha contribuito alla realizzazione del mio progetto di tesi, ma ha anche arricchito notevolmente le mie compotenze nell'ambito dell'analisi dei dati e dell'intelligenza artificiale. Ritengo di aver avuto la fortuna di essere guidato e accompagnato durante il mio percorso da figure altamente prfessionali, competenti e disponbili per ogni chiarimento. In questa fase della mia crescita, caratterizzata da un grande desiderio e motivazione di apprendere, ma anche da una forte mancanza di conoscenza ed esperienze pratiche, il supporto ricevuto è stato fondamentale. In particolare vorrei esprimere la mia gratitudine al mio tutor Mauro Carniel, grazie al quale ho potuto acquisire diversi concetti fondametali e apprendere la filosfia del ≪Continuos Learning≫, la quale mi accompagnerà per tutto il proseguo del mio percorso professionale.

Sinesy SRL

Sinesy SRL è una software house italiana specializzata nello sviluppo di soluzioni tecnologiche per il mondo Retail, Manufacturing, Sanità. Fondata con l'obiettivo di supportare i negozi e le aziende nel processo di digitalizzazione, Sinesy si distingue per l'offerta di software gestionali avanzati, tra cui piattaforme per la gestione e l'ottimizzazione delle vendite omnichannel.

L'azienda ha sviluppato strumenti che consentono ai retailer di unificare i diversi canali di vendita, sia online che fisici, in un'unica piattaforma, facilitando la gestione dell'inventario, degli ordini e delle vendite in modo efficiente. Sinesy si focalizza anche sull'eCommerce e sulle soluzioni integrate, offrendo strumenti come sistemi di cassa digitali e soluzioni per la gestione di magazzino.

Con una forte attenzione all'innovazione, Sinesy integra tecnologie moderne come l'intelligenza artificiale e i big data, permettendo ai clienti di sfruttare analisi avanzate per ottimizzare le operazioni commerciali.

1.3 Struttura della tesi

La tesi è suddivisa come segue:

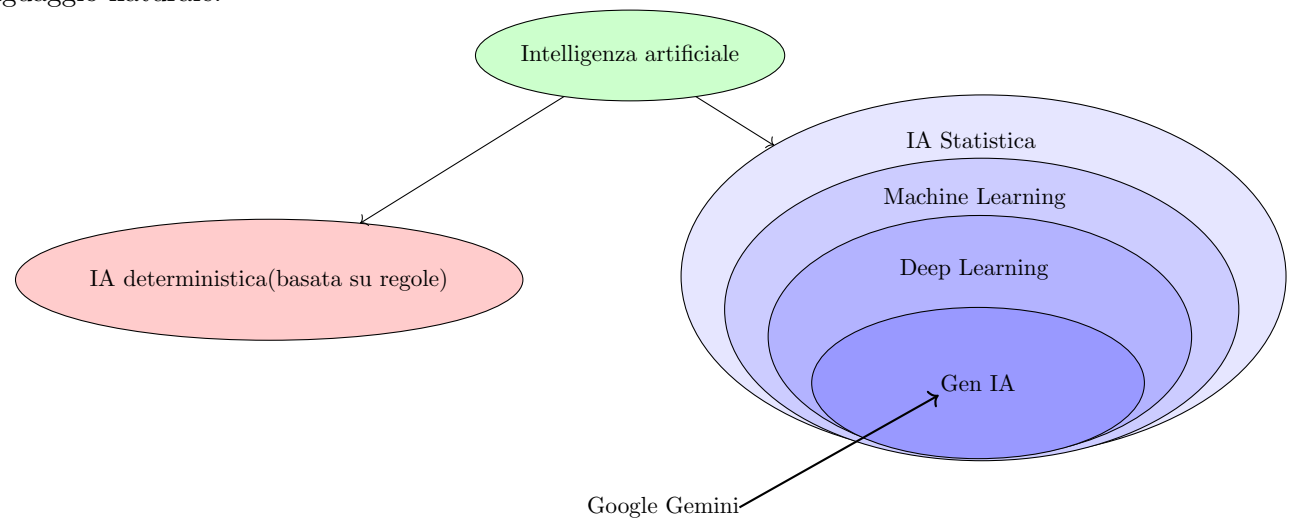
- Il **capitolo 2** si concentrerà nel fornire al lettore le nozioni teoriche necessarie per comprendere tutti gli aspetti che riguardano l'IA generativa: elencandone il funzionamento e descrivendo in maniera semplice ogni caratteristica.
- Il **capitolo 3** darà una panoramica sul progetto e lo introdurrà descrivendone le caratteristiche e le tecniche utilizzate.
- Il **capitolo 4** sarà dedicato al confronto fra i due LLM in termini di costo e velocità di risposta.
- Il **capitolo 5**, quello finale, fornirà un giudizio relativo al progetto e saranno presentati anche sviluppi e lavori futuri che potranno essere implementati utilizzando le tecniche citate.

2

Nozioni Tecniche sull'IA Generativa

2.1 Cosa è l'AI Generativa

L'intelligenza artificiale generativa (IA Generativa) rappresenta un ramo avanzato dell'intelligenza artificiale. [14] Quest'ultimo si distingue per la sua capacità di creare da zero contenuti nuovi e originali. Ciò che viene generato dal'Intelligenza Artificiale generativa consiste, in altre parole, in una ricombinazione dei dati utilizzati per addestrare l'algoritmo. Il risultato che otteniamo può sembrare un prodotto creativo ma è necessario tenere presente che questa percezione è dovuta principalmente a due fattori: da un lato, la quantità di dati utilizzata per addestrare gli algoritmi è enorme, molto oltre la capacità di elaborazione della mente umana e, dall'altro, i risultati prodotti contengono degli elementi casuali. Ne consegue che la varietà di risposte è infinita e ciò le fa sembrare ancor più originali. Questa tipologia di intelligenza artificiale sfrutta modelli di apprendimento profondo (Deep Learning) per identificare e analizzare i pattern nei set di dat esistenti. Il ruolo del Deep learning ha reso i modelli in intelligenza artificiale più sofisticati e potenti, permettendo loro di manipolare input sempre più complessi come il linguaggio naturale.



Sostanzialmente con il termine IA Generativa si intende un particolare sotto insieme dei modelli di Deep Learning (Per esempio i GPT), che si concentrano sul generare nuovi dati.

Caratteristica	IA	ML Tradizionale	IA Generativa
Scopo	Sviluppare sistemi informatici che possono eseguire compiti che tipicamente richiedono intelligenza umana.	Fare previsioni o prendere decisioni basate sui dati disponibili.	Generare nuovi campioni di dati che somigliano a un determinato set di dati di addestramento.
Interazione con i dati	I modelli utilizzano varie tecniche e strategie progettate per imitare l'intelligenza umana in una vasta gamma di applicazioni.	I modelli apprendono dai dati per fare previsioni o prendere decisioni su nuovi dati non visti.	I modelli producono nuovi dati che non facevano parte del set di dati originale ma condividono caratteristiche simili.

Tabella 2.1: tabella di confronto IA Vs ML tradizionale Vs IA generativa

2.2 Cosa è una Rete Neurale Artificiale?

Quando si tratta di compiti diversi dall'elaborazione di numeri, il cervello umano possiede molti vantaggi rispetto a un computer digitale; questo perchè siamo in grado di riconoscere rapidamente un volto, anche se visto di lato con una cattiva illuminazione in una stanza piena di altri oggetti, possiamo capire facilmente il parlato, anche quello di una persona sconosciuta in una stanza rumorosa. Nonostante anni di ricerche mirate, i computer sono lontani dal raggiungere questo livello. La caratteristica principale del cervello umano infatti, è quella di poter imparare: non ha bisogno di un programma definito o di aggiornamenti software [8].

Le reti neurali articiali (artificial neural networks, ANN) sono i modeli di apprendimento automatico più potenti a cui disponiamo oggi. Quest'ultime sono fortemente ispirate alla struttura e al funzionamento del cervello umano. Sono state progettate per ricononscere schemi e relazioni all'interno di dati complessi, utilizzando una struttura a strati di nodi interoconnessi,detti neuroni. Le reti neurali rappresentano uno degli strumenti fondamentali nel campo dell'intelligenza artificiale e sono alla base di molte applicazioni moderne, inclusi i Large Language Models (**LLM**) che verranno discussi nella prossima sezione.

2.2.1 Il Neurone

Si suppone che il neurone artificiale imiti l'azione di un neurone biologico accettarndo molti segnali diversi, x_i , da molti neuroni vicini e gli elabora in un modo semplice e predefinito. A seconda del risultato di questa elaborazione, il neurone j decide di emettere un segnale di uscita y_j o no. [16]

Come si può evincere dalla figura, alle variabili di input x_i, viene associato un peso w_i dopodiche vengono sommate tra di loro con l'aggiunta di un bias (in questo caso b) utile a indicare il punto ottimale di lavoro del neurone. Al termine di tutto ciò, il risultato viene usato da una funzione di attivazione (indicato in figura con b) per determinare l'output del neurone.

I pesi che vengono assegnati ad un collegamento determinano la forza di quest'ultimo. Inoltre questi pesi posso assumere un valore positivo o negativo.

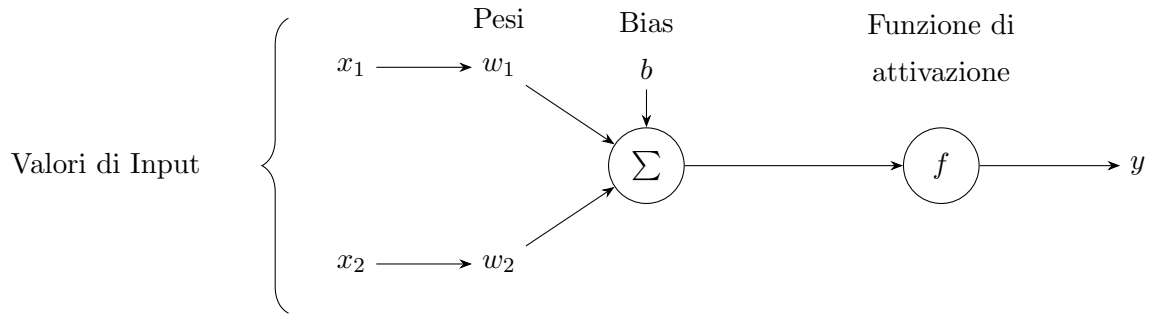


Figura 2.1: Struttura generale di un Neurone

2.2.2 Struttura

Come detto precedentemente le reti neurali artificiali sono composti da nodi, detti «neuroni». Quest'ultimi sono organizzati in strati, chiamati «layer», ognuno dei quali può avere un compito diverso. Sono suddivisi in 3 categorie:

- **Input Layer**(Strato di ingresso): Riceve i dati grezzi. Ogni nodo in questo strato rappresenta una caratteristica del dato in ingresso.
- **Hidden Layers**(Strati Nascosti): I nodi di questi strati elaborano i dati applicando delle funzioni matematiche, tipicamente non lineari, per catturare pattern e relazioni complesse nei dati.
- **Output Layer**(Strato di uscita): Fornisce l'output finale, che può rappresentare una classificazione, una previsione, o una decisione.

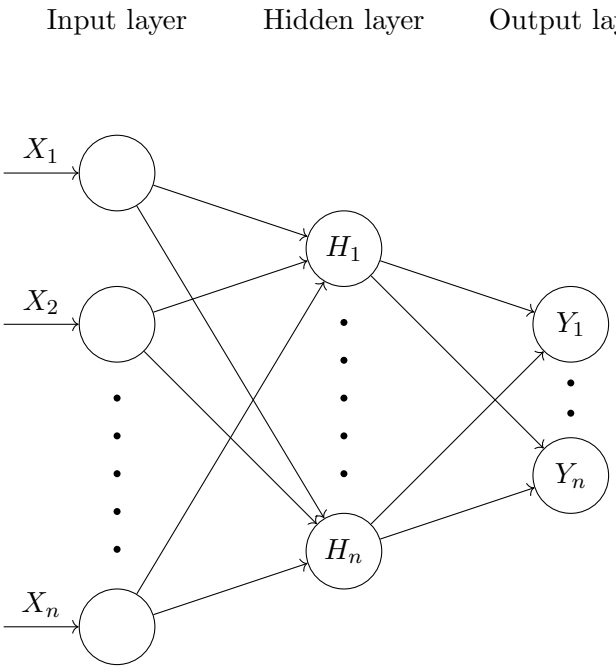


Figura 2.1: Struttura di una Rete Neurale

2.2.3 Come funziona una rete neurale

Per far si che una rete neurale funzioni, bisogna che quest'ultima venga addestrata con un set di dati. Durante la fase di training la rete neurale artificiale cerca di far coincidere il proprio valore di output calcolato con quello «corretto». Questo avviene tramite un apprendimento di tipo supervisionato (come nel caso dei LLM) che sta a indicare che per ogni valore di input ci sia il proprio valore di output. Viene presentato un esempio preso dal set di dati a disposizione e dopo aver calcolato l'errore tra l'output prodotto e quello desiderato, vengono aggiornati i pesi della rete in modo tale da minimizzare l' errore. Questo procedimento viene ripetuto per un numero di epoche finchè non si ottiene il risultato desiderato. La rete in questo modo apprende dagli errori e dagli esempi costruendo una mappatura input-output per risolvere il problema in maniera efficace.[5]
Il funzionamento di una rete neurale si basa su due fasi principali che avvengono durante la sua fase di training:

Feedforward

Feedforward è il processo in cui i dati passano attraverso la rete, dallo strato di ingresso fino allo strato di uscita. Questo vuol dire che l'output di uno strato è utilizzato come input del successivo.

Ogni neurone applica una funzione di attivazione;cosa molto importante perchè introduce non linearità al modello, ai dati che riceve e trasmette l'output ai neuroni del livello successivo
Le tre principali funzioni di attivazione sono:

- **ReLU**[11]: La funzione di attivazione ReLU (Rectified Linear Unit) è quella più usata e di maggior successo per i problemi di classifificazione offrendo prestazioni e generalizzazioni migliori nell'apprendimento rispetto alla funzione Sigmoide. Il vantaggio dell'utilizzo di questa funzione è che tutti i neuroni non vengono attivati contemporaneamente. Questo implica che un neurone verrà disattivato solo quando l'output della trasformazione lineare è zero. La funzione ReLU è definita come: $ReLU(x) = max(0, x)$

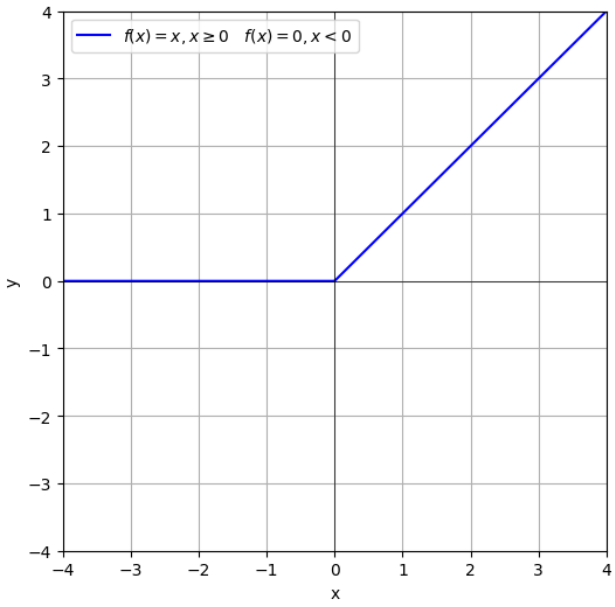


Figura 2.2: Grafico della funzione di attivazione ReLU

- **Sigmoid**[11]: La funzione sigmoide è stata una delle prime funzioni di attivazione utilizzate nelle reti neurali, soprattutto nei livelli di output per problemi di classificazione binaria. La funzione sigmoide è definita come: $\sigma(x) = \frac{1}{1+e^{-x}}$

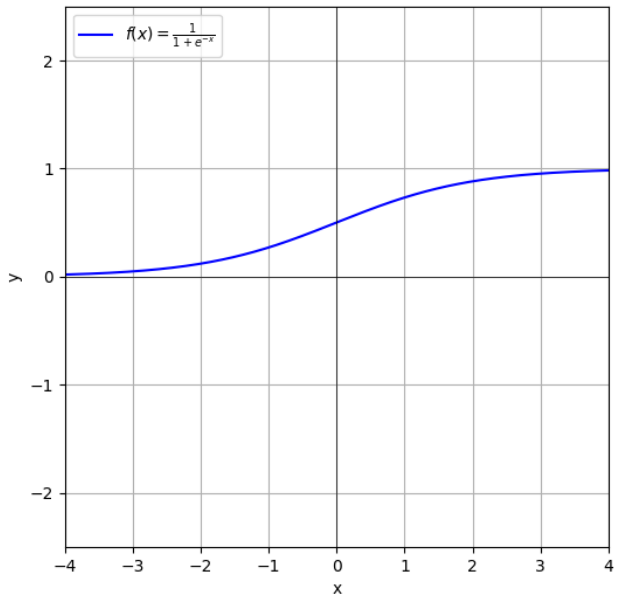


Figura 2.3: Grafico della funzione di attivazione Sigmoide

- **SoftMax**[11]:La funzione Softmax è tipicamente utilizzata nel livello di output delle reti neurali quando si ha a che fare con problemi di classificazione multiclasse. la funzione Softmax prende un vettore di valori e li trasforma in un vettore di probabilità, con i valori che sommati tra loro danno 1. È definita come: $\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$. Dove Z_i è l'input associato alla classe i e K è il numero totale di classi.

  La Softmax esalta le differenze tra i valori di input: i valori più grandi diventano più dominanti (più vicini a 1) e quelli più piccoli diventano molto vicini a 0.

BackPropagation

Il BackPropagation è un processo di apprendimento che consiste nell'aggiornamento dei pesi dei collegamenti tra i neuroni per ridurre l'errore(differenza tra l'output previsto dalla rete e il risultato atteso).

L'algoritmo più comunemente usato all'interno della BackPropogation è il Gradient Descent, il quale minimizza la funzione di perdita della rete.

2.2.4 Overfitting e Underfitting

Entrambi i fenomeni del Overfitting e Underfitting rappresentano dei problemi legati al bilanciamento tra il modello che si adatta troppo bene ai dati e la sua capacità di generalizzare a nuovi dati non visti.

- **Overfitting**: si verifica quando il modello si adatta troppo bene ai dati di addestramento. Questo comporta al fatto che il modello abbia delle performance eccellenti sui dati di addestramento, ma

non riesce a generalizzare sui dati di test nuovi. Questo perchè il modello è troppo complesso che si adatta eccesivamente ai dati di addestramento.

- **Underfitting**: Il problema inverso dell'underfitting ,si ha invece, quando il modello non è in grado di catturare la variabilità dei dati, essendo troppo semplice per i dati. Questo porta a una scarsa performance sia sui dati di addetsramento che su quelli di test

2.3 LLM

L'IA Generativa si basa su quelli che sono chiamati foundation model, che non sono altro che modelli di IA addestrati su enormi quantità di dati. Questi modelli sono in grado di comprendere la distribuzione dei dati di addestramento, studiarne la probabilità di sitribuzione e generare nuovi contenuti grazie che replicano le caratteristiche più probabili dei dati di addestramento.

I foundation model dell'IA Generativa prendono il nome di Large Language Models (LLM)[15], fanno parte di loro gli stessi GPT4 e Gemini. Quest'ultimi sono **reti neurali** addestrate su vasti corpus di testo per apprendere le regole e le strutture del linguaggio naturale. Grazie a miliardi di parametri, i LLM sono in grado di prevedere le parole successive in una frase, generare testo coerente e comprendere il contesto delle conversazioni.

2.3.1 Architetture dei LLM

L'IA Generativa può essere applicata a un ampia gamma di attività, questo implica che per ognua di esse richiede una diversa architterua di rete deep-learning per acquisire i pattern e le caratteristiche specifiche dei dati di addestramento. Proprio per questo si sono sviluppate diverse architteture di IA generativa.

- **Recurrent Neural Networks (RNN)**: È un modello di deep learning progettato per elaborare e trasformare una sequenza di dati in un'altra sequenza specifica. Questi modelli operano in modo sequenziale, grazie ai nodi che sono collegati lungo una sequenza temporale. Ogni nodo è dotato di una connessione ricorrente, che consente di mantenere una forma di memoria delle informazioni precedenti, influenzando così gli output futuri. Questo meccanismo di ricorrenza incorpora intrinsecamente il concetto di memoria. L'output di un neurone può influenzare se stesso in un momento successivo, oppure può influenzare altri neuroni all'interno della catena temporale, i quali a loro volta influenzeranno il comportamento del neurone, completando così un ciclo. Questo rappresenta una differenza fondamentale rispetto alle reti neurali feedforward, dove gli input e gli output sono trattati come indipendenti tra loro. Nella figura 2.4 è spiegata l'architteura di un neurone ricorrente a sinistra con un loop che indica la ricorrenza temporale, invece in quella di destra mostra la sequenza di neuroni ricorrenti srotolati nel tempo, evidenziando come la stessa unità ricorrente venga utilizzata per elaborare diverse parti di una sequenza nel tempo.

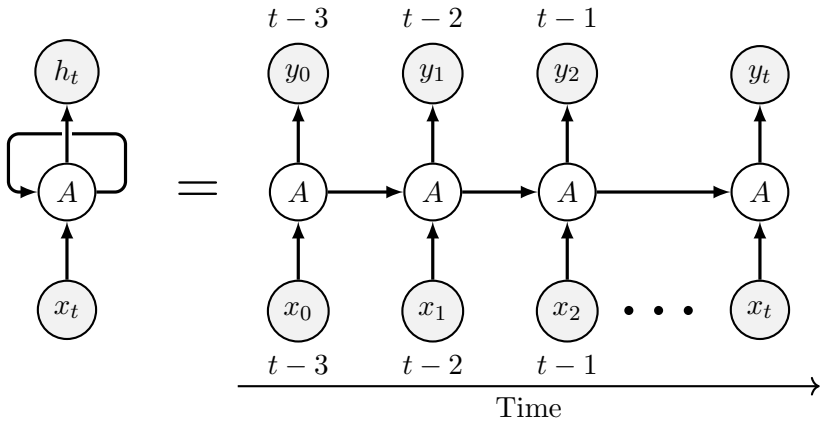


Figura 2.4: Recurrent Neuron (sinistra), unrolled through time (destra) [2]

- **Generative Adversarial Networks(GANs)**: le GANs sono una delle architteture più influenti nel campo dell'AI Generativa, svilluppate da Ian Goodfellow e i suoi colleghi nel 2014 [6].le GANs sono diventate uno strumento potente per la generazione di dati realistici, come immagini, suoni e testi. Le GAN sono delle architetture algoritmiche che sfruttano due reti neurali contrapponendole l'una all'altra con lo scopo di creare da zero nuovi dati potenzialmente simili alla realtà.
  Queste due reti neurali sono chiamate:

  – **Generatore**: Questa rete neurale ha il compito di creare nuovi dati a partire da input casuali(Rumore). Il suo obiettivo è di creare dati, i più simili ai reali, in modo tale da ingannare il discriminatore.

  – **Discriminatore**: è un classificatore che cerca di distinguere tra i dati reali, provenienti dal dataset di addestramento e i dati generati dal generatore. Sostanzialmente il suo obiettivo è classificare corretamente i dati come «reali» o «falsi», estraendone le caratteristiche.

  L'addestramento delle GANs è un processo iterativo, nel quale, sia il generatore che il discriminatore si migliorano reciprocamente. Al discriminatore vengono forniti un insieme di esempi reali dal dataset e un insieme di dati falsi dal generatore. dopodichè il discriminatore calcolerà la sua perdità in base alla sua capacità di distinguere correttamente tra esempi falsi e reali, dopodichè vengono aggiornati i pesi del discriminatore per minimizzare la perdita, migliorando la classificazione dei dati.
  D'altra parte se il discriminatore continua a migliorarsi anche il generatore deve continuare a migliorare per produrre dei dati che ingannino il discriminatore, di conseguenza la sua perdità è basata su quanto «ingannevoli» siano i suoi output, e anche in questo caso, i pesi verrano aggiornati minimizzando la perdita e migliorando la qualità dei dati generati.
  Questa tipologia di architettura presenta diverse sfide di implementazione. Ad esempio, se il discriminatore comincia a discriminare le immagini vere da quelle false troppo velocemente, il generatore potrebbe non recuperare mai più questo divario. Se invece il discriminatore è troppo lento a imparare, il generatore continuerà ad essere ricompensato per la generazione di immagini di scarsa qualità. Un altro problema che presenta questa tipologia di architettura è quella del Mode

Collapse. Questo avviene quando il generatore converge verso una distribuzione in cui genera solo un sottoinsieme di possibili output, ignorando altre modalità della distribuzione reale.

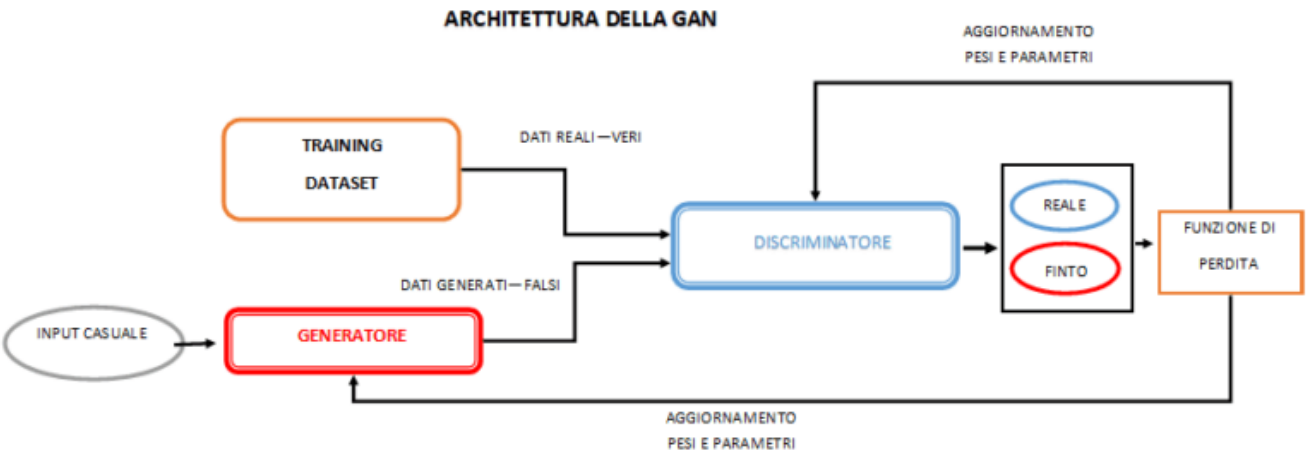


Figura 2.5: Architettura della GAN [10]

- **Variational Autoencoders (VAE):** Sono un modello di reti neurali utilizzati per apprendere una codifica (o rappresentazione latente) efficiente dei dati. Tuttavia, a differenza degli autoencoder tradizionali, quest'ultimi non si limitano a comprimere dati, ma ne generano nuovi. Anche questa architettura usa due reti neurali, come le GANs, queste sono classificate come:

  - **Encoder:** Accetta i dati in input e li riduce in una rappresentazione a bassa dimensionalità chiamata spazio latente, preservandone le caratteristiche più importanti.
  - **Decoder:** Prende questa rappresentazione dello spazio latente e ricostruisce i dati di input originali, ricostruendo il più possibile l'input processato dall'encoder.

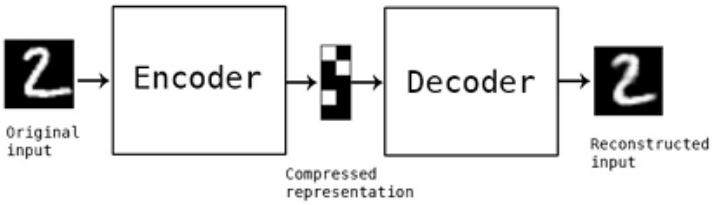


Figura 2.6: Architettura di un Variational Autoencoder [1]

I VAE sono diversi dai GANs poiché si concentrano sull'apprendimento della distribuzione sottostante dei dati di addestramento e generare nuovi dati campionando da questa distribuzione appresa. Questo approccio consente di generare VAE nuovi output preservando le caratteristiche dei dati di training. I Variational Autoencoders rappresentano una tecnica potente per la generazione e modellazione probabilistica dei dati. Grazie alla loro architettura e al modo in cui trattano lo spazio latente, i VAE offrono un approccio elegante e flessibile per generare nuovi dati che seguono la distribuzione appresa.

2.3.2 Architettura Transformers

I LLM, come detto precedentemente, sono modelli di AI generativa basati su un architettura di Rete Neurale specifica, detta, Transformers introdotta al pubblico nel 2017 grazie al paper "Attention Is All You Need" [13]. Quest'ultima è stata progettata per gestire i dati sequenziali, come il testo. La rete neurale in questione considera il signficato dell'intera frase nel suo complesso e non soltanto delle singole parole in modo sequenziale, come faceva il loro predecessore ovvero le Reti Neurali Ricorrenti.

Questo si è reso possibile grazie al meccanismo dell' ≪Attenzione≫, il quale permette ai transfomers di valutare l'importanza relativa di ogni parola all'interno della frase. Cosi facendo permette a quest'ultimi di capire meglio le dipendenze tra le parole,indipendentemente dalla loro posizione e lontananza all'interno della frase, e il significato complessivo della frase (Contesto). L'architettura dei transformers è basata su due componenti: Encoder e Decoder

L'encoder riceve un input e ne costruisce una rappresentazione, le features. Ciò significa che il modello è ottimizzato per la comprensione dell'input.

Il decoder utilizza la rappresentazione dell'encoder (le features) assieme ad ulteriori input per generare la sequenza target. Ciò significa che il modello è ottimizzato per la generazione di output. Questo **approccio parallelo** consente ai Transformer di catturare relazioni complesse tra le parole senza dipendere da una struttura sequenziale.

Funzionamento Transformers

I Transformers cominciano il processo di elaborazione del testo con la tokenizzazione. In questa fase, il testo viene suddiviso in token, ovvero unità numeriche che rappresentano parole, frammenti di parole, o anche singoli caratteri.I token sono le unità fondamentali con le quali un LLM lavora.

Successivamente, l'encoder esegue un processo chiamato embedding, che converte ogni token in un vettore numerico all'interno di uno spazio a dimensione ridotta. Questi vettori rappresentano le caratteristiche semantiche dei token, e token con significati simili tendono ad essere vicini tra loro in questo spazio. Questo è un processo dinamico, ciò significa che lo stesso token può avere rappresentazioni diverse a seconda del contesto in cui appare.

al fine di gestire l'ordine dei token, i Transformers applicano il positional encoding. Considerando il fatto che il modello non ha una consapevolezza intrinseca della sequenza dei token, vengono aggiunti ai vettori di embedding dei vettori posizionali, che forniscono informazioni sulla posizione di ogni token all'interno della sequenza.

A questo punto entra in gioco il meccanismo di autoattenzione, che è il nucleo dell'architettura dei Transformers. Questo meccanismo valuta quanto ogni token deve "prestare attenzione" agli altri token nella sequenza. Con il Multi-Head Attention, il modello è in grado di considerare diverse parti dell'input contemporaneamente, poiché ogni "testa" di attenzione si concentra su aspetti diversi della sequenza, permettendo di catturare relazioni diverse all'interno della stessa rappresentazione vettoriale.

Dopo il livello di attenzione, viene applicato uno strato di addizione e normalizzazione, che viene utilizzato per stabilizzare e migliorare l'apprendimento del modello. L'encoder include anche una rete feedforward che, attraverso trasformazioni lineari e funzioni di attivazione, aumenta la capacità del modello di rappresentare relazioni complesse.

L'output dell'encoder, il quale contiene informazioni strutturate e dettagliate sull'input, viene poi passato al decoder. Il decoder segue un processo simile: utilizza il livello di embedding e la codifica posizionale, seguiti da una variante dell'autoattenzione chiamata masked self-attention. Questa tecnica fa si che ogni token consideri solo i token precedenti nella sequenza, evitando di guardare ai token futuri durante la generazione del testo.

Il penultimo passo nel decoder è un layer lineare, che prepara l'output per la predizione finale. Infine, uno strato di softmax viene utilizzato per calcolare le probabilità dei possibili token successivi, consentendo al modello di generare il testo una parola alla volta, fino al completamento della sequenza.

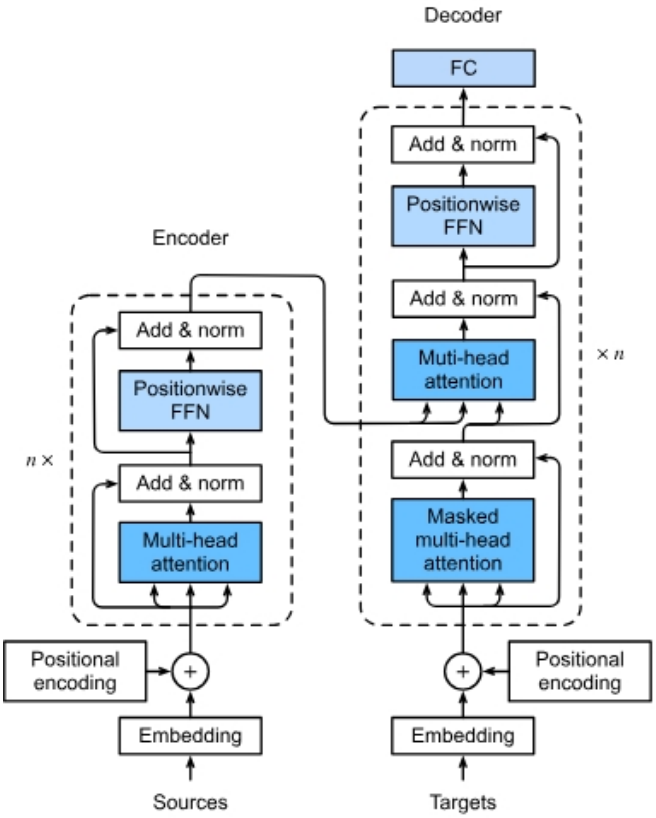


Figura 2.7: Architettura Tranfomers [13]

2.4 Miglioramento del Modello

Come si è potuto evincere dalle sezioni precedenti questi LLM sono pre-addestrati su un enorme corpus di dati comprendenti miliardi di parole. Questa fase fornisce al modello un'ampia conoscenza generale su linguaggio, argomenti presenti nei dati di addestramento.

Tuttavia, nonostante la loro incredibile ampiezza, questi set di dati mancano di competenze specializzate per settori o attività specifici, andando cosi ad influire sulla loro precisione e pertinenza in determinati contesti lavorativi.

Proprio per sopperire a queste mancanze, sono state create delle tecniche che potenziano le capacità dei modelli. Queste tecniche fan si che i LLM pre-addestrati si specializzino su determinati contesti d'uso andando così ad eccellere in determinate situazioni richieste da quel dominio applicativo.

2.4.1 Fine-Tuning

Il processo del Fine-Tuning si basa sul concetto di transfer learning, che è un principio centrale nel machine learning. Questo concetto prevede che un modello possa trasferire e adattare la conoscenza acquisita da un compito per migliorare le sue performance su un altro compito correlato. [4]

Quest'ultimo adatta un modello pre-addetsrato a un compito specifico o a un dominio in particolare. Di partenza i LLM, sono già addestrati su enormi dataset generici, il fine-tuning consente di specializzare il modello utilizzando un dataset più piccolo ma più pertinente al compito desiderato.

Durante il processo di Fine-Tuning, andando ad addestrare ulteriormente il modello su un nuovo set di dati più specifico, si modificheranno i parametri del LLM per ottimizzare le prestazioni in un contesto particolare (ad esempio nella traduzione del linguaggio naturale in Query SQL).

Poiché il modello ha già acquisito una vasta conoscenza durante la fase di pre-addestramento, il Fine-Tuning richiede meno risorse computazionali e meno dati rispetto all'addestramento da zero. Nonostante questo richiede dati etichettati e comporta costi computazionali non trascurabili per le aziende che decidono di implementarlo.

Figura 2.8: Processo Fine-tuning [12]

2.4.2 Retrieval Augmented Generation

La RAG, acronimo di Retrieval Augmented Generation, rappresenta un approccio innovativo nell'ambito dei LLM che sta rivoluzionando il modo in cui questi sistemi accedono e utilizzano le informazioni. [4]

La RAG si è fatta notare dagli sviluppatori di IA generativa dopo la pubblicazione di "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" [9], un articolo del 2020 pubblicato da Patrick Lewis e un team di Facebook AI Research. Il concetto di RAG è stato accettato da molti ricercatori accademici e del settore, i quali lo vedono come un modo per migliorare significativamente il valore dei sistemi di intelligenza artificiale generativa.

Questa metodologia sfrutta la capacità generativa degli LLM insieme alla possibilità di recuperare informazioni mirate da una base di conoscenza esterna. L'idea centrale alla base della RAG è semplice ma estremamente efficace: anziché dipendere esclusivamente dalla conoscenza "incorporata" nei parametri del modello durante l'addestramento, un sistema RAG può consultare dinamicamente fonti esterne durante la generazione delle risposte. Cosi facendo si riduce drasticamente il rischio di allucinazioni in un determinato contesto d'uso, il quale non era presente nei dati di addestramento del

LLM. Questo approccio è particolarmente utile in contesti dove è necessaria una conoscenza aggiornata e contestualizzata.

La pipeline RAG è la seguente:

1. **Importazione dati**: I dati vengono acquisiti da varie origini, come file locali, Cloud Storage e Google Drive.

2. **Trasformazione dei dati**: I dati vengono preparati per l'indicizzazione attraverso la conversione in blocchi.

3. **Embedding**: Le rappresentazioni numeriche delle parole o dei testi vengono create. Questi numeri catturano il significato semantico e il contesto del testo, con parole o testi simili che hanno embeddings simili e risultano più vicini nello spazio vettoriale ad alta dimensione.

4. **Indicizzazione dei dati**: La knowledge base viene strutturata in un corpus ottimizzato per la ricerca, simile a un sommario dettagliato per un vasto libro di riferimento.

5. **Recupero**: Quando l'utente pone una domanda o fornisce un prompt, il componente di recupero cerca nella knowledge base(Tipicamente un database vettoriale) le informazioni rilevanti per la query.

6. **Aumento**: Questa parte prevede il miglioramento e l'aggiunta di un contesto più pertinente alla risposta recuperata per la query dell'utente.

7. **Generazione**: Le informazioni recuperate sono utilizzate come contesto per guidare il modello di AI generativa nella produzione di risposte pertinenti e basate su dati concreti.

In sintesi, Quando viene posta una query al sistema, questa viene inizialmente utilizzata per cercare informazioni pertinenti in una base di conoscenza esterna. Le informazioni trovate vengono poi integrate nel prompt del modello linguistico, aggiungendole al contesto originale della query. Il modello linguistico quindi elabora una risposta, combinando la sua conoscenza interna con i dati specifici recuperati.

Un elemento essenziale per il funzionamento efficace del sistema RAG è il database vettoriale, o VectorDB. Questo tipo di database è progettato specificamente per gestire dati rappresentati come vettori ad alta dimensionalità, fondamentali per rendere comprensibile alle macchine il significato semantico dei testi.

Nel contesto della RAG, il VectorDB viene utilizzato per memorizzare e recuperare rapidamente "embeddings", che sono rappresentazioni numeriche di parole, frasi, documenti o altri media, capaci di catturare il loro significato semantico.

Quando una query viene inviata al sistema RAG, viene trasformata in un embedding e utilizzata per cercare nel VectorDB i documenti o frammenti di informazione più simili dal punto di vista semantico.

La potenza dei VectorDB risiede nella loro capacità di effettuare queste ricerche in maniera estremamente veloce, anche su grandi volumi di dati. L'integrazione dei VectorDB nel sistema RAG consente di combinare l'efficienza nel recupero delle informazioni con le capacità generative dei modelli linguistici, creando così sistemi più intelligenti e consapevoli del contesto.

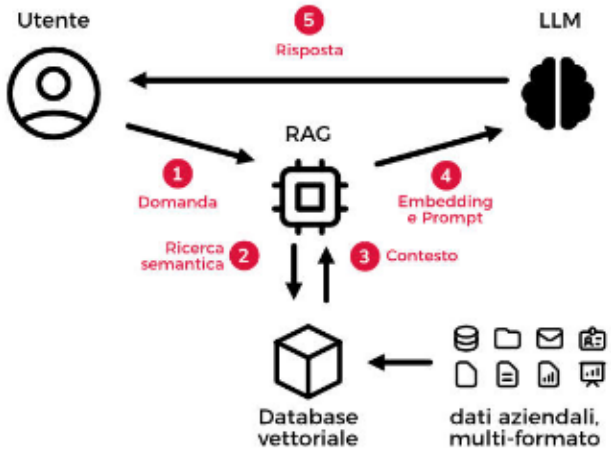


Figura 2.9: Architettura RAG

2.4.3 Prompt-Engineering

L'Ingegneria del Prompt è una nuova tecnica che si concentra sull'ottimizzazione degli input forniti al modello per produrre output più precisi e pertinenti, al contesto nel quale si vuole applicare il modello.

- **Formulazione Strategica del Prompt:** Le function call consentono di costruire query SQL con sintassi esatta e parametri definiti, garantendo che le interrogazioni siano comprese e correttamente eseguite dal sistema di gestione del database (DBMS). Ciò riduce l'ambiguità e aumenta l'accuratezza delle query, essenziale per operazioni che richiedono alta precisione.
- **Uso di Esempi e Contesto:** L'ingegneria del prompt richiede di riformulare i comandi o le domande in modo che il modello possa rispondere con maggiore precisione. Il modello può essere guidato verso la risposta desiderata scegliendo parole specifiche, l'ordine delle parole e altri dettagli rilevanti.
- **Riduzione della Dipendenza dal Fine-Tuning:** In alcuni casi, l'ingegneria del prompt può ridurre la necessità di fine-tuning, permettendo di ottenere risultati soddisfacenti anche senza modificare i parametri del modello.

L'ingegneria del prompt è un'arte che sfrutta la flessibilità del modello per massimizzare la qualità delle risposte senza necessità di ulteriori risorse computazionali, rendendola una strategia efficace e poco costosa per l'ottimizzazione del modello.

2.5 Allucinazioni dell'AI Generativa

Le allucinazioni nell'intelligenza artificiale (IA) si riferiscono a situazioni in cui un modello di IA, come i grandi modelli di linguaggio (LLM), genera risposte che sembrano plausibili ma sono in realtà inesatte, non fondate su dati reali, o completamente inventate. Questo fenomeno è particolarmente comune nei modelli di linguaggio avanzati, che sono addestrati per predire la parola successiva in una sequenza e quindi possono generare informazioni che non sono verificate o che non esistono affatto. Per mitigare

questo problema, è importante integrare meccanismi di verifica dei fatti e strumenti di recupero di informazioni affidabili (come il Retrieval-Augmented Generation, RAG) nei sistemi IA.

2.6 Temperatura dell'AI Generativa

La temperatura è un parametro di controllo utilizzato nei modelli di generazione del linguaggio per regolare la creatività o la casualità del testo generato. Questo parametro influisce sulla probabilità che il modello selezioni un output specifico tra le opzioni disponibili durante la generazione del testo.

- **Bassa Temperatura:** Quando la temperatura è vicina a zero, il modello tende a essere più deterministico e sceglie l'output con la probabilità più alta. Questo porta a generare risposte più prevedibili e coerenti, ma può anche risultare in output ripetitivi e meno creativi.
- **Alta Temperatura:** Con una temperatura più alta, le probabilità delle parole vengono "appiattite", il che aumenta la casualità nella selezione delle parole. Questo rende il modello più creativo, ma può anche aumentare il rischio di generare risposte incoerenti o meno rilevanti.

La temperatura è utile per bilanciare tra coerenza e creatività a seconda dell'applicazione. Ad esempio, in contesti dove è necessaria una risposta precisa e diretta (come nel question answering), si preferisce una temperatura bassa. In contesti creativi, come la generazione di poesia o storie, una temperatura più alta può essere desiderabile per ottenere output più variegati e interessanti.

3

SqlTalk: Descrizione del progetto e codice del progetto

3.1 Introduzione

Dopo aver spiegato le nozioni principali dell'IA Generativa, possiamo introdurre e spiegare il progetto SqlTalk, che si basa proprio su questi nozioni tecniche. L'applicativo è nato per fornire supporto ai clienti dell'azienda nella fase di analisi dati, facendo in modo che non si debbano sempre affidare ad un esperto. Quest'ultimo consente di trasformare le query da parte degli utenti ,in linguaggio naturale, in query SQL per interrogare il database e far si che si scoprino insights significativi.

L'applicativo fa uso LLM Gemini di Google. È stato preferito a GPT per il semplice motivo che una soluzione con quest'ultimo era già presente in azienda. SqlTalk è stato creato come un evoluzione di esso, per fare in modo da avere anche un confronto in termini di velocità di risposta e costi tra i due LLM, in modo tale da poter scegliere il più performante e adatto agli scopi aziendali. Grazie a Gemini, SqlTalk può comprendere e interpretare le richieste degli utenti, generando query SQL precise che possono essere eseguite su un database come BigQuery. Il sistema non solo genera le query necessarie, ma può anche analizzare il contesto delle richieste, creando grafici e visualizzazioni dei dati quando richiesto dall'utente. SqlTalk rappresenta un esempio pratico di come i Large Language Models (LLM) possano essere integrati in applicazioni aziendali per migliorare la capacità decisionale, semplificando l'accesso alle informazioni all'interno dei dati aziendali da parte dei clienti

In questa sezione, verranno esaminati i componenti principali di SqlTalk, come ad esempio l'uso di funzioni specializzate per l'interrogazione del database, e le tecniche implementate per migliorare la precisione e la sicurezza delle operazioni di query, oltre alle tecnologie e strumenti utlizzati.

3.2 Tecnologie e strumenti utilizzati

Per la creazione dell'applicativo ho dovuto mettere in pratica le competenze teoriche, di tecnologie e strumenti fondamentali come python e SQL, acquisite durante il mio percorso triennale. Queste due tecnologie sono il cuore pulsante dell'applicativo, qui di seguito fornisco una panoramica di entrambe le tecnologie e degli ulteriori strumenti utuilizzati al fine di completare il progetto.

3.2.1 Linguaggi di programmazione

Python

Python è un linguaggio di programmazione ad alto livello, interpretato e dinamico, noto per la sua sintassi semplice e leggibile, che lo rende ideale sia per principianti che per sviluppatori esperti. Utilizzato in una vasta gamma di applicazioni, Python è particolarmente popolare nei campi della scienza dei dati, del machine learning, dello sviluppo web e dell'automazione dei processi.

Uno dei punti di forza di Python è la sua vasta libreria di pacchetti e moduli, che estendono le sue funzionalità e semplificano compiti complessi come l'elaborazione dei dati, la manipolazione dei file, la comunicazione con API e altro ancora.

Python è anche noto per la sua versatilità e interoperabilità con altre tecnologie. All'interno dell'applicativo, Python è stato cruciale per automatizzare i processi di estrazione e analisi dei dati.

python è stato usato per tutto lo sviluppo del progetto.

SQL

SQL (Structured Query Language) è un linguaggio standardizzato per la gestione e manipolazione di database relazionali. È utilizzato per creare, leggere, aggiornare e cancellare dati (le cosiddette operazioni CRUD) all'interno di database come MySQL, PostgreSQL, Microsoft SQL Server e BigQuery.

Nel contesto del mio stage, ho utilizzato SQL per interrogare e manipolare grandi dataset memorizzati in BigQuery. L'integrazione con applicativi come SqlTalk mi ha permesso di tradurre richieste in linguaggio naturale in query SQL automatizzate, offrendo un modo intuitivo per interagire con i dati senza la necessità di scrivere manualmente complesse istruzioni SQL.

3.2.2 Tecnologie utilizzate

Come si potrà evincere dagli strumenti utilizzati per la creazione dell'applicativo la suite di servizi usati è Google Cloud, una suite di servizi cloud computing offerta da Google, progettata per soddisfare le esigenze di aziende e sviluppatori. L'azienda essendo partner Google, utilizza i servizi offerti da Google Cloud per le proprie soluzioni software.

BigQuery

BigQuery è un data warehouse cloud-based altamente scalabile e performante sviluppato da Google, progettato per l'analisi di grandi set di dati utilizzando SQL. È parte integrante della Google Cloud Platform (GCP) e si distingue per la capacità di elaborare petabyte di dati in modo rapido, sfruttando un'architettura serverless. Questa architettura significa che non è necessario gestire o configurare i server: Google si occupa della scalabilità e delle prestazioni. BigQuery consente di eseguire query complesse su enormi quantità di dati in tempo reale grazie alla sua struttura basata su colonne, che ottimizza la lettura di dati pertinenti. Viene comunemente utilizzato per analisi di grandi volumi di dati in settori come il retail, il marketing, la finanza e altri.

Google Cloud Storage

Google Cloud Storage è un servizio di archiviazione su cloud fornito da Google Cloud Platform, progettato per offrire una soluzione sicura, scalabile e durevole per l'archiviazione di file di qualsiasi dimensione. Permette di memorizzare oggetti in "bucket", che sono contenitori virtuali in cui si possono organizzare i file. Nel contesto del' applicativo, Google Cloud Storage viene utilizzato per memorizzare e distribuire file CSV e XLSX contenenti le risposte generate dall'interrogazione del database tramite SqlTalk.

Una volta che l'utente esegue una query e richiede il file CSV o XLSX, nel momento in cui riceve una risposta, i dati vengono esportati in formati leggibili come CSV o XLSX, i quali vengono salvati in bucket precedentemente inizializzati su Google Cloud Storage. Gli utenti possono quindi scaricare facilmente questi file per ulteriore analisi o integrazione con altri strumenti.

Gemini-1.5-Flash-Preview-0514

≪Gemini-1.5-Flash-Preview-0514≫ è un modello di linguaggio di grandi dimensioni (LLM) sviluppato da Google DeepMind, progettato per comprendere e generare testo in linguaggio naturale. Ha una finestra fino a 1,048,576 token per gestire grandi volumi di testo in una singola richiesta, con un limite di 8,192 token per l'output. Inoltre, è progettato per funzionare bene in applicazioni in cui costi e latenza sono fattori critici, offrendo una buona qualità a un costo inferiore rispetto ad altre varianti come il modello Pro. Questo lo rende ideale per chat assistant e contenuti generati on-demand In più questo modello è efficace nelle applicazioni in cui la traduzione di linguaggio naturale in linguaggi formali, come SQL, è il principio cardine.

Vertex AI Studio

Vertex AI è una piattaforma integrata offerta da Google Cloud per lo sviluppo e la gestione di modelli di intelligenza artificiale, inclusi i grandi modelli di linguaggio (LLM) come Gemini. Quest'ultima è stata progettata per dare alle aziende e alle organizzazioni un modo per creare e testare soluzioni che implementano l'AI.

In questo caso specifico abbiamo usato Vertex AI Studio per creare un notebook virtuale. Quest'ultimo non è che altro una macchina virtuale preconfigurata che consente agli sviluppatori e ai data scientist di lavorare su progetti di intelligenza artificiale e machine learning direttamente nel cloud. Questi notebook offrono un ambiente di sviluppo integrato (IDE) basato su Jupyter, uno strumento molto popolare per la scrittura ed esecuzione di codice Python e altre attività di analisi dei dati e addestramento dei modelli ML. In particolare, i notebook su Vertex AI sono particolarmente utili perché forniscono accesso immediato a risorse di calcolo scalabili e modelli di LLM. In questo caso specifico abbiamo usato il notebook di Vertex AI Studio per avere accesso illimitato al LLM ≪gemini-1.5-flash-preview-0514≫ [7] testando senza costi reali l'applicativo. Inoltre, sono strettamente integrati con i servizi di Google Cloud, come BigQuery e Google Cloud Storage.

Google Cloud Run

Google Cloud Run è un servizio completamente gestito di Google Cloud che consente di eseguire applicazioni containerizzate in modo scalabile e senza server. Cloud Run è progettato per semplificare

il deployment di applicazioni scritte in qualsiasi linguaggio o framework, che vengono racchiuse in un container Docker. Questo fornisce un'infrastruttura che scala in maniera automatica in base al traffico presente e permette di gestire le applicazioni senza dover gestire i server sottostanti. Cloud Run è stato creato per funzionare in modo ottimale insieme ad altri servizi su Google Cloud, consentendoti di creare applicazioni complete (ad esempio Google Cloud Storage, BigQuery, e Vertex AI) rendendolo una soluzione ideale per il deployment di applicativi.

3.3 Workflow AI generativa

In questa sezione esaminero il workflow generale dell'applicativo, quest'ultimo è diviso in diverse fasi le quali sono essenziali per capire il funzionamento dell'intero sistema. Più avanti esamineremo il workflow specifico delle function call (Figura 3.7), andando a spiegarne il funzionamento.

1. **Prompt:**
   Il processo inizia con un *prompt*, ovvero una richiesta formulata dall'utente in linguaggio naturale. Ad esempio, l'utente potrebbe chiedere: "Quali sono i clienti che hanno effettuato acquisti superiori a 1000 euro negli ultimi sei mesi?". Il prompt dell'utente costituisce il punto di partenza per il modello di IA generativa.

2. **Modelli di Base:**
   Il prompt viene inviato a un modello di IA generativa, in questo caso tramite l'API Gemini di Vertex AI. Gemini è particolarmente adatto per compiti complessi come il ragionamento avanzato e la generazione di codice; in questo caso, per la conversione di linguaggio naturale in SQL. Altri modelli disponibili, come l'API Imagen e MedLM, offrono funzionalità per la generazione di immagini e risposte mediche, rispettivamente, ma non sono rilevanti per questo specifico contesto.

3. **Personalizzazione del Modello:**
   Per ottenere risultati coerenti e pertinenti, il comportamento predefinito dei modelli può essere personalizzato attraverso un processo chiamato *ottimizzazione del modello*. Questo permette di adattare il modello alle esigenze specifiche dell'applicazione, riducendo la complessità dei prompt necessari per generare le risposte desiderate. Questo è stato fatto grazie alla tecnica del Prompt-Engineering. Così facendo il modello è stato ottimizzato e reso ultraperformante in questo specifico scenario d'uso

4. **Aumento dei Dati:**
   L'aumento dei dati gioca un ruolo cruciale nel migliorare la precisione delle risposte generate. ≪Aumentare i dati≫ fa si che le risposte prodotte dal modello siano all'interno di un contesto specifico, in questo è l'interrogazione del database aziendale. Questo processo avviene tramite le **Chiamate di Funzioni** nel nostro caso, quest'ultime vengono utilizzate per consentire al modello di interagire con il database in tempo reale. Questo permette al modello di eseguire query SQL direttamente sul database, restituendo informazioni aggiornate e precise.

5. **Verifica delle Citazioni:**
   Dopo aver generato la risposta, il processo verifica se la risposta richiede delle citazioni. Se una

parte significativa della risposta deriva da una fonte specifica, questa viene inclusa nei metadati come citazione, garantendo trasparenza e verificabilità.

6. **Filtri di Sicurezza:**
   Prima che la risposta venga restituita all'utente,l'applicativo applica dei *filtri di sicurezza*. Questi controlli verificano che né il prompt né la risposta contengano contenuti che superino le soglie di rischio definite. Se una soglia viene superata, la risposta viene bloccata e sostituita con una risposta di riserva per evitare potenziali danni o disinformazioni.

7. **Risposta:**
   Infine, se il prompt e la risposta superano i controlli di sicurezza, la risposta viene restituita all'utente. Nel contesto dell'applicazione SqlTalk, questo significa che l'utente riceverà una risposta alla sua richiesta in linguaggio naturale; ovviamente quest'ultima è stata tradotta in SQL, dal LLM Gemini, e poi processata all'interno del database associato al cliente al fine di ottenere una risposta sia consona ai requisiti di sicurezza, che alla domanda posta dal'utente.

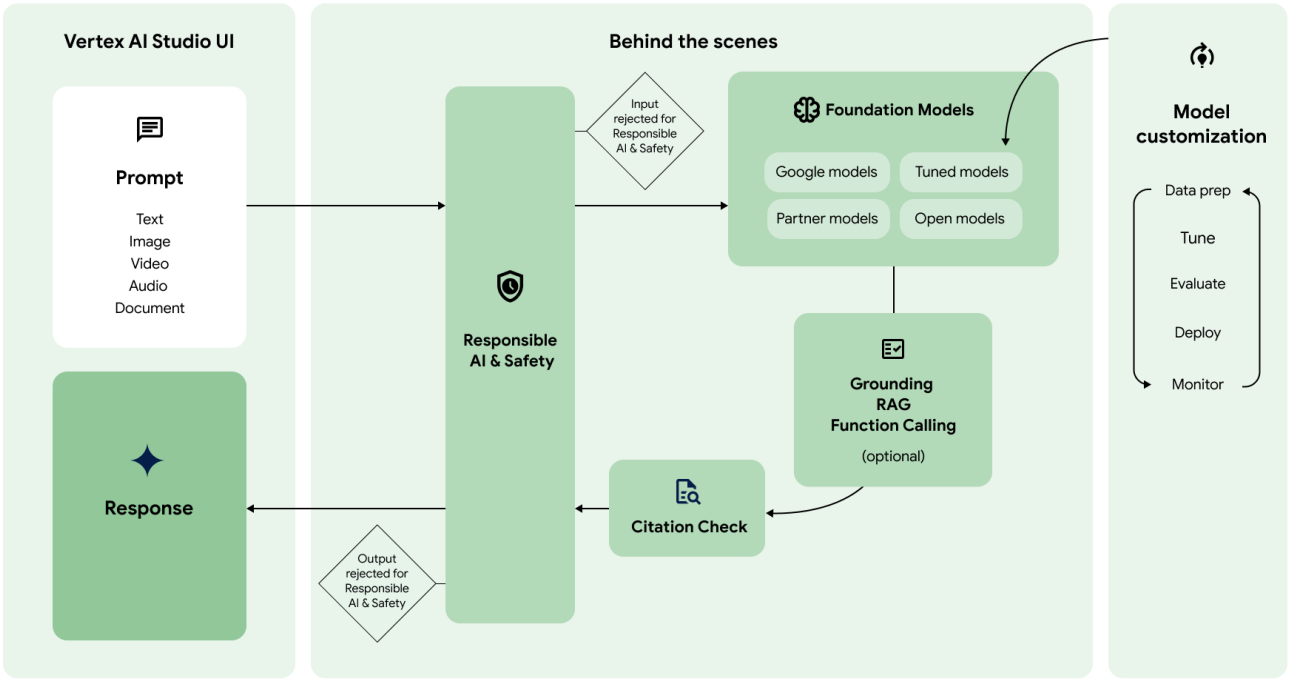


Figura 3.1: Workflow applicativo

Valutazioni personali

In termini di valutazione personale ho apprezzato molto il flusso logico dell'applicativo, sopratutto nella sezione dedicata alla verifica delle citazioni. Questa sezione è stata particolarmente rilevante per il controllo delle fonti dalle quali si estraevano i dati per la costruzione della risposta al prompt. Personalmente, ritengo che sia un valore aggiunto notevole per il sistema, poiché permette all'utente di verificare l'attendibilità delle risposte e delle fonti utilizzate dall'applicativo, contribuendo a ridurre potenziali rischi di disinformazione.

3.4 Creazione del Grafico

Per creare un applicativo in grado di soddisfare pienamente le esigenze dei clienti, fornendo loro un potente strumento per l'analisi dei dati, uno degli step fondamentali è stata l'integrazione della possibilità di analizzare i dati in modo visivo tramite l'utilizzo di grafici.

Al fine di poter dare agli utenti anche questa funzionalità, sono state implementate due funzioni. La prima di esse analizza il prompt dell'utente e capisce se è stato richiesto un grafico, la seconda in base all'output della prima funzione crea il grafico sulla base della risposta della query SQL. Per entrambe le funzioni è stata applicata la tecnica del Prompt-Engineering per ottimizzare il modello su un'attività specifica. La prima funzione è stata denominata Analyze prompt for graph, il modello LLM di Gemini analizza il prompt e determina se l'utente ha richiesto la creazione del grafico sulla base dei risultati della query SQL.

```python
#analizza se l'utente richiede un grafico
def analyze_prompt_for_graph(prompt):
    """
    Utilizza il modello LLM di Gemini per analizzare il prompt e determinare se l'utente ha richiesto un grafico.
    Restituisce si se è stato richiesto un grafico, altrimenti no.
    """
    # Creiamo un prompt per il modello per determinare se è stato richiesto un grafico
    analysis_prompt = f"""
    Utente: {prompt}
    Assistant: Analizza il prompt dell'utente e rispondi solo con 'Si' se l'utente ha richiesto la creazione di un grafico, altrimenti rispondi con 'No'.
    """


    # Inizializza una chat con il modello
    chat = model1.start_chat(history=chat_history)
    # Manda il prompt al modello
    response = chat.send_message(analysis_prompt)
    # Estrai la risposta dal modello
    response_text = response.candidates[0].content.parts[0].text.strip()

    # Determina se la risposta è 'Sì'
    return response_text.lower() == "si"
```

Figura 3.2: Analisi Prompt per richiesta grafo

La logica, come si può vedere dalla figura soprastante, segue un filo logico ben preciso analizzando il prompt e verifcando se l'utente ha richiesto la creazione di un grafo. In base a questa analisi , il risultato verrà passato alla funzione Generate Graph Code with Gemini che, sulla base dell'output della prima funzione, genererà o meno il codice per la creazione del grafo per l'analisi visiva dei dati.

```python
#creo il codice per il grafico
def generate_graph_code_with_gemini(query_result, prompt):
    """
    Utilizza Gemini per creare il codice del grafico e lo restituisce.
    """
    # Supponiamo che il risultato della query sia una stringa JSON
    df = pd.DataFrame(query_result)


    # Genera il prompt per Gemini
    graph_prompt = f"""
    crea SOLO IL CODICE ,in modo tale che io possa copiarlo e incollarlo su jupyter notebook togliendo tutti i simboli strani,
    CHE POI POSSA ESEGUIRE PER LA CREAZIONE DI UN GRAFICO APPROPRIATO IN BASE AL df che ti sto per passare e a quello che vuole l'utente.
    df= {df}
    per capire che tipologia di grafico deve rappresentare i dati ,per poi creare il codice con il grafico che li rappresenta, analizza sempre il prompt:{prompt}
    imposta una dimensione del grafico elevata e fai tutti gli accorgimenti del caso per facilitare la comprensione all'utente.
    """

    chat = model1.start_chat(history=chat_history)
    response = chat.send_message(graph_prompt)
    graph_code = response.candidates[0].content.parts[0].text.strip()
    graph_code = graph_code.replace(" ```python", "").replace("```", "").replace("python","")

    print(df)
    print(graph_code)


    # Esegui il codice del grafico
    exec(graph_code)
```

Figura 3.3: Generazione grafo con gemini

La logica della funzione trasforma il rsultato della query in un dataframe di Pandas (Una libreria di python per l'analisi dei dati), il quale lo passerà come argomento per la creazione del codice relativo al grafo. Inoltre analizzerà il prompt dell'utente per capire se è stata richiesta una tipolgia di grafo in specifico e in seguito eseguirà il codice per la creazione di quest'ultimo.

3.5 Sicurezza in SqlTalk

L'applicativo doveva soddisfare elevate aspettative di sicurezza, considerando il fatto che il sistema interagisce con un database dentro il quale ci sono informazioni, private, relative a più clienti. Nel contesto di un sistema del genere, il fatto che gli utenti possano accedere solo ai dati a cui sono autorizzati è di vitale importanza. Per implementare questo processo è stato implementato un filtro che permette alle aziende di accedere con i propri dati personali, inserendo all'interno dell'applicativo il loro identificativo. L'ID dell'azienda, e volendo anche il range delle date, viene richiesto all'inizio, e vengono integrate in una INNER QUERY che crea una tabella volatile filtrata per l'esecuzione della query, anche chiamata CTE (Common Table Expression) che vedremo in seguito.

```
# Organizzazione dal id del user
COMPANY_ID = input("Company ID: ").strip()

#Controllo se la data di inizio è minore della data di fine se è cosi il ciclo continua altrimenti richiede le date
while True:
    start_date = input("Data di inizio (DD-MM-YYYY): ").strip()
    if start_date.lower() == "esci":
        break

    end_date = input("Data di fine (DD-MM-YYYY): ").strip()
    if end_date.lower() == "esci":
        break


    start_date = time.strftime("%Y-%m-%d", time.strptime(start_date, "%d-%m-%Y"))
    end_date = time.strftime("%Y-%m-%d", time.strptime(end_date, "%d-%m-%Y"))


    if start_date > end_date:
        print("La data di inizio non può essere maggiore della data di fine. Riprova.")
    else:
        break
```

Figura 3.4: Filtro Id azienda e date

Durante il processo di sviluppo di questa parte dell'applicativo è stato trovato un potenziale rischio legato alla tecnica di hacking chiamata SQL Injection. Quest'ultima è un tipo di attacco alla sicurezza in cui un utente malevole inserisce del codice SQL malevole all'interno di una query, con l'intento di accedere a dati sensisbili non propri. Questa tecnica di hacking avviene generalmente quando un'applicazione consente di concatenare direttamente i dati inseriti dall'utente nella query SQL, per esempio attraverso il filtro citato in precedenza. Nella prossima sezione illustrerò la tecnica che è stata applicata, al fine di risolvere questa problematica relativa alla sicurezza dell'applicativo.

3.5.1 Parametri scalari per evitare possibili SQL Injection

Al fine di mitigare questo rischio, soddisfacendo le esigenze di sicurezza, e potendo dare ai clienti un'applicativo che, non solo fosse in grado di analizzare i loro dati sensibili, ma anche di proteggerli è

stata addottatta una tecnica chiamata **parametrizzazione**. La parametrizzazione consiste nell'inserire dei parametri scalari all'interno della query SQL, invece che concatenare i valori dinamici(Come ad esmepio l'id della azienda) nella stringa SQL, prevenendo attacchi di SQL Injection. L'implementazione dei parametri scalari è stata resa possibile grazie alla classe bigquery.ScalarQueryParameter fornita dalla libreria Google Cloud BigQuery. I parametri scalari sono stati implementati come in figura 3.5

```
if response.function_call.name == "sql_query":
    job_config = bigquery.QueryJobConfig(
        # In questo modo, i valori dei parametri sono passati in modo sicuro alla query, proteggendo l'applicazione da potenziali vulnerabilità di SQL injection
        query_parameters=[
            bigquery.ScalarQueryParameter("COMPANY_ID", "STRING", COMPANY_ID),
            bigquery.ScalarQueryParameter("start_date", "DATE", start_date),
            bigquery.ScalarQueryParameter("end_date", "DATE", end_date),
        ],
        maximum_bytes_billed=100000000,
    )
```

Figura 3.5: Implementazione parametri scalari

I parametri scalari vengono inseriti seguendo il flusso di lavoro definito dal codice soprastante. prima di tutto verifica se il nome della funzione chiamata è sql query. Dopodichè viene configurato il lavoro della query nel seguente modo: viene creato un oggetto QueryJobConfig che include i parametri di query definiti come ScalarQueryParameter. I parametri sono COMPANY ID (stringa), start date (data) e end date (data). infine viene anche impostato un limite massimo di byte fatturabili (100 MB), limitando la quantità massima di dati che possono essere elaborati in risposta a una query, proteggendo l'applicativo da eventuali query involontariamente costose o mal formulate che potrebbero processare grandi quantità di dati non necessari.

3.5.2 Common Table Expression

Un'altra problematica presente nel progetto era il fatto che all'interno del database contenente i dati dei clienti erano presenti molteplici articoli con ID uguale, ma con prezzo differente. Per far si che i dati presi in considerazione dall'interrogazione della query non fossero «vecchi», e si andasse a creare un problema di non veridicità dell'analisi, si è implementata all'interno dell'applicativo una Common Table Expression contenente solo i dati con Last Update più recente per ID. Una Common Table Expression (CTE) è una funzionalità SQL che permette di creare una tabella temporanea durante l'esecuzione di una query. Viene definita all'interno di una query SQL utilizzando la parola chiave WITH seguita dal nome della CTE e dalla query che definisce i dati da includere in essa. Le CTE funzionano come tabelle virtuali (con record e colonne) che vengono create al volo durante l'esecuzione di una query detta «principale», quest'ultime vengono consumate dalla query «principale» e distrutte dopo l'esecuzione della stessa.

L'implementazione della Common Table Expression (CTE) è avvenuta tramite la logica della figura 3.6.

```
try:
    # pulisci la query rendendola leggibile da big query
    cleaned_query = params["query"].replace("\\n", " ").replace("\n", "").replace("\\", "")

    # Inner query per filtrare i dati e ottenere solo i dati relativi agli articoli della company_ID con LAST_UPDATE più recente per ID
    inner_query = f"""
        WITH latest_records AS (
            SELECT *
            FROM (
                SELECT *, ROW_NUMBER() OVER (PARTITION BY ID ORDER BY LAST_UPDATE DESC) AS row_num
                FROM `{SPECIFIC_TABLE_ID_1}`
                WHERE COMPANY_ID = @COMPANY_ID
                AND DATE(DOCUMENT_DATE) >= @start_date
                AND DATE(DOCUMENT_DATE) <= @end_date
            )
            WHERE row_num = 1
        )
    """

    # far si che i dati relativi alla query più esterna venagno presi dalla query filtrata in precedenza
    if "FROM latest_records" not in cleaned_query:
        cleaned_query = cleaned_query.replace(
            f"FROM `{SPECIFIC_TABLE_ID_1}`",
            "FROM latest_records"
        )

    # Combinare le due query
    final_query = inner_query + cleaned_query
    print(final_query)

    # eseguire la final query
    query_job = client.query(final_query, job_config=job_config)
    api_response = query_job.result()
    query_result = [dict(row) for row in api_response]
    api_response = str(query_result)
    api_response = api_response.replace("\\", "").replace("\n", "")
    api_requests_and_responses.append(
        [response.function_call.name, params, api_response]
    )
```

Figura 3.6: Common Table Expression

Da quest'ultima si può evincere che la Common Table Expression (CTE) chiamata all'interno del codice «inner query», è statà implementata al fine di filtare i record in base all'ultimo aggiornamento (Last Update). Questa CTE seleziona tutti i campi e utilizza Row Number() per assegnare un numero di riga a ciascun record, partizionando per ID e ordinando per Last Update in ordine decrescente, mantenendo solo i record più recenti per ogni ID. Inoltre si può notare l'uso dei parametri scalari (company id, start date e end date) solo all'interno della «inner query», al fine di evitare le Sql Injection e , contemporanemente, evitando ridondanze e migliorando l'efficenza della query principale creata dal LLM.

Dopodichè la query principale viene modificata per utilizzare latest records come dataset principale e le due query (inner query e query principale) vengono combinate in una sola. Infine la query combinata viene eseguita utilizzando client.query().

3.6 Function call di Gemini

L'applicativo è costruito intorno alle «Function Call» di Gemini. Quest'ultime sono un modo sofisticato di interagire con i modelli di intelligenza artificiali, come Gemini, per eseguire funzioni specifiche (richieste di dati, calcoli o interazione con sistemi esterni).

Nel momento in cui il LLM elabora la query in linguaggio naturale, quest'ultimo non esegue direttamente le funzioni, ma genera un output che include il nome della fuinzione da chiamare e gli argomenti con cui eseguirla

Il modello, quindi, fornisce una descrizione strutturata di ciò che deve essere fatto, lasciando all'applicazione il compito di eseguire le funzioni attraverso API esterne. L'applicazione prende questo output e invoca le API corrispondenti (ad esempio, per interrogare un database SQL o per richiamare altre fonti di dati esterne). Una volta ottenuti i risultati dall'API, questi possono essere inviati nuovamente al modello, che li utilizza per completare la sua risposta finale alla query dell'utente.

Questa modalità di interazione con le function call offre diversi vantaggi, tra cui la possibilità di accedere a informazioni in tempo reale e di interfacciarsi con vari servizi esterni, come database SQL. In questo modo, gli LLM (Large Language Models) possono generare risposte più accurate e aggiornate, migliorando significativamente l'efficienza del processo di elaborazione delle query e garantendo che il modello possa ottenere informazioni che non erano presenti durante il suo addestramento.

3.6.1 Perchè le chiamate di funzioni e non la RAG?

La scelta progettuale di usare le Function Call piuttosto che la RAG è stata presa per un motivo semplice. In questo specifico caso in cui il database da interrogare è sql, quindi dati strutturati, l'interrogazione richiede una precisione molto alta. Le query sql, fornite direttamente a bigquery, forniscono una precisione più elevata piuttosto che la ricerca vettoriale per similarità che è all'interno del flusso RAG.

In più, generalmente, la RAG viene scelta come approccio progettuale in quei casi in cui i dati aziendali sono NON strutturati, ad esempio Documenti Testuali, video etc...

In sintesi ecco i vantaggi che hanno le Function Call rispetto alla RAG in questo specifico caso d'uso

- **Precisione e Specificità delle Query SQL:** Le function call permettono la costruzione di query SQL esatte e con parametri definiti, garantendo che le interrogazioni siano comprese dal database (DBMS). Questo fa si che aumenti l'accuratezza della query garantendo alta precisione e riducendo le ambiguità.

- **Limitazioni del Flusso RAG per SQL:** L'approccio RAG si basa sulla similarità del contenuto e spesso introduce ambiguità, soprattutto quando si tratta di generare query SQL strutturate. La ricerca vettoriale può restituire risultati pertinenti ma non esatti, mentre le query SQL richiedono precisione sia nei parametri che nella logica di interrogazione.

- **Controllo e Sicurezza:** Le function call permettono di avere un maggiore controllo sulla struttura della query. Consentono l'implementazione di validazioni rigorose sui dati in ingresso e logiche di accesso direttamente nelle funzioni SQL. Applicare questo livello di controllo e sicurezza non è facilmente implementabile con una ricerca vettoriale per similarità.

- **Efficienza e Prestazioni:** Le function call consentono di ottimizzare le query per prestazioni specifiche del database, utilizzando tecniche come indici e join ottimizzati. L'esecuzione diretta delle query SQL riduce il carico computazionale poiché non richiede l'analisi aggiuntiva di un corpus di documenti per trovare la risposta migliore. Questo è particolarmente vantaggioso in database di grandi dimensioni, dove i costi e i tempi degli embeddings possono essere elevati.

3.6.2 Funzionamento

Il funzionamento delle Function Call di Gemini si articola in quattro fasi, qui di seguito riportate:

- **Definizione della Funzione**: Le Function Call sono definite in anticipo con specifici parametri e azioni. Queste funzioni possono essere richiamate durante la conversazione per eseguire compiti specifici
- **Richiamo della Funzione**: Durante una conversazione, il modello di intelligenza artificiale può decidere di richiamare una funzione in risposta a una richiesta dell'utente. Questo richiamo è basato su trigger o parole chiave definite nelle istruzioni del modello.
- **Esecuzione della Funzione**: Quando viene richiamata una funzione, viene eseguito il codice associato. Questo può includere l'accesso a database, l'interazione con API esterne, o l'esecuzione di calcoli complessi.
- **Risposta al Richiamo**: Dopo l'esecuzione della funzione, il risultato viene ritornato all'utente sotto forma di risposta elaborata dal modello.

Flusso d'Uso delle Function Call

In questa sottosezione esamineremo in dettaglio il WorkFlow delle function call all'interno dell'applicativo

1. **Prompt dall'utente all'applicazione:** L'utente fornisce una richiesta in linguaggio naturale (ad esempio, "Dimmi le vendite per negozio").
2. **Prompt dall'applicazione al modello:** L'applicazione invia questa richiesta al modello di intelligenza artificiale, come Gemini.
3. **Suggerimento della funzione dal modello all'applicazione:** Il modello analizza la richiesta e suggerisce una o più function call specifiche, insieme ai parametri necessari (come il nome del negozio o la data).
4. **Richiesta API dall'applicazione all'API:** L'applicazione, in base ai suggerimenti del modello, invia una richiesta a un'API esterna, come BigQuery, per eseguire la query SQL.
5. **Risposta dell'API all'applicazione:** L'API restituisce i risultati della query SQL all'applicazione.
6. **Risposta dell'API dal modello all'applicazione:** L'applicazione fornisce i risultati dell'API al modello per ulteriori elaborazioni.
7. **Risposta del prompt dal modello all'applicazione:** Il modello elabora i dati forniti dall'API e restituisce una risposta completa e strutturata.
8. **Risposta finale all'utente:** Infine, l'applicazione fornisce la risposta completa e, se richiesto, visualizza anche un grafico o altre informazioni aggiuntive.

In seguito, mediante la figura 3.7, viene rappresentato graficamente il flusso di lavoro di una function call all'interno dell'applicativo SqlTalk.

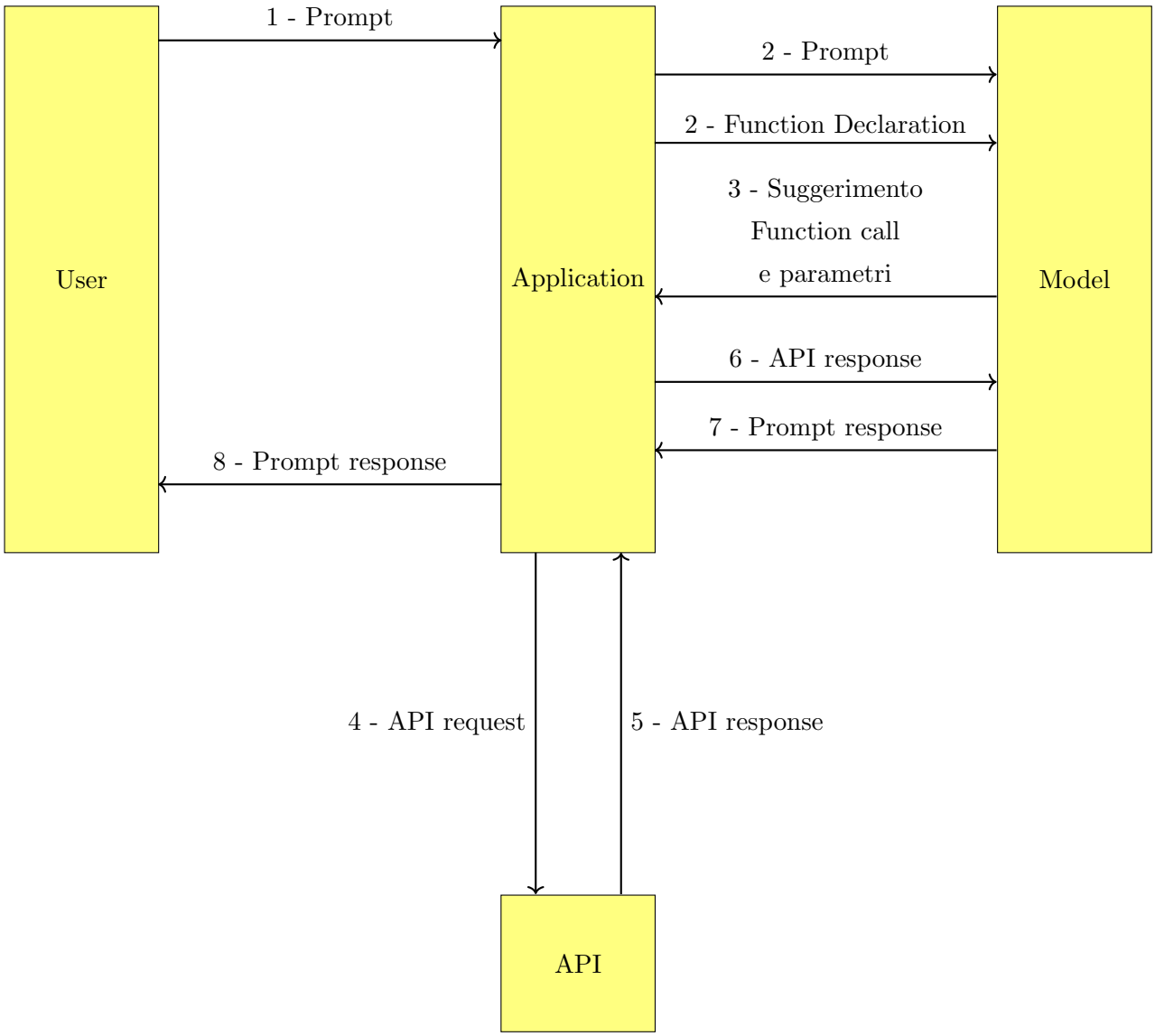


Figura 3.7: Flusso di lavoro Function Call

Valutazioni personali

Ho potuto notare che questo approccio innovativo delle Function call, con la suddetta suddivisione in fasi, ci ha consentito di isolare le diverse operazioni. Questo ha permesso un controllo maggiormente preciso sui vari processi, ad esmepio l'interazione con le API esterne e la ricezione dei risultati. Questa suddivisione di compiti ci ha permesso di capire dove si trovavano i problemi all'interno del flusso di lavoro.

Inoltre uno dei principali punti di forza delle Function call è la loro modularità. Ognuna di esse è progettata per eseguire un compito specifico. come il recupero del database o la creazione di query SQL, questo rende il codice estremamente modulare e facilmente riutilizzabile. Quest'ultimo concetto riprende il punto precedentemente spiegato, non solo si può andare a testare singolarmente le funzioni, rendendo il debug meno oneroso in termini di tempo, ma il codice è estremamente riutilizzabile. Questo fa sì che la struttura del codice consenta una manutenzione efficiente.

Concludendo le function call rappresentano un modo efficiente di gestire le risorse, in quanto per-

mettono di eseguire solo le operazioni necessarie al momento opportuno. il sistema invoca le function call solo quando richiesto evitando sprechi di risorse e migliorando la scalabilità del sistema.

Complessivamente, in termini di valutazione personale, ho aprezzato molto l'uso dell function call all'interno dell'applicativo, sopratutto per la loro modularità e facilità di debug dovuta alla suddisivione dei compiti tramite le diverse funzioni.

3.6.3 Implementazione Function call in SqlTalk

In questa sezione verrà approfondito SqlTalk, con l'analisi di parti di codice per comprendere il funzionamento e l'utilizzo delle Function Call di Gemini all'interno dell'applicativo.

Il primo passo che è stato effettuato nella costruzione dell'applicativo è stato definire le quattro funzioni di chiamata API di gemini.
Queste quattro funzioni definiscono la metodologia operativa del applicativo permettendo di strutturare Query SQL.

- **List dataset func**: Un comando BQ per elencare i set di dati disponibili in un determinato progetto Google Cloud

- **List tables func**: Un comando BQ per elencare le tabelle all'interno di un set di dati

- **Get table func**: Un comando BQ per ottenere dettagli su una tabella all'interno di un set di dati (ad esempio, campi e tipi di dati).

- **Sql query func**: Una query SQL da inviare a BQ per conoscere i dati e rispondere alla domanda dell'utente

Dopo aver definito le funzioni cardine dell'applicativo le racchiudiamo all'interno di un tool. Quest'ultimo verrà passato a gemini all'inizio di ogni chat con SqlTalk da parte di un utente, in modo tale che vengano usate sempre per formulare la risposta al prompt dell'utente.
Per gestire la chiamata di funzione all'interno dell'applicativo abbiamo usato la logica del if per determinare quale funzione gemini decide di chiamare. La logica viene rappresentata dalla seguente immagine:

```python
function_calling_in_process = True
while function_calling_in_process:
    try:
        # estrai i parametri dalle function call
        params = {}
        for key, value in response.function_call.args.items():
            params[key] = value

        if response.function_call.name == "list_datasets":
            api_response = client.list_datasets()
            api_response = [BIGQUERY_DATASET_ID]
            api_requests_and_responses.append(
                [response.function_call.name, params, api_response]
            )

        if response.function_call.name == "list_tables":
            api_response = client.list_tables(BIGQUERY_DATASET_ID)
            api_response = [SPECIFIC_TABLE_ID_1]
            api_requests_and_responses.append(
                [response.function_call.name, params, api_response]  # Params è vuoto perché non ci sono parametri
            )
```

Figura 3.8: Logica gestione delle chiamate

Una volta ricevute le risposte API ,per ogni function Call che Gemini ha invocato, quest'ultimo restituirà un riepilogo in linguaggio naturale della risposta API che risponde alla domanda dell'utente (in questo caso la Query SQL eseguita su BigQuery), oppure richiederà una successiva chiamata di funzione in modo da poter apprendere ulteriori informazioni ed esplorare i dati in modo più dettagliato. Per determinare il tipo di risposta restituita da Gemini, si è implementato un ciclo while che ispeziona l'oggetto dati strutturati risultante. Se il modello restituisce una chiamata di funzione, possiamo continuare a chiamare le funzioni. Se il modello restituisce una risposta di riepilogo del modello, possiamo interrompere la chiamata alle funzioni e passare la risposta di riepilogo all'utente finale.

Function call SQL Query Func

Di seguito fornisco un immagine del codice (3.9) e del flusso di lavoro (3.10) ad esso associato, al fine di comprendere meglio la Function Call fondamentale dell'applicativo.

```python
sql_query_func = FunctionDeclaration(
    name="sql_query",
    description="Get information from data in BigQuery using SQL queries",
    parameters={
        "type": "object",
        "properties": {
            "query": {"type": "string",
                      "description": "SQL query that will help give quantitative answers to the user's question when run on a BigQuery dataset and table.
        },
        "required": ["query"],
    },
)
```

Figura 3.9: Sql query func all'interno di SqlTalk

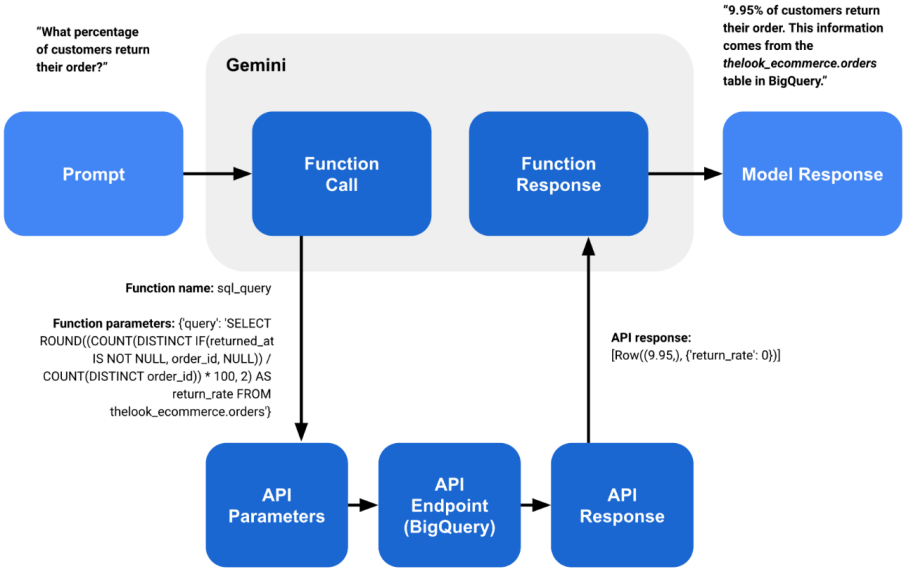


Figura 3.10: Workflow Sql Query Func

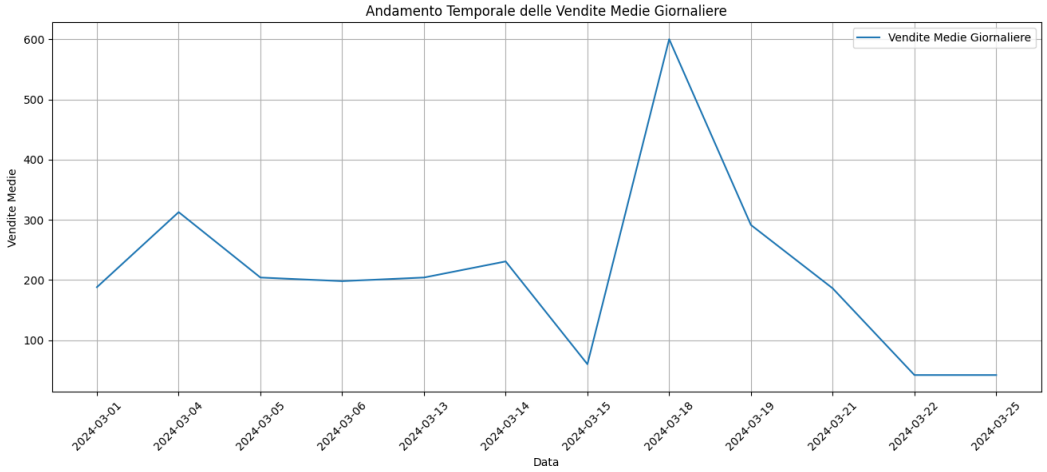
Come si può evincere dall'immagine il prompt dell'utente viene inviato a gemini, dopodiche gemini chiama la function call con i parametri necessari al fine di risolvere la richiesta da parte dell'utente (In questo caso i parametri includono la query SQL). Dopo che la query è stata mandata a BigQuery attraverso un endpoint API, la query viene eseguita e recupera i dati rilevanti. Infine attravero un API response restituisce il risultato della query a Gemini che elaborerà la risposta e la tornerà all'utente sotto forma di linguaggio naturale.

3.7 Esempio di risposta dell'applicativo

Nella seguente sezione fornisco un esempio di risposta dell'applicativo in seguito ad un prompt dell'utente.

```
SQL Talk con BigQuery - Console Version
Powered by Function Calling in Gemini
Scrivi 'esci' per uscire.
Sample prompts:
    - Dimmi le vendite per negozio
    - Qual è la quantità totale venduta il 11/03/2024
    - Dimmi la quantità Totale Venduta
    - Dimmi la quantità Totale Venduta da 20524032109995
    - Articoli Venduti da CARD_BARCODE: 2000013788504
    - Data vendita 12345PANTALONE01
    - Dimmi quale è l'articolo più venduto
    - Dimmi quale è l'articolo con il prezzo unitario più alto
    - Dimmi quale è l'articolo con il prezzo unitario più basso
    - Dimmi l'importo totale delle vendite
    - Dimmi la categoria di articolo più venduta
    - Dimmi quale è l'articolo più venduto per negozio
    - Dimmi la categoria di articolo con il prezzo unitario medio più alto
    - Dimmi la categoria di articolo con il prezzo unitario medio più basso

Company ID:  SVIL2
Data di inizio (DD-MM-YYYY):  1-03-2024
Data di fine (DD-MM-YYYY):  1-04-2024
Chiedimi qualcosa...  dimmi l'andamento temporale delle vendite medie  giornaliere e fammi il grafico
L'utente ha richiesto la creazione di un grafico.
```




```
SQL TALK:
  L'andamento temporale delle vendite medie giornaliere è il seguente:

2024-03-01: 188.00
2024-03-04: 312.67
2024-03-05: 204.00
2024-03-06: 198.00
2024-03-13: 204.00
2024-03-14: 230.74
2024-03-15: 60.00
2024-03-18: 600.00
2024-03-19: 291.33
2024-03-21: 186.03
2024-03-22: 41.98
2024-03-25: 41.98
```

Come si può notare dalla risposta dell'applicativo, in base al prompt dell'utente capisce o meno se è stato richiesto un grafico sulla base dei risultati della query SQL, dopodichè esegue in background tutto il flusso di lavoro spiegato in precedenza (Function Call, Common Table Expression) e restituisce all'utente la risposta al prompt scritto inizialmente, sia sotto forma testuale che visiva.

4

Risultati confronti LLM: Google Gemini 1.5 Flash e GPT 3.5 Turbo

L'applicativo SqlTalk è il risultato dell'evoluzione di una precedente soluzione software. Una delle principali modifiche è stata la transizione dall'utilizzo di GTT-3.5 Turbo a Google Gemini-1.5-Flash come LLM. Il precedente applicativo presenteva delle limitazioni sia in termini di costo che di velocità di risposta,inoltre la gestione dei dati avveniva attraverso snapshot (istantanee) del database, limitando la capacità di fornire agli utenti i dati più aggiornati in tempo reale.

4.1 GPT-3.5 Turbo

Il vecchio applicativo faceva uso di GPT-3.5 Turbo come LLM per eseguire query sul database, ma lavorava con istantanee del database per fornire le risposte agli utenti. Questo significava che, una volta creata un'istantanea, ogni successiva richiesta nella stessa sessione di chat si basava su quella stessa istantanea, impedendo l'accesso ai dati più recenti. Di conseguenza, se i dati del database venivano aggiornati durante la sessione, l'utente non poteva visualizzarli fino a quando non si apriva una nuova sessione o veniva aggiornata l'istantanea. Questa limitazione riduceva l'efficacia dell'analisi dei dati, specialmente in contesti che richiedono aggiornamenti continui. Inoltre GPT-3.5 TURBO, pur essendo un modello efficace, comportava costi significativi non trascurabili per l'azienda:

- **Costo per token di input**: 0.50$ per milione di token
- **Costo per token di output**: 1.50$ per milione di token

Anche in termini di velocità, GPT-3.5 Turbo elaborava circa 79.9 token al secondo e restituiva 100 token di output in circa 1.7 secondi, il che, pur soddisfacente, risultava meno competitivo in ambienti che richiedevano tempi di risposta più rapidi.

4.2 Google Gemini 1.5 Flash

Con lo sviluppo di SqlTalk usando come LLM Google Gemini 1.5 Flash, l'applicativo ha subito un notevole miglioramento sia in termini economici che di efficienza. Il nuovo LLM ha permesso, oltre

che ridurre i costi, di introdurre l'interrogazione dei dati in real-time eliminado le limitazioni imposte dalle istantanee del database. Cosi facendo l'utente che interroga il database, avrà sempre accesso alle informazioni più recenti e accurate. Inoltre i benefici economici di Google Gemini Flash 1.5 sono chiari:

- **Costo per token di input**: 0.08$ per milione di token
- **Costo per token di output**: 0.3$ per milione di token

in termini di prestazione, Gemini 1.5 Flash è in grado di generare 208 token al secondo e restituisce 100 token di output in 0.9 secondi, risultando più rapido e adatto a gestire carichi di lavoro in tempo reale, una caratteristica essenziale per il nuovo applicativo.

In conclusione il passaggio a Google Gemini 1.5 Flash ha rappresentato una svolta positiva per SqlTalk, rendendo il sistema più competitivo e adeguato alle esigenze delle moderne architetture di database, in cui l'accesso in tempo reale e i costi contenuti sono aspetti fondamentali per garantire la scalabilità e l'efficienza

4.3 Grafici di confronto

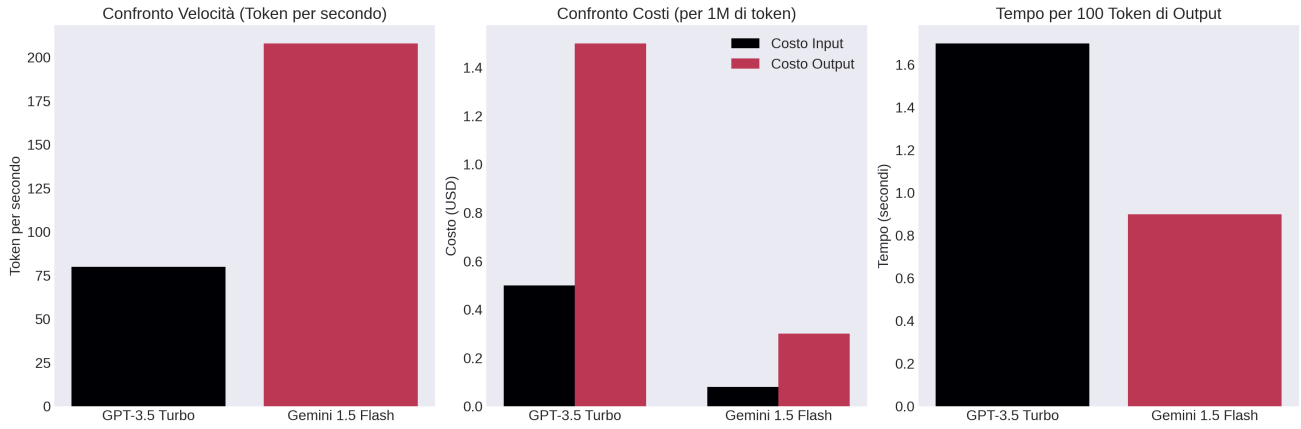


Figura 4.1: Grafici di confronto

5

Conclusioni e lavori futuri

In questa tesi si è affrontato il problema dell'integrazione dell'IA Generativa, e del suo impatto, all'interno del mondo aziendale, in particolare nel campo dell'analisi dei dati. Per fare ciò è stato svillupato un'applicativo in grado di offrire uno strumento più accessibile, versatile e sicuro per l'interrogazione di database aziendali, eliminando la barriera imposta dalla conoscenza tecnica di linguaggi di programmazione come SQL. Questo è stato reso possibile grazie all'integrazione di modelli linguistici avanzati, come Google Gemini, ha permesso di trasformare il linguaggio naturale in query SQL precise, fornendo risposte in tempo reale e garantendo una maggiore efficienza e velocità nell'analisi dei dati.

Uno dei principali aspetti emersi durante lo sviluppo del progetto è stato il confronto tra i diversi LLM, come GPT-3.5 Turbo e Google Gemini 1.5 Flash, in termini di prestazioni e costi. Quest'ultimi sono aspetti rilevanti da tenere in considerazione quando si sviluppa un applicativo, soprattutto per mantenere il rapporto costo-prestazioni il più favorevole possibile. Attraverso una serie di test e comparazioni, si è potuto osservare che il modello di Google offre una maggiore velocità di risposta a prezzi molto più contenuti per token rispetto al modello rivale di OpenAI. Questo risultato ha portato a scegliere Gemini 1.5 Flash come LLM preferito per la costruzione della soluzione software.

In più, oltre a essere cambiato il LLM, l'evoluzione del sistema ha implementato un'altro cambiamento importante basato sul interrogazione del database. Il passaggio da un sistema di interrogazione basato su snapshot del database a un'analisi in real-time, ha di fatto segnato il passo in avanti più importante per l'applicativo. Questa transizione ha risolto uno dei problemi più significativi riscontrati con la versione precedente del software, migliorando notevolmente l'accuratezza e l'affidabilità delle risposte fornite agli utenti.

Anche dal punto di vista della sicurezza, l'applicativo è risultato in linea con gli standard aziendali. l'adozione di tecniche come la parametrizzazione delle query SQL ha garantito una protezione efficace contro gli attacchi di SQL Injection, assicurando che l'accesso ai dati sensibili fosse regolato e limitato solo agli utenti autorizzati. Questo ha reso l'applicativo più robusto, rendendolo idoneno per un ambiente Multi-Tenant.

SqlTalk ha rappresentato, per me, una sfida complessa ma all stesso tempo molto interessante che mi ha permesso di esplorare delle tecnologie che ancora non conoscevo, In più mi ha permesso di capire cosa vuol dire essere immersi all'interno di un progetto di sviluppo software aziendale. Proprio per questo vorrei ringraziare ancora Sinesy S.R.L per l'opportunità concessami potendo lavorare a questo progetto.

Inoltre vorrei ringraziare il mio tutor aziendale Mauro Carniel, per avermi seguito con professionalità e pazienza, trasmettendomi tutta la sua passione e conoscenza in questo campo. Concludendo L'applicativo SqlTalk rappresenta un importante software aziendale da mettere a disposizione per i clienti, il quale offre ad essi un modo potente, sicuro e facile per l'analisi dei dati. Le tecnologie esplorate e implementate durante il progetto aprono la strada a futuri sviluppi, sia in termini di miglioramento delle prestazioni che di ampliamento delle funzionalità offerte, come l'integrazione con nuove tecnologie di AI e ulteriori ottimizzazioni nella gestione dei dati real-time.

5.1 Lavori futuri

Dalla conclusione di questa tesi è emerso un potenziale nuovo progetto che fa uso della tecnologia RAG(Retrieval Augmented Generation), precedentemente citata. Questa tecnologia potrebbe essere il punto cardine per la creazione di chatbot altamente specializzato sui PDF Aziendali, in particolare sulla documentazione dei prodotti software. un applicativo di questo tipo faciliterebbe notevolmente sia le operazioni di manutenzione interna che l'assistenza clienti, offrendo risposte rapide e precise su specifici dettagli tecnici e operativi dei prodotti. Questo approccio migliorerebbe l'efficienza, riducendo i tempi di ricerca e migliorando l'accesso alle informazioni, creando un sistema di supporto più intuitivo e completo.

Bibliografia

[1] Francois Chollet. Building autoencoder in keras. https://blog.keras.io/building-autoencoders-in-keras.html, 2016.

[2] Aurèlien Gèron. Neural networks and deep learning. https://www.oreilly.com/library/view/neural-networks-and/9781492037354/ch04.html, 2024.

[3] Carmen di Nardo. Intelligenza artificiale: le applicazioni che aumentano la produttività aziendale. https://deltalogix.blog/2021/10/13/intelligenza-artificiale-le-applicazioni-che-aumentano-la-produttivita-aziendale/, 2021.

[4] DataMasters. Llm: cosa sono i modelli linguistici di grandi dimensioni. https://datamasters.it/blog/llm-cosa-sono-i-modelli-linguistici-di-grandi-dimensioni/, 2024.

[5] Randall J Erb. Introduction to backpropagation neural network computation. *Pharmaceutical research*, 10:165–170, 1993.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[7] Google. Modelli google. https://cloud.google.com/vertex-ai/generative-ai/docs/learn/models?hl=it, 2024.

[8] Anders Krogh. What are artificial neural networks? *Nature Biotechnology*, 26(2):195–197, 2008.

[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[10] Nanni Bassetti. Gan: cosa sono, applicazioni e vantaggi. https://www.agendadigitale.eu/cultura-digitale/gan-generative-adversarial-networks-cosa-sono-applicazioni-e-vantaggi/, 2023.

[11] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[12] Turing. Fine-tuning llms : Overview, methods, and best practices. https://www.turing.com/resources/finetuning-large-language-models, 2024.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[14] Wikipedia contributors. Generative artificial intelligence. `https://en.wikipedia.org/wiki/Generative_artificial_intelligence`, 2024.

[15] Wikipedia contributors. Large language models. `https://en.wikipedia.org/wiki/Large_language_model`, 2024.

[16] Jure Zupan. Introduction to artificial neural network (ann) methods: what they are and how to use them. *Acta Chimica Slovenica*, 41(3):327, 1994.