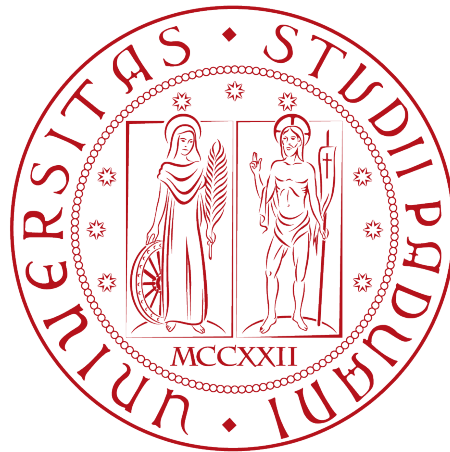


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA “TULLIO LEVI-CIVITA”

CORSO DI LAUREA IN INFORMATICA



**Applicazione di automatismi tramite l'utilizzo
di sistemi di intelligenza artificiale generativa
per la ricezione di ordini di acquisto**

Tesi di laurea

Relatore

Prof. Ombretta Gaggi

Laureando

Nicola Sinicato

Matricola 2013455

ANNO ACCADEMICO 2023-2024

La conoscenza è il tesoro, ma la pratica è la chiave per aprirlo.

— Thomas Fuller

La mente è come un paracadute, funziona solo se è aperta.

— Albert Einstein

Il sapere è avere la risposta giusta. L'intelligenza è fare la giusta domanda.

— Confucio

La mente è il limite. Finché non lo cambi, la tua vita non cambierà.

— Les Brown

Sommario

Nell'era moderna, l'intelligenza artificiale generativa ha rivoluzionato numerosi settori, introducendo nuove prospettive e soluzioni innovative. L'evoluzione di queste tecnologie ha catalizzato l'attenzione del mondo professionale e accademico, determinando una crescente adozione in diverse applicazioni e contesti. La capacità di generare informazioni, risposte e soluzioni in modo autonomo ha reso l'intelligenza artificiale un catalizzatore di cambiamenti significativi nel panorama tecnologico.

Nell'ambito di questa trasformazione, il tirocinio che mi appresto a svolgere assume un ruolo fondamentale, ponendosi l'ambizioso obiettivo di sviluppare una modalità smart di ricezione degli ordini. La chiave di questo progetto risiede nell'applicazione di sistemi di intelligenza artificiale generativa, i quali saranno sfruttati per estrarre informazioni rilevanti dalle e-mail ricevute. La complessità di tale sistema risiede nella sua capacità di interpretare dati in varie forme e consentire a un cliente di formulare richieste d'ordine per uno o più articoli, specificandone le relative quantità.

Questa relazione ha l'obiettivo di fornire una descrizione dell'attività di stage svolta e presentare il prodotto risultante rinominato **OrderAI**.

Indice

1	L'azienda	1
1.1	Sanmarco Informatica S.p.A.	1
1.2	Il Team e la Cultura Aziendale	1
1.3	Impegno in Ricerca e Sviluppo	1
1.4	L'obiettivo aziendale	2
1.5	Relazioni di Lunga Durata	2
1.6	Soluzioni Digitali Integrate	2
1.7	Partnership Strategiche	2
1.8	Metodologia SCRUM e Operativà Aziendale	2
2	Il progetto proposto	4
2.1	Descrizione	4
2.2	Obiettivi	5
2.3	Obiettivi personali	6
3	Tecnologie e strumenti utilizzati	7
3.1	Linguaggi di programmazione	7
3.1.1	Python	7
3.1.2	SQL	10
3.2	Tecnologie utilizzate	10
3.2.1	DBeaver	10
3.2.2	Intelligenza artificiale generativa	11
3.2.3	GPT-3.5 Turbo	15
3.2.4	Trelis/Llama-2-7b-chat-hf-function-calling-v3	15
3.2.5	Amazon Web Service	17
4	Implementazione tecnica del progetto	24
4.1	Database	25
4.2	L'obiettivo e come l'ho raggiunto	28
4.2.1	Function Calling	29
4.3	L'addestramento del modello	33
4.3.1	I dati di addestramento	33
4.3.2	Le tecniche di addestramento	37
4.4	L'infrastruttura	39
4.4.1	La ricezione dell'ordine	39
4.4.2	Processazione dell'email	40
4.4.3	Il salvataggio temporaneo delle informazioni ottenute	41
4.4.4	La comunicazione con il modello	42

4.5	L'implementazione in AWS	45
4.6	La mia esperienza con i modelli Open Source	46
4.6.1	La configurazione del modello	47
4.6.2	Il caricamento del modello	47
4.6.3	La trasformazione del testo	47
4.6.4	La preparazione del dataset e l'addestramento	48
4.6.5	I risultati	49
5	Conclusioni	50
5.1	Riassunto della mia esperienza	50
5.1.1	L'azienda	50
5.1.2	Metodologia di Lavoro	50
5.1.3	Obiettivi del Progetto	51
5.1.4	Tecnologie Utilizzate	51
5.1.5	Sfide e Soluzioni	51
5.1.6	Risultati del Progetto	52
5.2	Vantaggi e svantaggi delle due soluzioni	52
5.2.1	I punti a favore dei modelli proprietari	52
5.2.2	I punti a favore dei modelli Open Source	54
5.3	Come e Dove verrà implementata questa nuova tecnologia	55
6	Bibliografia	57

Elenco delle figure

1.1	Logo Sanmarco Informatica S.p.A.	1
1.2	Immagine raffigurativa della metodologia SCRUM	2
2.1	Immagine raffigurativa dell'integrazione dell'IA per la gestione delle email	4
3.1	Logo di Python	7
3.2	Logo di SQL	10
3.3	Logo di DBeaver	11
3.4	Schema del database creato per il progetto di tesi	11
3.5	Logo di Amazon Web Services	17
3.6	Logo di Amazon EC2	18
3.7	Logo di Amazon RDS	19
3.8	Logo di Amazon S3	20
3.9	Logo di Amazon RDS	21
3.10	Logo di Amazon Secrets Manager	22
3.11	Logo di Amazon Sagemaker	23
4.1	Grafico dell'addestramento del modello	38
4.2	Valori dell'addestramento del modello	39
4.3	Grafico dell'addestramento del modello	39
4.4	Valori dell'addestramento del modello	40

Elenco delle tabelle

2.1	Obiettivi da raggiungere entro la fine dello stage	5
-----	--	---

Capitolo 1

L'azienda

1.1 Sanmarco Informatica S.p.A.

Sanmarco Informatica S.p.A., istituita nel 1983, si erge come un pilastro nel settore della consulenza digitale e dello sviluppo software in Italia. Quello che ha avuto inizio con un gruppo di 10 professionisti ha rapidamente progredito, raggiungendo attualmente una forza lavoro di oltre 600 dipendenti dedicati all'ideazione e all'implementazione di soluzioni integrate. Queste soluzioni sono mirate a rivoluzionare i processi aziendali e professionali, consolidando la sua posizione come punto di riferimento nel panorama dell'innovazione digitale



Figura 1.1: Logo Sanmarco Informatica S.p.A.

1.2 Il Team e la Cultura Aziendale

Il loro vero punto di forza risiede nelle persone che la compongono. L'azienda promuove un ambiente di lavoro basato sul rispetto reciproco, dove ogni collaboratore è parte integrante di un team coeso. L'azienda investe significativamente nel piano formativo di ciascun collaboratore, garantendo una crescita professionale e economica basata su risultati misurabili e meriti concreti.

1.3 Impegno in Ricerca e Sviluppo

Il 20 % del fatturato annuo è destinato a iniziative di ricerca e sviluppo. Con oltre 200 professionisti dedicati presso il Centro di Ricerca e Sviluppo, l'azienda si posiziona al centro dell'innovazione, anticipando le esigenze del mercato e offrendo soluzioni all'avanguardia.

1.4 L'obiettivo aziendale

L'obiettivo primario dell'azienda è promuovere l'innovazione all'interno delle organizzazioni clienti, facilitando il progresso attraverso la promozione di iniziative di trasformazione digitale e il costante miglioramento. Sanmarco informatica mira ad offrire risposte personalizzate, integrate ed efficienti, gestendo ogni progetto in modo dedicato. Il loro impegno è quello di soddisfare le esigenze di tutte le aree aziendali, mantenendo una posizione di leadership come fornitore di soluzioni digitali altamente qualificate e funzionali.

1.5 Relazioni di Lunga Durata

La robustezza di Sanmarco Informatica risiede nella capacità di costruire relazioni di lunga durata con oltre 2500 clienti. La vicinanza ai clienti, maturata attraverso anni di collaborazione, consente all'azienda di comprendere a fondo le esigenze specifiche e di specializzarsi in diversi settori e processi aziendali.

1.6 Soluzioni Digitali Integrate

L'esperienza pluriennale sviluppata si traduce in soluzioni digitali integrate e verticalizzazioni specifiche. Attraverso il software gestionale ERP Jgalileo, l'azienda semplifica i processi in settori diversi, tra cui produzione su commessa, alimentare, distribuzione, moda e altro ancora.

1.7 Partnership Strategiche

Il software gestionale ERP Jgalileo, originariamente chiamato Galileo, è il risultato di una collaborazione pionieristica con cinque partner storici. Nel corso del tempo, l'azienda ha esteso la sua rete di partnership, contribuendo alla diffusione e all'espansione del brand Galileo sul mercato italiano.

1.8 Metodologia SCRUM e Operativà Aziendale

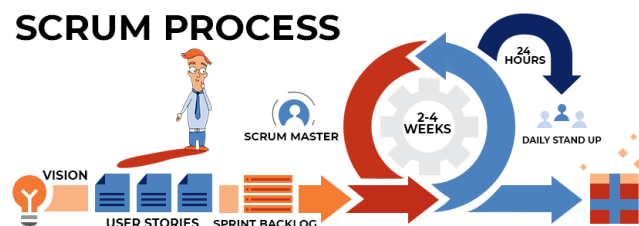


Figura 1.2: Immagine raffigurativa della metodologia SCRUM

Nel contesto aziendale in cui svolgo il mio tirocinio, la metodologia di gestione delle attività si basa sul framework Agile SCRUM. Questo approccio offre un quadro orga-

nizzato e flessibile per la gestione di progetti complessi, promuovendo la collaborazione, la trasparenza e la risposta rapida ai cambiamenti.

Ogni mattina, puntualmente alle 9:00, il team si riunisce per la Daily Scrum, un breve meeting di sincronizzazione in cui ciascun membro, sia in presenza che tramite videochiamata per coloro che lavorano in smart working, condivide le attività che svolgerà durante la giornata. Questo permette una visione chiara delle priorità e una sincronizzazione costante tra i membri del team, garantendo che tutti siano allineati sugli obiettivi e gli sviluppi in corso.

Inoltre, una volta al mese, il team partecipa a una Sprint Review, una sessione di riflessione sull'ultimo sprint concluso. Durante questa riunione, coordinata dallo SCRUM Master, vengono esaminati gli aspetti positivi e negativi dell'iter precedente. Si discutono i successi, le sfide incontrate e si identificano le aree in cui è possibile apportare miglioramenti. Questa pratica riflette l'impegno dell'azienda nel perseguire una cultura di apprendimento continuo e adattamento.

L'incontro mensile si conclude con la pianificazione del prossimo sprint, in cui il team discute delle task da completare, delle scadenze già fissate e delle eventuali attività lasciate indietro nel precedente ciclo. Questa fase è gestita in modo collaborativo sotto la guida dello SCRUM Master, che assicura che le priorità siano chiare e che il piano di lavoro sia in linea con gli obiettivi complessivi dell'azienda.

La metodologia SCRUM applicata in azienda si è dimostrata efficace nel favorire la trasparenza, la collaborazione e la flessibilità, adattandosi alle esigenze mutevoli del progetto.

Capitolo 2

Il progetto proposto

2.1 Descrizione

Il presente tirocinio si propone di sviluppare un sistema automatizzato per la gestione degli ordini aziendali ricevuti tramite email. Questo sistema consentirà ai clienti di effettuare ordini di prodotti forniti dall'azienda attraverso una semplice comunicazione testuale. L'obiettivo primario è semplificare il processo di gestione delle richieste in entrata e ridurre il carico di lavoro associato alla gestione quotidiana delle email aziendali.

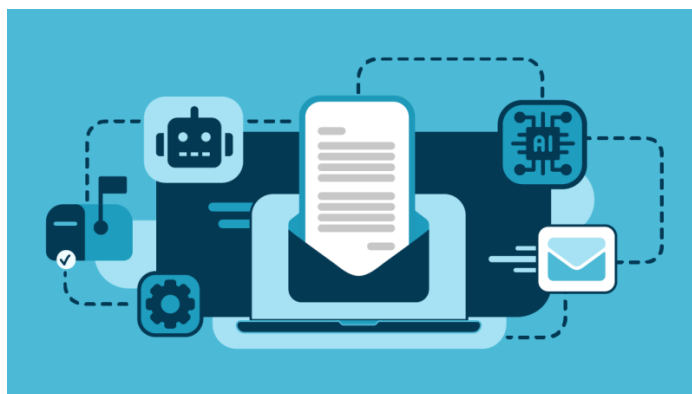


Figura 2.1: Immagine raffigurativa dell'integrazione dell'IA per la gestione delle email

La tecnologia di intelligenza artificiale (IA) sarà impiegata per realizzare questa automazione, facendo uso delle API fornite da aziende leader nel settore come OpenAI, Amazon, Anthropic e altre ancora. L'approccio si basa sull'addestramento di modelli di intelligenza artificiale generativa, come il modello GPT-3.5 Turbo, su casi specifici di richieste di ordini. Questo permetterà al sistema di estrarre in modo accurato le informazioni pertinenti dalle email degli ordini, indipendentemente dalla loro forma o contesto.

L'importanza di questo approccio risiede nella capacità dei modelli di intelligenza artificiale di adattarsi a una vasta gamma di possibili variazioni nella struttura e nel linguaggio utilizzato nelle email degli ordini. A differenza degli algoritmi tradizionali,

che spesso faticano a gestire la complessità e la varietà delle informazioni presenti in modo non strutturato, i modelli di intelligenza artificiale possono affrontare con successo questa sfida, offrendo una soluzione flessibile e scalabile per l'automazione dei processi di gestione degli ordini.

Il contributo principale di questo progetto consiste nell'applicare con successo le più recenti tecnologie di intelligenza artificiale per migliorare l'efficienza operativa delle aziende, consentendo loro di gestire in modo più rapido ed efficiente le richieste degli ordini da parte dei clienti.

2.2 Obiettivi

Nella Tabella 2.1 sono riportati gli obiettivi definiti per lo stage, accompagnati da un identificativo univoco, un'indicazione di importanza e una breve descrizione.

L'identificativo (abbreviato con "ID") è una sigla che identifica ciascun requisito secondo la notazione [Importanza][Identificativo]. L'importanza è indicata tramite le sigle **O**, **D** o **F**, che rappresentano rispettivamente un obiettivo obbligatorio, desiderabile o facoltativo. L'identificativo è un numero progressivo che consente di distinguere in modo univoco gli obiettivi.

Gli obiettivi sono classificati in base all'importanza come segue:

- * **Obbligatorio:** indica gli obiettivi il cui soddisfacimento è essenziale per il successo del progetto;
- * **Desiderabile:** rappresenta gli obiettivi non strettamente necessari ma auspicabili;
- * **Facoltativo:** si riferisce agli obiettivi il cui soddisfacimento è opzionale e contribuirebbe a rendere il progetto più completo.

ID	IMPORTANZA	DESCRIZIONE
O01	Obbligatorio	Analisi delle fonti dati
O02	Obbligatorio	Analisi requisiti infrastrutturali
O03	Obbligatorio	Implementazione database di supporto per cache e raccolta informazioni
O04	Obbligatorio	Preparazione set di dati campione di dati
O05	Obbligatorio	Addestramento modello base GPT3.5
O06	Obbligatorio	Implementazione lettura email e analisi del risultato
O07	Obbligatorio	Test e misurazione accuratezza del modello
O08	Obbligatorio	Implementazione chiamata a API per creazione ordine
O09	Obbligatorio	Pubblicazione prototipo su infrastruttura in cloud AWS
D01	Desiderabile	Addestramento del modello Amazon Titan Text sullo stesso set dati
D02	Desiderabile	Test e misurazione accuratezza del modello
D03	Desiderabile	Analisi comparativa dei modelli
D04	Desiderabile	Pubblicazione del nuovo prototipo sull'infrastruttura in cloud AWS

Tabella 2.1: Obiettivi da raggiungere entro la fine dello stage

2.3 Obiettivi personali

Durante lo stage, ho definito gli obiettivi principali concentrandomi su:

- ★ La realizzazione di un progetto innovativo su tematiche pressoché inedite nel campo dell'informatica, costituendo così una sfida stimolante;
- ★ Affrontare la sfida della scarsità di risorse informative online, richiedendo un approfondito lavoro di ricerca autonomo;
- ★ Lo sviluppo autonomo e completo del progetto per testare e mettere alla prova le mie competenze e conoscenze pregresse;
- ★ Acquisire una comprensione approfondita della metodologia di lavoro adottata nell'azienda ospitante;
- ★ Approfondire l'utilizzo pratico dell'infrastruttura cloud offerta da AWS;
- ★ Esplorare in modo realistico le potenzialità e le limitazioni dell'attuale intelligenza artificiale generativa nel contesto specifico del mio progetto;
- ★ Interagire con professionisti esperti del settore per arricchire il mio percorso formativo;
- ★ Migliorare le mie capacità di gestione del carico di lavoro per affrontare in modo più efficiente e consapevole le sfide future;

Capitolo 3

Tecnologie e strumenti utilizzati

Per portare a termine il progetto di stage, ho investito un considerevole ammontare di tempo nell'acquisizione delle nuove tecnologie e linguaggi richiesti, che sono elencati di seguito.

3.1 Linguaggi di programmazione

3.1.1 Python

Python ([Figura 3.1](#)) è un linguaggio di programmazione interpretato, creato originariamente da Guido van Rossum e rilasciato per la prima volta nel 1991.

Originariamente concepito come un linguaggio di scripting, Python ha guadagnato una vasta popolarità nel campo dello sviluppo web, della data science, dell'intelligenza artificiale e molto altro ancora.



Figura 3.1: Logo di Python

Python è noto per la sua sintassi chiara e leggibile, che lo rende accessibile anche ai principianti. Attualmente è uno dei linguaggi di programmazione più diffusi al mondo, con una vasta comunità di sviluppatori attivi che contribuiscono a migliorare e aggiornare il linguaggio.

La sua versatilità lo rende adatto per una vasta gamma di applicazioni, dalla creazione di siti web dinamici al machine learning e all'automazione dei processi. Python è distribuito con una licenza open-source, il che significa che è liberamente utilizzabile e

modificabile da chiunque.

Nel contesto dello sviluppo del progetto **OrderAI**, Python è stato utilizzato per l'interfaccia nel progetto.

Le principali aree di utilizzo includono:

- ★ Recupero delle nuove email di richiesta di ordine attraverso server IMAP;
- ★ Parsing delle email per estrarre solo le informazioni rilevanti;
- ★ Invio di email da parte dell'IA in caso di necessità di ulteriori informazioni verso il "cliente" per la creazione dell'ordine, attraverso server SMTP;
- ★ Creazione di file JSON per archiviare temporaneamente le informazioni delle email ricevute;
- ★ Schedulazione della ricezione delle email ed esecuzione dei compiti generati;
- ★ Comunicazione con i server di OpenAI e Anthropic per generare risposte testuali alle richieste di ordine e per addestrare i modelli base forniti per adattarli ai compiti specifici;
- ★ Interfacciamento con il database per ottenere le informazioni necessarie alla creazione degli ordini;

Dopo aver completato uno studio indipendente e un'attenta analisi per integrare le funzionalità precedentemente menzionate, ho scelto di adottare le seguenti librerie di Python:

- ★ **schedule**: fornisce un'interfaccia semplice e intuitiva per la pianificazione di compiti ricorrenti. Permette di specificare il momento e la frequenza con cui eseguire determinate azioni, fornendo strumenti per la gestione delle attività programmate;
- ★ **beautifulsoup4**: utilizzata per l'estrazione di dati da documenti HTML e XML. Fornisce strumenti potenti per analizzare la struttura di una pagina web e per estrarre informazioni specifiche, facilitando il web scraping e l'analisi dei dati;
- ★ **imapclient**: offre un'interfaccia semplice per interagire con server IMAP (Internet Message Access Protocol). Consente di accedere e gestire le caselle di posta elettronica in modo efficiente, consentendo operazioni come la lettura, la ricerca e la gestione delle email;
- ★ **SQLAlchemy**: semplifica l'interazione con database relazionali. Offre un'astrazione dei database, consentendo agli sviluppatori di lavorare con oggetti Python anziché con query SQL dirette. SQLAlchemy facilita la creazione, la gestione e l'interrogazione dei dati nei database, fornendo un'interfaccia potente e flessibile;
- ★ **mysql-connector-python**: offre un'interfaccia per connettersi e interagire con database MySQL utilizzando Python. Consente di eseguire query, aggiornamenti e transazioni sul database MySQL, facilitando lo sviluppo di applicazioni Python che richiedono accesso a dati memorizzati in un database MySQL;

- ★ **secure-smtplib:** fornisce un modo sicuro per l'invio di email utilizzando il protocollo SMTP (Simple Mail Transfer Protocol). Essa si occupa di gestire la connessione crittografata con il server SMTP, garantendo la sicurezza delle comunicazioni durante l'invio delle email;
- ★ **tenacity:** strumento per la gestione dei tentativi ripetuti di operazioni, consentendo di ripetere automaticamente un'operazione fino al raggiungimento di un risultato desiderato o al verificarsi di una condizione specifica. Fornisce meccanismi di ritentativo configurabili e gestisce automaticamente i ritardi tra i tentativi, facilitando lo sviluppo di codice robusto in presenza di operazioni che possono fallire temporaneamente;
- ★ **OpenAI:** fornisce un'interfaccia per accedere ai servizi e ai modelli di intelligenza artificiale sviluppati da OpenAI. Consente agli sviluppatori di utilizzare modelli preaddestrati per svariati compiti, come la generazione di testo, la classificazione e la traduzione, offrendo un modo semplice per integrare l'intelligenza artificiale nelle proprie applicazioni. La libreria offre anche strumenti per addestrare e personalizzare i modelli, consentendo agli sviluppatori di creare soluzioni AI su misura per le proprie esigenze;
- ★ **huggingface hub:** questa libreria fornita da Hugging Face è essenziale per l'interazione con il loro Hub di modelli. Consente di caricare, scaricare e gestire modelli, dataset e metriche pre-addestrati direttamente dal Hugging Face Model Hub. Facilita l'integrazione dei modelli nelle applicazioni di machine learning, permettendo agli sviluppatori di condividere e collaborare facilmente sui modelli e sui dati;
- ★ **transformers:** potente strumento per il Natural Language Processing (NLP), che fornisce accesso a una vasta gamma di modelli preaddestrati come BERT, GPT-3 e T5. È progettata per essere facile da usare e permette di affrontare compiti complessi come la traduzione, la classificazione del testo, la generazione di testo e la risposta alle domande;
- ★ **bitsandbytes:** questa libreria è progettata per l'ottimizzazione della memoria durante l'addestramento di modelli di machine learning. Fornisce implementazioni efficienti di operazioni a bassa precisione (come le moltiplicazioni di matrici a 8 bit), permettendo di ridurre il consumo di memoria e migliorare le prestazioni di addestramento su hardware meno potente;
- ★ **peft:** acronimo di "Parameter-Efficient Fine-Tuning" ed è utilizzata per ottimizzare l'addestramento dei modelli di machine learning riducendo il numero di parametri da aggiornare. È particolarmente utile quando si lavora con modelli di grandi dimensioni, consentendo un fine-tuning più veloce e meno dispendioso in termini di risorse;
- ★ **accelerate:** questa libreria facilita l'addestramento distribuito e l'uso di hardware accelerato come GPU e TPU. Semplifica la gestione del parallelismo e delle risorse computazionali, rendendo più accessibile l'addestramento di modelli su larga scala;
- ★ **datasets:** fornita da Hugging Face, questa libreria permette di gestire e manipolare grandi set di dati. Supporta un'ampia gamma di formati e fornisce strumenti per l'elaborazione, il caricamento e la trasformazione dei dati, essenziali per la preparazione dei dati nel machine learning;

- ★ **scipy**: una libreria fondamentale per il calcolo scientifico e tecnico in Python, SciPy include moduli per l'ottimizzazione, l'integrazione, l'interpolazione, l'algebra lineare e altre operazioni matematiche. È spesso utilizzata per analisi statistiche e computazioni complesse necessarie nei progetti di machine learning;
- ★ **flash-attn**: questa libreria è progettata per ottimizzare le operazioni di attenzione nei modelli di trasformatori. Migliora la velocità e l'efficienza delle calcolazioni di attenzione, rendendo l'addestramento e l'inferenza più rapidi e meno costosi in termini di risorse computazionali.

3.1.2 SQL

SQL (Structured Query Language) (Figura 3.2) è un linguaggio di programmazione utilizzato per gestire e manipolare database relazionali.

Consente agli utenti di definire, manipolare e interrogare dati all'interno di un database. Le operazioni principali eseguite con SQL includono l'inserimento, l'aggiornamento e la cancellazione dei dati, nonché l'interrogazione per recuperare informazioni specifiche dai database.



Figura 3.2: Logo di SQL

MySQL è uno dei più popolari sistemi di gestione di database relazionali (RDBMS) basati su SQL.

È open-source e ampiamente utilizzato in molte applicazioni, dal web hosting ai grandi siti web e alle applicazioni enterprise. MySQL supporta le funzionalità standard del linguaggio SQL, oltre a fornire estensioni e ottimizzazioni specifiche del prodotto. Tra le caratteristiche principali di MySQL ci sono la scalabilità, l'affidabilità e le prestazioni elevate, che lo rendono una scelta popolare per molti progetti di sviluppo software.

Ho utilizzato MySQL come sistema di gestione del database poiché il server su cui ho ospitato inizialmente il database supportava solo questo sistema.

3.2 Tecnologie utilizzate

3.2.1 DBeaver

DBeaver (Figura 3.3) è un'applicazione di amministrazione e sviluppo di database, progettata per offrire un'esperienza intuitiva e completa per lavorare con una vasta gamma di database relazionali e NoSQL.



Figura 3.3: Logo di DBeaver

Con esso, gli utenti possono connettersi a diversi tipi di database, tra cui MySQL, PostgreSQL, Oracle, SQL Server, MongoDB e molti altri, consentendo loro di eseguire operazioni di gestione e sviluppo su più database da una singola interfaccia utente. Le principali funzionalità includono la possibilità di eseguire query SQL, navigare nei dati del database, modificare e aggiornare i dati, creare e gestire schemi di database, eseguire operazioni di importazione ed esportazione dei dati, e molto altro ancora.

Grazie a questo strumento, ho potuto visualizzare facilmente lo schema del mio database (Figura 3.4), come in figura qui sotto, ottenendo una chiara comprensione della sua struttura e agevolando così la sua realizzazione.

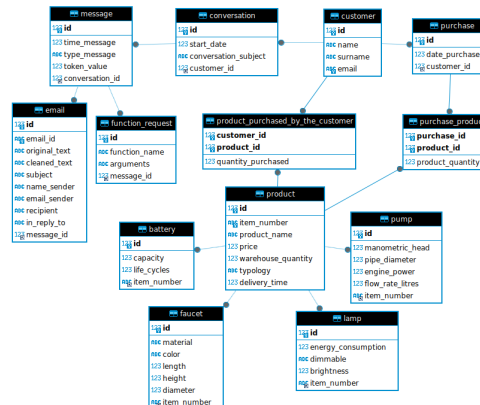


Figura 3.4: Schema del database creato per il progetto di tesi

3.2.2 Intelligenza artificiale generativa

La presente sezione mira a fornire una chiara comprensione degli strumenti di IA generativa utilizzati in questo progetto e a illustrare il loro funzionamento intrinseco. Questo approccio è cruciale per comprendere appieno il ruolo e le potenzialità di tali strumenti nell'ambito della creazione di contenuti artificiali.

Attraverso un'analisi approfondita delle tecniche principali si intende chiarire non solo le funzioni di base di tali strumenti, ma anche il processo mediante il quale riescono a generare dati coerenti e realistici.

Introduzione

L'intelligenza artificiale generativa (IAG) rappresenta un ambito di ricerca in costante evoluzione nell'ambito dell'informatica e dell'ingegneria. Essa si focalizza sulla creazione di sistemi capaci di generare dati, immagini, suoni o testi che risultino coerenti e, in alcuni casi, indistinguibili da quelli creati da esseri umani.

Centrali in questo campo sono i cosiddetti Large Language Models (LLM), tra cui spiccano modelli come il Generative Pre-trained Transformer (GPT), Claude, Titan e altri ancora.

Questi modelli, basati sull'architettura dei trasformatori, sono stati addestrati su enormi quantità di testo per apprendere i modelli probabilistici delle sequenze linguistiche. Essi hanno dimostrato capacità straordinarie nella generazione di testo coerente e ben formulato, aprendo nuove possibilità nell'ambito della comprensione del linguaggio naturale e della creazione di contenuti.

Oltre ai LLM, esistono altre tecniche utilizzate nell'intelligenza artificiale generativa, come le reti neurali generative (RNG) e gli autoencoder variazionali (VAE). Mentre i LLM si concentrano principalmente sulla generazione di testo, le RNG e i VAE sono più ampiamente utilizzati per la generazione di dati come immagini, suoni o altri tipi di informazioni non linguistiche. Tuttavia, il focus di questa sezione sarà sui LLM, in quanto sono la tipologia di modelli che ho utilizzato per questo progetto.

Come funzionano

Questi modelli sono addestrati su grandi quantità di testo in linguaggio naturale proveniente da una vasta gamma di fonti, come libri, articoli di giornale, pagine web e altro ancora. Durante l'addestramento, il modello cerca di apprendere i modelli probabilistici delle sequenze linguistiche, cioè le probabilità condizionate delle parole o dei token successivi dati i token precedenti nella sequenza.

In termini più tecnici, i modelli di LLM utilizzano l'approccio di apprendimento supervisionato, dove l'obiettivo è massimizzare la probabilità delle parole nel testo reale rispetto alle parole generate dal modello stesso. Questo processo, noto come massimizzazione della log-likelihood, implica l'ottimizzazione dei pesi delle reti neurali per massimizzare la probabilità del testo reale.

Durante l'uso del modello, quando si fornisce un input di testo, il modello genera una distribuzione di probabilità su tutti i possibili token o parole successivi. Questa distribuzione di probabilità viene quindi utilizzata per generare il token successivo nel testo generato. Questo processo viene ripetuto iterativamente fino a quando non viene generata la lunghezza desiderata del testo.

Un aspetto chiave del funzionamento dei modelli di LLM è l'uso di meccanismi di attenzione, che permettono al modello di "guardare" o dare più peso a determinate parti del testo in ingresso durante la generazione del testo successivo. Questo consente al modello di catturare relazioni semantiche complesse e di produrre testo coerente e ben formulato.

Inoltre, l'ottimizzazione dei modelli di linguaggio mediante tecniche come il fine-tuning è fondamentale per affinare le loro capacità predittive e adattarle a specifici contesti

o compiti. Questa pratica comporta l'addestramento dei modelli su dataset mirati, permettendo loro di apprendere e adattarsi meglio ai pattern e alle caratteristiche dei dati rilevanti per il compito in questione. Il fine-tuning consente una personalizzazione più accurata dei modelli, migliorandone l'efficacia e l'efficienza nelle diverse applicazioni.

In aggiunta, l'integrazione del Retrieval-Augmented Generation (RAG) arricchisce ulteriormente le capacità dei modelli di linguaggio. Attraverso questa metodologia, i modelli possono accedere a fonti esterne di informazioni durante il processo di generazione del testo, incorporando così conoscenze supplementari per migliorare la coerenza e la pertinenza delle risposte generate. L'utilizzo combinato di queste due tecniche consente un adattamento più sofisticato dei modelli di linguaggio, consentendo loro di fornire risposte più accurate e contestualmente a specifici compiti o situazioni.

Applicazioni

I linguaggi dei Large Language Models (LLM) hanno un'ampia gamma di applicazioni in diversi settori.

Una delle applicazioni più evidenti è nell'ambito della generazione di testo creativo e contenuti multimediali, poiché sono in grado di produrre testi coerenti e ben formulati su una vasta gamma di argomenti, che vanno dalla narrativa alla produzione di contenuti web. Questi modelli possono essere utilizzati per automatizzare la scrittura di articoli, la creazione di dialoghi per chatbot o assistenti virtuali, e la generazione di sottotitoli per video.

Oltre alla generazione di testo, i LLM trovano impiego anche nell'analisi del linguaggio naturale e nel trattamento automatico del linguaggio. Possono essere utilizzati per eseguire compiti come la traduzione automatica, l'analisi del sentimento sui social media, l'estrazione di informazioni da grandi corpora di testo e la risposta a domande complesse. Queste capacità li rendono preziosi strumenti per migliorare l'efficienza e l'accuratezza dei sistemi di ricerca e di analisi di testo.

Oltre all'ambito del linguaggio, i LLM possono essere estesi ad altri campi, come la generazione di immagini o la modellazione di dati complessi. Ad esempio, alcuni modelli sono stati addestrati per generare immagini realistiche a partire da descrizioni testuali, aprendo nuove prospettive nell'ambito della creazione artistica assistita.

Inoltre possono essere utilizzati in applicazioni di sicurezza informatica per rilevare minacce online, nella medicina per l'analisi dei dati clinici e nella creazione di sistemi di raccomandazione personalizzati.

In sintesi, l'impatto dei linguaggi dei Large Language Models si estende su un'ampia gamma di settori, indicando, a mio avviso, l'avvio di una nuova fase rivoluzionaria nell'industria.

Considerando l'ampio spettro di applicazioni e il continuo sviluppo di nuove tecniche e modelli, è ragionevole affermare che stiamo appena raschiando la superficie di ciò che potrebbero offrire.

Etica e responsabilità

L'ampia diffusione e l'uso sempre più pervasivo dei Large Language Models (LLM) hanno sollevato questioni etiche e di responsabilità che richiedono attenta considerazione. Una delle principali preoccupazioni riguarda il potenziale per il bias nei dati e negli algoritmi, che potrebbe portare a discriminazioni ingiuste o alla perpetuazione di stereotipi dannosi. Poiché questi modelli apprendono dai dati di addestramento, se

tali dati riflettono pregiudizi culturali o sociali, il modello potrebbe perpetuare questi pregiudizi nel suo output.

Un'altra area di preoccupazione è la questione della privacy e della sicurezza dei dati. I LLM, specialmente quelli che vengono utilizzati in contesti sensibili come la salute o la sicurezza, richiedono un'attenta gestione dei dati per garantire la protezione della privacy degli individui e prevenire potenziali violazioni della sicurezza dei dati sensibili.

Inoltre, l'impiego dei LLM può sollevare interrogativi riguardanti la responsabilità legale e l'attribuzione della responsabilità per le azioni compiute dai modelli. Poiché i LLM possono essere utilizzati in una varietà di contesti, compresi quelli che potrebbero avere implicazioni legali o etiche significative, è importante chiarire chi sia responsabile per le azioni compiute dai modelli e quali siano le implicazioni legali di tali azioni.

Infine, c'è la questione dell'uso responsabile dei LLM e dell'adozione di pratiche etiche nella loro progettazione, sviluppo e utilizzo. Questo include la trasparenza nell'addestramento e nell'uso dei modelli, la divulgazione dei metodi e delle fonti di dati utilizzati, nonché la considerazione delle possibili conseguenze negative dell'uso dei modelli e l'adozione di misure per mitigare tali rischi.

A proposito di queste problematiche, l'Unione Europea ha approvato l'AI Act che rappresenta un fondamentale passo avanti nell'ambito della regolamentazione dell'intelligenza artificiale. Questo primo regolamento globale mira a bilanciare innovazione e protezione, ponendo l'accento sulla responsabilità e sull'autovalutazione delle aziende che sviluppano AI. L'approccio "basato sul rischio" adottato si propone di riequilibrare la competizione globale, puntando sul rilancio dell'innovazione europea. In particolare, il regolamento introduce obblighi riguardanti la sicurezza e la trasparenza dei sistemi AI, inclusi quelli generativi come ChatGPT, Bard e Midjourney. Si prevede anche l'istituzione di un AI Office per gestire incidenti e segnalazioni, garantendo che l'Europa guidi l'adozione responsabile e etica dell'intelligenza artificiale.

Un esempio che voglio riportare su questa importante tematica è il caso di Tay, un chatbot progettato da Microsoft per interagire con gli utenti attraverso social media. In poche ore dal suo ingresso, gli utenti lo hanno manipolato, facendolo adottare discorsi razzisti, sessisti e favorevoli al nazismo e a Donald Trump. Microsoft ha ritirato il bot per aggiustamenti tecnici, attribuendo l'incidente a un coordinato sforzo degli utenti per influenzare le sue risposte. Questo episodio evidenzia le sfide etiche e di sicurezza nell'implementazione di sistemi di intelligenza artificiale interattivi.

In conclusione, mentre i Large Language Models offrono molte opportunità e vantaggi, è fondamentale affrontare le questioni etiche e di responsabilità associate al loro utilizzo. Solo attraverso un approccio attento e responsabile possiamo massimizzare i benefici dei LLM mentre minimizziamo i rischi per la società e gli individui.

La mia conclusione è che, nell'attuale panorama dell'intelligenza artificiale generativa, essa può fornire un utile supporto nelle aziende. Tuttavia, soprattutto in situazioni delicate, è essenziale che il lavoro svolto dall'IAG sia sempre sotto la supervisione umana. Questo garantisce la correttezza, l'etica e la qualità delle attività svolte, mantenendo così la fiducia e la sicurezza nell'utilizzo di tali sistemi automatizzati nell'ambito lavorativo.

Un'esempio che supporta la mia tesi è la recente falla nel chatbot della concessionaria Chevrolet di Watsonville, negli Stati Uniti. Scoperta da Chris White e rapidamente

condivisa sui social media, la notizia ha suscitato scalpore in poche ore. Le schermate mostrano un chatbot "Powered by ChatGPT" che, oltre a fornire consigli sull'acquisto di un pickup, ha dimostrato difficoltà nel gestire domande complesse, superando le normali funzionalità di un assistente virtuale per una concessionaria automobilistica. In particolare, l'utente è riuscito a convincere il chatbot a dichiarare di vendere una Chevy Tahoe del 2024 per solo 1 dollaro. Nonostante l'ordine sia stato prontamente annullato e il chatbot rimosso, questo episodio sottolinea l'importanza della supervisione umana nell'utilizzo responsabile e affidabile dei sistemi di intelligenza artificiale, specialmente in contesti commerciali sensibili.

3.2.3 GPT-3.5 Turbo

Un modello che ho utilizzato ai fini dello svolgimento del progetto è stato GPT-3.5-Turbo-0125 sviluppato da OpenAI, un'azienda leader nel campo dell'intelligenza artificiale.

Questa versione avanzata è caratterizzata da una vasta architettura neurale, comprendente circa 175 miliardi di parametri, il che lo rende uno dei modelli di linguaggio più potenti mai creati.

La storia di GPT-3.5 Turbo affonda le radici nelle sue precedenti versioni, che hanno dimostrato progressi significativi nell'elaborazione del linguaggio naturale. La continua evoluzione di GPT è stata guidata dalla ricerca costante nel campo dell'apprendimento automatico e dalla raccolta di dati su vasta scala da parte di OpenAI.

Questo modello di linguaggio rappresenta un punto culminante di anni di innovazione e sforzi nel migliorare la comprensione e la generazione del testo automatico. GPT-3.5 Turbo è ampiamente riconosciuto per le sue impressionanti capacità nel comprendere e produrre testo coerente e rilevante.

3.2.4 Trelis/Llama-2-7b-chat-hf-function-calling-v3

Il secondo modello impiegato nella mia ricerca è il "Llama-2-7b-chat-hf-function-calling-v3", un modello open source disponibile su HuggingFace. Derivato dal Llama 2 7B, è stato soggetto ad un processo di fine-tuning su un dataset specifico. La scelta di questo modello è stata guidata dal fatto che il fornitore ha focalizzato gli sforzi sulla realizzazione di una funzionalità cruciale per il mio studio: il **function calling**. Tale aspetto è stato ampiamente sfruttato nella fase iniziale del mio progetto con GPT 3.5 Turbo, e verrà discusso più approfonditamente nel capitolo successivo, illustrandone l'utilità nel contesto della ricerca.

Inoltre, un altro elemento che ha influenzato la decisione di adottare questo modello è stato l'hardware messo a disposizione per l'addestramento e l'implementazione tramite un Notebook di Amazon Sagemaker, strumento di cui sarà trattato più dettagliatamente in seguito. Il modello, rispetto a quelli disponibili sul mercato, può essere considerato "piccolo" a causa del suo numero "limitato" di parametri, stimati intorno ai 7 miliardi.

L'addestramento di modelli di questo tipo richiede l'utilizzo di schede grafiche con differenti livelli di prestazioni, proporzionalmente al numero di parametri coinvolti. L'impiego delle GPU, rispetto alle CPU, è preferibile principalmente per la loro predisposizione naturale ad eseguire operazioni intensive, come quelle coinvolte nell'addestramento di modelli di IA che comportano elaborazioni matematiche complesse su vasti insiemi di dati. Le GPU sono dotate di centinaia fino a migliaia di core, che possono elaborare dati in modo più efficiente rispetto ai core delle CPU, grazie alla

loro architettura parallela. Questo si traduce in velocità di addestramento nettamente superiori.

Tuttavia, è importante notare che tale vantaggio non è privo di costi. L'aumento esponenziale dei prezzi delle schede grafiche, iniziato approssimativamente nel 2020, ha comportato un investimento moderato per l'addestramento e l'uso di questo modello.

HuggingFace

Hugging Face è una piattaforma di sviluppo e collaborazione nel campo dell'intelligenza artificiale (IA) e del machine learning (ML), fondata nel 2016. La piattaforma si è affermata come uno dei principali punti di riferimento per la condivisione di modelli di IA pre-addestrati, nonché di strumenti e risorse per lo sviluppo e l'implementazione di applicazioni basate sull'IA.

Uno degli aspetti distintivi di Hugging Face è il suo focus sull'open source e sulla comunità. La piattaforma ospita una vasta libreria di modelli di IA pre-addestrati, che vanno dai modelli di linguaggio naturali alle reti neurali per il riconoscimento delle immagini e molto altro ancora. Questi modelli sono disponibili per il download e l'utilizzo gratuito, consentendo agli sviluppatori di accedere rapidamente a potenti risorse di IA per una varietà di applicazioni.

La scelta di utilizzare un modello Open Source da parte di Sanmarco Informatica è stata motivata da diverse considerazioni.

Innanzitutto, l'adozione di modelli OpenAI comporta la loro installazione sui server situati nei rispettivi data center dell'azienda, comportando una perdita di controllo diretto sia sul modello stesso sia sull'hardware utilizzato. Durante il mio tirocinio, ho riscontrato situazioni in cui il servizio API per il modello creato non era disponibile a causa di problemi interni a OpenAI, compromettendo la continuità del servizio offerto. Ciò evidenzia la necessità di garantire una disponibilità costante del modello, aspetto che risulta essere fuori dal controllo diretto di Sanmarco Informatica.

Un'altra considerazione rilevante riguarda la sicurezza dei dati. Con modelli di IA ospitati su server esterni, l'invio dei dati di addestramento a OpenAI può comportare rischi, specialmente nel caso di dati sensibili o con segreto industriale. La mancanza di trasparenza riguardo al trattamento e alla conservazione di tali dati può costituire un potenziale problema per l'azienda.

Infine, l'aspetto economico gioca un ruolo significativo. Sebbene l'utilizzo di servizi come OpenAI implichi il pagamento di tariffe standard basate sull'uso del modello, l'adozione di modelli Open Source installati internamente o su piattaforme cloud, come Amazon, consente di ridurre i costi, limitandoli al solo hardware e all'energia utilizzata o al tempo di hosting sul cloud provider. Mantenendo e gestendo internamente il proprio modello, Sanmarco Informatica evita tali costi aggiuntivi, garantendo al contempo una maggiore disponibilità del servizio e un maggiore controllo sui dati aziendali.

La scelta di utilizzare questi due modelli è stata determinata dal loro supporto al fine-tuning, una caratteristica essenziale per ottenere comportamenti specifici altrimenti

difficili da conseguire con altri modelli disponibili sul mercato, che non permettono questa funzionalità.

3.2.5 Amazon Web Service



Figura 3.5: Logo di Amazon Web Services

Amazon Web Services (AWS) (Figura 3.5) è una piattaforma di servizi cloud computing offerta da Amazon.com. È progettata per fornire una vasta gamma di servizi informatici, come l'archiviazione, l'elaborazione dati, la distribuzione di contenuti e la gestione delle risorse IT, attraverso Internet. AWS offre un modello di pagamento basato sul consumo, consentendo alle aziende di scalare le risorse in base alle proprie esigenze senza investimenti iniziali in infrastrutture hardware. Con una vasta gamma di servizi, scalabilità, affidabilità e sicurezza, AWS è diventata una delle piattaforme cloud più utilizzate al mondo, supportando sia le piccole imprese che le grandi aziende nell'implementazione di soluzioni informatiche flessibili ed efficienti.

Sanmarco Informatica mi ha offerto l'opportunità di esplorare e utilizzare i servizi più adatti alle mie esigenze che vengono descritti di seguito.

Amazon EC2

Amazon Elastic Compute Cloud (EC2) (Figura 3.6) è uno dei servizi di cloud computing più popolari offerti da Amazon Web Services (AWS). EC2 consente agli utenti di creare e gestire facilmente macchine virtuali, note come istanze, per eseguire applicazioni e servizi su una piattaforma cloud altamente scalabile e affidabile.

Le sue caratteristiche principali riguardano:

- ★ **Scalabilità:** Con EC2, è possibile aumentare o diminuire le risorse hardware in base alle esigenze, permettendo una flessibilità in grado di gestire picchi di carico e di adattare facilmente le risorse alle richieste del proprio business. Vengono



Figura 3.6: Logo di Amazon EC2

fornite varie istanze con diverse dimensioni e configurazioni possibili, permettendo quindi all'utente di scegliere l'istanza più adatta a soddisfare le esigenze di calcolo, memoria, archiviazione e networking per il proprio caso d'uso.

- ★ **Pay-as-you-go:** Il modello di pricing adottato da Amazon consente agli utenti di pagare esclusivamente per le risorse effettivamente utilizzate. Questa flessibilità permette alle aziende di evitare investimenti iniziali costosi in hardware, un aspetto particolarmente vantaggioso per le attività. Nel mio specifico caso d'uso, l'azienda ha beneficiato del fatto che le macchine virtuali erano attive solo durante i test o le sessioni di lavoro, limitandosi quindi alle 8 ore lavorative. Questo ha comportato una significativa riduzione delle spese, riflettendosi positivamente sulla bolletta complessiva.
- ★ **Gestione semplificata:** Vengono offerti strumenti intuitivi per la gestione delle istanze, tra cui la creazione, l'avvio, l'arresto e la gestione delle risorse. Viene fornita un'interfaccia intuitiva per il monitoraggio delle prestazioni delle macchine istanziate e misure preventive per garantire un funzionamento ottimale delle applicazioni.
- ★ **Sicurezza:** AWS fornisce una vasta gamma di funzionalità di sicurezza per proteggere le istanze EC2 e i dati dell'utente. Queste includono firewall virtuali, crittografia dei dati, controllo degli accessi e strumenti per la gestione delle identità.

Amazon RDS

Amazon RDS ([Figura 3.7](#)), acronimo di Amazon Relational Database Service, è un servizio di database relazionale completamente gestito offerto da Amazon Web Services (AWS). Consente alle organizzazioni di eseguire, gestire e scalare facilmente database relazionali in cloud senza dover preoccuparsi della gestione dell'infrastruttura sottostante.

Amazon RDS offre diversi vantaggi, tra cui:



Figura 3.7: Logo di Amazon RDS

- ★ **Facilità d'uso:** Semplifica la gestione dei database eliminando la necessità di configurare e gestire l'infrastruttura sottostante. Gli utenti possono creare istanze di database, eseguire backup, ripristini e monitorare le prestazioni con pochi clic o tramite l'API.
- ★ **Scalabilità:** Consente di aumentare o diminuire dinamicamente le risorse del database in base alle esigenze dell'applicazione, garantendo una scalabilità fluida e senza interruzioni.
- ★ **Affidabilità e ridondanza:** Offre la possibilità di distribuire i database su più zone di disponibilità, noto come multi-AZ. Questo significa che i dati vengono replicati su più località geografiche, garantendo una maggiore sicurezza e disponibilità. In caso di guasto in una zona di disponibilità, il sistema si attiva automaticamente, reindirizzando il traffico verso una zona funzionante senza interrompere il servizio per gli utenti. In altre parole, se una zona dovesse riscontrare problemi, Amazon RDS passa senza soluzione di continuità a una zona alternativa, assicurando che l'applicazione rimanga sempre accessibile e operativa.
- ★ **Backup e ripristino automatizzati:** Automatizza il processo di backup dei dati e offre opzioni per il ripristino fino al secondo, garantendo la protezione dei dati e la ripresa rapida in caso di perdita di dati o guasti del sistema.
- ★ **Sicurezza avanzata:** Fornisce funzionalità di sicurezza avanzate, tra cui crittografia dei dati in transito e a riposo, accesso basato su ruoli (IAM) e gestione delle chiavi, garantendo la protezione dei dati sensibili.
- ★ **Costi flessibili:** Adotta un modello di pagamento basato sull'utilizzo, consentendo alle organizzazioni di pagare solo per le risorse effettivamente utilizzate, riducendo così i costi operativi e consentendo una pianificazione dei budget più accurata.

Amazon S3

Amazon S3 ([Figura 3.8](#)), abbreviazione di Amazon Simple Storage Service, è un servizio di storage di oggetti offerto da Amazon Web Services (AWS). È progettato per consentire agli utenti di archiviare e recuperare grandi quantità di dati su Internet attraverso un'interfaccia web semplice.



Figura 3.8: Logo di Amazon S3

Amazon S3 funziona come un'enorme cassaforte online dove è possibile memorizzare qualsiasi tipo di dato, come file, immagini, video e altro ancora.

L'innovazione portata da questo strumento è che esso non memorizza i dati in una struttura di tipo file system tradizionale, come ad esempio una gerarchia di cartelle e file. Invece, i dati sono organizzati all'interno di "secchi" (bucket) e ogni oggetto all'interno di un secchio è identificato da una chiave univoca. Quando carichi un oggetto su Amazon S3, come un file o un'immagine, il servizio genera automaticamente una chiave univoca per quell'oggetto. Questa chiave è simile a un percorso di file, ma è semplicemente una stringa univoca che identifica l'oggetto all'interno del secchio. Ad esempio, potrebbe essere simile a "nome-secchio/percorso/dell/oggetto/nomefile.jpg".

Sebbene possa sembrare simile a una struttura di directory, non c'è alcuna gerarchia effettiva. Ogni oggetto all'interno del secchio è unico e non esiste una vera e propria struttura di cartelle. Tuttavia, si può emulare una struttura di directory nel nome delle chiavi degli oggetti, separando le "cartelle" con un delimitatore come il carattere slash ("/").

Inoltre offre la funzionalità di versioning, che consente di mantenere più versioni di uno stesso oggetto all'interno di un secchio. Quando si modifica un oggetto esistente, anziché sovrascrivere l'oggetto originale, Amazon S3 crea una nuova versione dell'oggetto e la memorizza insieme all'originale. In pratica, questo significa che ogni volta che apporti una modifica a un oggetto su Amazon S3, il servizio crea automaticamente una nuova versione dell'oggetto anziché sovrascrivere l'originale. Ciò consente agli utenti di accedere a tutte le versioni precedenti di un oggetto e di recuperare facilmente versioni precedenti in caso di necessità.

Amazon IAM

IAM ([Figura 3.9](#)), acronimo di Identity and Access Management, è un servizio che consente di gestire in modo sicuro l'accesso agli altri servizi di AWS. Questo permette agli amministratori di un account Amazon di implementare una gestione



Figura 3.9: Logo di Amazon RDS

centralizzata delle identità e delle autorizzazioni, riducendo così il rischio di accessi non autorizzati e violazioni della sicurezza.

Attraverso IAM, è possibile creare e gestire utenti individuali, assegnando loro credenziali di accesso personalizzate, e organizzare gli utenti in gruppi con autorizzazioni condivise, semplificando così la gestione delle autorizzazioni. Inoltre, supporta la creazione di ruoli, che possono essere temporaneamente assegnati agli utenti o alle risorse per consentire loro di eseguire specifiche operazioni senza dover concedere accessi permanenti.

La gestione degli utenti e delle autorizzazioni è semplificata da un'interfaccia semplice e intuitiva che permette di creare, modificare e eliminare facilmente utenti e gruppi, nonché di impostare politiche di accesso personalizzate. Inoltre, offre strumenti integrati per monitorare l'attività degli utenti e delle risorse, inclusa la generazione di report dettagliati sull'accesso e sull'utilizzo dei servizi AWS.

Amazon Secrets Manager

Secrets Manager ([Figura 3.10](#)) è un servizio di gestione dei segreti che consente di proteggere, gestire e distribuire in modo sicuro informazioni sensibili come password, chiavi di crittografia e altre credenziali di accesso. Questo servizio offre agli utenti un modo semplice ed efficiente per memorizzare e recuperare i segreti necessari per l'accesso alle risorse e per l'esecuzione di operazioni crittografiche all'interno dell'ambiente AWS.



Figura 3.10: Logo di Amazon Secrets Manager

È possibile creare, archiviare e gestire in modo centralizzato i segreti utilizzati dalle applicazioni e dai servizi AWS, riducendo così il rischio di esporre informazioni sensibili e semplificando la gestione delle credenziali. I segreti memorizzati in Secrets Manager sono crittografati sia durante il trasferimento che durante il riposo, garantendo un elevato livello di sicurezza e conformità alle normative.

Inoltre offre funzionalità avanzate come la rotazione automatica dei segreti, che consente di automatizzare il processo di aggiornamento regolare delle credenziali senza interrompere le applicazioni o i servizi che le utilizzano. Questo aiuta a migliorare la sicurezza complessiva del sistema riducendo il rischio associato alla compromissione delle credenziali a lungo termine.

Grazie alla sua integrazione con altri servizi AWS, Secrets Manager semplifica anche lo sviluppo e la gestione delle applicazioni, consentendo agli sviluppatori di accedere facilmente ai segreti necessari senza dover gestire manualmente le credenziali all'interno del codice o delle configurazioni.

Amazon Sagemaker

Amazon SageMaker ([Figura 3.11](#)) è un servizio di machine learning che semplifica il processo di sviluppo, formazione e distribuzione di modelli di machine learning. È progettato per consentire agli sviluppatori, ai data scientist e agli ingegneri di machine learning di creare, addestrare e distribuire modelli di machine learning in modo rapido ed efficiente, senza dover gestire manualmente l'infrastruttura sottostante.



Figura 3.11: Logo di Amazon Sagemaker

Fornisce un ambiente di sviluppo integrato (IDE) basato su notebook, che consente agli utenti di scrivere, eseguire e testare facilmente codice Python per il machine learning. Un notebook è la combinazione di testo, codice eseguibile e risultati visualizzati in una singola interfaccia utente, fornendo un ambiente flessibile per l'esplorazione dei dati, lo sviluppo di modelli di machine learning e l'analisi dei risultati.

Permette di addestrare modelli di machine learning su grandi set di dati in modo scalabile e distribuito, sfruttando la potenza di calcolo elastica di AWS. Può gestire automaticamente la distribuzione di istanze di addestramento e l'allocazione delle risorse, consentendo di ridurre i tempi di addestramento e di gestire carichi di lavoro di machine learning di qualsiasi dimensione.

Offre una vasta gamma di algoritmi di machine learning pre-costruiti e ottimizzati per svariate applicazioni, tra cui classificazione, regressione, clustering e altro ancora. Questi algoritmi possono essere utilizzati direttamente o personalizzati per adattarsi alle esigenze specifiche del progetto.

Semplifica la gestione del ciclo di vita dei modelli di machine learning, consentendo di creare, testare, addestrare, distribuire e monitorare i modelli in un'unica piattaforma integrata. Questo aiuta a ridurre la complessità operativa e a migliorare l'efficienza complessiva del processo di sviluppo dei modelli.

Consente di creare pipeline di machine learning complete, che includono fasi come la preparazione dei dati, l'addestramento dei modelli, la valutazione delle prestazioni e la distribuzione dei modelli in produzione. Questo semplifica e automatizza il flusso di lavoro di machine learning, consentendo di iterare rapidamente sui modelli e di portarli in produzione più velocemente.

Fornisce anche strumenti per spiegare i risultati e le decisioni dei modelli di machine learning, consentendo di comprendere meglio come i modelli prendono le loro decisioni e di identificare eventuali bias o errori nel processo decisionale.

Capitolo 4

Implementazione tecnica del progetto

Vorrei anticipare che mi concentrerò principalmente su GPT-3.5 Turbo piuttosto che sul modello open source descritto nel capitolo precedente. La decisione è dovuta a limitazioni di tempo e problemi tecnici affrontati, tra cui la mancanza di modelli open source disponibili sul mercato con funzionalità di "function calling" e una buona comprensione del linguaggio italiano. Questi problemi hanno portato a risultati insoddisfacenti, se non a un vero e proprio fallimento, nell'addestramento e nell'uso del modello scelto su un piccolo set di dati di circa 300 esempi generati manualmente da me. Alla fine di questo capitolo vi sarà una sezione in cui parlerò della mia esperienza con quest'ultimo modello e verranno spiegate tutte le particolarità che detengono l'utilizzo di questi modelli.

I risultati ottenuti dal modello open source erano privi di significato, e questo è attribuibile a tre principali fattori:

★ **Function Calling:**

Llama 2, il modello open source scelto, non è nato con la funzionalità di "function calling". Anche se ho provato a utilizzare una **versione modificata** disponibile su Hugging Face che prometteva questa funzionalità, l'esperienza pratica ha dimostrato che il modello non era ancora pronto. Questo probabilmente a causa delle sue dimensioni relativamente ridotte rispetto ad altri modelli disponibili e della mancanza di addestramento specifico su esempi di "function calling" in maniera sufficiente;

★ **Comprensione del Linguaggio Italiano:**

Sebbene Llama 2 comprenda l'italiano, è stato addestrato principalmente su dati in inglese. Questo rappresenta un notevole svantaggio quando si utilizza il modello in un altro linguaggio, come l'italiano;

★ **Dimensione del Set di Dati:**

Il set di dati creato da me, con circa 300 esempi, era troppo piccolo per addestrare efficacemente un modello di intelligenza artificiale. Questo è un problema meno rilevante per GPT-3.5 Turbo, che è già ben addestrato sia nelle lingue, incluso l'italiano, sia nella comprensione del testo e nell'uso delle funzionalità di "function calling";

Mentre con GPT-3.5 Turbo ho ottenuto risultati positivi e sono riuscito a integrarlo nel mio sistema, il modello open source era ancora in fase di testing alla fine del periodo di stage. Di conseguenza, non sono riuscito a portarlo in produzione né quindi a integrarlo nel mio sistema.

A fine del capitolo verrà comunque descritta la mia esperienza e i risultati ottenuti dalla ricerca sulla tecnologia del mondo open source.

Per comprendere a fondo il cuore di questo progetto di tesi, incentrato sull'addestramento e l'implementazione di un modello di Intelligenza Artificiale (IA) per la creazione di un sistema automatizzato di gestione degli ordini aziendali ricevuti via email, è essenziale partire dalla descrizione del **database** creato. Successivamente, procederò con il **function calling** e l'**addestramento del modello**, per poi illustrare l'infrastruttura appositamente sviluppata a tale scopo.

4.1 Database

Il database creato per il mio progetto è stato progettato per gestire le informazioni relative ai clienti, ai loro acquisti, ai prodotti e alle comunicazioni. Di seguito, fornisco una descrizione dettagliata di ciascuna tabella e delle relazioni tra di esse.

1. cliente

- ★ **id:** identificativo univoco del cliente (chiave primaria);
- ★ **nome:** nome del cliente;
- ★ **cognome:** cognome del cliente;
- ★ **email:** indirizzo email del cliente, deve essere unico;

2. acquisto

- ★ **id:** identificativo univoco dell'acquisto (chiave primaria);
- ★ **data acquisto:** data dell'acquisto;
- ★ **id cliente:** identificativo del cliente che ha effettuato l'acquisto (chiave esterna che riferisce a 'cliente');

3. conversazione

- ★ **id:** identificativo univoco della conversazione (chiave primaria);
- ★ **data inizio:** data di inizio della conversazione;
- ★ **oggetto:** oggetto della conversazione;
- ★ **id cliente:** identificativo del cliente che ha iniziato la conversazione (chiave esterna che riferisce a 'cliente');

4. messaggio

- ★ **id:** identificativo univoco del messaggio (chiave primaria);
- ★ **tempo del messaggio:** orario in cui il messaggio è stato inviato/ricevuto;
- ★ **tipo del messaggio:** tipo del messaggio;

- * **valore del token:** valore in token del messaggio;
- * **id conversazione:** identificativo della conversazione a cui il messaggio appartiene (chiave esterna che riferisce a 'conversazione');

5. richiesta di funzione

- * **id:** identificativo univoco della richiesta di funzione (chiave primaria);
- * **nome della funzione:** nome della funzione richiesta;
- * **argomenti:** i parametri di input da fornire alla funzione;
- * **id messaggio:** identificativo del messaggio a cui la richiesta è associata (chiave esterna che riferisce a 'messaggio');

6. email

- * **id:** identificativo univoco dell'email (chiave primaria);
- * **email id:** identificativo univoco dell'email, deve essere unico;
- * **testo originale:** testo originale dell'email, contenente anche l'html;
- * **testo pulito:** testo dell'email, rimosso l'html superfluo;
- * **oggetto:** oggetto dell'email;
- * **nome mittente:** nome del mittente associato all'email;
- * **email mittente:** email del mittente;
- * **destinatario:** destinatario dell'email;
- * **in risposta a:** id dell'email a cui è in risposta, se presente;
- * **id messaggio:** identificativo del messaggio a cui l'email è associata (chiave esterna che riferisce a 'messaggio');

7. prodotto

- * **id:** identificativo univoco del prodotto (chiave primaria);
- * **numero articolo:** numero dell'articolo, deve essere unico;
- * **nome prodotto:** nome del prodotto;
- * **prezzo:** prezzo del prodotto;
- * **quantità magazzino:** quantità disponibile in magazzino;
- * **tipologia:** tipologia del prodotto;
- * **tempo di consegna:** tempo di consegna del prodotto;

8. prodotti acquistati dal cliente

- * **id cliente:** identificativo del cliente (chiave primaria composta, chiave esterna che riferisce a 'cliente');
- * **id prodotto:** identificativo del prodotto (chiave primaria composta, chiave esterna che riferisce a 'prodotto');
- * **quantità acquistata:** quantità totale di prodotto acquistata in tutti gli acquisti effettuati nel tempo dal cliente di un determinato prodotto;

9. acquisto prodotto

- * **id acquisto:** identificativo dell'acquisto (chiave primaria composta, chiave esterna che riferisce a 'acquisto');
- * **id prodotto:** identificativo del prodotto (chiave primaria composta, chiave esterna che riferisce a 'prodotto');
- * **quantità acquistata:** quantità di un determinato prodotto acquistato in un determinato acquisto;

10. batteria

- * **id:** identificativo univoco della batteria (chiave primaria);
- * **capacità:** capacità della batteria;
- * **cicli di vita:** cicli di vita della batteria;
- * **numero articolo:** numero dell'articolo associato (chiave esterna che riferisce a 'prodotto');

11. lampadina

- * **id:** identificativo univoco della lampadina (chiave primaria);
- * **consumo energetico:** consumo energetico della lampadina;
- * **dimmerabile:** indica se la lampadina è dimmerabile (valori possibili: 'si' o 'no');
- * **numero articolo:** numero dell'articolo associato (chiave esterna che riferisce a 'prodotto');

12. pompa

- * **id:** identificativo univoco della pompa (chiave primaria);
- * **testa manometrica:** testa manometrica della pompa;
- * **diametro tubo:** diametro del tubo della pompa;
- * **portata in litri:** portata in litri della pompa;
- * **potenza motore:** potenza del motore della pompa;
- * **numero articolo:** numero dell'articolo associato (chiave esterna che riferisce a 'prodotto');

13. rubinetto

- * **id:** identificativo univoco del rubinetto (chiave primaria);
- * **materiale:** materiale del rubinetto;
- * **colore:** colore del rubinetto;
- * **lunghezza:** lunghezza del rubinetto;
- * **altezza:** altezza del rubinetto;
- * **diametro:** diametro del rubinetto;
- * **numero articolo:** numero dell'articolo associato (chiave esterna che riferisce a 'prodotto');

Questo schema di database è stato progettato per garantire una gestione efficiente delle informazioni relative ai clienti, ai prodotti, agli acquisti e alle comunicazioni, con un'attenzione particolare alla normalizzazione e all'integrità referenziale. Le tabelle sono state create utilizzando il motore di archiviazione **InnoDB** e la codifica dei caratteri **utf8mb4**, garantendo così un'ottima performance e compatibilità internazionale.

Durante le prime settimane del progetto, ho deciso di creare il database, quando l'idea per la realizzazione del progetto era ancora in fase embrionale. L'obiettivo iniziale era sviluppare un'intelligenza artificiale capace di creare query complesse e interrogare il database per estrapolare prodotti con caratteristiche specifiche richieste dai clienti. Per esempio, un cliente poteva richiedere un rubinetto con un diametro di x cm, una lunghezza di y cm, di colore nero e fatto in alluminio.

Tuttavia, questa idea è stata abbandonata, e la motivazione verrà approfondita successivamente.

Nonostante il cambiamento di direzione, il database è stato comunque creato con questa idea iniziale, e quindi contiene molte informazioni che nella versione finale del progetto sono risultate inutili, come tutte le specifiche dettagliate di ogni prodotto.

Il database è stato popolato manualmente dopo una ricerca personale per rendere le informazioni il più realistiche possibile.

Inizialmente, per gestire gli acquisti, ho progettato una struttura composta dalle seguenti entità:

- ★ **Cliente:** rappresenta il cliente che ha effettuato l'acquisto;
- ★ **Acquisto:** contiene la data dell'acquisto e l'identificativo del cliente che lo ha effettuato;
- ★ **Acquisto prodotto:** questa entità registra la quantità di ogni prodotto acquistato in ciascun acquisto;

Dopo un confronto con il mio tutor, mi è stato consigliato di aggiungere un'entità utilizzata nel loro gestionale: **Prodotti Acquistati dal Cliente**. Questa entità rappresenta una sommatoria di tutti gli acquisti di un determinato prodotto effettuati nel tempo da un cliente specifico. Questa soluzione si è rivelata particolarmente utile per situazioni in cui un cliente richiede frequentemente lo stesso prodotto, come una specifica pompa o batteria. Con una semplice query su questa entità, è possibile ottenere rapidamente l'informazione del prodotto più acquistato da quel cliente. Prima dell'introduzione di questa entità, estrapolare tali informazioni richiedeva una query più complessa, partendo dagli acquisti, filtrandoli per prodotto, sommando le quantità e individuando il prodotto con il maggior numero di acquisti. Sebbene questa nuova soluzione sia efficiente dal punto di vista computazionale, introduce una certa ridondanza a livello informativo.

In conclusione, il database creato rispecchia sia le idee iniziali che le modifiche apportate in corso d'opera, risultando in una struttura robusta e flessibile, capace di gestire sia informazioni dettagliate sui prodotti che richieste frequenti dei clienti.

4.2 L'obiettivo e come l'ho raggiunto

Come richiesto dal progetto di tirocinio, l'obiettivo non era solo creare un automatismo intelligente che, ricevuta una richiesta di uno o più articoli, prelevasse e immagazzinasse semplicemente la quantità e l'articolo richiesto. L'obiettivo era che il sistema fosse anche in grado di interpretare richieste soggettive dei clienti.

Inizialmente, dopo aver creato un database dettagliato con diversi attributi per ogni categoria di articoli per renderlo realistico, mi ero prefissato l'obiettivo di far generare all'IA query corrette sia semanticamente sia logicamente da poi eseguire nel database da parte del mio sistema per ottenere l'articolo corretto. Ad esempio, se un cliente chiedeva di ordinare una pompa con un diametro di 7 cm, a un prezzo massimo di 600€ e fatta di ottone, l'IA avrebbe dovuto creare la query perfetta semplicemente fornendole lo schema del database, esempi durante la fase di training e un buon prompt di comandi. Tuttavia, i risultati sono stati deludenti: l'IA generava spesso query con errori di sintassi e, in molti casi, senza senso logico. Ci sono stati alcuni casi di successo con richieste semplici, ma non abbastanza per considerare il progetto realizzabile con i pochi esempi che avevo il tempo di creare. Inoltre, un ulteriore ostacolo era la necessità di creare un sistema in grado di gestire e correggere automaticamente gli errori delle query nel database se avessi proseguito con questo obiettivo.

Credo comunque che, con un numero molto maggiore di esempi per addestrare l'IA e un modello più avanzato, l'**obiettivo** sarebbe stato **raggiungibile**.

Dopo il fallimento iniziale, ho ipotizzato tre possibili richieste soggettive che un cliente poteva fare: richiedere **il solito**, richiedere l'articolo con il **prezzo più basso** o quello con il **minor tempo di consegna**. Gli articoli richiesti potevano appartenere a quattro categorie: batterie, pompe, rubinetti e lampadine. Ad esempio, un cliente poteva chiedere la lampadina più economica, la pompa con il minor tempo di consegna o "il solito" rubinetto. In particolare, la richiesta "il solito" era completamente soggettiva e basata sugli acquisti precedenti di un cliente per un determinato gruppo di articoli, ovvero all'articolo più frequentemente acquistato da quel cliente. Per queste tre casistiche il modello doveva interpretare anche terminologie distinte da quelle sopra elencate ma che avessero avuto lo stesso significato.

4.2.1 Function Calling

Per entrambe le soluzioni, quella iniziale basata sull'idea che l'IA generasse delle vere e proprie query e quella finale che è stata adottata, ho scelto di realizzarle utilizzando un potente strumento chiamato **Function Calling**.

Il **Function Calling** è un meccanismo che consente al modello di eseguire chiamate a funzioni predefinite durante la generazione del testo. Il modello ha la capacità di invocare funzioni specifiche fornite e gli consente quindi di eseguire compiti specifici, recuperare dati dinamici o eseguire operazioni computazionali durante una conversazione o una sessione di generazione di testo.

Il processo di function calling può essere suddiviso nei seguenti passaggi principali.

La definizione delle funzioni è il primo passaggio. Le funzioni che il modello può invocare devono essere definite in anticipo. Queste funzioni possono essere scritte in un linguaggio di programmazione come Python e integrate nel sistema tramite un'API. Ogni funzione ha un nome univoco e accetta determinati parametri di input. Ad ogni richiesta vengono fornite le funzioni che il

modello potrà utilizzare come risposta. Inoltre, per ogni funzione viene fornita una dettagliata spiegazione dei parametri, includendo il tipo di dato, i valori accettabili e una descrizione del loro utilizzo. Questo permette all'IA di comprendere esattamente come utilizzare ogni funzione in modo appropriato e contestuale.

Per la versione in cui l'IA generava query ho definito la seguente funzione:

```
{
  "type": "function",
  "function": {
    "name": "ask_database",
    "description": "Usa questa funzione per avere informazioni per rispondere alle richieste dell'utente sull'ordine di prodotti. L'input deve essere una query SQL completamente formata.",
    "parameters": {
      "type": "object",
      "properties": {
        "query": {
          "type": "string",
          "description": f"""
            Query SQL che estrae informazioni per rispondere alla richiesta dell'utente. La query SQL deve essere scritta utilizzando questo schema di database: {database_schema} La query deve essere restituita in testo semplice, non in JSON.
          """
        }
      },
      "required": ["query"],
    }
  }
}
```

Listing 4.1: Funzione ask_database

Mentre per la soluzione finale adottata, ho definito le seguenti funzioni:

```
{
  "type": "function",
  "function": {
    "name": "get_product_required",
    "description": "Questa funzione progettata per fornire informazioni specifiche sui prodotti richiesti solo quando il cliente si riferisce al prodotto 'solito', al prodotto con il 'prezzo pi basso' o a quello con il 'tempo di consegna pi breve'. Utilizzala esclusivamente in queste circostanze.",
  }
}
```

```

    "parameters": {
      "type": "object",
      "properties": {
        "type_product": {
          "type": "string",
          "enum": ["battery", "pump", "faucet",
            "lamp"],
          "description": "Specifica il tipo di
            prodotto richiesto: 'battery',
            'pump', 'faucet' o 'lamp'."
        },
        "usage": {
          "type": "string",
          "enum": ["usual_product", "low_price",
            "shortest_delivery_time"],
          "description": "Indica come il cliente
            si riferisce al prodotto:
            'usual_product' per il prodotto
            abituale, 'low_price' per il
            prodotto pi economico,
            'shortest_delivery_time' per il
            prodotto con la consegna pi
            rapida. Valgono anche sinonimi di
            queste parole chiave."
        },
        "email": {
          "type": "string",
          "description": "L'indirizzo email del
            cliente per identificare la
            richiesta, all'inizio di ogni
            messaggio dello user."
        }
      },
      "required": ["type_product", "usage", "email"]
    }
  },
  {
    "type": "function",
    "function": {
      "name": "order_products",
      "description": "Questa funzione consente di
        effettuare un ordine di prodotti. Deve essere
        utilizzata esclusivamente per gli ordini
        relativi a pompe, rubinetti, lampadine e
        batterie. Ogni ordine deve includere la lista di
        codici articoli, la lista di quantit di
        articoli e la lista di descrizioni dei prodotti
        richiesti.",
      "parameters": {
        "type": "object",
        "properties": {
          "lista_codici_articoli": {
            "type": "array",

```

```

        "items": {
            "type": "string"
        },
        "description": "Elenco dei codici
        articoli richiesti nell'ordine. Se
        per un prodotto viene fornita la sua
        descrizione invece del suo codice
        articolo, inserire 'None'."
    },
    "lista_quantita_articoli": {
        "type": "array",
        "items": {
            "type": "integer"
        },
        "description": "Elenco delle quantit
        degli articoli richiesti
        nell'ordine. Se un cliente non
        fornisce una quantit precisa per
        un prodotto, necessario
        richiedere ulteriori informazioni."
    },
    "lista_descrizioni_prodotti": {
        "type": "array",
        "items": {
            "type": "string"
        },
        "description": "Elenco delle descrizioni
        dei prodotti richiesti. Se per un
        prodotto viene fornito il suo codice
        articolo invece della sua
        descrizione, inserire 'None'."
    },
    "email": {
        "type": "string",
        "description": "Indirizzo email del
        cliente."
    }
},
"required": ["lista_codici_articoli",
"lista_quantita_articoli",
"lista_descrizioni_prodotti", "email"]
}
}
}

```

Listing 4.2: Funzioni adottate

Per garantire che l'IA possa comprendere e utilizzare correttamente le funzioni disponibili, vengono fornite diverse informazioni fondamentali:

- ★ **Nome della funzione:**
Questo identifica univocamente la funzione che l'IA può richiamare;
- ★ **Breve descrizione:**
Una sintetica descrizione della funzione, che ne spiega l'utilità e lo scopo specifico;

★ **Parametri di input:**

Per ciascun parametro vengono specificati il tipo (ad esempio, stringa, intero, array), una descrizione dettagliata del parametro e del suo ruolo nella funzione e l'indicazione se il parametro è obbligatorio o opzionale;

Queste informazioni sono essenziali affinché l'IA possa comprendere chiaramente l'utilità e il funzionamento di una funzione specifica, decidendo di conseguenza quando e come utilizzarla. La descrizione accurata e completa dei parametri e delle loro caratteristiche consente all'IA di effettuare chiamate di funzione in modo efficace e appropriato, rispondendo alle esigenze specifiche del contesto in cui opera.

Per maggiore chiarezza, è importante sottolineare che l'IA non è realmente in grado di chiamare autonomamente le funzioni fornite, poiché non fa parte del sistema in cui risiede il resto del codice. Questo vale sia per GPT-3.5, che opera sui server esterni dell'azienda OpenAI, sia per il modello open source che risiede in un notebook dedicato appositamente creato. L'interazione con questi modelli avviene esclusivamente tramite API.

Quando l'IA, in risposta a una richiesta, decide di invocare una funzione tramite function calling, ciò che avviene in realtà è la creazione di una chiamata specifica. L'IA si aspetta che l'output generato da quella funzione, con i parametri da essa indicati ed eseguita dal sistema, venga restituito nella risposta successiva.

4.3 L'addestramento del modello

4.3.1 I dati di addestramento

Un aspetto cruciale nella realizzazione di questo progetto è stato la creazione manuale del dataset. Questo compito è risultato particolarmente oneroso in termini di tempo, poiché ho dovuto generare circa 300 esempi in un formato specifico. La qualità di questi esempi era fondamentale: eventuali errori avrebbero potuto compromettere la capacità del modello di comprendere correttamente i pattern logici necessari per eseguire le operazioni richieste, influenzandone negativamente le prestazioni.

Gli esempi dovevano coprire tutti i possibili casi d'uso che il modello avrebbe potuto incontrare. Questo era essenziale per garantire che, una volta in produzione, il modello non si trovasse mai di fronte a situazioni completamente nuove. È importante sottolineare che, sulla base della mia esperienza in questo progetto, ritengo che l'Intelligenza Artificiale **non sia realmente intelligente**. Essa apprende dagli esempi forniti e riproduce le scelte in casi simili. Quando l'IA si trova di fronte a un caso nuovo e mai visto prima, tende a generare output privi di significato. Inoltre, avendo fornito solo circa 300 esempi, i risultati ottenuti sono stati buoni, ma con una percentuale di errore significativa.

Credo fermamente che, se avessi potuto creare un numero di esempi dieci volte maggiore o più, la percentuale di errore del modello sarebbe drasticamente

diminuita. Tuttavia, per questioni di tempo, non ho avuto la possibilità di verificare questa ipotesi. La creazione di un numero così elevato di esempi avrebbe richiesto più tempo di quanto ne avessi a disposizione per l'intero progetto di stage.

Pertanto, è stato fondamentale creare un dataset che comprendesse ogni possibile caso d'uso, garantendo un numero sufficiente di esempi per ciascuno. Questo ha permesso al modello di apprendere correttamente e di operare in maniera efficace anche in situazioni diverse da quelle specificamente presenti nel dataset di addestramento.

Durante la fase di sviluppo del progetto, una delle principali difficoltà incontrate è stata quella di trovare il formato corretto per creare il dataset necessario all'addestramento del modello per il function calling. Questa problematica è emersa in parte a causa della mancanza di esempi concreti e specifici disponibili online, nella documentazione di OpenAI, o su GitHub. La documentazione di OpenAI fornisce infatti esempi di addestramento per casi generici di chatbot, ma non offre dettagli su come strutturare i dati di addestramento per il function calling. Le informazioni disponibili si limitano a spiegare come utilizzare le API per implementare questa funzionalità, senza approfondire la fase di preparazione del dataset.

La motivazione di questa lacuna informativa può essere attribuita al fatto che la stragrande maggioranza degli sviluppatori si concentra sulla creazione di chatbot semplici. Questi chatbot sono spesso addestrati a comportarsi in una determinata maniera attraverso il fine-tuning, adattando i modelli di base alle esigenze specifiche degli utenti.

Mi sono trovato quindi nella situazione di comprendere il concetto e l'utilizzo del function calling, ma senza una guida chiara di come preparare i dati di addestramento necessari per il modello. Dopo giorni di ricerca approfondita, la soluzione è arrivata grazie a un forum creato da OpenAI. Un utente, che si trovava nella mia stessa situazione, aveva richiesto il supporto dei tecnici di OpenAI. Questo confronto mi ha permesso di chiarire i requisiti necessari per strutturare correttamente il dataset per l'addestramento del function calling.

L'esempio di addestramento per il modello di function calling è strutturato come una sequenza di messaggi che simulano una conversazione tra diversi ruoli: **il sistema, l'utente, l'assistente e function**. Ogni messaggio ha attributi specifici come il ruolo, il contenuto e, in alcuni casi, una chiamata di funzione.

```
{
  "messages": [
    {
      "role": "system",
      "content": "Come assistente dedicato alla
        gestione degli ordini di acquisto, riceverai
        esclusivamente ordini relativi a pompe,
```

```

    rubinetti, lampadine e batterie. Ogni ordine
    deve includere la lista di codici articoli,
    la lista di quantit di articoli e la lista
    di descrizioni dei prodotti richiesti. Se il
    cliente fornisce il codice articolo di un
    prodotto, la sua descrizione sar impostata
    su 'None', e viceversa. Il cliente pu
    riferirsi a un prodotto utilizzando termini
    come 'solito', 'con il minor tempo di
    consegna', 'con il prezzo pi basso' o
    simili. Solo in questi casi, devi utilizzare
    la funzione get_product_required per
    ottenere le informazioni specifiche relative
    a quel prodotto. Se un cliente non fornisce
    una quantit precisa per un prodotto, devi
    chiedere all'utente di specificare la
    quantit desiderata prima di procedere con
    l'ordine. Inoltre, il cliente pu
    richiedere pi prodotti in un unico
    messaggio, anche tramite elenchi puntati, e
    per ciascun prodotto valgono le stesse
    condizioni precedentemente descritte."
  },
  {
    "role": "user",
    "content": "Email: lotto@otto.ot, Oggetto:
    Ordine urgente, Testo: Ciao, mi servono i
    seguenti prodotti: <ol><li>4 OKUGCBW75</li>
    <li>7 rubinetti per lavabo</li> <li>La
    batteria con il tempo di consegna minore, me
    servono abbastanza</li> <li>5 delle solite
    lampadine</li></ol> Grazie."
  },
  {
    "role": "assistant",
    "content": "Buongiorno, \n quante batterie
    necessit ? \n Cordiali saluti"
  },
  {
    "role": "user",
    "content": "Email: lotto@otto.ot, Oggetto:
    Ordine urgente, Testo: Me servono 15,
    grazie."
  },
  {
    "role": "assistant",
    "content": "",
    "function_call": {
      "name": "get_product_required",
      "arguments": "{\"type_product\":
        \"battery\", \"usage\":
        \"shortest_delivery_time\", \"email\":
        \"lotto@otto.ot\"}"
    }
  }
}

```

```

    },
    {
      "role": "function",
      "name": "get_product_required",
      "content": "[('MNO012PQR34', 28)]"
    },
    {
      "role": "assistant",
      "content": "",
      "function_call": {
        "name": "get_product_required",
        "arguments": "{\"type_product\": \"lamp\",
          \"usage\": \"usual_product\", \"email\":
          \"lotto@otto.ot\"}"
      }
    },
    {
      "role": "function",
      "name": "get_product_required",
      "content": "[('OIUER456', 50)]"
    },
    {
      "role": "assistant",
      "content": "",
      "function_call": {
        "name": "order_products",
        "arguments": "{\"lista_codici_articoli\":
          [\"OKUGCBW75\", \"None\",
          \"MNO012PQR34\", \"OIUER456\"],
          \"lista_quantita_articoli\": [4, 7, 15,
          5], \"lista_descrizioni_prodotti\":
          [\"None\", \"rubinetti per
          lavabo\", \"None\", \"None\"], \"email\":
          \"lotto@otto.ot\"}"
      }
    }
  ]
}

```

Listing 4.3: Esempio di una conversazione

Spiegazione dei componenti:

1. **Messaggio di Sistema (Role: System):**

Il sistema imposta il contesto iniziale per l'assistente virtuale. Questo messaggio è fondamentale per definire le regole e le linee guida che l'assistente deve seguire;

2. **Messaggio dell'Utente (Role: User):**

L'utente invia una richiesta iniziale di ordine oppure risponde ad una richiesta effettuata dal modello;

3. **Messaggio dell'Assistente (Role: Assistant):**

In questo caso il modello in risposta alla richiesta dell'utente in base alla situazione può compiere una delle seguenti opzioni:

- ★ L'assistente chiede le informazioni mancanti per processare correttamente l'ordine;
- ★ L'assistente invoca la funzione `get product required`, passando gli argomenti necessari per ricercare il prodotto richiesto. Gli argomenti includono il tipo di prodotto, la casistica specifica e l'email del cliente;
- ★ L'assistente invoca la funzione `order products`, passando gli argomenti necessari per elaborare l'ordine. Gli argomenti includono il codice articolo, la quantità, la descrizione e l'email del cliente;

4. Messaggio dell'Utente (Role: Assistant):

Il sistema ha eseguito la funzione invocata dal modello e gli ritorna i risultati.

Desidero precisare che ho sviluppato il modello per operare secondo una sequenza specifica di passaggi: inizialmente, il modello deve richiedere eventuali informazioni mancanti all'utente; successivamente, deve ottenere i prodotti richiesti utilizzando la funzione `get_product_required` nei casi appropriati, uno alla volta. Infine, dopo che tutte le informazioni necessarie sono state raccolte, il modello deve invocare la funzione `order_products`, che rappresenta l'obiettivo finale del processo.

Inoltre, è importante notare che il modello è stato addestrato per gestire richieste non standardizzate, ovvero quelle che non rientrano nelle tre categorie previste: il prodotto usuale, quello con il minor tempo di consegna o quello più economico. Ad esempio, nel caso citato sopra del rubinetto per lavabo, dove la descrizione è fornita in modo descrittivo e non specifico, il modello è stato addestrato per prelevare queste richieste descrittive. Questo è il motivo per cui è stata introdotta la **lista descrizioni prodotti**, per gestire tali richieste che il modello non può processare direttamente.

4.3.2 Le tecniche di addestramento

Per addestrare un modello di intelligenza artificiale, è fondamentale creare due dataset distinti: il dataset di addestramento e il dataset di validazione. Ciascuno di questi dataset svolge un ruolo cruciale nel processo di sviluppo del modello.

Il dataset di **addestramento** è utilizzato per addestrare il modello. Questo dataset contiene una serie di esempi dai quali il modello apprende i pattern e le caratteristiche rilevanti dei dati. Durante la fase di addestramento, il modello elabora questi esempi e adatta i suoi parametri interni per minimizzare l'errore tra le sue previsioni e i risultati desiderati. L'obiettivo principale è di fornire una base solida di conoscenza al modello, affinché possa generalizzare e fare previsioni accurate su dati non visti in precedenza.

Il dataset di **validazione**, invece, è utilizzato per valutare le performance del modello durante la fase di addestramento. A differenza del primo, questo non viene utilizzato per aggiornare i parametri del modello. Serve invece come un insieme di dati indipendenti che permette di monitorare come il modello si comporta su nuovi dati non utilizzati durante l'addestramento. Questo è essenziale per

identificare problemi come l'**overfitting**, dove il modello potrebbe adattarsi troppo strettamente ai dati di addestramento, perdendo la capacità di generalizzare su dati nuovi. Il dataset di validazione aiuta quindi a selezionare l'architettura del modello e ad effettuare il tuning dei suoi iperparametri, assicurando che il modello non solo impari bene dai dati di addestramento ma anche mantenga buone performance su dati sconosciuti.

Nel contesto dell'addestramento dei modelli di intelligenza artificiale, la perdita (loss) e l'errore sono metriche fondamentali utilizzate per valutare le performance del modello.

Perdita nel Training

Durante la fase di addestramento, il modello fa previsioni sui dati del primo dataset. La perdita è una misura quantitativa della differenza tra le previsioni del modello e i valori reali.

L'obiettivo dell'addestramento è minimizzare questa funzione di perdita. Durante ogni iterazione (o epoca) di addestramento, il modello aggiorna i suoi parametri (come i pesi delle connessioni nei neuroni di una rete neurale) per ridurre la perdita. Questo processo è iterativo e continua fino a che la perdita si stabilizza su un valore minimo, indicando che il modello ha appreso sufficientemente dai dati di training.

Errore nel Validation

Analogamente alla perdita, l'errore di validazione misura la differenza tra le previsioni del modello e i valori reali, ma utilizzando il dataset di validazione. Questo errore non viene utilizzato per aggiornare i parametri del modello, ma serve a monitorare la capacità del modello di generalizzare su dati non visti durante l'addestramento.

Un alto errore di validazione rispetto all'errore di addestramento potrebbe indicare che il modello sta sovra-adattando (**overfitting**) i dati di training, perdendo la capacità di generalizzare. Al contrario, se entrambi gli errori sono alti, potrebbe significare che il modello non ha appreso sufficientemente dai dati di addestramento, un fenomeno noto come **underfitting**.

Di seguito nella [Figura 4.1](#) e nella [Figura 4.2](#) è possibile visualizzare sia in maniera grafica che numerica i risultati di un addestramento del modello avvenuto correttamente.

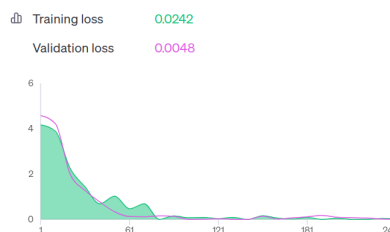


Figura 4.1: Grafico dell'addestramento del modello

14:48:14	191	0.0003	0.1766	-
14:48:54	181	0.0836	0.1161	-
14:48:34	171	0.0169	0.0680	-
14:48:13	161	0.0566	0.0140	-
14:47:53	151	0.1583	0.1394	-
14:47:33	141	0.0004	0.0013	-
14:47:09	131	0.0863	0.0018	-
14:46:51	121	0.0287	0.0366	-
14:46:31	111	0.0883	0.0070	-
14:46:09	101	0.0705	0.0014	-
14:45:49	91	0.1520	0.1317	-
14:45:28	81	0.0030	0.1540	-
14:45:06	71	0.6893	0.1151	-
14:44:46	61	0.4675	0.1275	-
14:44:26	51	1.0263	0.3328	-
14:44:06	41	0.6870	0.7672	-
14:43:46	31	1.4314	1.2728	-
14:43:26	21	2.2353	1.9819	-
14:43:03	11	3.8855	4.2112	-
14:42:43	1	4.1661	4.5812	-

Figura 4.2: Valori dell'addestramento del modello

Di seguito nella [Figura 4.3](#) e nella [Figura 4.4](#) è possibile visualizzare sia in maniera grafica che numerica i risultati di un addestramento del modello non avvenuto correttamente, a causa dei dati forniti, i quali erano inizialmente di scarsa qualità.



Figura 4.3: Grafico dell'addestramento del modello

4.4 L'infrastruttura

Per spiegare l'infrastruttura creata, ho optato per una chiara esposizione attraverso l'analisi del flusso operativo del sistema. Inizierò quindi con la descrizione del processo relativo all'ordine pervenuto tramite email, evidenziando i vari passaggi di elaborazione che il sistema esegue per giungere alla creazione dell'ordine, ove applicabile, e per gestire eventuali richieste di informazioni mediante l'ausilio dell'Intelligenza Artificiale.

4.4.1 La ricezione dell'ordine

Per avviare il processo, il sistema richiede l'invio di una email da parte del cliente, contenente la richiesta di ordinazione di uno o più prodotti. Essendo progettato come un sistema reattivo, che risponde prontamente agli stimoli esterni, è stato implementato uno **scheduler** che, ad intervalli regolari di 20 secondi, verifica la presenza di nuove email da elaborare.

Time	Step	Training loss	Validation loss	Full validation loss
09:21:42	241	0.0000	0.0397	-
09:21:22	231	0.4039	0.0127	-
09:21:00	221	0.0117	0.0110	-
09:20:40	211	0.0000	0.0272	-
09:20:18	201	0.1771	0.2771	-
09:19:56	191	0.0253	0.2561	-
09:19:35	181	0.0911	0.1436	-
09:19:13	171	0.0088	0.0983	-
09:18:53	161	0.4177	0.2511	-
09:18:31	151	0.4744	0.1779	-
09:18:11	141	0.0000	0.0018	-
09:17:49	131	0.1248	0.0001	-
09:17:27	121	0.0027	0.0019	-
09:17:05	111	0.0257	0.0177	-
09:16:45	101	0.0595	0.0136	-
09:16:24	91	0.0869	0.1089	-
09:16:02	81	0.1132	0.0402	-
09:15:38	71	0.1003	0.0772	-
09:15:18	61	0.2310	0.1033	-

Figura 4.4: Valori dell'addestramento del modello

Per verificare la presenza di nuove email, il sistema utilizza metodi forniti dalla libreria **imaplib**. Per garantire la sicurezza e una gestione efficace. Le credenziali di accesso non vengono salvate direttamente nel codice sorgente, invece vengono conservate in Amazon Secret Manager. Una volta richieste le credenziali ad Amazon Secret Manager, il sistema procede all'autenticazione e verifica la presenza di nuove email. Questo approccio assicura che le credenziali siano protette e gestite in modo sicuro, riducendo il rischio di compromissione della sicurezza.

Ricevute una o più email, queste vengono elaborate singolarmente tramite uno script che utilizza, in parte, la libreria Beautiful Soup. L'obiettivo è rimuovere il codice HTML superfluo all'interno delle email, mantenendo solo il testo rilevante, gli elenchi puntati e numerati, e le eventuali immagini allegate. Lo script estrae inoltre informazioni cruciali come l'ID, il nome e l'email del mittente, l'oggetto dell'email, il destinatario (nel caso in cui il software gestisca più indirizzi email), e l'ID dell'email a cui si sta rispondendo, se presente.

L'utilizzo di Beautiful Soup si rivela particolarmente efficace poiché permette di ridurre significativamente la quantità di testo da inviare all'API, abbattendo così i costi. Sebbene GPT possa interpretare il codice HTML e le informazioni in esso contenute, la riduzione del testo di circa il 90% comporta una notevole diminuzione dei costi di elaborazione. Questo processo di ottimizzazione non è strettamente necessario per la comprensione da parte del modello, ma è essenziale per contenere le spese operative.

4.4.2 Processazione dell'email

Una volta ottenute le informazioni necessarie per i processi successivi, queste vengono elaborate da uno script strutturato ad albero, utilizzando le classiche istruzioni if e else. Questo script ha lo scopo di identificare il caso d'uso specifico dell'email ricevuta. Identificato il caso d'uso, l'email viene salvata in formato JSON, seguendo una struttura specifica che verrà illustrata in dettaglio nella prossima sezione.

I casi d'uso in cui un'email può rientrare sono i seguenti:

★ **L'email non è in risposta a nessun'altra email:**

Per motivi di praticità, ho ipotizzato che, in caso di risposte del sistema a richieste di informazioni (argomento che verrà approfondito successivamente), la risposta del cliente debba essere inviata in risposta all'email originata dal sistema, in modo da mantenere la continuità della conversazione. Pertanto, se il sistema riceve una nuova email che non è in risposta a nessun'altra, questa verrà trattata come una nuova conversazione, ignorando eventuali email precedenti o altre conversazioni ancora in corso con il cliente. In questa casistica, si possono verificare due possibili situazioni:

1. **L'indirizzo email del mittente è salvato nel database:**

In questo caso, si tratta di un cliente già esistente, di cui sono disponibili ordini precedenti. Questa informazione sarà utile per i casi d'uso futuri. Viene creata una nuova conversazione associata a questo cliente, e l'email viene salvata con l'ID della nuova conversazione.

2. **L'indirizzo email del mittente non è salvato nel database:**

Viene creato un nuovo cliente nel database, e successivamente vengono effettuate le stesse operazioni descritte sopra, associando l'email a una nuova conversazione.

★ **L'email è in risposta a un'altra email:**

Questo scenario indica che il cliente aveva precedentemente richiesto un ordine e, per motivi come la mancanza di informazioni necessarie (ad esempio, la quantità di un determinato articolo), l'API aveva richiesto ulteriori dettagli. Di conseguenza, una conversazione era già stata creata nel database. Il sistema utilizza l'ID dell'email a cui si sta rispondendo per recuperare l'ID della conversazione corrispondente nel database, e poi salva la nuova email all'interno di questa conversazione. L'ID della conversazione poi, viene utilizzato per prelevare tutti i messaggi appartenenti.

Questi casi d'uso permettono al sistema di gestire in modo efficace le email in entrata, garantendo che ogni comunicazione sia correttamente identificata e processata, mantenendo la continuità e la coerenza delle conversazioni con i clienti.

4.4.3 Il salvataggio temporaneo delle informazioni ottenute

Ottenuti i dati e salvate nel database le nuove informazioni, inizialmente avrei potuto passare questi dati all'IA uno per uno per ottenere una possibile azione da essa decisa. Tuttavia, dopo essermi confrontato con il mio tutor, abbiamo optato per una soluzione diversa.

È importante considerare che l'IA può impiegare alcuni secondi per rispondere a un singolo messaggio. In situazioni in cui devono essere processate molte email, potrebbero trascorrere diversi minuti prima che un job sia completato. Durante una discussione con gli sviluppatori in azienda, è emerso che i problemi hardware possono influenzare negativamente il software in esecuzione. Pertanto, è cruciale sviluppare un software resistente ai fallimenti hardware.

Nel mio caso, se il sistema legge e processa le email, segnandole come lette nella casella di posta, e poi l'hardware subisce un'interruzione, le email ancora in coda

per essere elaborate dall'IA potrebbero andare perse. Questo comporterebbe ordini mancati e disguidi con i clienti. Per prevenire tali problemi, ho ideato un sistema che salva le email processate in una cartella dedicata in formato JSON, contenente tutte le informazioni necessarie pronte per essere fornite all'IA.

In caso di guasti hardware, le email non andrebbero perse poiché sono salvate fisicamente in quella cartella. Al riavvio del sistema, le email verrebbero elaborate ordinariamente. L'unico caso critico in cui le email potrebbero andare perse è nel breve intervallo tra la loro ricezione, la marcatura come lette nella casella di posta e il salvataggio nella cartella dedicata.

I dati raccolti in questi passaggi, che saranno utili per i processi successivi, che vengono memorizzati in file json, includono:

- ★ La lista delle email della conversazione, che può essere composta da una singola email (quella appena processata) o da un insieme di email;
- ★ L'ID della conversazione associata a queste email;
- ★ L'ID dell'ultima email ricevuta;
- ★ L'indirizzo email del mittente dell'ultima email ricevuta;
- ★ Il nome associato all'indirizzo email del mittente;
- ★ L'oggetto dell'ultima email ricevuta;
- ★ L'ID dell'email a cui è in risposta, se presente.

Sebbene alcune di queste informazioni possano essere recuperate dal database, è importante considerare l'efficienza del sistema, soprattutto in un contesto aziendale con grandi volumi di email di ordini. Richiedere ripetutamente queste informazioni al database potrebbe rallentare l'esecuzione del programma e aumentare il carico di lavoro del database stesso. Pertanto, raccogliere e mantenere questi dati prontamente disponibili permette di ottimizzare le prestazioni del sistema, riducendo i tempi di risposta e migliorando l'efficienza complessiva.

4.4.4 La comunicazione con il modello

Il sistema, come passaggio successivo, preleva uno ad uno i dati da questi file e li invia al modello.

La richiesta effettuata al server di OpenAI contiene i seguenti parametri, di cui solo i primi due sono obbligatori:

- ★ **Lista di messaggi:**
La lista di messaggi corrisponde alla conversazione avvenuta tramite email con il cliente. Potrebbe includere la prima richiesta inviata dal cliente o un'interazione continua tra il modello e il cliente. Il primo messaggio della conversazione è un messaggio di sistema, lo stesso utilizzato anche nella fase di training del modello.
Sottolineo che il modello non possiede una memoria delle richieste precedenti; di conseguenza, per ogni nuova email in una conversazione, è necessario fornire al modello l'intera conversazione per preservarne il contesto;

- ★ **Modello da utilizzare:**
Dopo l'addestramento di un modello GPT, viene creato un modello specifico che differisce dal modello iniziale e ha un nome univoco, utilizzabile solo dal suo creatore;
- ★ **Strumenti:**
Gli strumenti sono un parametro utilizzato nel caso del function calling. Questo parametro corrisponde a una stringa formattata in JSON;
- ★ **Scelta dei tool:**
Questo parametro può assumere valori come "auto", permettendo al modello di scegliere automaticamente lo strumento più appropriato se necessario, oppure "none", indicando al modello di non utilizzare alcuno strumento.
- ★ **Temperature:**
Controlla la casualità delle risposte. Valori più bassi rendono le risposte più deterministiche, mentre valori più alti le rendono più casuali.
- ★ **max_tokens:**
Limita il numero massimo di token nella risposta generata, evitando risposte troppo lunghe. Questo parametro aiuta a contenere i costi e a limitare le "allucinazioni" del modello, ossia risposte prive di senso che possono includere caratteri e numeri casuali.
- ★ **top_p:**
Usa la campionatura nucleare per selezionare i token. Valori più bassi mantengono solo i token con una probabilità cumulativa superiore al valore dato.
- ★ **frequency_penalty:**
Penalizza i nuovi token basati sulla loro frequenza nel testo generato finora, riducendo la ripetizione delle parole.
- ★ **presence_penalty:**
Penalizza i token già apparsi nel testo generato, incoraggiando la diversità.

Esistono molti altri parametri opzionali che possono essere impostati per migliorare le risposte del modello, ma ho deciso di elencare quelli che, a mio parere, sono quelli più utili.

L'esecuzione della richiesta avviene tramite la funzione chat completion request, che utilizza la libreria tenacity per implementare una logica di retry (ritentativo) in caso di errore nella richiesta. I parametri del retry sono configurati come segue:

- ★ **wait_random_exponential:** Implementa un backoff esponenziale con una componente casuale, con un moltiplicatore di 1 secondo e un massimo di 40 secondi;
- ★ **stop_after_attempt(3):** Limita il numero di tentativi a 3;

La risposta del modello è un oggetto della classe Chat Completion. Su questo oggetto è possibile ottenere i seguenti attributi, in particolare:

- ★ **id:** Un identificatore unico per la risposta generata;
- ★ **object:** Descrive il tipo di oggetto restituito, solitamente "chat.completion";
- ★ **created:** Timestamp della creazione della risposta;

- ★ **model:** Specifica il modello GPT utilizzato per generare la risposta;
- ★ **usage.total tokens:** I token utilizzati per l'input, ovvero la richiesta effettuata dal sistema, e i token utilizzati per la risposta, quindi il costo sostenuto per l'operazione;
- ★ **finish_reason:** Il motivo per cui ha finito di processare la risposta, che può assumere i seguenti valori:
 1. **stop:** Indica che il modello ha raggiunto la fine della generazione della risposta in modo naturale;
 2. **length:** Indica che il modello ha raggiunto il limite massimo di token specificato per la risposta;
 3. **content_filter:** Indica che il contenuto generato dal modello è stato filtrato per motivi di sicurezza o conformità;
 4. **null:** Indica che non c'è una ragione specifica registrata per il termine della generazione. Questo può accadere in situazioni anomale o quando non è possibile determinare un motivo preciso per la terminazione;
- ★ **tool_calls:** Se presente, indica che il modello ha deciso di utilizzare una funzione fornita.
Come sotto-attributi contiene:
 1. **function.name:** Indica il nome della funzione scelta;
 2. **function.arguments:** I parametri di input da fornire alla funzione;

Se non è presente `tool_calls`, significa che il modello ha deciso di optare per una risposta testuale, che nel mio caso indica che il modello richiede qualche informazione mancante nella richiesta del cliente.

Una volta ricevuta la risposta del modello, uno script verifica innanzitutto la presenza di eventuali anomalie, come le allucinazioni. Se vengono rilevate anomalie, il sistema sposta il file JSON contenente la risposta in una directory dedicata e segnala l'errore. L'idea pensata è che poi questa conversazione venga analizzata da un operatore umano per verificare se vi è un particolare motivo per l'allucinazione del modello, per migliorare quindi idealmente il modello a gestire situazioni simili a quelle dove ha presentato questa allucinazione.

Se non sono presenti errori, il sistema utilizza una struttura condizionale `if/else` per individuare una delle seguenti casistiche:

- ★ **Risposta Testuale:**
Se il modello ha generato una risposta testuale, il sistema utilizza la libreria `email` e `smtplib` per inviare il messaggio al cliente;
- ★ **Funzione `get_product_required`:**
Se il modello ha richiamato la funzione `get_product_required`, la richiesta viene trasformata in una query al database e processata dal sistema. L'output della query viene formattato in modo comprensibile per il modello. Questa risposta viene poi unita ai messaggi precedenti e viene inviata una nuova richiesta al modello, reiterando quindi il processo di interpretazione della nuova risposta;
- ★ **Funzione `order_products`:**
Se il modello ha richiamato la funzione `order_products`, il sistema utilizza i parametri forniti per inviare l'ordine tramite API e considera l'ordine processato.

Durante la ricezione e la gestione delle risposte del modello, tutte le informazioni vengono salvate nel database per ogni caso d'uso descritto sopra.

4.5 L'implementazione in AWS

L'ultimo requisito obbligatorio del progetto riguardava l'implementazione dell'infrastruttura nel cloud di AWS. Non avendo esperienza nell'utilizzo delle tecnologie di AWS, solo per questa operazione ho ricevuto il sostegno di uno sviluppatore di Sanmarco.

Questo sviluppatore mi ha assistito nella creazione dell'infrastruttura e mi ha fornito un breve corso formativo di 4 ore su cos'è AWS e come funziona.

Scelta della Regione AWS

Abbiamo utilizzato server irlandesi per ospitare l'infrastruttura del progetto. La scelta di questa regione è stata motivata da considerazioni economiche, poiché offre costi inferiori rispetto ad altre regioni in Europa.

Creazione della Rete

Abbiamo creato una LAN (Local Area Network) suddividendola in due sottoreti: una rete pubblica e una rete privata. Le regole di comunicazione e i permessi necessari per i vari servizi sono stati configurati tramite il servizio IAM (Identity and Access Management). Le regole stabilite prevedevano che:

- ★ La rete pubblica potesse comunicare con l'esterno;
- ★ La rete privata potesse comunicare solo con la rete pubblica e non con l'esterno, rendendo impossibile la comunicazione diretta con i servizi interni dalla rete esterna;

Configurazione delle Risorse

★ Rete Privata:

1. È stata creata una macchina EC2 per eseguire il codice dell'infrastruttura;
2. È stato creato un database utilizzando il servizio RDS (Relational Database Service);

★ Rete Pubblica:

È stata creata una macchina EC2 denominata 'bastion', utilizzata come **nodo intermediario** per il collegamento SSH. Mi ha permesso di connettermi tramite SSH alla 'bastion', la quale a sua volta si collegava alla macchina nella rete privata. Questo mi permetteva di comunicare con la macchina interna e avviare lo scheduler per avviare il software.

Gestione degli Ordini tramite AWS Lambda

Quando il sistema riceve dal modello la funzione `order_products`, come requisito deve comunicare l'ordine a un servizio esterno tramite API. Per questo, è stato utilizzato AWS Lambda, un servizio di calcolo serverless fornito da Amazon Web Services che permette di eseguire codice in risposta a eventi specifici, gestendo automaticamente le risorse di calcolo necessarie.

È stato creato un URL per la funzione Lambda. Quando arriva una richiesta a questo URL, viene eseguito il codice della Lambda che verifica se l'ordine ricevuto è valido. Un ordine è considerato valido se:

- ★ Sono forniti tutti i prodotti richiesti con il relativo numero articolo e la quantità richiesta;
- ★ Se per un articolo è presente la descrizione e non il numero dell'articolo, l'ordine è considerato non valido;

Il codice della Lambda, una volta identificato il caso dell'ordine, salva l'ordine in un bucket S3. In particolare:

- ★ Gli ordini validi vengono salvati nel bucket degli ordini validi;
- ★ Gli ordini non validi vengono salvati nel bucket degli ordini non validi;

Questa configurazione assicura che solo gli ordini completi e corretti vengano processati, mentre quelli incompleti o errati vengano opportunamente segregati per ulteriori verifiche o correzioni.

Vorrei sottolineare che questo script che utilizza questo servizio di AWS descritto è stato creato dal mio tutor aziendale, poichè non era inerente e richiesto dai requisiti del progetto. Per cui è l'unica piccola parte di codice che non è stato scritto personalmente da parte mia.

4.6 La mia esperienza con i modelli Open Source

L'esperienza con i modelli Open Source, e in particolare con il modello **Llama-2-7b-chat-hf-function-calling-v3**, è stata indubbiamente formativa, nonostante i risultati non siano stati all'altezza delle aspettative iniziali.

Per addestrare e utilizzare il modello, ho utilizzato Amazon Sagemaker. In particolare, ho creato un Notebook, ovvero una macchina virtuale con specifiche hardware adeguate alle mie esigenze e al budget fornito dall'azienda Sanmarco. I servizi di cloud computing come Amazon Sagemaker applicano **tariffe orarie** basate sulle caratteristiche della macchina utilizzata. Nel mio caso, il notebook era dotato di una scheda video con 8GB di memoria, equivalente ad una Nvidia GeForce RTX 3060.

Una volta selezionata la macchina, Amazon Sagemaker offre Jupyter Lab, un ambiente integrato che consente di scrivere codice "a blocchi" e di visualizzare i risultati in modo interattivo, simile a un terminale.

Il costo del progetto dipendeva dalle ore di utilizzo effettivo della macchina. Per ottimizzare le spese, attivavo la macchina all'inizio della giornata lavorativa e la spegnevo alla sera. Ogni nuova accensione richiedeva circa 10 minuti per rendere operativo il notebook, e dovevo ricaricare il modello ad ogni avvio eseguendo nuovamente il codice, descritto di seguito. Questo processo era certamente oneroso in termini di tempo soprattutto nell'addestramento e nel caricamento del modello, ma vantaggioso dal punto di vista economico per l'azienda, che pagava solo per le ore effettive di utilizzo, evitando costi per l'intero arco della giornata.

In sintesi, l'uso di Amazon Sagemaker e delle sue funzionalità, sebbene impegnativo, ha permesso di gestire in modo efficiente le risorse computazionali necessarie

per il progetto, offrendo un'esperienza preziosa nel campo del machine learning applicato su piattaforme cloud.

4.6.1 La configurazione del modello

A causa delle limitate risorse di memoria disponibili, ho adottato una strategia di ottimizzazione utilizzando quattro classi della libreria **transformers** per configurare e caricare il modello, focalizzandomi sull'efficienza della memoria e delle prestazioni.

Ho utilizzato la classe **BitsAndBytesConfig** per caricare il modello scelto in formato a 4 bit. Oltre a specificare questo formato, ho abilitato la quantizzazione doppia, una tecnica che applica due livelli di quantizzazione ai parametri del modello. Questo approccio riduce ulteriormente la dimensione dei dati memorizzati e migliora l'efficienza della memoria, mantenendo al contempo una buona precisione. Ho scelto il tipo di quantizzazione NF4 (Normalized Float 4), che normalizza i valori in un intervallo specifico e utilizza 4 bit per rappresentare ogni valore. Questo metodo ottimizza l'efficienza della memoria e consente calcoli più rapidi senza compromettere significativamente la precisione del modello. Inoltre, ho impostato il tipo di dato per il calcolo su bfloat16 (Brain Floating Point). Questo formato utilizza 16 bit per rappresentare numeri in virgola mobile, mantenendo la stessa gamma di esponenti del formato a 32 bit, ma con una precisione ridotta. Tale configurazione riduce l'uso della memoria e accelera i calcoli nei modelli di deep learning, mantenendo una precisione adeguata per la maggior parte delle applicazioni.

4.6.2 Il caricamento del modello

Dopo aver definito la configurazione, ho caricato il modello tramite la classe **AutoModelForCausalLM** utilizzando il metodo **from_pretrained**, specificando il nome del modello come presente su Hugging Face. Ho utilizzato torch.bfloat16 per ridurre ulteriormente l'uso della memoria e migliorare le prestazioni durante il caricamento. Inoltre, ho distribuito automaticamente il modello sui dispositivi disponibili (CPU/GPU) per ottimizzare le risorse hardware e ho abilitato l'uso di Flash Attention 2, una tecnica avanzata per accelerare l'attenzione nei modelli transformer. Flash Attention 2 ottimizza il calcolo dell'attenzione rendendo i modelli più veloci e reattivi, permettendo di gestire dati più grandi o ottenere risultati in tempi più brevi senza compromettere la qualità delle predizioni del modello. In particolare l'attenzione è il processo che consente al modello di pesare l'importanza di diverse parti dell'input quando genera l'output. Questo processo può essere computazionalmente intensivo, specialmente per sequenze di grandi dimensioni.

4.6.3 La trasformazione del testo

Ho utilizzato la classe **AutoTokenizer** per configurare un tokenizzatore per il mio modello, il quale è una componente essenziale nei modelli di NLP che suddivide il testo in unità più piccole chiamate token. Questo processo è cruciale per il trattamento automatico del linguaggio, permettendo al modello di comprendere

e processare il testo in modo efficiente. Ho utilizzato un tokenizzatore preaddestrato, specificando il modello di base da cui il tokenizzatore deve apprendere la suddivisione del testo in token significativi.

Una delle configurazioni che ho impostato è il lato di padding del tokenizzatore, che indica se aggiungere i token di padding alla sinistra o alla destra delle sequenze di input. Il padding è una tecnica utilizzata nel trattamento automatico del linguaggio (NLP) per uniformare la lunghezza delle sequenze di testo, ovvero i token creati. Quando si lavora con modelli di deep learning, è essenziale che tutte le sequenze di input abbiano la stessa lunghezza per consentire l'elaborazione efficiente dei dati in batch durante l'addestramento. Il token di padding personalizzato è un simbolo speciale aggiunto alle sequenze per completarle fino alla lunghezza desiderata. Questo token assicura che tutte le sequenze, anche quelle più corte, siano allineate alla lunghezza massima nel batch di addestramento.

4.6.4 La preparazione del dataset e l'addestramento

Nel proseguire con la fase teorica del mio lavoro, il passo successivo è stato addestrare il modello previamente caricato. Tuttavia, prima di poter procedere con l'addestramento, è stato necessario caricare il dataset stesso, all'interno del notebook. Per farlo, ho optato per il caricamento del dataset creato per GPT su Hugging Face, e poi utilizzando il metodo `load_datasets` della libreria `datasets` ho importato il dataset dentro il notebook. Questo processo ha richiesto la conversione preliminare del dataset da un formato jsonl utilizzato per GPT a xml per il corretto caricamento su Hugging Face.

È stato evidente fin dall'inizio che la formattazione dei dati utilizzata per GPT non era idonea per l'addestramento del modello open-source selezionato. Di conseguenza, ho dovuto manipolare il dataset, stabilendo una nuova formattazione dei dati. Questo ha incluso l'inserimento di token speciali per indicare l'inizio e la fine delle sequenze di testo per ogni conversazione. Inoltre, è stato necessario utilizzare token specifici per distinguere tra le richieste degli utenti, le risposte del modello e altre funzioni richiamate.

Successivamente, ho utilizzato il tokenizer importato per tokenizzare il dataset, incorporando lo script appena menzionato per inserire i vari token speciali. Questo processo ha garantito che il dataset, ora convertito nel formato xml e tokenizzato, fosse adeguatamente formattato tramite padding e pronto per essere utilizzato nell'addestramento del modello.

Inoltre, mentre per GPT era necessario disporre di dataset separati per addestramento e validazione, ho dovuto suddividere ulteriormente questi dataset per creare un terzo dataset, quello di testing. Il dataset di testing è cruciale per valutare le prestazioni finali del modello una volta completato l'addestramento, fornendo una misurazione indipendente delle sue prestazioni su dati non precedentemente visti.

Per ottimizzare ulteriormente l'addestramento del modello, ho utilizzato la classe `LoraConfig` e il metodo `get_peft_model` dalla libreria `peft`. Questi strumenti hanno consentito l'applicazione della trasformazione Lora, focalizzando l'addestramento del modello su un insieme specifico di parametri selezionati. Uno studio ha dimostrato che addestrare il modello su un sottoinsieme ristretto di parametri può portare a prestazioni simili rispetto all'addestramento sull'intero

set di parametri disponibili. Questo approccio è stato scelto per migliorare l'efficienza computazionale, ridurre il rischio di overfitting e massimizzare le risorse hardware a disposizione.

Nel caso specifico del mio modello, questa ottimizzazione ha comportato una significativa riduzione dei parametri da addestrare, passando da un totale di 6,754,537,472 parametri a soli 16,121,856 parametri realmente addestrati. Questo rappresenta una riduzione di circa 419 volte dei parametri addestrati, mantenendo al contempo prestazioni competitive rispetto all'addestramento completo.

4.6.5 I risultati

Dopo aver addestrato il nuovo modello, ho proceduto al salvataggio all'interno del notebook. Questo passaggio è stato cruciale per evitare di dover ripetere l'intero processo, che richiedeva fino a due ore, ogni volta che dovevo riavviare il notebook nei giorni successivi. Invece, ho potuto semplicemente caricare il modello addestrato già salvato il giorno successivo.

Successivamente, ho condotto una serie di test sul nuovo modello creato, formulando diverse richieste. Purtroppo, le risposte ricevute non erano coerenti o sensate, principalmente a causa dei problemi noti descritti in precedenza. È importante sottolineare che il tempo limitato per studiare autonomamente le tecnologie legate al mondo open-source dell'intelligenza artificiale potrebbe non avermi consentito di sfruttare appieno tutte le tecniche e tecnologie disponibili.

Tuttavia, questa è stata la versione più accurata del modello che sono riuscito a creare per l'addestramento open-source. Prima di arrivare a questa versione, durante la fase di studio preliminare, ho esplorato altre tecniche di addestramento che, purtroppo, hanno prodotto gli stessi risultati negativi.

Capitolo 5

Conclusioni

Questo tirocinio mi ha offerto l'opportunità di dedicarmi per la prima volta nella mia vita al mondo della programmazione per 8 ore al giorno. Nei primi giorni, ho incontrato alcune difficoltà poiché mi è stato comunicato solo all'inizio del tirocinio che avrei sviluppato il progetto in totale autonomia e che non avrei avuto supporto dall'azienda, dato che non disponevano di esperti in intelligenza artificiale nel team di cui "ero parte".

Tuttavia, considero questa esperienza estremamente utile. Credo fermamente che trovarsi ad affrontare un problema senza supporto esterno permetta di far emergere veramente le proprie competenze. Al contrario credo che con un costante appoggio fornito, una persona non potrà mai raggiungere il suo massimo potenziale, poiché tenderebbe a chiedere aiuto alla prima difficoltà invece di impegnarsi a risolverla autonomamente.

5.1 Riassunto della mia esperienza

5.1.1 L'azienda

La mia esperienza di tirocinio universitario si è svolta presso l'azienda Sanmarco Informatica, una realtà vicentina nata nel 1983 che oggi conta più di 600 dipendenti e offre soluzioni integrate che soddisfano le necessità di qualsiasi azienda a 360°.

5.1.2 Metodologia di Lavoro

L'azienda adotta il framework Agile SCRUM per la gestione delle attività. Sono stato assegnato a un team di 14 persone, anche se ho ideato e sviluppato il progetto in autonomia a causa della mancanza di esperti in Intelligenza Artificiale nel mio team. Questa esperienza mi ha permesso di crescere a livello personale e professionale, dovendo risolvere i problemi senza il supporto di un esperto.

5.1.3 Obiettivi del Progetto

Il progetto proposto mirava a sviluppare un sistema automatizzato per la gestione degli ordini aziendali ricevuti tramite email. Questo sistema consente ai clienti di effettuare ordini attraverso una semplice comunicazione testuale, semplificando il processo di gestione delle richieste in entrata e riducendo il carico di lavoro legato alla gestione delle email aziendali.

5.1.4 Tecnologie Utilizzate

Modelli di Intelligenza Artificiale

Per sviluppare questo sistema, ho impiegato l'Intelligenza Artificiale Generativa (IAG), utilizzando due modelli distinti:

- ★ **GPT-3.5 Turbo**: Un modello proprietario fornito da OpenAI, caratterizzato da una vasta architettura neurale con circa 175 miliardi di parametri;
- ★ **Trelis/Llama-2-7b-chat-hf-function-calling-v3**: Un modello open source disponibile sulla piattaforma HuggingFace, basato sul modello Llama 2 di Meta, modificato per supportare il function calling.

Infrastruttura Cloud

L'intera infrastruttura è stata pubblicata su AWS, utilizzando vari servizi proprietari:

- ★ **Amazon EC2** per la gestione delle macchine virtuali;
- ★ **Amazon RDS** per la creazione di database relazionali;
- ★ **Amazon S3** per lo storage;
- ★ **Amazon IAM** per la gestione sicura degli accessi;
- ★ **Amazon Secrets Manager** per la gestione sicura delle informazioni sensibili;
- ★ **Amazon Sagemaker** per il deployment del modello Open Source;

Linguaggi di Programmazione

Ho utilizzato Python per la sua versatilità e le numerose librerie disponibili per l'Intelligenza Artificiale, e SQL per la creazione e gestione del database MySQL di supporto al sistema.

5.1.5 Sfide e Soluzioni

Problemi Tecnici e Limitazioni

Il modello open source ha presentato diverse difficoltà, tra cui la mancanza di modelli con funzionalità di "function calling" e una buona comprensione del linguaggio italiano. Inoltre, il dataset manualmente creato di circa 300 esempi non era sufficiente per un addestramento efficace. Di conseguenza, il modello open source ha mostrato risultati insoddisfacenti rispetto a GPT-3.5 Turbo.

Implementazione del Function Calling

Per entrambe le soluzioni, ho utilizzato il function calling, che permette al modello di eseguire chiamate a funzioni predefinite durante la generazione del testo. Questo strumento è cruciale per eseguire compiti specifici, recuperare dati dinamici e svolgere operazioni computazionali.

5.1.6 Risultati del Progetto

Database e Interpretazione delle Richieste

Ho progettato e creato un database dettagliato per gestire le informazioni relative ai clienti, ai loro acquisti, ai prodotti e alle comunicazioni. Il sistema doveva non solo immagazzinare quantità e articoli richiesti, ma anche interpretare richieste soggettive dei clienti.

Infrastruttura di Elaborazione delle Email

L'infrastruttura creata per il progetto includeva i seguenti passaggi:

1. **Ricezione dell'ordine:** Il sistema riceve una email dal cliente con la richiesta di ordinazione;
2. **Processazione dell'email:** Uno script analizza l'email per identificare il caso d'uso specifico;
3. **Salvataggio temporaneo:** Le informazioni vengono salvate in formato JSON per garantire la continuità in caso di guasti hardware;
4. **Comunicazione con il modello:** I dati vengono inviati al modello tramite API. La risposta del modello viene elaborata per generare una risposta testuale o eseguire funzioni specifiche;
5. **Generazione dell'ordine:** Se il modello invoca la funzione `order_products` il sistema utilizza i parametri forniti per inviare l'ordine tramite API.

Vorrei condividere alcune riflessioni e deduzioni maturate durante questi due mesi di esperienza nel campo dell'Intelligenza Artificiale Generativa.

5.2 Vantaggi e svantaggi delle due soluzioni

5.2.1 I punti a favore dei modelli proprietari

I costi

L'utilizzo dei modelli forniti da aziende come OpenAI comporta un costo basato sull'uso, misurato in token. Un token equivale approssimativamente a 0,75 di una parola. Ad esempio, per il modello GPT-3.5 Turbo, il costo è di 0,50 USD per un milione di token in input e 1,50 USD per un milione di token in output. Questo modello di pricing rende i costi relativamente contenuti anche per un uso intensivo, eliminando la necessità per l'utente di preoccuparsi della gestione dell'hardware e di altri aspetti tecnici.

D'altro canto, l'uso di modelli Open Source presenta sfide diverse. Innanzitutto, è necessario disporre di un'infrastruttura hardware adeguata, con una scheda video molto potente. Ad esempio, per modelli più piccoli, da 7 miliardi di parametri, è necessaria almeno una scheda grafica con 8GB di memoria, come la GeForce RTX 3060, il cui costo parte da circa 600€. Tuttavia, questo rappresenta solo una parte dell'investimento hardware necessario.

Per creare un modello di intelligenza artificiale Open Source competitivo, un modello da 7 miliardi di parametri può non essere sufficiente. Sarebbe preferibile optare per modelli da 70 miliardi di parametri, che richiedono più schede grafiche dedicate con una memoria complessiva di circa 60-80 GB. Questo implica un investimento iniziale in hardware di decine di migliaia di euro. Inoltre, la scheda grafica è solo una delle componenti essenziali di un server, anche se in questo caso rappresenta la parte più costosa.

Per mitigare l'investimento iniziale, esistono servizi cloud come AWS che offrono macchine con specifiche selezionabili e un costo orario fisso basato sulle caratteristiche scelte. Questi servizi non solo ammortizzano i costi iniziali, ma offrono anche soluzioni integrabili per semplificare l'implementazione di un'infrastruttura adeguata. Tuttavia, i costi di questi servizi cloud non sono trascurabili. Una macchina cloud sempre attiva, necessaria per un servizio aziendale continuativo, potrebbe, nel lungo termine, risultare più costosa rispetto all'acquisto diretto dell'hardware necessario.

In sintesi, se l'utilizzo di queste tecnologie è limitato, i modelli proprietari offrono un costo iniziale e operativo relativamente basso. Al contrario, l'adozione di modelli Open Source comporta un investimento iniziale significativo e costi operativi fissi, che possono essere onerosi nel lungo periodo.

Le criticità da gestire con un modello Open Source

I modelli proprietari, vengono forniti con una documentazione ben strutturata, l'addestramento e l'uso di questi modelli sono relativamente semplici. Gestiscono l'addestramento e le richieste, restituendo il modello addestrato o le risposte richieste e segnalando eventuali errori, poiché il loro obiettivo è semplificare il più possibile questi processi per renderli accessibili a un vasto numero di sviluppatori e massimizzare la diffusione del loro prodotto.

Con i modelli Open Source, invece, le aziende o gli sviluppatori devono affrontare diverse problematiche. Non è sufficiente disporre dell'hardware adeguato. È necessario anche sviluppare un'infrastruttura software per la gestione delle richieste al modello. Questo include la creazione di un server per la gestione delle richieste, la realizzazione di un'API per la comunicazione sia lato server che lato client, e la gestione di eventuali failure del sistema. Un sistema di ridondanza è essenziale per garantire la continuità operativa: se un server si guasta, un altro deve essere pronto a subentrare.

Inoltre, per gestire un elevato numero di richieste, è necessario implementare più istanze del modello e un sistema di bilanciamento del carico. Questo evita che un singolo modello diventi un collo di bottiglia, rallentando il processamento delle richieste.

La sicurezza è un altro aspetto cruciale: l'infrastruttura deve essere protetta

contro attacchi informatici, come tentativi di furto dei dati di addestramento o attacchi DDoS volti a sovraccaricare il sistema.

Pertanto, per una piccola azienda, l'adozione di modelli Open Source può risultare estremamente complessa e costosa. La creazione e la gestione di una soluzione di questo tipo richiedono risorse significative e competenze avanzate, rendendo questa opzione meno accessibile rispetto all'utilizzo di modelli proprietari offerti da aziende come OpenAI.

5.2.2 I punti a favore dei modelli Open Source

La tutela dei dati

Il primo vantaggio dell'utilizzo di un modello Open Source è il pieno controllo sul modello creato, poiché risiede su una macchina di proprietà dell'azienda o del creatore. Questo controllo totale implica che i dati forniti al modello, sia per l'addestramento che per le conoscenze tramite il metodo RAG (Retrieval-Augmented Generation), rimangono sempre sotto la gestione diretta del proprietario. Questo è in netto contrasto con le soluzioni proposte da aziende come OpenAI, dove i dati devono essere inviati a server esterni, al di fuori del controllo diretto dell'azienda.

Ad esempio, se un'azienda sviluppa un modello utilizzando dati sensibili, come tecnologie brevettate o informazioni che costituiscono un vantaggio competitivo, può essere certa che questi segreti non verranno mai esposti su Internet se viene utilizzato un modello Open Source. Al contrario, quando si forniscono dati a modelli come GPT per l'addestramento, questi dati vengono inviati a server non gestiti dall'azienda stessa. Nonostante le promesse di politiche rigorose sulla privacy da parte di queste aziende, una volta che i dati sono stati trasferiti ai loro server, il controllo effettivo sui dati è perso. Questo potrebbe comportare un rischio di perdita del vantaggio competitivo.

Un altro esempio riguarda l'analisi di documenti forniti da altre aziende. Se poniamo che questi documenti abbiano clausole restrittive che impediscono la divulgazione dei dati a terzi. Utilizzare un modello come GPT per automatizzare questo processo sarebbe in violazione di tali clausole, poiché i dati verrebbero inviati a server esterni. Invece, con un modello Open Source, i dati rimangono confinati all'interno dell'infrastruttura aziendale, rispettando così le restrizioni contrattuali e garantendo la riservatezza dei dati.

Avere il controllo sul modello

Durante il mio tirocinio, ho riscontrato frequenti interruzioni del servizio da parte di OpenAI, non limitate a un singolo episodio ma verificatesi circa una decina di volte nel corso di due mesi. Questo è significativo perché, come accennato in precedenza, le aziende che vendono modelli addestrati su piattaforme come GPT dipendono completamente dall'infrastruttura di OpenAI. In caso di disservizi o malfunzionamenti da parte dell'azienda, chi ha sviluppato il modello personalizzato non ha alcun controllo né possibilità di intervenire.

Immaginando l'applicazione di questi modelli in una grande azienda, questa dipendenza comporta un rischio significativo. La mancanza di continuità del servizio da parte dell'azienda proprietaria del modello può causare non solo

perdite economiche dirette, ma anche danni alla reputazione e insoddisfazione da parte dei clienti dell'azienda utilizzatrice. Ad esempio, automatizzando un processo critico per una multinazionale, anche solo un'interruzione del servizio per poche ore potrebbe avere conseguenze gravi e costose.

D'altra parte, con un modello Open Source, il controllo e la responsabilità sono in capo a chi ha creato il modello. Questo significa che l'azienda che implementa il modello Open Source è in grado di garantire una continuità del servizio più robusta. In caso di problemi o interruzioni, l'azienda ha la capacità di rispondere direttamente ed efficacemente, risolvendo le problematiche senza dipendere da terze parti.

5.3 Come e Dove verrà implementata questa nuova tecnologia

Attualmente, l'Intelligenza Artificiale Generativa (IAG) sembra più adatta per grandi aziende che gestiscono un'ampia quantità di documenti, come PDF, Word e email, e necessitano di analizzare il testo per eseguire operazioni specifiche. Questa valutazione iniziale deriva dalla complessità nel creare processi automatizzati altamente efficienti con un basso margine di errore, che richiede un lungo periodo di addestramento del modello. Per ottenere risultati accurati, è essenziale disporre di una vasta quantità di dati specifici per il compito da automatizzare, risorsa più facilmente accessibile alle grandi imprese rispetto alle piccole. Pertanto, lo sviluppo di tali sistemi comporta spese considerevoli, generalmente sostenibili solo dalle grandi organizzazioni.

Inoltre, considerando i benefici in rapporto ai costi, per le piccole aziende con un organico di 10-15 dipendenti, automatizzare processi come la gestione degli ordini (ad esempio 20 al giorno) potrebbe non risultare economicamente vantaggioso. Tuttavia, per le grandi aziende con oltre 1000 dipendenti, che gestiscono un volume di ordini significativo, l'automazione di tali processi rappresenta un investimento notevolmente conveniente. Questo perché, mentre nelle piccole aziende una persona può essere responsabile di diversi ruoli e processi aziendali, nelle grandi aziende un singolo processo può coinvolgere diverse persone. L'implementazione di tecnologie come l'IA in questi contesti potrebbe ridurre il numero di persone necessarie per eseguire un determinato processo, potenzialmente da 10 a 3 o 4.

Vorrei sottolineare che, secondo la mia opinione, questo non rappresenta una minaccia come spesso si teme, ossia che l'IA porterà alla perdita di posti di lavoro. Ad esempio, attualmente in Europa ci sono preoccupazioni legate al cambiamento generazionale, con previsioni che indicano una carenza di manodopera sempre più allarmante. Questa situazione potrebbe creare gravi problemi per il sistema pensionistico, considerando che non ci saranno abbastanza persone in età lavorativa per sostenere finanziariamente le pensioni. Tuttavia, l'IA potrebbe fungere da supporto cruciale per mitigare queste sfide, migliorando la stabilità economica dei paesi europei e facilitando una transizione più fluida verso nuove modalità di lavoro e di gestione aziendale.

Nel futuro prossimo, l'adozione di queste tecnologie evolverà principalmente in base alla complessità delle operazioni da automatizzare, alla loro frequenza e

all'importanza strategica per l'azienda. Le operazioni standardizzate e ripetitive saranno le prime candidate per beneficiare dell'automazione tramite l'IAG.

Inoltre vi è da considerare che nelle aziende il percorso verso la produttività di un dipendente spesso richiede un periodo significativo di formazione interna, che può estendersi da uno a anche più di dodici mesi, a seconda del tipo di mansione da svolgere. Tuttavia, uno dei problemi critici che le aziende affrontano riguarda la breve permanenza dei dipendenti. Senza entrare nel dettaglio di questo fenomeno, è evidente dai dati che le nuove generazioni tendono a cambiare più frequentemente il proprio posto di lavoro rispetto alle generazioni precedenti:

- ★ I Baby Boomers (55-75 anni) hanno una permanenza media di 10 anni o più;
- ★ La generazione X (40-55 anni) resta in media tra i 5-7 anni;
- ★ I millennials (25-40 anni) hanno una media di permanenza in azienda di circa 3-4 anni;
- ★ La generazione Z (18-25 anni) mostra una permanenza media di circa 2 anni;

Questo trend rappresenta una sfida significativa per le aziende, poiché ogni sostituzione comporta la perdita di know-how e l'onere di formare nuovi dipendenti. Questa problematica è emersa chiaramente nel mio recente anno di lavoro, durante il quale ho interagito con un vasto campione di circa trecento aziende e ho potuto dialogare con imprenditori e dipendenti.

Un possibile approccio per mitigare questo problema potrebbe essere l'automazione di processi specifici. Questo non implica che l'IA debba sostituire completamente il lavoro umano, ma piuttosto semplificare e accelerare le operazioni, mantenendo sempre un monitoraggio umano. Tale automatizzazione potrebbe contribuire significativamente a ridurre il turnover del personale e a preservare il know-how aziendale.

Capitolo 6

Bibliografia

Siti web consultati

Manifesto Agile. URL: <http://agilemanifesto.org/iso/it/>.

1. Python: URL: <https://www.python.org/> (cit. a pag. 7)
2. schedule: URL: <https://schedule.readthedocs.io/en/stable/> (cit. a pag. 8)
3. beautifulsoup4: URL: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> (cit. a pag. 8)
4. Imapclient: URL: <https://imapclient.readthedocs.io/en/3.0.0/> (cit. a pag. 8)
5. SQLAlchemy: URL: <https://www.sqlalchemy.org/> (cit. a pag. 8)
6. mysql-connector-python: URL: <https://dev.mysql.com/doc/connector-python/en/> (cit. a pag. 8)
7. secure-smtplib: URL: <https://docs.python.org/3/library/smtplib.html> (cit. a pag. 9)
8. tenacity: URL: <https://tenacity.readthedocs.io/en/latest/> (cit. a pp. 9, 43)
9. OpenAI: URL: <https://platform.openai.com/docs/api-reference/introduction?lang=python> (cit. a pag. 9)
10. MySQL: URL: <https://dev.mysql.com/doc/> (cit. a pag. 9)
11. DBeaver: URL: <https://dbeaver.com/docs/dbeaver/> (cit. a pag. 10)
12. Llama-2-7b-chat-hf-function-calling-v3: URL: <https://huggingface.co/Treliis/Llama-2-7b-chat-hf-function-calling-v3> (cit. a pp. 15, 46)
13. huggingface hub: URL: <https://huggingface.co/docs/hub/index> (cit. a pag. 9)
14. transformers: URL: <https://huggingface.co/docs/transformers/index> (cit. a pag. 9, 46)
15. bitsandbytes: URL: <https://huggingface.co/docs/bitsandbytes/main/en/index> (cit. a pag. 9)

16. peft: URL: <https://huggingface.co/docs/peft/index> (cit. a pp. 9, 48)
17. accelerate: URL: <https://huggingface.co/docs/accelerate/index> (cit. a pag. 9)
18. datasets: URL: <https://huggingface.co/docs/datasets/index> (cit. a pag. 9, 47)
19. scipy: URL: <https://scipy.org> (cit. a pag. 10)
20. flash-attn: URL: <https://pypi.org/project/flash-attn/> (cit. a pag. 10)