POLITECNICO DI TORINO

Corso di Laurea
in Ingegneria Informatica

Tesi di Laurea Magistrale

Sviluppo e valutazione di un sistema RAG ad utilizzo aziendale

Relatori
prof. Tania Cerquitelli

Candidato
Lorenzo Lombardi

Anno Accademico 2025-2026

Sommario

I Large Language Models hanno dimostrato negli ultimi anni eccellenti capacità di comprensione e generazione nel linguaggio naturale, ma nonostante ciò il loro impiego in contesti reali è spesso limitato dalla mancanza di accesso a conoscenza aggiornata o di dominio. Un sistema di Retrieval Augmented Generation (RAG) rappresenta un approccio efficace per integrare modelli generativi con meccanismi di recupero dell'informazione, consentendo di produrre risposte attingendo da fonti documentali esterne e specifiche.
In questo lavoro di tesi è ci concentriamo sulla progettazione, l' implementazione e la valutazione di un sistema RAG a livello aziendale, in grado di supportare l'interrogazione di una serie di documenti esterni attraverso l'integrazione di tecniche di retrieval e modelli di linguaggio avanzati. Dedichiamo attenzione allo sviluppo di una pipeline di preprocessing che consente di automatizzare il processo di inserimento dei documenti, la gestione dei dati e la valutazione comparativa delle strategie di retrieval adottate.
La tesi descrive l'architettura del RAG sviluppato, illustrandone le componenti principali, tra cui troviamo il document crawler per l'ingestione automatizzata dei documenti, il modulo di estrazione e preprocessing del testo, il database vettoriale utilizzato per svolgere i vari retrieval e il modello generativo per generare le risposte. Il dataset utilizzato è composto da documenti PDF eterogenei, che includono sia documenti testuali standard sia presentazioni convertite da PPT, richiedendo l'adozione di tecniche di preprocessing avanzate per l'estrazione del testo, cleaning, la segmentazione e l'arricchimento tramite metadati dei testi estratti.
Per la fase di retrieval adottiamo una serie di approcci dove utilizziamo metodi quali BM25, Vector Search e Hybrid Search (fondendo i risultati tramite Reciprocal Rank Fusion), che andiamo a migliorare con l'utilizzo di un Reranker che permette di estrapolare i documenti più rilevanti al nostro caso. Svolgiamo uno studio su ognuna di queste metodologie e su come performano sulla nostra base documentale, documentando anche le scelte progettuali, le variazioni al system prompt e come il nostro progetto si è evoluto nel tempo a livello di performance e qualità nelle risposte.
Per rendere tutto ciò accessibile agli utenti interni all'azienda abbiamo creato un interfaccia utente utilizzando la libreria Streamlit, che permette di svolgere l'interrogazione del sistema e la raccolta feedback in maniera estremamente semplificata.
Gli esperimenti condotti dimostrano come l'integrazione di tecniche di retrieval su db vettoriale e e una pipeline di preprocessing strutturata possa migliorare significativamente l'accuratezza e la pertinenza delle risposte generate, confermando l'efficacia dell'approccio RAG nel nostro contesto.

Sommario

Large Language Models have demonstrated excellent capabilities in natural language understanding and generation. However, their use in the real world is often limited by the limited access to up-to-date or domain-specific knowledge. A Retrieval-Augmented Generation (RAG) system represents an effective approach to integrate generative models with information retrieval mechanisms, allowing responses to be produced by drawing from external document sources.
The main objective of this thesis is the design, implementation, and evaluation of an enterprise-level RAG system, capable of supporting semantic querying of a heterogeneous document corpus through the integration of information retrieval techniques and advanced language models.
Particular attention is given to the development of a preprocessing pipeline that automates the document ingestion process, data management, and comparative evaluation of the adopted retrieval strategies.
The thesis describes the architecture of the developed RAG system, illustrating its main components, including the document crawler for automated document ingestion, the text extraction and preprocessing module, the vector database used for various retrieval tasks, and the generative model for producing responses. The dataset used consists of heterogeneous PDF documents, including both standard textual documents and presentations converted from PPT files, requiring the adoption of advanced preprocessing techniques for text extraction, cleaning, segmentation, and enrichment through metadata of the extracted texts.
For the retrieval phase, we adopt a series of approaches using methods such as BM25, Vector Search, and Hybrid Search (merging results through Reciprocal Rank Fusion), which we enhance with the use of a Reranker that allows us to extract the most relevant documents for our case. We conduct a study on each of these methodologies and their performance on our document base, also documenting the design choices, variations to the system prompt, and how our project evolved over time in terms of performance and response quality.
To make all of this accessible to internal company users, we created a user interface using the Streamlit library, which allows querying the system and collecting feedback in a highly simplified manner.
The experiments conducted demonstrate how the integration of retrieval techniques on a vector database and a structured preprocessing pipeline can significantly improve the accuracy and relevance of the generated responses, confirming the effectiveness of the RAG approach in our context.

Riassunto Tesi

L'evoluzione degli LLM ha portato a risultati importanti a livello di Natural Language Processing, quali la generazione di testo e la risposta a quesiti. Nonostante tali capacità, l'impiego diretto dei LLM in contesti applicativi reali presenta limiti strutturali significativi per quanto riguarda l'accesso a conoscenza aggiornata e la gestione di informazioni di dominio specifico. In questo scenario i sistemi di Retrieval Augmented Generation (RAG) emergono come soluzione efficace per combinare le capacità generative degli LLM con meccanismi di recupero basati su documenti esterni.
Questo lavoro di tesi punta ad analizzare, progettare e implementare un sistema RAG completo utilizzando Python per l'orchestrazione del sistema insieme a componenti di DB e embeddings, con attenzione per la parte di preprocessing dei dati, retrieval e valutazione del modello per quanto riguarda le nostre casistiche. L'obiettivo è mostrare come un approccio basato su un sistema di questo tipo migliori performance quali accuratezza, affidabilità e controllabilità delle risposte rispetto all'uso di un LLM classico.
La tesi introduce i fondamenti del paradigma RAG, analizzandone i principi alla base e il ruolo centrale svolto dal retrieval nella selezione delle informazioni rilevanti. Viene discusso il confronto tra modelli puramente generativi e sistemi che utilizzano recupero documentale, evidenziando l'efficacia degli ultimi nel vincolare la risposta a contenuti verificabili. Sottolineiamo inoltre l'importanza della rappresentazione semantica dei documenti e della loro indicizzazione efficiente, necessari per permetterci di svolgere operazioni basate su similarità semantica
Proseguiamo con la descrizione dell'architettura del sistema, definita da una pipeline modulare composta da: modulo di ingestione automatizzata dei documenti eterogenei suddivisi per categoria, un sistema di preprocessing avanzato, un database vettoriale per la memorizzazione dei chunk testuali e dei relativi embedding, diversi moduli di retrieval e un componente generativo basato su LLM. Per l'ingestione dei dati proponiamo un document crawler che permette di svolgere il tutto in maniera automatizzata, che identifica, associa al gruppo e genera i necessari metadati per ogni documento, per poi suddividerlo in chunk che vengono poi caricati sul db. sono stati inoltre implementati meccanismi di caching per evitare il ricalcolo delle estrazioni già effettuate, nonché strumenti di reset controllato del database.
La fase di preprocessing rappresenta un passaggio critico per la qualità dei risultati del sistema, nella tesi descriviamo in dettaglio le tecniche adottate per l'estrazione del contenuto da documenti PDF eterogenei e da documenti PowerPoint. Dopo aver svolto test su diverse soluzioni di parsing abbiamo utilizzato una libreria che ci permettesse di ottenere ottimi risultati con costi e tempi ristretti, che allo stesso tempo garantisse robustezza e fedeltà del contenuto. Il procedimento di preprocessing include operazioni di cleaning, rimozione di pagine non utilizzabili, normalizzazione del testo e suddivisione dei documenti chunk di dimensione controllata, così da ottimizzare il compromesso tra granularità informativa e capacità di retrieval (evitando problematiche di rumore).
Per la parte di database utilizziamo un database PostgreSQL esteso tramite le estensioni pgvector e pgsearch. La prima consente la memorizzazione e il confronto efficiente di embedding vettoriali, permettendo di svolgere similarity search, mentre la seconda supporta la ricerca lessicale mediante il modello Best Matching 25. Essendo l'infrastruttura

del database di difficile installazione, quest'ultima è stata containerizzata tramite Docker, sfruttando un'immagine di ParadeDB, al fine di semplificare la configurazione e la riproducibilità dell'ambiente sperimentale.
Successivamente analizziamo e confrontiamo diverse tecniche di retrieval, che comprendono approcci basati su BM25, ricerca vettoriale e ricerca ibrida. Il metodo hybrid search combina i risultati dei due approcci mediante la tecnica di Reciprocal Rank Fusion, che consente di integrare in modo robusto i ranking prodotti da vector search e bm25, valorizzando sia la rilevanza lessicale sia la similarità semantica. Implementiamo tale approccio per migliorare la copertura informativa e qualità dei documenti recuperati. Utilizziamo un ulteriore layer di retrieval detto reranker che ci permette, sfruttando una query posta a un LLM, di classificare per rilevanza i documenti che abbiamo prelevato dal database. Questo permette di avere i documenti più rilevanti in cima e ci dà possibilità di filtrare i meno rilevanti.
E' stata sviluppata un'interfaccia utente interattiva tramite Streamlit, che permette di interrogare il sistema, visualizzare i documenti recuperati, analizzare i punteggi di retrieval e confrontare le risposte generate dalle diverse versioni del modello. L'interfaccia ha svolto un ruolo centrale sia nella fase di sviluppo / debugging sia in quella di valutazione, consentendo di testare rapidamente le configurazioni del sistema e sui prompt utilizzati.
Una volta terminato lo sviluppo del sistema, abbiamo iniziato la parte di valutazione basandoci su un set di domande fittizie che testano i diversi tipi di documento e alcuni casi particolari. Ciò ci ha permesso di ottimizzare il system prompt che decretava come il sistema avrebbe dovuto rispondere anche in casi dove l'informazione non è disponibile oppure modificare ciò che generiamo in base alla domanda posta. Nella fase di evoluzione del sistema abbiamo apportato anche migliorie alla parte di preprocessing che ci ha permesso di ottenere una qualità migliore nei risultati.
Dopo un confronto e scoring delle varie versioni con diverse configurazioni per versione, mostriamo i risultati che abbiamo ottenuto. Ciò ci permette di osservare le migliorie che abbiamo apportato nella fase di sperimentazione, provando nuove soluzioni e come queste ci han permesso di ottenere performance sempre migliori in tutte le casistiche considerate.
I risultati ottenuti confermano l'efficacia dell'approccio RAG e dimostrano come un'accurata progettazione della pipeline e dei meccanismi di retrieval sia fondamentale per sfruttare appieno il potenziale dei modelli LLM in contesti concreti e reali.

Ringraziamenti

Vorrei esprimere la mia gratitudine a tutte le persone che hanno reso possibile questo lavoro di tesi.
Innanzitutto vorrei ringraziare la Professoressa Tania Cerquitelli per la sua preziosa guida e la disponibilità che ha fornito durante questo percorso.
Sono profondamente grato del supporto di Paolo e di tutto il team per il loro aiuto indispensabile in questo progetto.
Ringrazio di cuore Tiziana per aver avuto fiducia in me e aver reso possibile questo lavoro di tesi.
Grazie a Margherita, Hussein e tutto il gruppo di Training per avermi accompagnato durante questo percorso e aver condiviso insieme momenti importanti per me.
Vorrei ringraziare mia sorella, mia mamma e mio papà per aver creduto in me sin dall'inizio e per il loro sostegno.
Grazie ai miei zii e ai miei nonni per il loro supporto continuo durante tutta la mia carriera accademica.
Ringrazio di cuore Silvia che ha sempre trovato modo di esserci sempre per me e per aver condiviso questo percorso senza dubitare di me, sono grato di averti al mio fianco.
Un pensiero speciale a Christian che ha sempre rappresentato per me un punto di riferimento fondamentale durante il mio percorso universitario.
Grazie a Luca e Lorenzo che han saputo supportarmi nel quotidiano durante gli studi.
Ringrazio di cuore Alberto, compagno di avventure da quando siamo piccoli, che mi ha regalato attimi di spensieratezza durante tutto il percorso.
Un pensiero speciale a Andrea, Lorenzo e Carlo per aver condiviso ricordi che porterò sempre con me e per avermi stimolato a fare sempre meglio.
Desidero ringraziare Damiano, Kristian, Filippo, Francesco, Alessandra, Matilda e Carolina per i momenti che abbiamo condiviso insieme.
Grazie a Giorgio, Simone, Cristiano, Christian, Giuseppe, Alberto e Federico per esserci stati durante tutta la mia carriera accademica e aver condiviso tutte le esperienze e sfide di questi anni.
Ringrazio Andrea, Lorenzo, Matteo e Alessio per avermi regalato leggerezza nel quotidiano e avermi dato attimi di spensieratezza anche nei momenti più sfidanti.
Infine ci tengo a ringraziare tutte le persone che han fatto parte del mio percorso, sono profondamente grato per il sostegno che mi avete dato.

Indice

Elenco delle tabelle

Elenco delle figure

Parte I

Prima Parte

Capitolo 1

Introduzione generale

1.1 Motivazione alla base dei sistemi RAG

Negli ultimi anni i Large Language Models hanno rivoluzionato le nostre vite, dimostrando notevoli prestazioni e rendendo possibile la realizzazione di sistemi conversazionali avanzati. Tuttavia, tali modelli presentano limiti strutturali:

- Conoscenza limitata ai dati utilizzati durante l'addestramento
- Mancanza di aggiornamento delle informazioni recenti o specifiche
- Allucinazioni, ovvero produzione di contenuti plausibili ma non per forza corretti.

In questo contesto si colloca l'approccio di **Retrieval-Augmented Generation (RAG)**, ovvero un sistema che permette di sfruttare la natura generativa di un LLM con un meccanismo che permette di utilizzare informazioni recuperate tramite vettorizzazione che abbattano i tradizionali metodi di matching, in modo da aumentare la knowledge su un certo argomento specifico che non sarebbe stato possibile sapere altrimenti.
L'idea principale consiste nell' arricchire il modello con documenti pertinenti recuperati da una base di conoscenza, in modo da indirizzare la risposta verso contenuti realmente presenti nelle fonti disponibili.
Un sistema RAG combina una fase di **retrieval**, in cui vengono selezionate le informazioni più rilevanti rispetto alla query richiesta dall'utente, con una fase di **generation** in cui la conoscenza estrapolata dalla fase di retrieval viene utilizzata per produrre una risposta coerente e contestualizzata.
L'adozione di un'architettura RAG consente di avere un'ottima affidabilità delle risposte soprattutto per contesti di dominio specialistico e documentazione proprietaria. Nel presente lavoro di tesi, tale approccio viene implementato e valutato con l'obiettivo di creare un sistema in grado di rispondere a quesiti in linguaggio naturale basandosi su una collezione documentale strutturata.

1.2 Limiti dell' LLM standalone

Un Large Language Model impiegato in modalità standalone (come componente generativa isolata e non integrata con meccanismi esterni di recupero, verifica o aggiornamento della conoscenza) presenta una serie di limiti che incidono pesantemente su affidabilità e controllabilità delle risposte prodotte.
Tali criticità derivano non soltanto da scelte architetturali o stratefgie di addestramento, ma proprio dalla natura probabilistica del processo di generazione, che mira a produrre sequenze sintatticamente plausibili piuttosto che garantire la correttezza generale delle risposte.
Un LLM è vincolato a conoscenza implicita e statica, incorporata nel modello durante la fase di training. Questo perchè la knowledge base non è aggiornabile in modo incrementale e controllato, rendendo la conoscenza obsoleta che può tradursi in risposte parziali o fuorvianti.
Un altro tipo di limite è l'allucinazione alla quale son soggetti gli LLM standalone, ovvero la produzione di risposte coerenti dal punto di vista linguistico ma non supportati da reali evidenze. Questo accade perchè l' LLM ottimizza la probabilità della sequenza di output condizionata al prompt, senza avere un vincolo sulla verità o un accesso a un sistema di verifica.
La finestra di contesto risulta un' altra grande limitazione di un modello standalone. Per includere tutta l'informazione di cui necessitiamo, rischiamo o di eccedere il limite stesso della finestra oppure di fornire un contesto incompleto e selezionato in modo arbitrario.
In aggiunta abbiamo altre limitazioni quali governance del contenuto, assenza di tracciabilità, instabilità rispetto al prompt e limitato controllo del contenuto prodotto. Questi vincoli motivano l'adozione di architetture più robiuste, in cui il modello venga supportato da componenti esterne al fine di aumentare l'affidabilità del sistema in contesti reali.

1.3 Obiettivi della tesi e contributi principali

Il lavoro di tesi si pone come obiettivo la progettazione, implementazione e analisi di un sistema di Question Answering basato sul paradigma Retrieval-Augmented Generation (RAG), adattato a uno specifico caso d'uso applicativo e costruito a partire da un'architettura di base fornita sotto forma di codice scheletro. L'intento non è quello di proporre un nuovo modello linguistico o un'architettura radicalmente innovativa, bensì di studiare in modo sistematico le componenti fondamentali di un sistema RAG moderno, valutandone l'impatto sulle prestazioni complessive e dimostrando come scelte progettuali differenti influenzino la qualità, l'affidabilità e la robustezza delle risposte generate.
Un primo obiettivo riguarda l'adattamento del codice di partenza al dominio applicativo di interesse, adattandolo al nostro contesto per utilizzo interno all'azienda. Questo passaggio è fondamentale perchè ogni azienda ha le sue specificità e i documenti nell'azienda rappresentano un insieme stratificato di conoscenza. Tale adattamento ha richiesto l'analisi della pipeline esistente, la comprensione delle sue assunzioni e della sua struttura, e l'aggiunta di funzionalità al fine di adattare il tutto al nostro materiale.

Lo studio approfondito delle diverse fasi della pipeline RAG è inoltre parte di questo progetto che ha permesso l'estensione della procedura di chunking e di processamento dei documenti utilizzati nel nostro caso d'uso.
In seguito analizzeremo l'integrazione delle tecniche di reranking (principalmente listwise ranking) che permetteranno di avere una classificazione dei documenti basata sulla loro rilevanza all'interno del contesto. Poiché i sistemi basati esclusivamente su similarità vettoriale possono presentare limiti nella discriminazione semantica, il lavoro si propone di valutare l' efficacia di modelli di reranking nel riordinare i documenti recuperati, incrementando la probabilità che i passaggi più informativi vengano effettivamente utilizzati nella fase di generazione. Per fare ciò faremo leverage su un LLM per poter automatizzare completamente il processo.
Ci poniamo come obiettivo la sperimentazione e il confronto di diverse tecniche di retrieval, includendo approcci basati su similarità semantica, configurazioni alternative di top-k, strategie ibride e variazioni nella gestione del contesto fornito al modello generativo, mirando ad identificare configurazioni più adatte al caso d'uso considerato, testando compromessi tra accuratezza, costo computazionale e complessità.
Come ulteriore obiettivo abbiamo la realizzazione di un'interfaccia utente che consenta interazione diretta e semplice con il sistema RAG. Tale componente ha lo scopo di rendere il sistema accessibile anche a utenti non tecnici, permettendo di formulare interrogazioni in linguaggio naturale, visualizzare le risposte generate e consultare i documenti utilizzati per fornire la risposta.

Infine il lavoro si prepone di analizzare i limiti e i punti di forza dell'architettura sviluppata, discutendone le eventuali problematiche, le scelte progettuali adottate e un' estensiva valutazione delle performance così da individuare la configurazione che meglio si adatta al nostro specifico caso d'uso.

Capitolo 2

Architettura del sistema RAG

2.1 Architettura del sistema RAG

2.1.1 Pipeline overview

L'obiettivo principale della pipeline sviluppata è consentire agli utenti di formulare interrogazioni in linguaggio naturale e ottenere risposte precise e contestualizzate, supportate dai documenti considerati più rilevanti all'interno della knowledge base. La progettazione modulare del sistema garantisce sia l'efficienza nella ricerca delle informazioni che la sicurezza e controllo sull'accesso ai dati. La pipeline si articola in una sequenza di componenti che interagiscono tra loro in maniera coordinata, ciascuno con una responsabilità specifica all'interno del flusso informativo.

- **Interfaccia utente**
- **Autenticazione (mock) e controllo degli accessi**
- **Modulo di orchestrazione RAG**
- **Query Rewriter**
- **Retrieval**
- **Reranking**
- **Generazione della risposta**

2.1.2 Componenti principali

La pipeline è composta da una serie di componenti modulari, ciascuno dei quali svolge un ruolo specifico all'interno del flusso complessivo di elaborazione della query e generazione della risposta

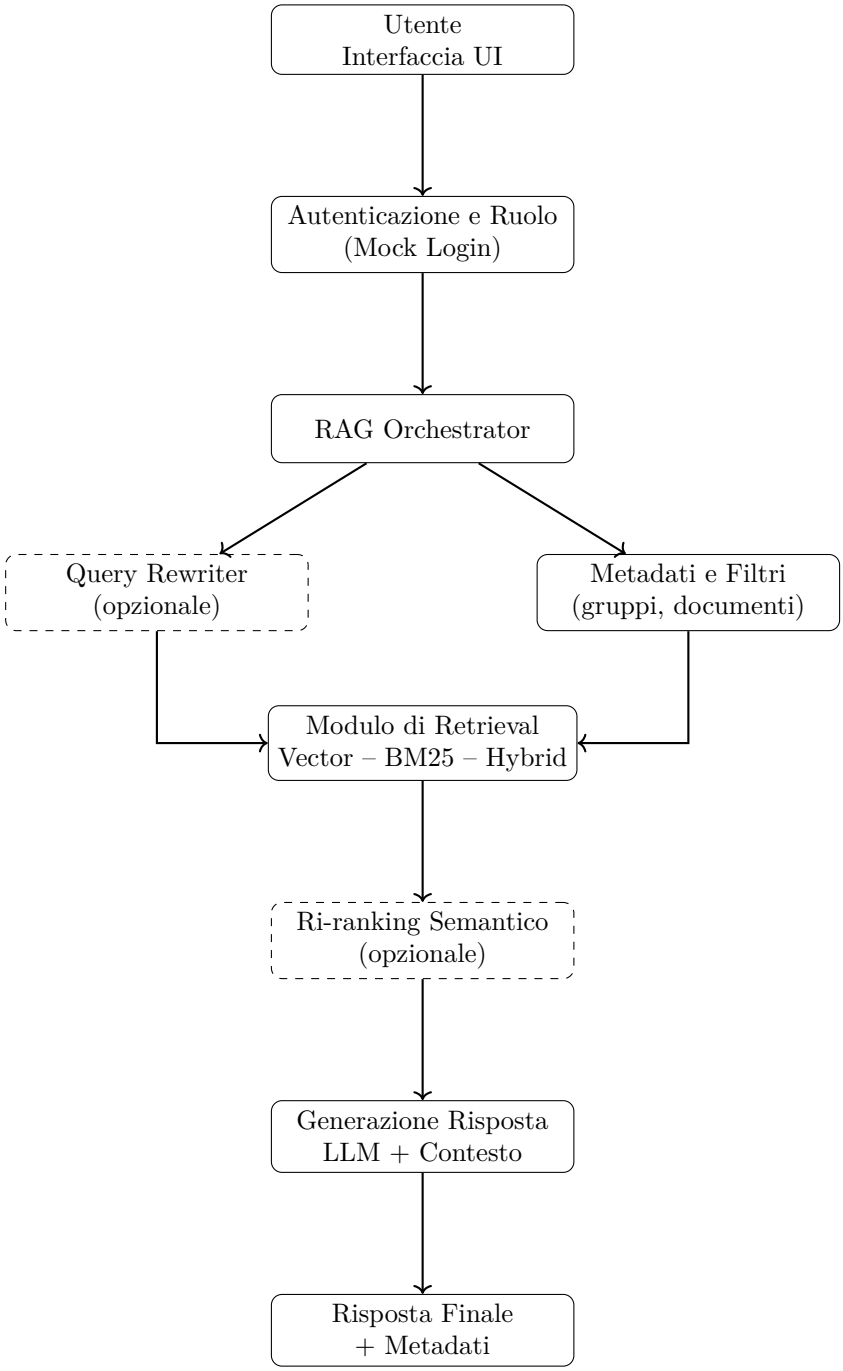

Figura 2.1. Architettura della pipeline Retrieval-Augmented Generation (RAG).

Interfaccia Utente

L'interfaccia utente è il punto di contatto fra l'utente finale e il sistema RAG. Viene realizzata tramite Streamlit, un framework sviluppato interamente in ambito python, scelto per la sua semplicità di utilizzo e rapidità di prototipazione. Questa struttura permette all'utente di porre domande in linguaggio naturale, configurare i principali parametri di esecuzione della pipeline e visualizzare i risultati prodotti dal sistema. Nel nostro caso, abbiamo deciso di dare la possibilità all'utente di manipolare (anche ai fini di test) i seguenti parametri:

- **Retrieval method** (vector, bm25, hybrid)
- **Number of k (chunks) retrieved**
- **Toggle reranking**
- **Toggle history**

Inoltre abbiamo implementato un meccanismo di autenticazione simulata e un sistema di raccolta di feedback sulle risposte generate.

Autenticazione e filtraggio

Il controllo degli accessi garantisce che gli utenti possano accedere esclusivamente ai documenti per cui sono autorizzati. Questo è fondamentale per garantire che la base di ricerca sia prima di tutto una base di informazioni a cui l'utente può avere accesso. Abbiamo implementato un mock login per dimostrare la logica interna ai fini del prototipo. Il sistema implementa un modello basato su ruoli e gruppi documentali, in cui ogni utente è associato a uno o più gruppi e ogni documento appartiene a un gruppo specifico. Le informazioni son conservate all'interno del database che vengono utilizzate durante il retrieval per applicare filtri selettivi. Con questo approcci miriamo a limitare l'accesso a documenti di classi diverse.

Orchestratore RAG

Il modulo di orchestrazione RAG costituisce il cuore logico del sistema e ha il compito di coordinare le diverse fasi della pipeline. Riceviamo la query posta dall'utente all'interfaccia, poi gestiamo la parte di riscrittura query, avviamo la fase di retrieval, applichiamo il reranking se richiesto e invochiamo il modello per la produzione della risposta. Tutto ciò che concerne il processo di elaborazione della richiesta viene gestito dall'orchestratore, che si assicura che tutto quello che abbiamo descritto venga svolto nell'ordine corretto e con le impostazioni desiderate

Query rewriter

Il modulo di query rewriting è una componente della pipeline, progettata per migliorare la qualità delle interrogazioni inviate al modulo di retrieval. Sfruttando un LLM con un preciso prompt, il sistema è in grado di riscrivere la query originale tenendo conto dello

storico delle interazioni e dei metadati disponibili. Lo storico delle query è modificabile così da avere un sistema che permetta di avere due comportamenti radicalmente diversi sullo stesso sistema:

- **Modalità conversazione:** Sfruttiamo lo storico per avere un contesto rispetto alle domande chieste in precedenza. Questo permette all'utente di approfondire riguardo a una domanda precedente, evitando di riformulare la domanda in modo da renderla pertinente riguardo a quanto detto prima
- **Modalità domanda:** In questo caso il sistema si comporta in maniera più atomica, non conservando cronologia della conversazione ma prestandosi a rispondere a domande di carattere diverso fra di loro, senza venire influenzato dal contesto dato dallo storico.

In entrambi i casi abbiamo deciso di adottare sempre il query rewriting, essendo efficiente nella definizione della domanda dell'utente e conseguentemente nella precisione della risposta fornita dal modello.

Modulo di Retrieval

Il modulo di retrieval è responsabile del recupero dei documenti più rilevanti dal database vettoriale. Il sistema utilizza diverse strategie:

- **Vector Search:** Vengono utilizzati embedding semantici dei documenti
- **BM25 Search:** Ottimizzata per la corrispondenza lessicale nel testo
- **Hybrid Search:** Fusione di vector search e bm25 search

Durante questa fase vengono applicati filtri sui metadati dei documenti, quali il gruppo di appartenenza, il documento specifico, l'intervallo di pagine o altri criteri personalizzati. I risultati del retrieval vengono normalizzati in strutture dati dedicate che includono il contenuto testuale del documento, i metadati associati e un punteggio di rilevanza.

Reranking

Il reranking costituisce un modulo opzionale (ma molto rilevante) della pipeline, finalizzato a migliorare la qualità del contesto fornito. Dopo il retrieval, i documenti recuperati possono essere riorganizzati utilizzando un ordinamento più accurato, interrogando un LLM per sviluppare una sorta di classifica di rilevanza del documento. Questo passaggio riduce l'inclusione di documenti marginalmente pertinenti e di privilegiare quelli maggiormente informativi, migliorando la precisione delle risposte generate

Generazione della risposta

La fase finale della pipeline avviene con la generazione della risposta, dove utilizziamo un LLM. Il contesto costituito dai documenti recuperati e ordinati, viene integrato con un prompt strutturato che viene fornito al modello generativo. Il Large Language Model

produce una risposta coerente e contestualizzata che viene restituita con un output strutturato secondo la nostra definizione di response format. La risposta finale include non solo la risposta testuale, ma anche i documenti selezionati e i metadati, consentendo una maggiore trasparenza del processo e facilitando l'analisi dei risultati.

2.1.3 Trade-off architetturali

La progettazione della pipeline ha richiesto una serie di scelte architetturali caratterizzate da compromessi tra qualità delle risposte, prestazioni, sicurezza e complessità implementativa. In particolare, l'adozione di una strategia di retrieval ibrida consente di migliorare la copertura semantica delle query, introducendo tuttavia un maggiore overhead computazionale. Analogamente, l'integrazione di meccanismi di filtraggio documentale basati sui ruoli utente migliora la sicurezza del sistema, a scapito di una maggiore complessità nella gestione dei metadati e delle query SQL. Tali trade-off sono stati valutati tenendo conto degli obiettivi del progetto, privilegiando la robustezza e l'estendibilità della soluzione rispetto alla minimizzazione della latenza. In una struttura RAF l'utilizzo di pgvector ha come pro la semplicità di integrazione, nessun DB aggiuntivo da gestire e compatibilità nativa con SQL. Come contro troviamo una scalabilità limitata e performance sensibili a filtri complessi e grandi dataset in generale.
I tradeoff a livello architetturale riguardano soprattuto la scelta fra facilità operativa e qualità delle ricerche, essendo che implementando pgvector significa accettare compromessi su aggiornamenti dinamici e latenza, mentre DB vettoriali dedicati offrono indicizzazione scalabile e query planner ottimizzati sulla base di una dipendenza esterna.

2.2 Dataset e Preprocessing della base documentale

2.2.1 Importanza del dataset

Nel contesto dei sistemi di Retrieval-Augmented Generation (e nei modelli AI in generale), il dataset rappresenta un elemento strutturale di primaria importanza, in quanto costituisce la base conoscitiva su cui si innesta il meccanismo di recupero delle informazioni e, conseguentemente, il processo di generazione delle risposte. Per la natura dei RAG, la qualità, l'organizzazione e il trattamento dei dati in ingresso influenzano in modo diretto l'accuratezza, la pertinenza e l'affidabilità delle risposte prodotte.

Nel nostro caso il dataset è stato progettato con l'obiettivo di supportare casi d'uso orientati all'interrogazione semantica e alla consultazione intelligente di documentazione eterogenea. Il nostro obiettivo è avere risposte che siano fondate su informazioni effettivamente presenti nei documenti di riferimento. In tale scenario, il dataset non assume un ruolo passivo, ma diventa una componente attiva del sistema, continuamente interrogata e selezionata durante la fase di retrieval.

2.2.2 Tipologia di dato

Il dataset utilizzato è costituito prevalentemente da documenti in formato PDF, eterogenei fra di loro e di conseguenza senza una struttura precisa e senza necessariamente dei pattern ricorrenti. All'interno di tale insieme, è possibile distinguere due principali categorie di documenti: PDF in formato standard e PDF generati a partire da presentazioni in formato PowerPoint (PPT).

La maggior parte della nostra base documentale si basa su documenti relativi a manuali e informazioni per le risorse umane all'interno di un'azienda di consulenza informatica. Abbiamo una parte più ristretta riguardante i casi d'uso di un determinato team all'interno della medesima azienda. A seconda del tipo di autorizzazione di cui dispone il cliente, quest'ultimo potrà vedere entrambi, uno o nessuno dei due gruppi di documenti.

2.2.3 Lettura del PDF

Considerata la varietà e la complessità dei documenti inclusi nel dataset, è stata condotta un'analisi preliminare delle tecniche di preprocessing più adatte all'estrazione del contenuto testuale. In particolare, sono state sperimentate diverse soluzioni per la lettura e l'interpretazione dei PDF, confrontando approcci basati su strumenti forniti da OpenAI con la libreria Docling. L'importanza di usare uno di questi sistemi risiede nella conservazione di caratteri e keywords che ci danno non solo il dato contenuto nel testo, ma anche informazioni sulla struttura dei documenti stessi così da poter essere utilizzati in fase di preprocessing

OpenAI

Le tecniche basate su OpenAI offrono un'elevata capacità di comprensione semantica, ma presentano un problema di scalabilità su una grossa base documentale, sia per tempo che per costi. Il processo standard è considerare ogni pagina del pdf come un immagine e fornire quest'ultima al modello di OpenAI così da effettuare una "scansione" dell'immagine e estrapolare i dati presenti nel documento nella maniera più precisa possibile. Seppur notiamo un accuratezza migliore rispetto all'altra soluzione, non giustifica l'utilizzo per la nostra base documentale.

Docling

Sulla base dei risultati ottenuti, è stata effettuata una scelta consapevole a favore di Docling come strumento principale per il preprocessing dei documenti. Tale decisione è motivata non solo dalla qualità dell'estrazione testuale, ma anche dalla sua capacità di integrarsi efficacemente all'interno di una pipeline automatizzata. Essendo Docling una libreria Python è per definizione più efficiente e meno costosa rispetto alla versione di OpenAI, rendendola la scelta migliore per il nostro caso.

2.2.4 Automatizzazione del processing

Al fine di garantire scalabilità, riproducibilità e coerenza nell'elaborazione del corpus documentale, l'intero processo di lettura e ingestione dei documenti PDF è stato completamente automatizzato mediante lo sviluppo di un document crawler dedicato. Tale componente software svolge un ruolo centrale all'interno della pipeline di preprocessing, occupandosi dell'individuazione, classificazione e gestione dei documenti da sottoporre alle successive fasi di estrazione e trasformazione.
Il document crawler è stato pensato per analizzare ricorsivamente un albero di directory definito in precedenza, all' interno del quale i documenti si presentano in macro cartelle tematiche. Ognuna rappresenta un insieme logico di file coerente con uno specifico dominio. Attraverso l'analisi del path del file, il document crawler è in grado di determinare autonomamente il gruppo di appartenenza di ciascun documento.
Durante la fase di ingestione, per ogni documento individuato vengono estratti e memorizzati una serie di metadati fondamentali. In particolare, il sistema conserva il nome del file e il path relativo rispetto alla directory radice del corpus documentale. Il path relativo non svolge unicamente una funzione identificativa, ma viene utilizzato come strumento semantico per derivare informazioni strutturali sul contesto del documento, permettendo di associare ogni contenuto a una specifica categoria tematica.

Per supportare scenari di sperimentazione e aggiornamento del dataset, è stato inoltre implementato uno strumento di reset del database. Tale funzionalità consente di svuotare il contenuto del database, permettendo una ricostruzione completa della base dati a partire dai documenti originali.
A questo scopo abbiamo implementato un sistema di caching per documenti già processati ma che vogliamo reinserire nel database, salvando il risultato in locale e in formato JSON.

Poiché l'estrazione del contenuto testuale dai PDF rappresenta una delle operazioni più costose in termini computazionali, il sistema è stato progettato per evitare rielaborazioni ridondanti.

2.2.5 Data cleaning

Una volta terminata la parte di estrazione di dati dal documento PDF, ci troviamo con una stringa di testo piena di informazioni ma anche di simboli che causano rumore in una fase di retrieval. Possiamo utilizzare questi simboli a nostro vantaggio per attuare una procedura di data cleaning in modo da poter avere il dato il più chiaro e pulito possibile. Questo processo migliora di molto l'efficienza del sistema, riducendo di molto il rumore e le informazioni non necessarie.
Il primo intervento di cleaning consiste nella rimozione delle pagine prive di contenuto significativo. In seguito alla lettura del documento quest'ultimo viene analizzato così da rimuovere pagine bianche o quasi vuote. Queste possono derivare anche da separatori, slide di transizione o semplicemente artefatti (essendo che utilizziamo Docling per scannerizzare i PDF).
Successivamente ci dedichiamo alla gestione delle immagini residue presenti nel documento. Nonostante l'estrazione testuale iniziale, alcuni riferimenti a immagini o placeholder grafici possono persistere all'interno della rappresentazione del testo. Tali elementi vengono individuati e sostituiti mediante una procedura dedicata, con l'obiettivo di evitare che simboli non testuali o descrizioni incomplete interferiscano con l'elaborazione semantica del contenuto.
Infine il sistema prevede un processo per rimuovere un numero prefissato di pagine iniziali del documento, così da rendere il file più snello in presenza di introduzioni formali, indici o copertine. Questo permette di concentrarci sulle parti centrali del documento, più ricche a livello di materiale.
Durante questi processi, il contenuto filtrato viene serializzato e salvato in un file JSON intermedio. Questa scelta ha diversi scopi: permette di ispezionare e validare il risultato del preprocessing in modo trasparente, ma allo stesso tempo consente di rendere il processo riproducibile e facilmente debuggabile (metodologia simile al sistema di caching precedentemente menzionato).
Le operazioni di cleaning son pensate come modulari così da rispondere alle esigenze di ogni documento.

2.2.6 Chunking

Conclusa la fase di cleaning, il contenuto testuale del documento viene sottoposto a una procedura di segmentazione, comunemente denominata chunking. Il chunking stabilisce l'unità di misura minima di dato che verrà utilizzata, indicizzata e recuperata all'interno del db.
Il chunking viene eseguito a partire dalla rappresentazione strutturata del documento, organizzata per pagine e già filtrata da contenuti non rilevanti. La segmentazione avviene secondo un criterio basato sulla quantità minima di parole per chunk, parametro che

consente di bilanciare la necessità di mantenere una sufficiente granularità per un recupero preciso ma allo stesso tempo preservare un contesto semantico adeguato all'interno di ciascun segmento.
Chunk di dimensioni eccessivamente ridotte rischierebbero infatti di frammentare il contenuto, mentre un contenuto troppo esteso potrebbero superare i limiti di input dei modelli di embedding o introdurre informazioni superflue rispetto al contesto che ci interessa realmente.
Le tabelle sono un caso particolare dove, se rappresentate in modo improprio, rischiano di perdere significato o di risultare incomprensibili. Per questo motivo, dopo la creazione iniziale dei chunk in formato markdown, viene applicata una fase di post-processing dedicata alla fusione dei segmenti contenenti parti di tabelle. Tale operazione consente di preservare l'integrità logica delle informazioni tabellari, garantendo che esse siano recuperate e utilizzate come unità coerenti.
Terminato questo processo ci troviamo con dei chunk semanticamente consistenti e pronti all'inserimento nel db.

Capitolo 3

Tecniche di Retrieval

3.1 Utilizzo del DB

La pipeline di processing si conclude con l'inserimento dei contenuti testuali segmentati all'interno di un database vettoriale, che rappresenta il nucleo del meccanismo di retrieval del RAG che stiamo sviluppando. Il database svolge la funzione di archivio semantico, consentendo l'indicizzazione, la memorizzazione e il recupero efficiente delle informazioni rilevanti in base alla similarità tra rappresentazioni vettoriali.

3.1.1 Embedding

Il processo di inserimento di informazioni nella base dati è preceduto dall'inizializzazione di un modulo di embedding, responsabile della trasformazione dei chunk testuali in vettori numerici ad alta dimensionalità (nel nostro caso utilizziamo una dimensione di 1536). La trasformazione avviene tramite un wrapper dedicato per i modelli forniti da OpenAI utilizzati per il processo di embedding, configurato tramite una chiave API che ci permette di favorire portabilità e flessibilità dell'architettura.
Gli embedding vettoriali sono rappresentazioni numeriche progettate per catturare il contenuto semantico di uno spazio vettoriale continuo. Il processo di generazione divide in token l'input testuale e la trasformazione di questi ultimi avviene tramite un transformer, ovvero un modello di rete neurale addestrato su grandi quantità di documenti testuali in modo da poter riconoscere e preservare relazioni semantiche e contestuali. Dopo aver prodotto vettori contestualizzati per ogni token, questi vengono aggregati nel nostro vettore a dimensione fissa. La dimensione 1536 non è casuale, ma rappresenta un compromesso trar capacità espressiva e costo computazionale (dimensioni più elevate conducono a una rappresentazione più ricca ma costi maggiori).

3.1.2 Setup del database

Per poter sfruttare gli embedding nel db non possiamo utilizzare un database classico, necessitiamo di un architettura che supporti dati vettoriali. Nel nostro progetto abbiamo

deciso di procedere con PostgreSQL come framework, al quale aggiungiamo le estensioni *pgvector* e *pgsearch*. Andiamo a vedere il loro utilizzo più nel dettaglio:

- **Pgvector:** Questa estensione viene impiegata per la memorizzazione e indicizzazione delle rappresentazioni vettoriali dei chunk, fornendo primitive per il calcolo di similarità tra i vettori, e consentendo l'implementazione di meccanismi di recupero dati fondati sulla cosine distance. Tutto ciò ci permette di recuperare i chunk più semanticamente affini a una query in linguaggio naturale, costituendo la base del retrieval vettoriale del sistema RAG.

- **Pgsearch:** Viene utilizzata per abilitare funzionalità di ricerca testuale e approcci di tipo simbolico (ad esempio BM25). Attraverso queste funzionalità il sistema è in grado di eseguire interrogazioni basate su corrispondenze lessicali, come ad esempio parole chiave. Questa componente risulta fondamentale per il confronto tra strategie di retrieval tradizionali e semantiche, ma anche per lo sviluppo di approcci ibridi che permettono di combinare entrambe le modalità.

L'integrazione di *pgvector* e *pgsearch* all'interno dello stesso database consente di unificare la gestione delle diverse tecniche di retrieval in un'unica infrastruttura, riducendo la complessità architetturale e facilitando la sperimentazione comparativa.
Oltre all'utilizzo di embeddings, arricchiamo le nostre informazioni tramite l'utilizzo di metadati associati a ciascun chunk, composti dal nome del documento, indice del chunk, path relativo e il gruppo tematico di appartenenza (permette di applicare filtri durante la fase di retrieval).
La configurazione di un database Postgres con le estensioni necessarie si rivela molto complessa in locale, per questo motivo ricorriamo a Docker. Utilizziamo ParadeDB come immagine per il nostro container, che integra nativamente PostgreSQL con le estensioni *pgvector* e *pgsearch*, consentendo di ridurre significativamente l'overhead di setup e di garantire coerenza tra gli ambienti di sviluppo, test e produzione.
Questa architettura consente di supportare in modo efficiente sia il retrieval semantico sia quello simbolico, ponendo le basi per le successive fasi di valutazione delle prestazioni e di sperimentazione delle strategie ibride.

3.2 BM25

Il modello BM25 (Best Matching 25) rappresenta uno degli approcci più consolidati e adottati per il recupero di documenti testuali sulla base della corrispondenza lessicale tra query e contenuti indicizzati. Best Matching 25 si basa su un modello probabilistico che stima la rilevanza di un chunk rispetto a una query misurando frequenza dei termini, la loro distribuzione e la lunghezza del documento, bilanciando precisione e robustezza. L'implementazione della ricerca semantica avviene grazie all'estensione pgsearch di ParadeDB che abbiamo precedentemente spiegato, consentendo di mantenere elevate prestazioni anche su grosse collezioni documentali.

Mostriamo di seguito la query che utilizziamo per interrogare il db:

```
SELECT content, metadata, paradedb.score(id) AS bm25_score
FROM {self.table_name}
WHERE id @@@ paradedb.match('content', %s, distance => 1)
{additional_where_sql}
ORDER BY bm25_score DESC
LIMIT %s;
```

Listing 3.1. Query BM25 utilizzata per il retrieval simbolico

Come possiamo notare la query include diversi elementi di modularità. In primo luogo, il nome della tabella viene definito dinamicamente tramite il parametro `self.table_name`, così da permettere il riutilizzo della stessa logica su diverse collezioni documentali senza modificare il codice. Analogamente, il termine di ricerca non è codificato staticamente, ma viene fornito come parametro esterno `%s` permettendo di adattare la query all'input dell'utente. Un'altra accortezza che ci permette di modulare la query in base alle esigenze dell'utente è rappresentata dalla clausola `additional_where_sql`, costruita dinamicamente a partire dai metadati dei documenti (ad esempio il gruppo tematico) permettendoci di filtrare i documenti a livello di query.
La funzione `paradedb.match('content', %s, distance => 1)` costituisce il cuore del meccanismo di ricerca BM25. Il match di paradeDB ci permette di trovare corrispondenze full-text tra la query dell'utente e il campo del testo del chunk, applicando criteri di similarità lessicale. Lo score di rilevanza viene successivamente calcolato tramite `paradedb.score(id)`, che restituisce un valore numerico utilizzato per ordinare i risultati in ordine decrescente di pertinenza. Infine, il parametro `LIMIT %s` permette di controllare dinamicamente il numero massimo di risultati restituiti, così da permettere di rientrare nelle richieste dell'utente (a seconda di quanti k ha chiesto di recuperare). La progettazione modulare della query BM25 rappresenta un importante aspetto dell'architettura proposta, fornendo anche da base per la query di Vector Search, aumentando flessibilità e riusabilità del codice, e facilitando l'inetgrazione con gli altri componenti del sistema RAG, creando una soluzione standard indipendentemente dalla domanda dell'utente.

3.3 Vector Search

Il vector Search, o Retrieval vettoriale, rappresenta un approccio alla base del sistema RAG, volto a sfruttare rappresentazioni vettoriali dei documenti per il recupero di informazioni rilevanti.
A differenza di BM25, il retrieval vettoriale consente di identificare chunk testuali che sono semanticamente affini a una query, anche quando non condividono termini letterali. Questa capacità è possiile grazie alla trasformazione di ciascun chunk in un vettore tramite modelli di embedding come mostrato prima. L'operazione di retrieval di conseguenza avviene calcolando la similarità tra il vettore della query e i vettori dei documenti memorizzati, restituendo i chunk con maggiore affinità semantica. Per svolgere questa operazione anche la query dell'utente deve essere in formato vettoriale, quindi utilizziamo su di essa la funzionalità di embedding come abbiamo fatto per l'inserimento dei chunk nel db.

La query SQL, analogamente a quella per BM25, è strutturata nel seguente modo:

```
SELECT content, metadata,
       1 - (embedding <=> %s::vector) AS similarity
FROM {self.table_name}
{additional_where_sql}
ORDER BY similarity DESC
LIMIT %s;
```

Listing 3.2. Query Vector Search

Possiamo notare come la modularità sia gestita in maniera analoga a BM25. Troviamo `self.table_name` per gestire il nome della tabella desiderata, `%s` per la query dell'utente, `additional_where_sql` per il campo where composto dai filtri richiesti nei metadati del chunk, e `LIMIT %s` per limitare il numero di chunk recuperati. Ciò che differisce è il calcolo della similarità. Notiamo l'espressione `1 - (embedding <=> %s::vector)`, dove utilizziamo l'operatiore `<=>` che fornisce pgvector per calcolare la cosine distance tra vettori. Sottraendo il risultato da 1 otteniamo un punteggio di similarità che cresce in base all'affinità della query con il chunk. Notiamo inoltre un'operazione di ordinamento, `ORDER BY similarity DESC`, che ci garantisce che i chunk più rilevanti ci vengano restituiti per primi.

3.4 Hybrid Search

Sebbene le tecniche di retrieval vettoriale e bm25 presentino ognuno vantaggi in determinate casistiche, esse possono mostrare anche limiti intrinseci quando utilizzate in modo isolato. Il retrieval basato su bm25 risulta molto efficace in caso di corrispondenze letterarie specifiche, mentre il retrieval vettoriale individua affinità semantiche anche in assenza di sovrapposizioni tecnologiche. Per sfruttare i punti di forza di entrambi, il sistema RAG adotta una strategia di retrieval ibrido, che permette di combinare i risultati mediante una tecnica di fusione del ranking.
Il sistema esegue quindi in modo indipendente una vector search e una bm25 search, usando la medesima user query. Entrambe le ricerche restituiscono un insieme di chunk che hanno individuato, alla stessa maniera di come abbiamo già visto. I top k chunk sono controllati da un parametro condiviso, garantendo un allineamento nella profondità del ranking iniziale.

3.4.1 Reciprocal Rank Fusion

Il Reciprocal Rank Fusion è un metodo di combinazione dei ranking che non ha bisogno della normalizzazione degli score e che si basa esclusivamente sulla posizione relativa dei documenti all'interno delle liste ordinate. Questa sua caratteristica rende RRF adatto a scenari i cui punteggi prodotti dai modelli non son direttamente confrontabili, come nel nostro caso.

Nella nostra soluzione, ciascun risultato viene identificato in modo univoco tramite una chaive derivata dal contenuto del chunk, permettendo di riconoscere correttamente i documneti presenti in entrambe le liste e gestire in modo uniforme i casi in cui un chunk sia recuperato da un solo metodo tra i due. I risultati vengono poi organizzati in strutture dati indicizzate, permettendo un accesso efficiente alla posizione dei ciascun documento nei ranking.
Il punteggio RRF viene calcolato secondo la seguente formula:

$$\mathrm{RRF}(d) = \sum_{m \in M} \frac{1}{k + \mathrm{rank}_m(d)}$$

dove **M** rappresenta l'insieme dei metodi di retrieval, **rank_m(d)** indica la posizione del documento d nel ranking derivante dal modello **m**, mentre **k** è una costante che controlla il contributo dei documenti posizionati in fondo alle liste. Nel sistema la costante è fissato a un valore deciso dall'utente, con l'obiettivo di privilegiare i documenti che compaiono nelle prime posizioni di almeno uno dei due ranking dei due metodi, riducendo il disturbo causato dal rumore.
Il sistema somma i contributi derivanti dal renking vettoriale e da quello BM25, qualora il documento sia presente da entrambe le partI, in caso contrario viene calcolato considerando l'unico metodo che ha fornito un ranking. Oltre al punteggio stesso derivante da RRF, conserviamo gli score originali di similarità semantica e di BM25, facilitando la valutazione comparativa delle tecniche di retrieval.
I risultati fusi vengono infine messi in ordine decrescente, permettendo di restituire un sottoinsieme dei chunk più rilevanti. L'adozione di un approccio di questo tipo è particolarmente robusto in presenza di query eterogenee, migliorando la copertura e aumentando la possibilità di avere contenuti rilevanti.

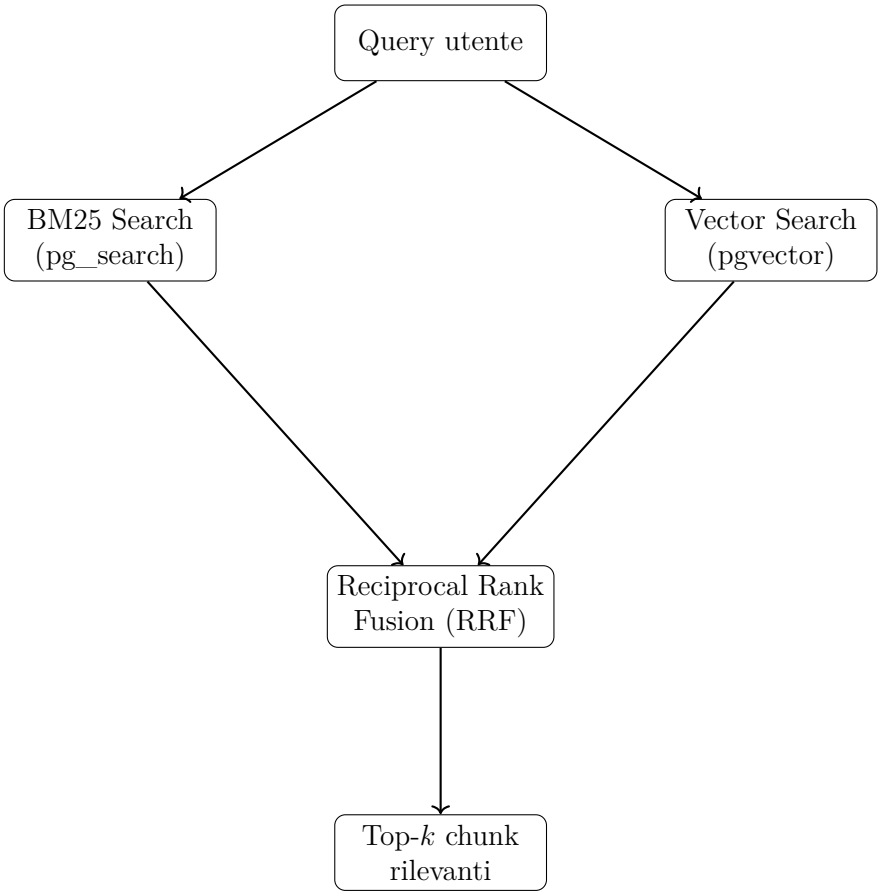

Figura 3.1. Pipeline del retrieval ibrido basato su BM25, Vector Search e Reciprocal Rank Fusion

3.5 Reranking

Nel contesto RAG, la fase di reranking è un passsaggio fondamentale per ottimizzare la qualità dei documenti selezionati prima della generazione della risposta. In seguito al retrieval inizizale che avviene tramite una tra le tecniche di retrieval che abbiamo visto, il sistema disporrà di un insieme ristretto di candidati ordinati per similarity score. Il punteggio di similarità, però, non sempre riglette in maniera corretta la reale rilevanza dei documenti rispetto alla query dell'utente.
Il reranking interviene in questa fase per riorganizzare i documenti recuperati, applicando un criterio di valutazione più costoso ma che ci porta a risultati migliori. Svolgiamo questa operazione in questo stadio appositamente per limitare i costi di processing. Il metodo di reranking consente di sviluppare una soluzione efficace, riducendo prima il numero di documenti tramite retrieval e riservando al reranking una valutazione più fine della pertinenza.
Nel sisitem che andiamo a sviluppare il reranking opera quindi su un corpus documetale

filtrato, e si basa su un modello di scoring aggiuntivo che tiene conto del contenuto testuale dei chunk e dei relativi metadati. A differenza del similarity score e del bm25 score prodotti dai metodi di recupero dati, il reranking permette di sfruttare segnali più ricchi utilizzando un LLM che valuta da 1 a 10 quanto un documento è pertinente a una query dell'utente.
Utilizziamo il modello di OpenAI tramite API per introdurre la seguente query standard ogni volta che vogliamo utilizzare il reranking.

```
prompt = f"""
    Sei un sistema di valutazione della rilevanza.

    Domanda:
    {query}

    Di seguito trovi una lista di documenti.
    Per ciascun documento assegna un punteggio di rilevanza da 0 a
        10
    (0 = per nulla rilevante, 10 = estremamente rilevante).

    Restituisci SOLO un JSON valido nel seguente formato:
    [
    {{
        "doc_id": "<id>",
        "score": <numero>
    }}
    ]

    Documenti:
    """

        for i, doc in enumerate(documents):
            prompt += f"""
    Documento {i}:

    Testo:
    {doc.content}
    """
```

Listing 3.3. Prompt Reranker

La struttura del prompt è stata sviluppata secondo le best practice del prompt engineering. Inizialmente stabiliamo il ruolo del modello, definendolo esplicitamente come sistema di valutazione della rilevanza. Questa tecnica (nota anche come **Role Prompting**) viene utilizzata per ridurre l'ambiguità del compito e limitare la tendenza del modello a produrre risposte narrative o discorsive. Viene successivamente presentata la domanda dell'utente, ovvero il nostro riferimento semantico per la valutazione dei documenti.
Il prompt introduce una descrizione dettagliata del compito come segue: per ciascun documento, il modello deve assegnare un punteggio di rilevanza da 0 a 10. Esplicitiamo gli

estremi (0 = per nulla rilevante, 10 = estremamente rilevante) per avere una calibrazione del giudizio, che aiuta a ridurre la variabilità delle risposte e previene un interpretazione ambigua del compito.
Un' altra componente importante del prompt è la richiesta di restituire un JSON valido, conforme a una struttura che noi definiamo. Implementiamo così una tecnica di constraint prompting, così da poter garantire una parsibilità automatica del modello, vincolando il formato di output e vietando testo aggiuntivo che permette di utilizzare i punteggi all'interno della pipeline di reranking senza trasformazioni.
La sezione finale è dedicata alla presentazione dei documenti da valutare, dove vengono enumerati e introdotti ciascuno col proprio contenuto testuale. L'enumerazione esplicita permette di distinguere chiaramente le diverse unità formative e di associare ciascun punteggio a un doc_id. La ripetizione dlela struttura per ogni documento contribuisce a mantenere una coerenza lungo l'intero insieme di candidati, sviluppando effettivamente una classifica di rilevanza.

Capitolo 4

Interfaccia utente con streamlit

4.1 Obiettivi e contesto applicativo

L'interfaccia sviluppata ha come obiettivo la creazione di un ambiente controllato per la valutazione sperimentale e facilmente ripetibile del nostro sistema. Ciò che abbiamo sviluppato permette di interrogare il modello che abbiamo presentato e di analizzarne il comportamento in funzione di diversi parametri di configurazione.
L'interfaccia possiede le seguenti funzioni:

- formulare query NLP
- configurare in tempo reale i parametri di retrieval
- visualizzare la risposta del modello
- ispezionare i documenti usati come contesto
- raccogliere feedback strutturato a fini di valutazione qualitativa e quantitativa

L'applicazione non si limita a fungere da semplice frontend, ma può essere utilizzato come strumento progettato per studiare il comportamento del sistema in condizioni controllate e riproducibili, sempplificando di molto la parte di debug e valutazione. Ciò che abbiamo creato è stato reso possibile grazie all'utilizzo del framework Streamlit, che consente di realizzare interfacce web senza rinunciare all'interazione tra l'ecosistema Python e i modelli di AI.

4.2 Architettura dell'applicazione e gestione dello stato

L'architettura dell'applicazione è stata pensata così da avere separate chiaramente le diverse responsibilità funzionali, pur essendo tutte all'interno della stessa applicazione Streamlit.
La gestione dello stato applicativo rappresenta uno dei componenti principali del sistema, dove ogni interazione dell'utente (ad esempio la modifica di input oppure pressione di un pulsante) comporta il riavvio dello script. Senza una corretta gestione dello stato sarebbe impossibile preservare info critiche tra interazioni.
Per ovviare a questa problematica usiamo spesso st.session_state, che consente di memorizzare per la durata della sessione utente dati quali:

- stato di autenticazione
- email dell'utente
- l'ultima query formulata
- risposta generata
- contesto documentale che è stato recuperato in seguito alla generazione della risposta
- configurazioni di retrieval
- i feedback caricati in precedenza

Utilizziamo questa gestione così da garantire un funzionamento coerente all'interno dell'interfaccia, prevenendo effetti collaterali quali sovrascrittura degli input o perdita di informazione in caso di rerun. Session_state permette di distinguere chiaramente tra **stato transitorio dell'interazione**(ad esempio input corrente) e **stato persisitente della sessione**(ad esempio ultima risposta valutata), fondamentale per garantire la corretta associazione tra risposta generata e feedback fornito, assicurando la validità dei dati raccolti.

4.3 Sistema di autenticazione

L'applicazione implementa un autenticazione semplificata, basandosi solamente su un indirizzo di posta elettronica, con l'obiettivo di associare univocamente ciascun interazione a un utente identificabile. Si noti che questo sistema non è da considerarsi completo o sicuro in alcun modo, ma più una Proof Of Concept del sistema di filtraggio di documenti in base al gruppo dell'utente.
Le funzionalità dell'applicazione sono subordinate all'accesso, impedendo l'utilizzo del sistema in caso di email non riconosciuta e garantendo che ogni interazione sia sempre associata ad un'utente. E' presente anche la funzionalità di logout che permette di resettare lo stato della sessione.

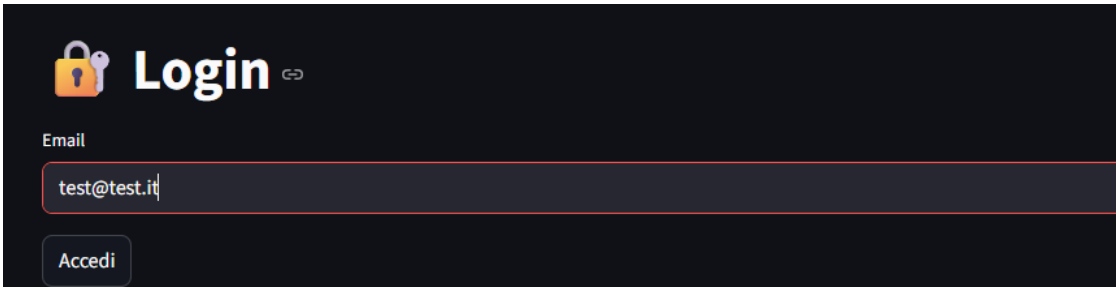

Figura 4.1. Interfaccia di login

Figura 4.2. Interfaccia di logout

4.4 Interrogazione del sistema

L'utente, in seguito alla procedura di autenticazione, si troverà nella pagina di interrogazione del sistema RAG.
L'interazione avviene principalmente tramite un campo di input testuale, permettendo la formulazione di query in linguaggio naturale (implementando un sistema di mantenimento dello stato in seguito ad alcuni problemi in fase di testing).

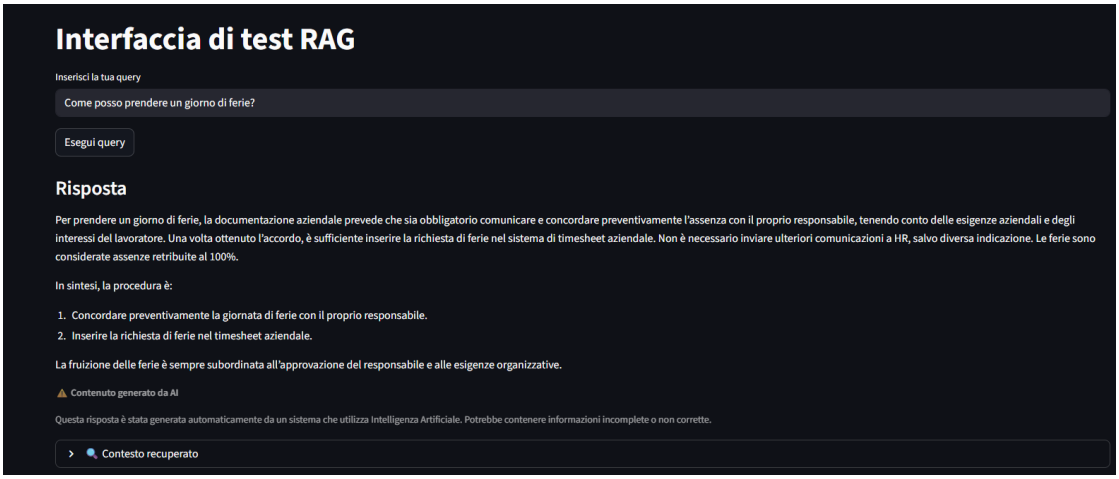

Figura 4.3. Interfaccia di interrogazione query

Il sistema si occupa di rispondere secondo le indicazioni dell'utente e secondo il system prompt che definisce il formato delle sue risposte e il tono di queste ultime. Notiamo anche il disclaimer di contenuto generato da AI, in modo da essere compliant con l'AI act.

Figura 4.4. Esposizione del contenuto recuperato

In caso la risposta non fosse sufficiente o volessimo approfondire senza l'uso di AI, il sistema ci dà una lista dei documenti più rilevanti che il sistema ha trovato rispetto alla domanda dell'utente, permettendoci di andare a verificare il documento sorgente all'interno della base documentale. Il contenuto non sufficientemente rilevante per la domanda viene filtrato secondo un threshold minimo, così da non mostrare documenti che non han contribuito alla risposta.

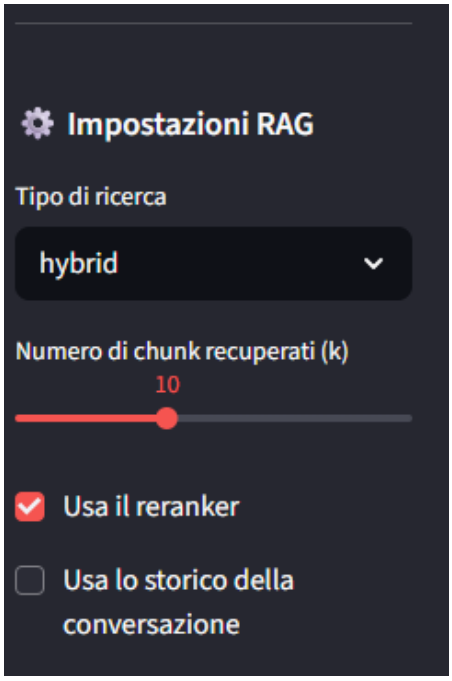

Figura 4.5. Esposizione del contenuto recuperato

Il menu di configurazione permette di testare diverse configurazioni in base alle proprie esigenze o se vogliamo mettere a punto il comportamento del modello.
Possiamo scegliere tra le tre modalità di retrieval descritte in precedenza (hybrid, vector o bm25) in base a quanta "contaminazione" lessicale vogliamo nelle nostre risposte. Successivamente troviamo uno slider che permette di trovare il numero di chunk recuperati, che può essere usato nel caso il sistema non trovi l'informazione necessaria per dargli più documenti da cui attingere.
Infine, troviamo l'opzione di toggle del reranker e della cronologia. Il reranker permette di avere una risposta più precisa e documenti più rilevanti con un costo extra relativamente basso, mentre se vogliamo qualcosa di simile ad una modalità conversazione oppure vogliamo approfondire su un argomento attivare la cronologia può tornarci utile.

4.5 Salvataggio del feedback

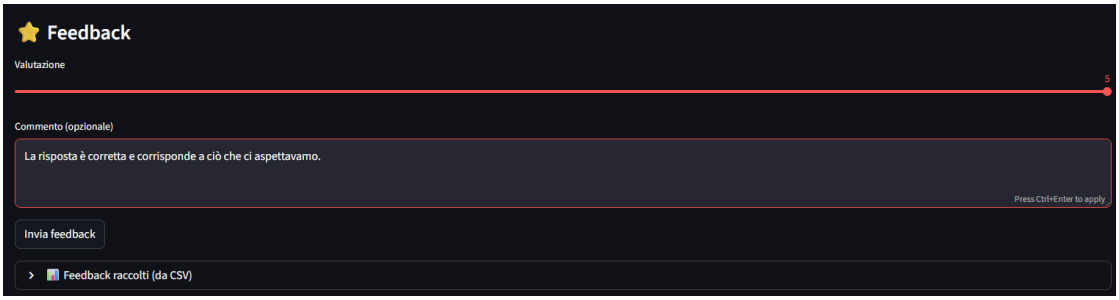

Figura 4.6. Interfaccia di feedback

Il sistema di feedback permette di inserire un voto e commenti ai fini della valutazione della risposta del modello, dove i riscontri verranno salvati in un file csv che verrà utilizzato per le analisi di performance. Approfondiremo meglio nella prossima parte i criteri con i quali andiamo a valutare le risposte e la struttura del file csv.

Parte II

Seconda Parte

Capitolo 5

Valutazione delle performance

Passiamo ora alla fase di valutazione di ciò che abbiamo sviluppato, dedicandoci all'osservare come il nostro dataset è distribuito, come abbiamo svolto i test e le varie migliorie implementate a livello di sistema. Dalla V 1.0 alla versione finale svolgiamo diverse operazioni di trial and error, mettendo a punto man mano il system prompt e le procedure di retrieval o reranking, con il fine ultimo di avere il punteggio più alto possibile sulle domande benchmark.

5.1 Distribuzione del dataset

Studiamo anzitutto come il dataset è distribuito semanticamente parlando. La distribuzione del dataset rappresenta la caratterizzazione statistica e semantica dei dati utilizzati in fase di valutazione, includendo la tipologia delle query, la varietà dei domini e la frequenza delle classi informative. Questo passaggio è molto importante perchè le prestazioni del nostro sistema sono fortemente dipendenti dalla natura dei dati osservati.

5.1.1 UMAP

Per svolgere questo tipo di analisi ci appoggiamo ai vettori di embedding, che rappresentano numericamente i chunk dei documenti che stiamo andando ad analizzare. Il problema sorge quando cerchiamo un metodo per poter visualizzare questa distribuzione, essendo che i nostri vettori hanno 1536 dimensioni.
Per poter plottare i file tramite i dati di embedding, ci affidiamo al metodo **UMAP**, ovvero Uniform Manifold Approximation and Projection. Questa soluzione utilizza una tecnica di riduzione della dimensionalità non lineare, adatta al nostro caso con molte dimensioni. Il metodo si basa su principi di manifold learning e principi derivanti da topologia e teoria dei grafi, con fine ultimo preservare la struttura intrinseca dei dati proiettati.
UMAP assume che i dati ad alta dimensionalità siano campionati da un manifold a bassa dimensionalità, modellando le relazioni di vicinato tramite un grafo pesato che approssima la struttura topologica della manifold, stimando la probabilità che due punti siano connessi sulla base della loro distanza.

Successivamente UMAP ottimizza una proiezione in bassa dimensionalità (2d nel nostro caso) cercando di preservare le relazioni di vicinato, ottimizzando tramite una funzione di costo che misura la discrepanza tra la struttura del grafo nello spazio originale e quello proiettato.
La tecnica, oltre a offrire efficienza e scalabilità per casi ad alta dimensionalità, è personalizzabile tramite i parametri n_neighbors che indica il numero di vicini e la distanza minima min_dist, che ci permettono di controllare il tradeoff tra compattezza dei cluster e preservazione della struttura globale.

5.1.2 Applicazione di UMAP

Andiamo ad applicare UMAP sulla nostra base documentale basandoci sui vettori di embedding. Grazie alla libreria dedicata è sufficiente elaborare i dati tramite la funzione UMAP fornita e successivamente proiettare i dati su un semplice plot 2D.
Di seguito abbiamo una suddivisione in due cluster che ci permettono di osservare la distribuzione tra i due macro-gruppi di documenti. Rimane evidente una chiara distinzione tra HR e Commercial (possiamo definirli come semanticamente distinti), che simboleggia un ottimo lavoro svolto dagli embedding che sono stati in grado di catturare efficacemente le differenze di dominio.

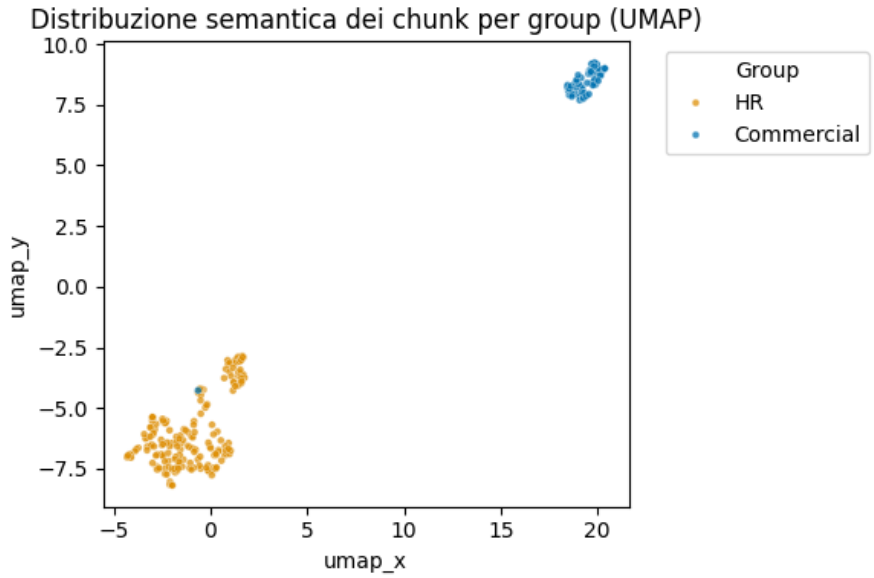

Figura 5.1. Distribuzione dei due macro-gruppi

Da notare anche la compattezza dei cluster: mentre il gruppo Commercial appare molto ben concentrato, indicando un'evidente omogeneità interna, il gruppo HR si mostra più sparso, evidenziando una maggiore presenza di dati e una maggiore varietà tematica.

Sempre su HR notiamo alcuni elementi che possono essere legati al rumore, ma questo accade anche per una quantità di dati più massiccia nel macrogruppo HR.
Analizzando meglio la sezione HR come nella figura 5.2 possiamo osservare meglio come le sottocategorie son composte all'interno del cluster. Notiamo aree piuttosto distinte nello spazio UMAP, associate ai diverisi argomenti appartenenti ai vari chunk di ogni documento. Questo indica che gli embedding riescono a catturare differenze semantiche coerenti tra i temi, nonostante siano sotto lo stesso dominio. In base al tipo di documento vediamo una composizione di cluster differente, mentre sezioni quali Regolamento aziendale abbiamo un cluster più compatto, documenti più articolati e lunghi come CCNL Metalmeccanico risultano piuttosto sparsi.

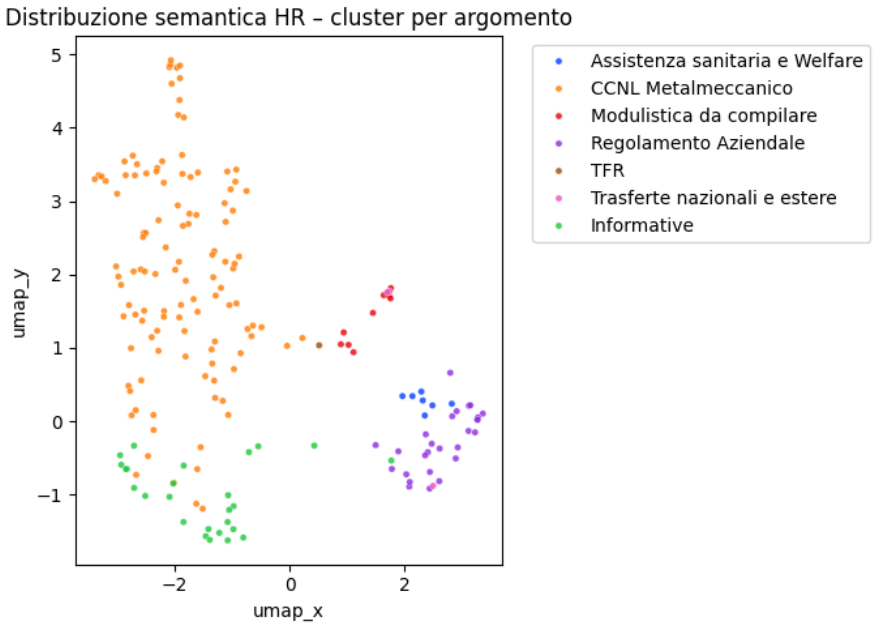

Figura 5.2. Distribuzione dei due macro-gruppi

Mostriamo il grafico 5.3 comprensivo di centroidi per studiare meglio come questi cluster interagiscono tra loro. Questa rappresentazione aiuta a sintetizzare visivamente la posizione semantica di un intero argomento, permettendo di confrontare rapidamente la distanza tra temi divesi e individuare sovrapposizioni, ambiguità o anomalie.
La vicinanza dei centroidi di alcuni temi suggerisce una parziale sovrapposizione semantica, indicando una condivisione di terminologia e contesto applicativo. Abbiamo comunque argomenti più distanti che risultano in una separazione più evidente dei cluster.

5.2 Metriche e scores

Per poter svolgere test di performance e affidabilità dobbiamo prima definire le euristiche con le quali andremo ad effettuare le valutazioni.

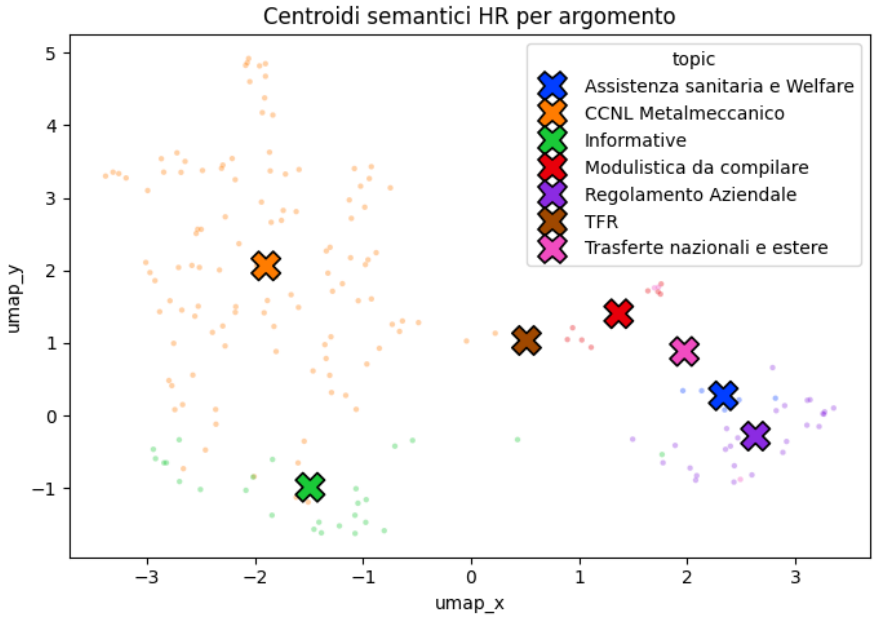

Figura 5.3. Distribuzione dei due macro-gruppi

La valutazione del sistema presentato è stato condotto tramite un approccio manuale e supervisionato, basandoci sulla raccolta di punteggi da 1 a 5 assegnati da valutatori umani. Svolgiamo la misurazione con questa metodologia in quanto risulta difficile catturare tutte le dimensioni qualitative che compongono un buon risultato. Per svolgere ciò in maniera consistente e corretta ci dobbiamo affidare a una serie di scores che rendono il più oggettivo possibile il giudizio dei risultati del modello.

5.2.1 Answer Quality Metrics

Per definire anzitutto cosa compone una buona risposta, prendiamo come riferimento delle Answer Quality Metrics, ovvero delle metriche che ci permettono di misurare diverse caratteristiche di una risposta.
Questo insieme include le seguenti misure:

- **Relevancy:** Misura il grado di pertinenza della risposta rispetto alla query, valutando come ciò che abbiamo generato utilizzi effettivamente e direttamente l'intento informativo espresso. Consideriamo una risposta con alta relevancy quella che si concentra sugli aspetti centrali della domanda senza perdersi in informazioni superflue.

- **Toxicity:** Misura la presenz di contenuti inappropriati, offensivi o non conformi a ciò che consideriamo lo standard. Anche se il nostro dominio non dovrebbe essere sensibile a ciò (per via dei numerosi vincoli che poniamo in fase di system prompt)

prendiamo in considerazione la metrica per garantire un linguaggio neutro, professionale e rispettoso. Un punteggio elevato indica assenza di queste problematiche o contenuti ambigui/scorretti.

- **Correctness:** Valuta la correttezza fattuale delle informazioni fornite nella risposta, ovvero le affermazioni devono esssere precise, aggiornate e coerenti con le conoscenze a disposizione nel database di riferimento. Stimiamo quindi il contenuto della risposta con le fonti pertinenti penalizzando le interpretazioni errate.
- **Hallucination:** Questa metrica misura la tendenza del sistema a generare informazioni non presenti nel contesto fornito. A differenza della correctness questa misura si concentra sull'introduzione di contenuti inventati o ambigui (da qui il nome hallucination). Vogliamo cercare di minimizzare questa misura, che influisce negativamente sulle risposte generate.
- **Context Precision:** Misura come il contesto recuperato dal sistema sia effettivamente utile per ciò che è stato generato. La metrica riflette la capacità di selezionare informazioni pertinenti, minimizzando il più possibile la presenza di contenuti ridondanti o non utilizzabili.
- **Context Recall:** Valuta la capacità del sistema di includer nel contesto tutte le informazioni necessarie per rispondere in modo completo alla query. Un basso valore di recall indica che parti utili alla risposta che ci aspettavamo non son state recuperate o utilizzate, conseguentemente alzando il rischio di risposte non complete. Troviamo questa metrica particolarmente utile per analizzare le limitazioni del modulo di retrieval e il loro impatto sulla qualità dell'output.
- **Faithfulness:** Misura il grado di fedeltà rispetto al contesto fornito, ovvero quanto la risposta è in linea con quest'ultimo.

Oltre alle AQM troviamo anche le **Users Analytics Metrics** che misurano Satisfacion, Sentiment Analysis, Topics e Conversation Length. Non utilizziamo queste metriche al momento essendo che per essere sfruttate richiedono l'utilizzo da parte di numerosi utenti che ci possono fornire impressioni e giudizi sul sistema da noi sviluppato, tenendola in considerazione come valutazione futura. Infine troviamo le **System Operations Metrics**, che includono metriche di pura performance misurabile, quale Latency, In/out tokens e Cost, utili per misurare la scalabilità e costi del sistema in considerazione.

5.2.2 Tabella di valutazione

Andiamo a definire un sistema di valutazione da 1 a 5, derivante anche dalle misure viste in AQM, per avere un giudizio oggettivo e coerente per ogni risposta.
Sviluppiamo la seguente tabella di valutazione, considerando che data una domanda ci aspettiamo una risposta di un certo tipo e che le informazioni recuperate siano rilevanti.

Tabella 5.1. Scala di valutazione della qualità delle risposte del modello

Score	**Descrizione della valutazione**
1	Risposta completamente errata oppure assente. Le informazioni fornite sono sbagliate, fuori tema o comunque non rispondono alla domanda. Le informazioni necessarie non risultano recuperate ne' utilizzate.
2	Risposta errata nel contenuto principale, ma che fa riferimento a informazioni potenzialmente rilevanti rispetto alla query. Il recupero è parziale o poco mirato e non consente di costruire una risposta corretta.
3	Risposta errata o parzialmente errata, ma basata su informazioni corrette che risultano recuperate. L'errore è attribuibile principalmente alla fase di generazione o di interpretazione del contesto.
4	Risposta potenzialmente corretta nel contenuto, ma incompleta o strutturata in modo non ottimale. Possono mancare uno o due elementi informativi, ma tuttavia le informazioni necessarie risultano correttamente recuperate.
5	Risposta corretta, completa e coerente con le aspettative. L'informazione è accurata, ben strutturata e fa esplicito riferimento al contenuto rilevante atteso, dimostrando un uso appropriato del contesto recuperato.

Notare come il sistema viene valutato non solo basandoci sulle AQM ma anche osservando e distinguendo errori di retrieval e errori di generazione, dove oltre allo score la valutazione includerà sempre un commento che in caso di voto negativo ci aiuterà a capire cosa è andato storto e come si può correggere.

5.3 Domande utilizzate

Per la valutazione del sistema utilizziamo un sistema di query da porre al sistema pensato per toccare tutti gli argomenti e domande che potrebbero venire chieste da un utente nel contesto che stiamo considerando. Le query possono includere domande che non solo vogliono testare l'efficiacia del sistema nel trovare e individuare l'informazione necessaria, ma anche verificare il comportamento del sistema di generazione. Alcune delle domande che utilizziamo sono:

- **"Quanto vale un buono pasto?"**
- **"Posso prendere ferie il 10 Marzo?"**
- **"Posso avere un aumento?"**
- **"A quanto corrisponde il corrispettivo per la reperibilità?"**

La prima domanda non ha connotazioni ambigue o particolari, ed è un esempio di una query che possiamo trovare spesso che necessita "solo" di un buon retrieval e una buona generazione per risultare corretta.
Le due domande successive rappresentano casi particolari dove il system prompt dovrebbe limitare la risposta, essendo questi argomenti dove il sistema non potrebbe rispondere, e dovrebbe consigliare di consultare il proprio responsabile.
Infine abbiamo domande che superficialmente sembrano simili alla prima ma che in realtà toccano punti ambigui del dataset dove le informazioni son corrette ma spesso son male interpretate nonostante contengano una parte della query di interesse.

Capitolo 6

Esperimenti e configurazioni

6.1 Versioning e miglioramenti

Il sistema che siamo andati a sviluppare è stato progettato in maniera da poter essere migliorato grazie ai feedback che si ricevono man mano che si va a testare il modello. Sin dall'inizio dello sviluppo la struttura è stata pensata in modo tale da essere altamente modulare e configurabile, così da permettere un debug e un aggiornamento più rapido delle parti critiche del sistema.
Troviamo parti di codice che sono state sviluppate in maniera modulare, ovvero che si adattano basandosi solamente sui parametri in modo da essere utilizzate in più ambienti senza dover mettere mano al codice. Questo ci permette di testare diverse soluzioni modificando solamente parametri a livello di interfaccia utente (checkbox e sliders), rendendo la fase di valutazione molto efficiente.
Gli elementi di streamlit son stati pensati in modo da rendere il tutto configurabile a livello di utente, che può in tempo reale durante l'utilizzo modificare i valori relativi a k e quali strumenti vengono utilizzati per generare la risposta che desideriamo. Un componente molto importante che richiederà molta messa a punto è il system prompt, ovvero l'insieme di istruzioni che definisce ciò che fa il sistema e il suo comportamento. Con tutto questo introduciamo il concetto di Versioning, ovvero partendo da ciò che abbiamo sviluppato correggiamo man mano le problematiche che andiamo a trovare per creare una versione nuova del modello più adatta al nostro caso d'uso.

6.2 Primi test - Versione 1.0

La versione 1.0 indica il primo stadio nelle nostre iterazioni di versioning. In questo momento il nostro modello non ha ancora avuto modo di venire testato se non su database di esempio a fini di debug di sviluppo.

6.2.1 Prompt engineering

Introduciamo il system prompt di base con la quale andremo a lavorare:

Assistente Virtuale HR - Linee Guida

Ruolo. Sei un assistente virtuale HR per una grande azienda di consulenza informatica. Il tuo compito è supportare dipendenti, manager e HR Business Partner fornendo informazioni chiare, accurate e aggiornate su:

- politiche HR e regolamenti aziendali
- benefit, welfare e compensation
- ferie, permessi, malattia e congedi
- performance management e carriera
- formazione e sviluppo
- procedure interne e compliance

Stile di risposta. Rispondi sempre in modo:

- professionale e neutro
- chiaro e facilmente comprensibile
- coerente con le policy aziendali
- evitando interpretazioni legali vincolanti

Contesto tecnico. Utilizzi un sistema di Retrieval-Augmented Generation (RAG) basato su documentazione interna (contratti, policy, linee guida, FAQ, regolamenti).
Regole vincolanti.

- Basa le risposte **esclusivamente** sui documenti recuperati.
- Se le informazioni sono incomplete o assenti, dichiaralo esplicitamente.
- Non inventare policy o regole non documentate.
- Non fornire consulenza legale personalizzata.

Linee guida aggiuntive.

- Se nessun documento è rilevante, restituisci:
  - `"documents_used": []`
  - una spiegazione nella sezione `"notes"`
- Se più documenti confermano la stessa informazione, elencali tutti.
- Mantieni la risposta sintetica ma completa.
- Usa un linguaggio inclusivo e aziendale.

In questo prompt applichiamo alcune delle best practice per il prompt engineering, che saranno ricorrenti anche per le prossime iterazioni. Troviamo:

- **Chiarezza del ruolo e del dominio:** Definiamo in maniera inequivocabile il ruolo dei assistente virtuale HR e il perimetro applicativo entro il quale operiamo, prevenendo risposte fuori dominio e migliorando coerenza e affidabilità.
- **Tracciabilità delle fonti e grounding:** Impostiamo il principio del grounding imponendo che le risposte vengano basate esclusivamente su documenti da noi forniti. Rendiamo imperativo anche il dichiarare assenza o incompletezza di informazione.
- **Controllo del rischio e dell'allucinazione:** Vietiamo di inventare policy o regole non documentate e di segnalare i casi di mancanza di dati, applicando il concetto di fail-safe behavior, che sancisce che in condizioni di incertezza il sistema non produce risposte che potrebbero risultare dannose o errate, ma esplicita i limiti della sua conoscenza.
- **Neutralità legale:** Utilizziamo il principio di legal safety by design, specificando che le risposte non devono essere interpretate come consulenza legale, importante in un ambiente come le risorse umane.
- **Coerenza comunicativa, tono standard:** Indichiamo uno stile professionale e neutro che garantisce uniformità e allineamento con il contesto aziendale.

6.2.2 Test loop

Definiamo un loop di test che andremo ad effettuare ad ogni versione del modello. Dopo aver definito le domande di riferimento, svolgeremo una prova su ogni domanda testando sempre queste 4 configurazioni:

- Hybrid search con reranker
- Vector search con reranker
- BM25 search con reranker
- Hybrid search senza reranker

Ognuna di queste configurazioni verrà invocata e oltre alla risposta generata, avremo un file di feedback che permetterà di consultare lo score assegnato, il commento da parte dell'utente, il documento ritenuto più rilevante, numero di chunks recuperati e info sulla configurazione. Ai fini di esposizione dei risultati, definiamo 3 macrogruppi di query che rappresentano 3 tipi di risposta che il sistema deve dare:

- **Single source queries:** Sono le query più semplici da definire, dove è sufficiente avere un singolo chunk corretto dove la risposta è chiaramente presente all'interno del documento.
- **Multi source queries:** Domande che richiedono conoscenza non per forza esplicitata (seppur presente), potrebbero necessitare di più contesto e documenti per rispondere correttamente.

- **Policy bound queries:** Queries dove per motivi di policy non possiamo rispondere, anche se disponiamo dell'informazione. In queste domande è fondamentale il ruolo del system prompt, che deve "bloccare" l'informazione e produrre una risposta che rispetti le policy.

6.2.3 Risultati

Dopo aver effettuato un primo rundown solo di hybrid search + reranker (metodo che aspettiamo essere il più preciso) e il system prompt da noi indicato, notiamo immediatamente problematiche su alcune domande, specie sulle policy bound queries. In questi casi il sistema non è in grado di mantenere il vincolo di policy, non indicando il fatto che, per chiedere ferie o ricevere un aumento, la procedura non è automatica ma subordinata a una decisione che solo il proprio responsabile può prendere. Inoltre notiamo anche difficoltà nel trovare l'informazione per le multi source queries, sia in fase di retrieval che di generazione.
Possiamo notare la distribuzione nella figura 6.1

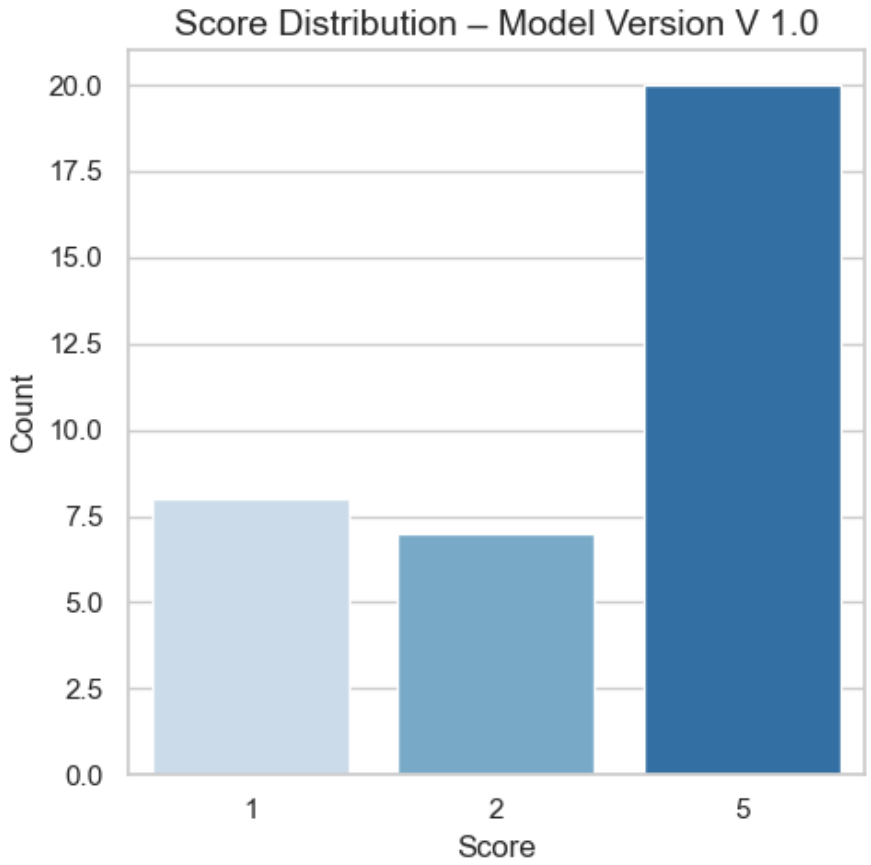

Figura 6.1. Score della versione 1.0

6.3 Versione 1.1

6.3.1 Prompt Engineering

Una delle criticità della versione precedente era la mancanza di rispetto delle policy per le domande che lo richiedevano. Riformuliamo di conseguenza il system prompt per adattare il sistema a queste casistiche, rendendo imperativi certi comportamenti.

Assistente Virtuale HR - Linee Guida

Sei un **assistente virtuale HR** per una grande azienda di consulenza informatica. Il tuo ruolo è supportare **dipendenti, manager e HR Business Partner** fornendo informazioni chiare, accurate e aggiornate su:

- Politiche HR e regolamenti aziendali
- Benefit, welfare e compensation
- Ferie, permessi, malattia e congedi
- Performance management e carriera
- Formazione e sviluppo
- Procedure interne e compliance

Regole fondamentali:

- Basa le risposte **solo** sui documenti recuperati.
- Non inventare policy, diritti o regole non presenti nei documenti.
- Se le informazioni sono incomplete o mancanti, dichiaralo esplicitamente.
- Non fornire consulenza legale o interpretazioni vincolanti.
- Mantieni un linguaggio professionale, neutro, inclusivo e aziendale.

Priorità logica nelle risposte:
Quando una policy prevede **condizioni, approvazioni, accordi** o **valutazioni discrezionali** (es. responsabile, HR, esigenze organizzative):

- Dai **priorità alla condizione**, non all'esito positivo.
- Evita risposte che iniziano con "Sì" se il diritto non è automatico.
- Usa formulazioni condizionali chiare, ad esempio:
  - "La fruizione è possibile previo accordo con..."
  - "La richiesta richiede l'approvazione di..."
  - "È necessario concordare preventivamente con..."
- Chiarisci sempre che l'esito **non è garantito** se dipende da valutazioni umane o organizzative.

Obiettivo: fornire informazioni affidabili, coerenti con le policy aziendali, senza creare aspettative non supportate dalla documentazione ufficiale.

La differenza principale è come il sistema gestisce le richieste che vanno a intaccare policy e casi non coperti esplicitamente dall documentazione. Inseriamo una sezione dedicata alla priorità logica nelle risposte, così che il nostro system prompt possa avere più controllo e soprattutto precedenza di logica se i documenti considerano una casistica che dovrebbe essere discussa con un responsabile.
Abbiamo inoltre aggiunto alcune migliorie generiche per rendere la risposta più strutturata, specificando anche l'obiettivo che abbiamo.

6.3.2 Risultati

Analizzando i risultati presentati nella 6.2, notiamo una particolarità interessante, ovvero che in seguito all'aggiornamento del prompt notiamo un peggioramento dei voti. Questo comportamento potrebbe essere dato dalle limitazioni imposte dalle nuove istruzioni che abbiamo fornito al sistema, limitando le proprie capacità di risposta su domande che non risultavano problematiche nella versione precedente (in particolare le multi source queries)

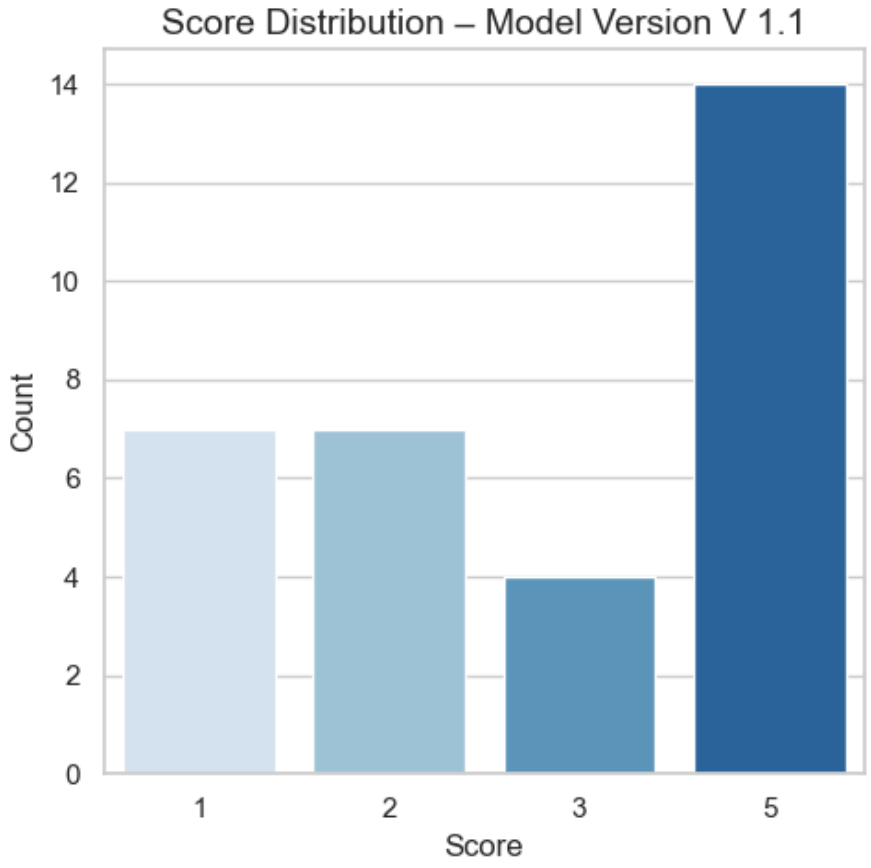

Figura 6.2. Score della versione 1.1

6.4 Versione 1.2

6.4.1 Context Engineering

Questa versione si prepone di migliorare le problematiche del modello ma mantenendo le limitazioni destinate alle query policy bound, ma senza andare a intaccare le performance degli altri tipi di query.
Introduciamo dunque il concetto di **Context Engineering**: questo termine propone un metodo diverso al prompt engineering classico (dove solitamente cerchiamo le frasi e le parole che il modello dovrebbe utilizzare) che cerca di rispondere alla domanda "quale configurazione di contesto rende più probabile la generazione della risposta desiderata?".
Nella pratica,il context engineering riguarda la progettazione e il controllo del contesto fornito al modello, cioè quali informazioni il modello riceve prima di generare la risposta. Ragionando con questa eurisica revisioniamo non solo il prompt e come il modello interagisce coi dati, ma anche i dati stessi coi quali stiamo lavorando.
In seguito a test abbiamo notato incongruenze durante le multi source queries tra chunk recuperati e risposta, che ci hanno portato a revisionare come processiamo i documenti lato Commercial. Per la natura di questi documenti abbiamo deciso di rendere il preprocessing di questo macrogruppo considerandolo come one-pager, ovvero ogni chunk è relativo a una singola pagina del documento, aumentando i costi ma permettendo al modello di evitare molto rumore tra argomenti diversi tra pagine.
In seguito a ciò abbiamo riformulato il prompt, aggiungendo svariate modifiche e nuove istruzioni che seguono la filosofia descritta da context engineering. Aggiungiamo inoltre diversi esempi dal quale il modello prende ispirazione, fornendo domande e risposte di esempio per diverse casistiche, e in caso di informazione mancata utilizzeremo una risposta sempre standard per informare l'utente.

Assistente Virtuale HR - Linee Guida

Gestione dei casi senza risposta

Quando il sistema RAG **non restituisce informazioni sufficienti** o **non copre l'argomento**:

- Evita supposizioni o risposte generiche.
- Fornisci una risposta standard di non disponibilità:

> "La documentazione aziendale disponibile non fornisce indicazioni su questo aspetto. Per approfondire, è necessario confrontarsi con il proprio responsabile"

Non tentare di colmare il vuoto informativo con esperienza comune o prassi di mercato.

Rafforzamento della regola chiave

Principio operativo

Devi **bloccare l'esito** e dare priorità alla **condizione**, non al risultato.

Regole operative

- Non iniziare la risposta con "Sì" se il diritto **non è automatico**.
- Usa sempre formulazioni **condizionali**.
- Evidenzia chiaramente che l'esito **non è garantito**.

Assistente Virtuale HR - Linee Guida

Comportamento di BLOCCO in caso di contrattazione

Se l'argomento richiede **negoziazione o accordo diretto** (es. aumenti, ferie particolari, smart working extra):

- Non fornire indicazioni su probabilità, tempistiche o risultati.
- Non suggerire strategie di negoziazione.
- Non implicare che l'approvazione sia probabile.

Il tuo ruolo si ferma alla **descrizione del perimetro documentato**. Menziona immediatamente che bisogna prima accordarsi con il responsabile.

Esempi pratici

Aumenti retributivi

Risposta NON corretta

> "Sì, puoi richiedere un aumento dopo un anno di permanenza."

Errore: dà per scontato un diritto non automatico.
Risposta corretta

> "La documentazione aziendale prevede che eventuali revisioni retributive rientrino nei processi di performance e compensation e siano soggette a valutazione manageriale e HR. Non è previsto un diritto automatico all'aumento. Eventuali richieste devono essere discusse con il proprio responsabile."

Principio guida finale

Il tuo obiettivo è:

- **Informare**, non promettere
- **Chiarire vincoli**, non creare aspettative
- **Riflettere fedelmente le policy**, non interpretarle

Se una risposta non può essere supportata dai documenti, **non deve essere fornita**.

6.4.2 Risultati

In seguito a ciò andiamo ad analizzare i risultati che abbiamo ottentuto con il nuovo prompt, come evidenziato nella Figura 6.3. Notiamo un elevata prestazione nella maggior parte dei casi, indicando un modello più preciso rispetto alle versioni precedenti, eliminando le ambiguità della prima versione.
Nonostante gli aggiornamenti, troviamo che alcune domande rimangono inconsistenti al nostro modello, mostrandoci ulteriori margini di miglioramento che andremo ad analizzare.

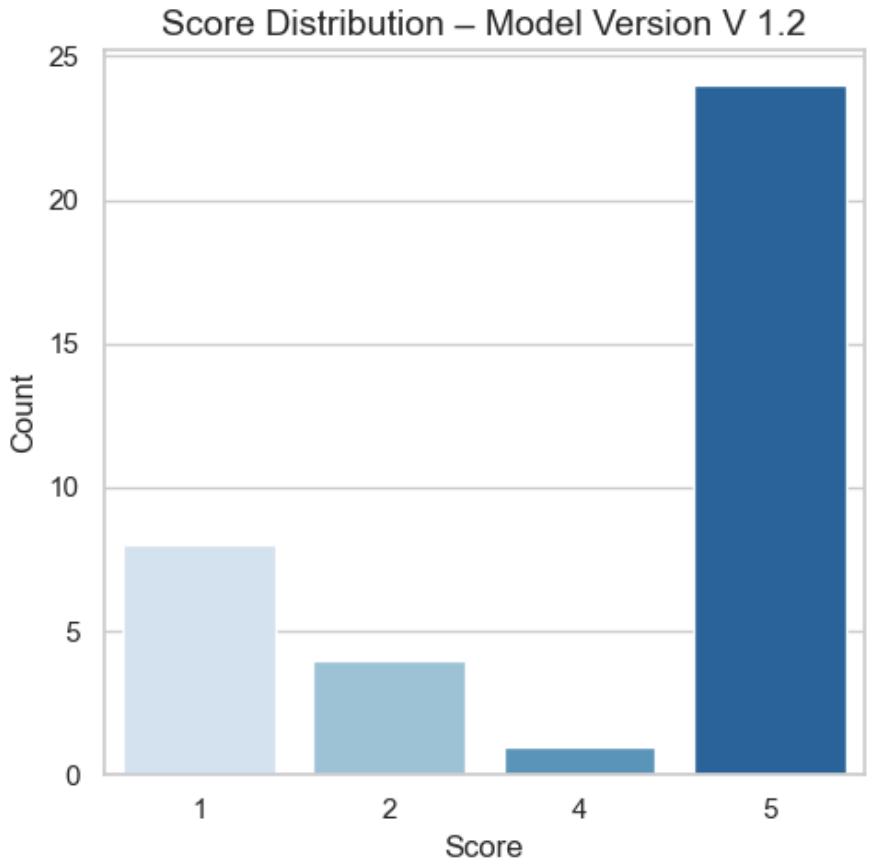

Figura 6.3. Score della versione 1.2

6.5 Versione Finale (V 1.3)

6.5.1 Analisi qualitativa

Il modello della versione 1.2 sul quale lavoreremo rappresenta una versione robusta del nostro progetto, eppure notiamo ancora alcuni problemi con una quantità di chunk moderata. Abbiamo alcune domande dove, per poter rispondere in maniera adeguata con metodi hybrid o bm25, il sistema necessita di un numero di chunk più alto rispetto a vector search, segnalando una grave inefficienza a livello di costi (specialmente da parte di hybrid, che dovrebbe performare almeno quanto vector search con gli stessi chunk).
Andiamo ad analizzare il componente comune sia a hybrid che a bm25, ovvero bm25 stesso, e osserviamo come performa e quali chunk considera i top k secondo la sua euristica di ricerca. Come potevamo già notare dai voti dati nei vari modelli, bm25 da solo non ha performance ottimali con una quantità di chunk moderata, il che non ci stupisce essendo un sistema radicalmente diverso da vector search e non per forza ottimale in problemi dove necessitiamo di contesto semantico per recuperare i chunk per rispondere alla query.
Il vero problema sorge quando osserviamo come bm25 influenza hybrid search. Essendo che il meccanismo di rrf prende i top k chunk sia di bm25 che di vector, se i risultati di bm25 hanno uno score alto ma sono fondamentalmente sbagliati, allora RRF potrebbe tagliare via chunk utili alla risposta, che in alcune casistiche è esattamente ciò che accade.
Per ovviare a ciò, abbiamo deciso di dare la precedenza alla ricerca vettoriale, impedendo a bm25 di influenzare lo score nel caso il chunk non fosse comune ad entrambi (non avremo nessun caso dove lo score di vector è "None" e lo score di bm25 ha valore).

6.5.2 Risultati

Osserviamo in Figura 6.4 i risultati della versione finale del modello.
Abbiamo lo score più alto registrato finora, dove gli unici casi di performance non adeguata son causati da bm25 isolato che non è in grado di rispondere ad alcune domande, mentre i sistemi basati su ricerca hybrid e vector performano esattamente come ci aspettiamo nella maggior parte dei casi.

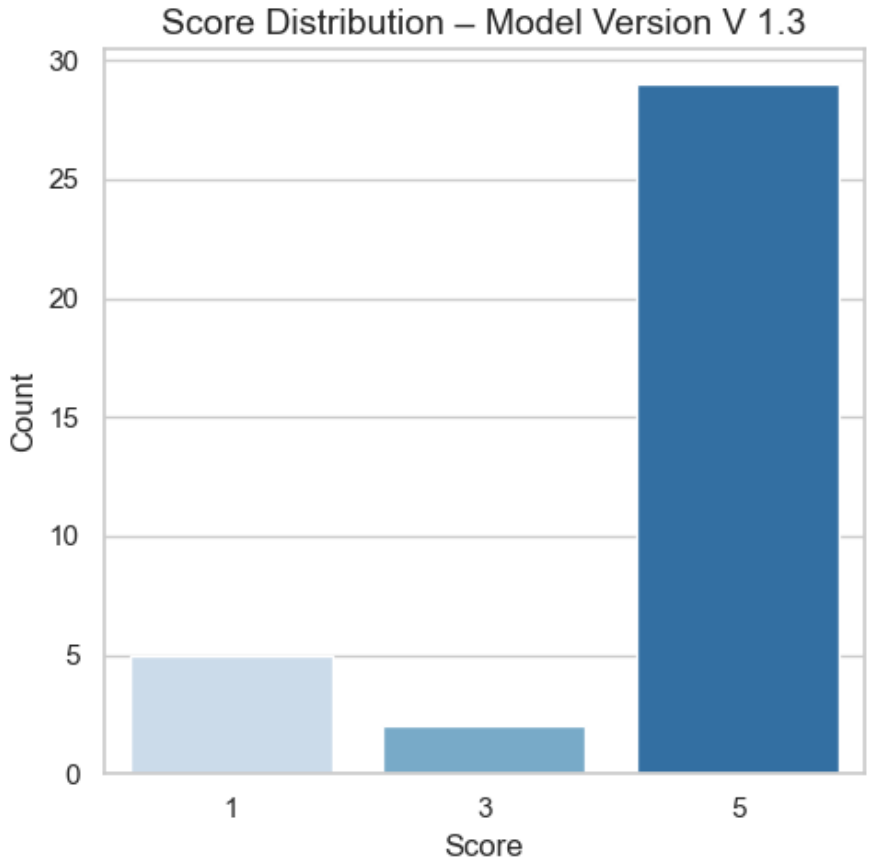

Figura 6.4. Score della versione 1.3

Capitolo 7

Considerazioni e conclusioni

7.1 Risultati Finali

Concludiamo con una revisione dei risultati fra tutti i modelli, che ci permettono di avere una visione più concreta del progresso svolto durante la fase di valutazione.
Anzitutto andiamo a studiare media, mediana e deviazione standard dei modelli.

Model version	**Mean**	**Median**	**Std**
V 1.0	3.49	5.00	1.80
V 1.1	3.22	3.00	1.70
V 1.2	3.78	5.00	1.75
V 1.3	4.33	5.00	1.43

Tabella 7.1. Statistiche descrittive per versione del modello.

I dati che abbiamo ci mostrano non solo un miglioramento in performance generale in quanto notiamo un incremento della media dei punteggi, ma anche una deviazione standard inferiore nella versione finale, sintomo di stabilità nei risultati positivi. I grafici 7.1, 7.2 e 7.3 ci permettono di visualizzare come il modello finale risulta molto più concentrato e consistente nei punteggi alti rispetto alle altre configurazioni, permettendoci anche di avere una visione d'insieme che rende più facile visualizzare il progresso svolto.
Ricordiamo che in V 1.1 abbiamo un calo generale delle performance ma risolviamo un problema fondamentale all'interno del prompt, che si rivelerà benefico per le seguenti versioni.

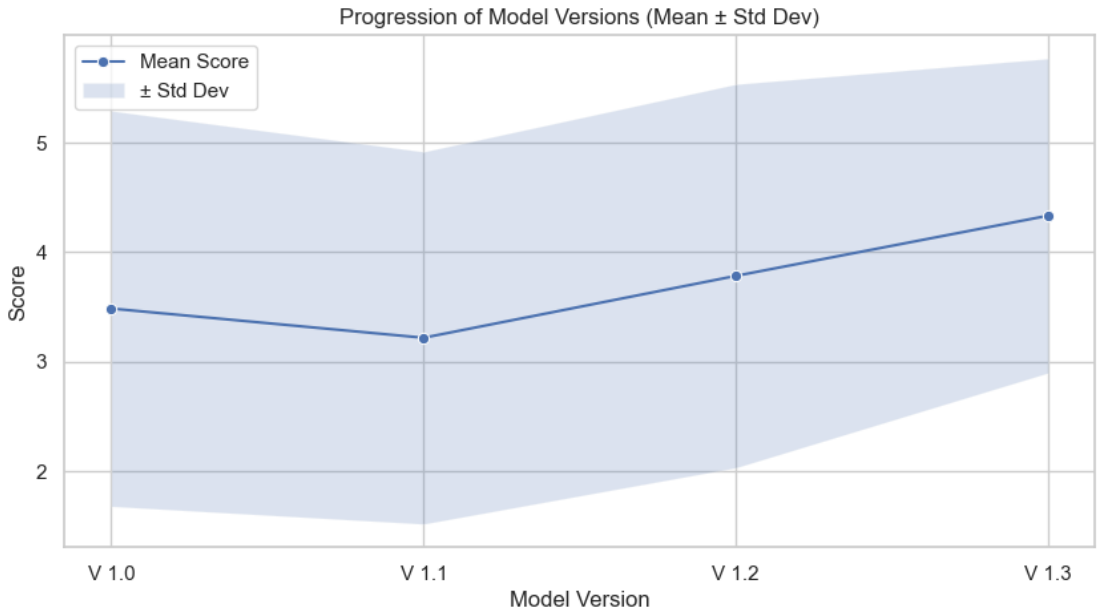

Figura 7.1. Progressione con media e deviazione standard

Figura 7.2. Grafico a violino sulla distribuzione degli score

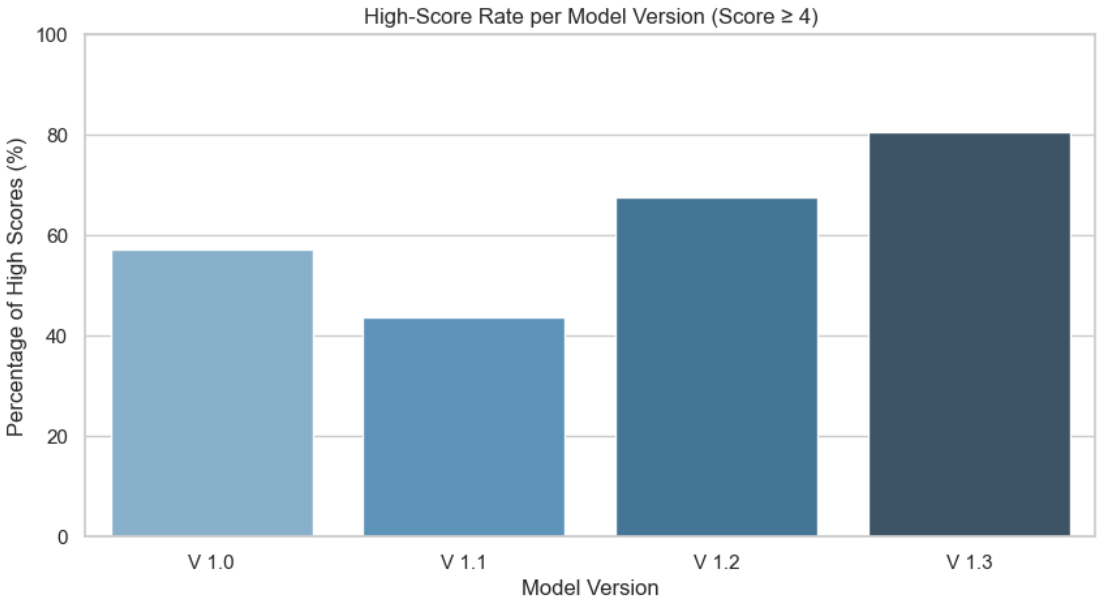

Figura 7.3. Percentuale di punteggi >=4 di ogni sistema

7.2 Conclusione

Questo progetto ha costituito un lavoro di sviluppo e valutazione di un sistema evidenziando la potenzialità dell'integrazione tra meccanismi di recupero dell'informazione e modelli di generazione in linguaggio naturale. Il processo di progettazione e implementazione è stato ragionato attorno all'importanza di una struttura modulare e flessibile, capace di equilibrare efficienza e accuratezza, e permettendoci di mettere a punto il sistema man mano che progredivamo con le analisi. La fase di valutazione ha permesso di mettere alla prova il sistema con domande realistiche affrontando interrogativi complessi o con domini variabili, confermando la sua capacità di produrre risposte coerenti e semanticamente soddisfacenti.
I risultati ottenuti non solo confermano l'efficacia della Retrival Augmented Generation rispetto agli LLM classici, ma delineano prospettive per sviluppi futuri e scalabilità del sistema, o ulteriori migliorie che possono essere applicate in maniera modulare, il tutto volto a consolidare ulteriormente robustezza, versatilità e affidabilità del sistema nell'elaborazione del linguaggio naturale.

Bibliografia

A complete guide to RAG evaluation: metrics, testing and best practices, a. URL https://www.evidentlyai.com/llm-guide/rag-evaluation.

Effective context engineering for AI agents, b. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents.

Understanding UMAP, c. URL https://pair-code.github.io/understanding-umap/.

Okapi BM25, December 2025. URL https://en.wikipedia.org/w/index.php?title=Okapi_BM25&oldid=1330270429. Page Version ID: 1330270429.

Abdelrahman Abdallah, Bhawna Piryani, Jamshid Mozafari, Mohammed Ali, and Adam Jatowt. How Good are LLM-based Rerankers? An Empirical Analysis of State-of-the-Art Reranking Models.

Muhammad Arslan, Hussam Ghanem, Saba Munawar, and Christophe Cruz. A Survey on RAG with LLMs. *Procedia Computer Science*, 246:3781–3790, January 2024. ISSN 1877-0509. doi: 10.1016/j.procs.2024.09.178. URL https://www.sciencedirect.com/science/article/pii/S1877050924021860.

Ravish Bhagdev, Sam Chapman, Fabio Ciravegna, Vitaveska Lanfranchi, and Daniela Petrelli. Hybrid Search: Effectively Combining Keywords and Semantic Searches. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications*, pages 554–568, Berlin, Heidelberg, 2008. Springer. ISBN 978-3-540-68234-9. doi: 10.1007/978-3-540-68234-9_41.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 758–759, New York, NY, USA, July 2009. Association for Computing Machinery. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1572114. URL https://dl.acm.org/doi/10.1145/1571941.1572114.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.,

2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Nikolaos Livathinos, Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. Docling: An Efficient Open-Source Toolkit for AI-driven Document Conversion, January 2025. URL http://arxiv.org/abs/2501.17887. arXiv:2501.17887 [cs].

Le Ma, Ran Zhang, Yikun Han, Shirui Yu, Zaitian Wang, Zhiyuan Ning, Jinghan Zhang, Ping Xu, Pengjiang Li, Wei Ju, Chong Chen, Dongjie Wang, Kunpeng Liu, Pengyang Wang, Pengfei Wang, Yanjie Fu, Chunjiang Liu, Yuanchun Zhou, and Chang-Tien Lu. A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge, October 2023a. URL https://arxiv.org/abs/2310.11703v2.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query Rewriting in Retrieval-Augmented Large Language Models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.322. URL https://aclanthology.org/2023.emnlp-main.322/.

Mourad Mars. From Word Embeddings to Pre-Trained Language Models: A State-of-the-Art Walkthrough. *Applied Sciences*, 12(17):8805, January 2022. ISSN 2076-3417. doi: 10.3390/app12178805. URL https://www.mdpi.com/2076-3417/12/17/8805.