UNIVERSITÀ DEGLI STUDI DI BARI
"ALDO MORO"

DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA TRIENNALE IN INFORMATICA

TESI DI LAUREA

In Metodi per il Ritrovamento dell'Informazione

Large Language Models e Retrieval-Augmented Generation: un caso di studio per l'accesso ai bandi regionali della Puglia

Relatori:
Prof. Pasquale Lops
Prof. Pierpaolo Basile

Laureando:
Giannantonio Sanrocco

ANNO ACCADEMICO 2023/2024

Indice

Elenco delle figure **vii**

Elenco delle tabelle **ix**

1 Introduzione **1**

2 Stato dell'arte **5**

2.1 Introduzione 5

2.2 Retrieval Augmented Generation 6

2.2.1 Implementazione RAG 7

2.2.2 Applicazioni della RAG 8

2.2.3 Architettura RAG e funzionamento 8

Generatori 9

2.2.4 Retriever 9

Sparse Retriever 9

Dense Retriever 10

2.2.5 Tipi di Integrazione: 10

2.2.6 Classificazione dei sistemi RAG: 11

Query-based RAG 11

Latent Representation-based RAG 11

Logit-based RAG 12

Speculative RAG 12

2.2.7 Metodi per migliorare le performance della RAG 12

Miglioramento dell'input: 13

Miglioramento del Retriever: 13

Miglioramento del Generatore: 14

Miglioramento del risultato: 14

Miglioramento della pipeline RAG: 14

2.3 Fine-tuning dei LLM 15

2.3.1 Fasi del Fine-Tuning: 15

Pre-addestramento Iniziale: 15

Fine-Tuning Specifico per il Compito: 15

2.3.2 Tipi di Fine-Tuning: . . . 16
Full Fine-Tuning: . . . 16
Parameter-Efficient Fine-Tuning (PEFT): . . . 16
2.3.3 Vantaggi del Fine-Tuning . . . 16
Migliore performance su compiti specifici: . . . 16
Riduzione dei Bias: . . . 16
2.3.4 Svantaggi del Fine-Tuning . . . 16
Catastrophic Forgetting: . . . 16
Overfitting: . . . 16
Costi computazionali e di memoria: . . . 17
2.4 Prompt Engineering . . . 17
2.5 Un confronto delle diverse tecniche . . . 18
2.6 Problematiche aperte . . . 20
2.6.1 Knowledge shortcut . . . 21
2.6.2 Riconoscimento dei Limiti della Conoscenza . . . 21
2.6.3 Bilanciamento tra Creatività e Fattualità . . . 21
2.6.4 Conoscenze Obsolete . . . 21
2.6.5 Gestione di scenari di ragionamento complessi . . . 21
2.7 Sistemi basati su LLM nel contesto della Pubblica Amministrazione e dei governi . . . 22
2.7.1 Sistemi basati su LLaMandement . . . 23
2.7.2 CensusGPT . . . 24
2.7.3 Wobby . . . 25
2.7.4 NEPAccess . . . 26
2.7.5 Parla . . . 27

3 Metodologia **29**
3.1 Framework . . . 29
3.2 Componenti principali di LLamaIndex . . . 29
3.3 Architettura e funzionamento applicazione . . . 31

4 Implementazione **33**
4.1 Configurazione del LLM per il sistema RAG . . . 33
4.2 Gradio . . . 34
4.3 Caricamento dei documenti . . . 34
Ottimizzazione dei chunks . . . 35
4.4 Pinecone . . . 35
4.5 Creazione del Retriever Ibrido . . . 38
4.6 Response mode e Prompt Template . . . 38

4.7 Filtraggio con metadati 39
4.8 Streaming 40

5 Sperimentazione **41**
5.1 Caratteristiche principali del sistema: 41
5.2 Modalità di utilizzo: 41
5.3 Documenti indicizzati 42
5.4 Tempi di recupero nodi 43
5.5 Tempi di risposta 44
5.6 Risultati Sperimentazione 45

6 Conclusioni **51**

Bibliografia **53**

Elenco delle figure

2.1 Architettura RAG. 6
2.2 Tassonomia delle applicazioni della RAG 8
2.3 Architettura RAG. 8
2.4 Classificazione dei sistemi RAG 11
2.5 Metodi per migliorare le performance della RAG 12
2.6 Fine-Tuning di un LLM 15
2.7 Prompt Engineering 17

3.1 Architettura RAG. 31

4.1 Configurazione parametri LLM 33
4.2 Costruzione indice ibrido 37
4.3 Query su indice ibrido 37
4.4 Response Mode "Compact" 38
4.5 Response Mode "Tree Summarize" 39

Elenco delle tabelle

2.1 Confronto tra RAG, Fine-Tuning e Prompt Engineering 18

Capitolo 1

Introduzione

Nell'era digitale in cui viviamo, l'accesso istantaneo a informazioni accurate e aggiornate è diventato cruciale per molte sfere della vita quotidiana e professionale. Tuttavia, la capacità dei modelli di linguaggio (LLM) di generare risposte precise e contestualmente rilevanti è spesso limitata dalla portata dei dati su cui sono stati addestrati. Questo divario tra la conoscenza incorporata nei modelli e l'evoluzione costante delle informazioni disponibili può compromettere l'efficacia delle risposte generate. Altri limiti dei LLM (Large Language Model) includono:

- **Mancanza di contesto**: Senza informazioni aggiuntive, i LLM possono avere difficoltà a comprendere il contesto di una richiesta o a interpretare correttamente il significato implicito di una frase. Questo può portare a risposte imprecise o non pertinenti.
- **Generazione di testo non affidabile**: A causa della natura probabilistica dei LLM, le risposte generate possono essere inaccurate o fuorvianti, soprattutto quando si tratta di informazioni tecniche o specifiche di dominio.
- **Difficoltà con l'interpretazione e la spiegazione**: I LLM possono avere difficoltà a spiegare il ragionamento dietro alle loro risposte o fornire le fonti utilizzate per generare la risposta, rendendo difficile per gli utenti comprendere e fidarsi delle risposte generate.
- **Presentazione di informazioni false quando non si ha la risposta.**
- **Presentazione di informazioni generiche quando l'utente si aspetta una risposta specifica e attuale.**

Per affrontare questo problema utilizzeremo la Retrieval-Augmented Generation (RAG), una metodologia che ottimizza l'output dei LLM incorporando dati esterni aggiornati prima di generare una risposta. La RAG agisce come un

ponte tra la vasta conoscenza dei LLM e le informazioni più recenti provenienti da fonti esterne come database, API, archivi di documenti e pagine web. In questa tesi, esamineremo da vicino i vantaggi della RAG nell'ambito specifico dei bandi regionali della Puglia, illustrando come questa metodologia possa migliorare l'accesso a informazioni rilevanti e mitigare i limiti dei LLM precedentemente citati. E' fondamentale che la regione e i cittadini possano accedere e recuperare velocemente informazioni affidabili e sempre aggiornate sui bandi disponibili poichè i bandi regionali possono essere soggetti a frequenti aggiornamenti normativi e modifiche nei criteri di selezione. L'implementazione di un'applicazione basata su RAG (Retrieval Augmented Generation) permetterà alla regione Puglia e ai cittadini di porre domande sui bandi al sistema e ricevere risposte contestualmente accurate e aggiornate grazie alla capacità della RAG di integrare le conoscenze di un LLM pre-addestrato (LLaMAntino, sviluppato internamente dall'Università degli Studi di Bari Aldo Moro) con dati specifici di dominio memorizzati in un repository di conoscenze facilmente aggiornabile. Inoltre, il sistema basato su RAG permetterà agli utilizzatori di visualizzare le fonti utilizzate per generare la risposta, così come verranno mostrati gli specifici paragrafi di testo recuperati dal repository della RAG per migliorare l'output generato dal LLM. In questo modo, andremo ad aumentare la "trasparenza" del sistema e la fiducia degli utilizzatori nei confronti delle risposte generate. Nel Capitolo 2 sullo stato dell'arte analizzeremo i meccanismi di funzionamento del sistema Retrieval-Augmented Generation (RAG), discutendo in dettaglio come un sistema RAG può essere implementato, la sua architettura e i componenti fondamentali che lo costituiscono. Classificheremo i diversi tipi di RAG e i metodi attualmente usati per migliorare le performance di un sistema RAG, includendo un confronto tra la tecnica RAG e altre alternative per la mitigazione dei limiti dei Large Language Models (LLM), come il Fine-Tuning e il Prompt Engineering. Inoltre, esamineremo i sistemi attualmente esistenti basati su LLM che supportano cittadini e dipendenti di enti pubblici attraverso risposte generate utilizzando dati governativi o prodotti da vari enti pubblici. Nel Capitolo 3 descriveremo la metodologia adottata per risolvere il problema inizialmente individuato. Esploreremo l'architettura del sistema RAG sviluppato, le sue varie componenti e il framework utilizzato, LlamaIndex. Questo capitolo fornirà una panoramica dettagliata dei passi seguiti per progettare e implementare il sistema, mettendo in luce le decisioni chiave prese durante lo sviluppo. Nel Capitolo 4 ci concentreremo sui dettagli implementativi. Spiegheremo le librerie utilizzate per la creazione dell'interfaccia, il processo di caricamento e di indicizzazione dei documenti, il database vettoriale scelto e il metodo di creazione

dell'indice. Discuteremo il tipo di indice implementato, il processo di creazione del retriever e le sue caratteristiche. Verranno inoltre illustrati i dettagli tecnici relativi alla configurazione del LLM per il sistema RAG, inclusi i parametri di quantizzazione e generazione delle risposte. Descriveremo l'integrazione con Pinecone per la gestione dei dati vettoriali e l'ottimizzazione del retrieval tramite la ricerca ibrida. Infine, discuteremo l'importanza dello streaming delle risposte per migliorare l'interazione con l'utente e l'efficienza del sistema.

Capitolo 2

Stato dell'arte

2.1 Introduzione

Negli ultimi anni, la generazione di contenuti basata su Intelligenza Artificiale ha avuto una trasformazione radicale. Questo progresso è stato principalmente guidato dallo sviluppo di modelli di intelligenza artificiale sempre più potenti e versatili, capaci di produrre una grande varietà di contenuti che va da testi scritti a immagini, video e oltre. Alcuni esempi sono i LLM (modelli di linguaggio di grandi dimensioni), come la serie GPT (Generative Pre-Trained Transformer) e BERT (Bidirectional Encoder Representations from Transformers), i quali hanno rivoluzionato il settore della generazione di testo. Altri modelli generativi come DALL-E e Stable Diffusion sono in grado di generare immagini a partire da descrizioni testuali semplici, mentre modelli come Sora possono generare video o modificare quelli esistenti. L'introduzione di algoritmi innovativi e di nuove architetture e ha permesso ai modelli generativi di raggiungere prestazioni notevoli. In particolare, l'architettura dei modelli, inizialmente costituita da milioni di parametri è adesso in grado di incorporare miliardi di parametri. Questi avanzamenti sono stati ulteriormente rafforzati dalla disponibilità di dataset di alta qualità e ricchi di informazioni, i quali forniscono esempi di training per ottimizzare pienamente i parametri del modello. Nonostante gli incredibili progressi fatti dagli avanzati modelli generativi, il settore continua a incontrare sfide significative, tra cui la difficoltà nel mantenere conoscenze aggiornate e approfondite e il rischio di fuga di dati sensibili. I modelli di linguaggio pre-addestrati hanno dimostrato di esser capaci di apprendere una quantità sostanziale di conoscenze approfondite dai dati usati per l'addestramento del modello. Possono farlo senza accedere a una memoria esterna. Questi modelli, però, hanno anche degli svantaggi: non possono facilmente espandere o rivedere la loro memoria e possono produrre "allucinazioni". I modelli ibridi che combinano memoria parametrica con memorie non parametriche (basate sul recupero) possono affrontare alcuni di questi problemi perché la

conoscenza può essere direttamente rivista ed espansa, mentre la conoscenza accessibile può essere ispezionata e interpretata.

2.2 Retrieval Augmented Generation

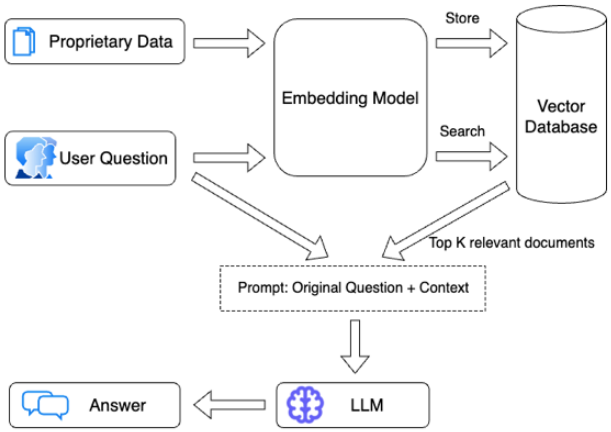


FIGURA 2.1: Architettura RAG.

La Retrieval-Augmented Generation(RAG) è il processo di ottimizzazione dell'output di un modello di grandi dimensioni, in modo che faccia rifermento a dati esterni prima di generare una risposta. I nuovi dati al di fuori del set di dati su cui il LLM è stato addestrato sono chiamati dati esterni, possono provenire da molte sorgenti come API, database o archivi di documenti. Questi dati esterni sono convertiti in una rappresentazione numerica grazie ad un embedding model e memorizzati in un database vettoriale. Con la RAG, viene introdotto un componente di recupero delle informazioni che utilizza l'input dell'utente per estrarre informazioni da una nuova sorgente dati. La richiesta dell'utente e le informazioni rilevanti al contesto sono fornite al LLM. Il LLM utilizza le nuove conoscenze e i suoi dati di training per creare risposte più aggiornate, accurate e contestuali. Un vantaggio significativo della RAG è la sua capacità di ridurre la dimensione complessiva dei modelli generativi. Integrare il recupero delle informazioni direttamente nel processo di generazione del testo elimina la necessità di includere tutte le conoscenze all'interno dei parametri del modello stesso. Questo approccio ottimizza le risorse computazionali, migliorando l'efficienza e la velocità di elaborazione, senza compromettere la qualità delle risposte generate. Come citato negli articoli [2] e [6] la Retrieval Augmented Generation riduce significativamente le allucinazioni del LLM e la produzione

di informazioni false. Inoltre, l'integrazione delle informazioni recuperate dalla base di conoscenza esterna permette di generare risposte contestualmente rilevanti contenenti le informazioni più accurate e recenti.

2.2.1 Implementazione RAG

Per poter implementare un sistema basato su RAG è necessario l'utilizzo di:

- **Framework**: Essi forniscono librerie per sviluppare applicazioni in modo più efficiente e coerente. Facilitano l'integrazione dei vari componenti del sistema permettendo agli sviluppatori di concentrarsi sulla logica dell'applicazione senza dover gestire i dettagli di basso livello. Esempi di framework comunemente utilizzati per implementare sistemi RAG includono LLamaIndex e LangChain.

- **Modello embedding**: Il modello embedding è responsabile della trasformazione di porzioni di testo in rappresentazioni numeriche dense, chiamate embedding. Queste rappresentazioni vettoriali sono fondamentali per confrontare e cercare somiglianze tra i testi. Modelli come BERT o altri modelli di embedding pre-addestrati sono comunemente utilizzati per questo scopo, poiché riescono a catturare efficacemente il significato semantico delle parole e delle frasi.

- **Database vettoriale (vector store)**: Il database vettoriale è un sistema di archiviazione ottimizzato per la gestione e la ricerca di dati vettoriali ad alta dimensionalità. Questo tipo di database è essenziale per memorizzare gli embedding e permette di eseguire ricerche efficienti basate sulla similarità.

- **LLM (Large Language Model)**: Il Large Language Model è un modello di linguaggio di grandi dimensioni che viene utilizzato per generare risposte e fornire informazioni contestuali in base ai dati di input. Questi modelli, come GPT-3 o GPT-4, sono addestrati su enormi quantità di dati testuali e sono in grado di comprendere e generare testo in modo altamente coerente e rilevante.

2.2.2 Applicazioni della RAG

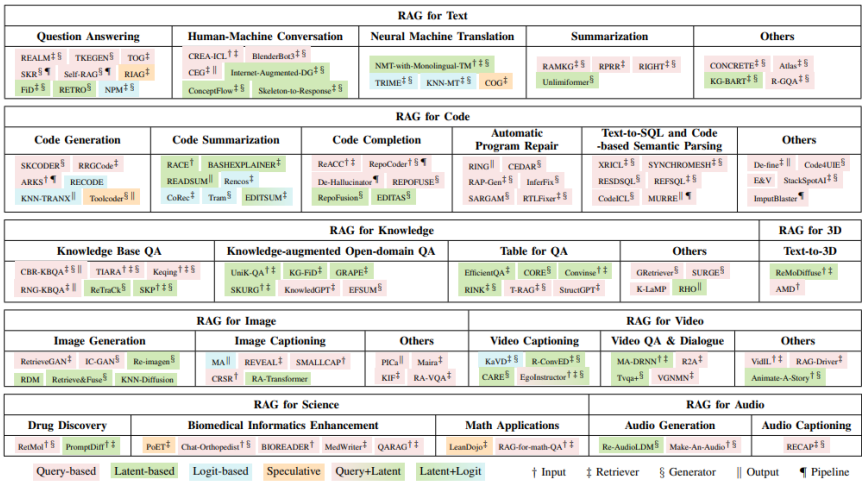


FIGURA 2.2: Tassonomia delle applicazioni della RAG

La RAG (Retrieval-Augmented Generation) trova applicazione in diversi settori, migliorando l'efficienza e l'efficacia dei modelli generativi. La RAG può essere usata in vari campi:

- **Generazione di testo**
- **Generazione di immagini**
- **Generazione di codice**
- **Generazione di audio**
- **Generazione di video**
- **Generazione di contenuti 3D**

2.2.3 Architettura RAG e funzionamento

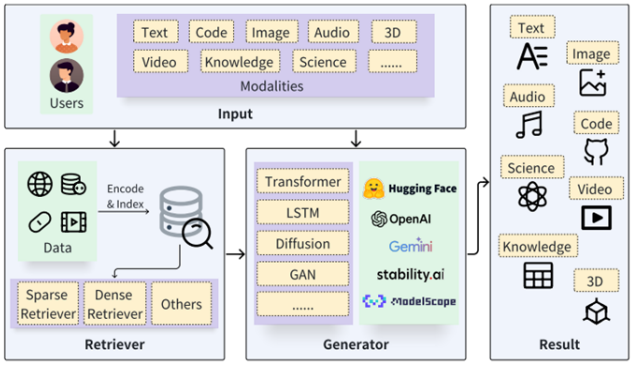


FIGURA 2.3: Architettura RAG.

Il tipico processo RAG è il seguente: data una query di input, il retriever individua sorgenti di dati pertinenti, successivamente i risultati recuperati interagiscono con il generatore per migliorare il processo generale di generazione. I risultati recuperati possono interagire con il generatore in modi diversi: possono servire per aumentare l'input fornito al generatore, possono essere usati nello stato intermedio di generazione come rappresentazioni latenti, oppure influenzare o omettere alcuni step di generazione. Nonostante il concetto di RAG è inizialmente emerso nell'area della generazione di testo, esso è stato anche adattato a vari domini, come generazione di codice, audio, immagini, video, 3D e conoscenza.

Generatori

I generatori nella RAG trasformano le informazioni recuperate in risposte comprensibili, utilizzando le informazioni come input supplementare. Per ogni scenario viene scelto il modello generativo più adatto. Per esempio, transformer model per task text-to-text , VisualGPT per task image-to-text, Stable Diffusion per task text-to-image, e Codex per task text-to-code.

2.2.4 Retriever

Un retriever è un componente all'interno di un sistema RAG che ha il compito di cercare e recuperare informazioni pertinenti da una grande base di dati in risposta a una query specifica. La funzione principale di un retriever è identificare e ottenere risorse informative che soddisfano un bisogno informativo, fornendo contesto e supporto al generatore per migliorare l'accuratezza e la rilevanza del contenuto generato.

Sparse Retriever

Gli Sparse Retriever sono ampiamente utilizzati nel recupero di documenti. Questi retriever sfruttano metriche di corrispondenza dei termini come TF-IDF (Term Frequency-Inverse Document Frequency) per recuperare documenti a partire da una query. TF-IDF calcola la rilevanza di un termine in un documento basandosi sulla frequenza del termine nel documento e sulla rarità del termine nell'intero corpus. Si calcola un vettore TF-IDF per la query e un vettore TF-IDF per il documento, dopodichè, si calcola la similarità tra il vettore TF-IDF della query e i vettori TF-IDF dei documenti per determinare la rilevanza dei documenti.

Dense Retriever

Diversamente dagli Sparse Retriever, i Dense Retriever rappresentano query e documenti usando vettori embedding densi. Le query e i documenti vengono convertiti in vettori densi tramite modelli di embedding pre-addestrati, come BERT o altri modelli di linguaggio basati su Transformer. Questi modelli codificano il testo in vettori che catturano le informazioni semantiche complesse. Gli embedding dei documenti vengono indicizzati utilizzando strutture di dati avanzate come indici di nearest neighbor approssimati (ANN) che permettono di cercare rapidamente e accuratamente i vettori embedding più simili all'interno di uno spazio vettoriale ad alta dimensione.

2.2.5 Tipi di Integrazione:

- **Input Augmentation**: L'informazione recuperata viene integrata direttamente all'inizio del processo di generazione come parte dell'input.
- **Latent Representation**: Le rappresentazioni latenti dei dati recuperati possono essere combinate durante la generazione per migliorare la qualità e la pertinenza del contenuto generato.
- **Logits Integration**: Durante la decodifica, i logit derivati dalle informazioni recuperate vengono combinati con quelli del modello per influenzare la generazione finale.

Come citato nell'articolo [7] solitamente si integrano sequenzialmente la query iniziale e i documenti recuperati, per poi passare tutto a un modello generativo pre-addestrato.

2.2.6 Classificazione dei sistemi RAG:

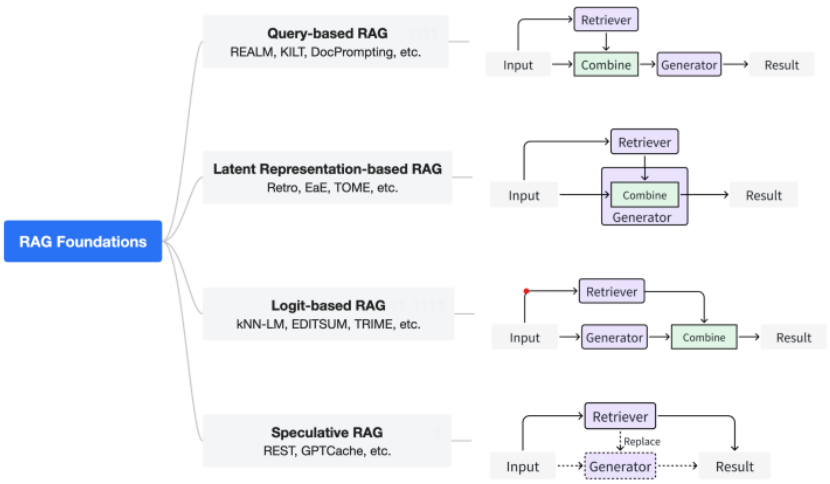


FIGURA 2.4: Classificazione dei sistemi RAG

Come definito nell'articolo di Wentao Zhang [10], possiamo classificare i sistemi RAG in: Query-based RAG, Latent Representation-based RAG , Logit-based RAG, Speculative RAG.

Query-based RAG

Il query-based RAG, anche chiamato Prompt Augmentation, integra la query dell'utente con informazioni rilevanti recuperate dai documenti durante il processo di retrieval. Quindi, una volta recuperati i documenti rilevanti, il loro contenuto viene combinato con la query originale dell'utente per migliorare l'input inserito in un modello linguistico pre-addestrato per generare risposte.

Latent Representation-based RAG

Il Latent Representation-based RAG (Retrieval-Augmented Generation basato su rappresentazioni latenti) è un paradigma in cui i modelli generativi interagiscono con le rappresentazioni latenti degli oggetti recuperati. Il processo è il seguente: I paragrafi o le altre unità informative vengono recuperate dal retriever, ogni paragrafo recuperato viene trasformato in una rappresentazione latente (vettore numerico che cattura le caratteristiche semantiche del testo) dall'encoder del modello generativo. Il decoder del modello generativo prende le rappresentazioni latenti prodotte e le utilizza per generare l'output finale.

Logit-based RAG

Nel Logit-based Retrieval-Augmented Generation (RAG), i logit generati dal LLM vengono combinati con i logit delle informazioni recuperate (per esempio documenti). Questo processo aiuta a integrare conoscenze esterne nel processo di generazione, migliorando così la qualità delle risposte generate dal modello. Nell'ambito dei modelli di linguaggio, i logit sono fondamentali per il processo di generazione del testo poiché essi sono utilizzati nella scelta del token successivo durante la generazione. Essi forniscono una misura di quanto il modello di generazione considera plausibile ogni possibile token. Il Generatore calcola i logit per i possibili token di risposta utilizzando i documenti recuperati. Successivamente i logit dei vari documenti vengono aggregati, così, utilizzando i logit aggregati, il modello di generazione può determinare la sequenza di token con le probabilità più alte per formare la risposta finale.

Speculative RAG

La Speculative RAG mira a risparmiare risorse e accelerare la velocità di risposta cercando opportunità per usare il recupero piuttosto che una generazione pura. Per esempio, GPTChache (libreria open-source) affronta il problema della latenza costruendo una cache semantica per memorizzare le risposte del LLM, rendendo più veloci le risposte future.

2.2.7 Metodi per migliorare le performance della RAG

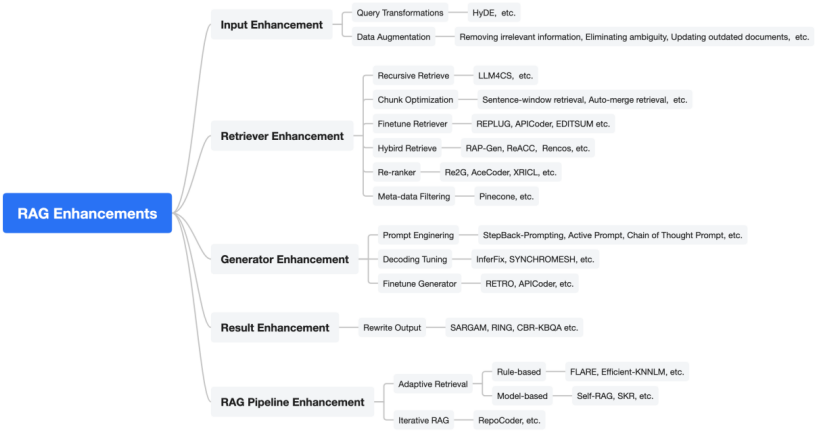


FIGURA 2.5: Metodi per migliorare le performance della RAG

Come definito nell'articolo di Wentao Zhang [10], esistono diverse tecniche per migliorare le prestazioni di un sistema RAG.

Miglioramento dell'input:

Con input si fa riferimento alla query dell'utente, la quale viene fornita al retriever. La qualità dell'input impatta significativamente il risultato finale.

- **Trasformazione della query:** Questa strategia migliora il risultato del recupero modificando la query inserita dall'utente. Query2doc e HyDE sono esempi di tecniche che usano LLM per generare pseudo-documenti espandendo la query originale con questi documenti generati.
- **Data Augmentation:** Questa strategia migliora le prestazioni del sistema RAG migliorando i dati prima del recupero attraverso operazioni come la rimozione di informazioni irrilevanti, eliminazione dell'ambiguità, oppure aggiornamento di documenti obsoleti.

Miglioramento del Retriever:

In un sistema RAG, il processo di recupero è cruciale, per questo è fondamentale migliorare l'efficacia del processo di recupero. Migliore è la qualità del retriever, migliori saranno le informazioni fornite al Generatore.

- **Recupero ricorsivo:** Il recupero ricorsivo è il processo di dividere una query in più parti prima del recupero dei documenti. Grazie a questa divisione della query vengono effettuate più ricerche per recuperare contenuti di qualità migliori.
- **Ottimizzazione dei chunk:** È una tecnica che ha lo scopo di raggiungere migliori risultati di recupero attraverso l'ottimizzazione della grandezza dei chunk.
- **Fine-Tuning del Retriever:** La qualità del modello embedding utilizzato è fondamentale per l'efficacia complessiva del sistema RAG. È possibile effettuare il fine-tuning del modello embedding usando dati di dominio di alta qualità migliorando così le capacità del retriever stesso.
- **Recupero ibrido:** Per recupero ibrido si intende l'utilizzo di più tipi di retriever. Per esempio, si può pensare di usare sia lo Sparse Retriever che il Dense Retriever per migliorare la qualità del recupero.
- **Re-ranking:** La tecnica del Re-Ranking si basa sul riordinare gli elementi recuperati in modo da raggiungere risultati migliori e diversificati.

- **Filtraggio dei metadati:** Si tratta di un metodo che filtra i documenti recuperati usando i metadati, nonché informazioni contestuali di un documento come autore, categoria e data di pubblicazione.

Miglioramento del Generatore:

In un sistema RAG la qualità del Generatore determina la qualità dell'output finale.

- **Prompt Engineering:** l'arte di ottimizzare i prompt forniti ai LLM.
- **Decoding Tuning:** Si riferisce a tecniche utilizzate durante la fase di decodifica dei modelli di generazione per migliorare la qualità delle risposte. Queste tecniche consistono nell'aggiustare iperparametri come la temperatura o nel limitare il vocabolario di output del decodificatore. (La fase di decodifica di un modello di linguaggio riguarda il processo attraverso il quale il modello genera il testo finale a partire dalle rappresentazioni interne apprese durante l'addestramento).
- **Fine-Tuning del Generatore:** Il Fine-Tuning è essenziale per migliorare ulteriormente i LLM per prestazioni ottimali su nuovi dataset e task.

Miglioramento del risultato:

Si tratta di tecniche che si assicurano che il risultato generato sia di qualità.

Miglioramento della pipeline RAG:

Si focalizza sull'ottimizzazione dell'interazione tra il processo di recupero e di generazione.

2.3 Fine-tuning dei LLM

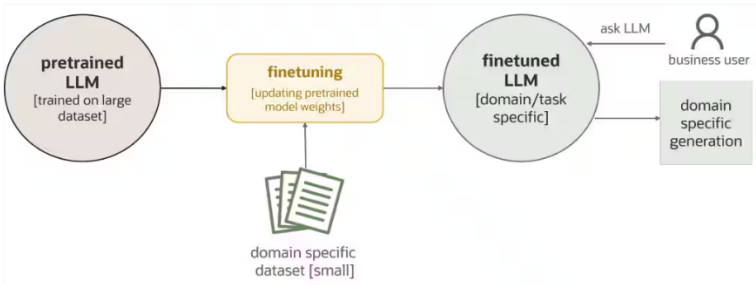


FIGURA 2.6: Fine-Tuning di un LLM

Un'altra tecnica utilizzata per superare i limiti dei LLM è il Fine-tuning. Questa tecnica permette di migliorare le prestazioni di un LLM per specifici domini o compiti apportando piccole modifiche ai suoi parametri. Questa tecnica consente di sfruttare la conoscenza già acquisita durante il pre-addestramento su un ampio e generale dataset e di affinarla ulteriormente utilizzando un dataset più piccolo e specifico per il compito. Come riportato nell'articolo [9] il Fine-Tuning ha diversi vantaggi e svantaggi. Inoltre, come definito nell'articolo [8] esistono diversi tipi di Fine-Tuning.

2.3.1 Fasi del Fine-Tuning:

Pre-addestramento Iniziale:

Un grande modello linguistico (LLM) viene addestrato su un ampio dataset, spesso comprendente enormi quantità di dati testuali. Questa fase comporta l'apprendimento di schemi linguistici generali, grammatica, fatti e alcune capacità di ragionamento.

Fine-Tuning Specifico per il Compito:

Il modello pre-addestrato viene poi addestrato su un dataset più piccolo e specifico per il compito. Questo comporta una leggera regolazione dei parametri del modello per migliorare le prestazioni del LLM su uno specifico dominio di conoscenza senza perdere la comprensione linguistica generale acquisita durante il pre-addestramento.

2.3.2 Tipi di Fine-Tuning:

Full Fine-Tuning:

Questo comporta l'aggiornamento di tutti i parametri del modello pre-addestrato. Sebbene possa portare a elevate prestazioni sul compito, è computazionalmente costoso e soggetto a overfitting, soprattutto con dataset piccoli.

Parameter-Efficient Fine-Tuning (PEFT):

Solo un sottoinsieme dei parametri del modello viene regolato. Questo approccio riduce i costi computazionali e i requisiti di memoria.

2.3.3 Vantaggi del Fine-Tuning

Migliore performance su compiti specifici:

Il fine-tuning consente al modello di diventare più preciso su un determinato compito. Per esempio il modello grazie al fine-tuning può migliorare su compiti specifici come analisi del sentiment, traduzione automatica e generazione di risposte.

Riduzione dei Bias:

Il modello può essere adattato per minimizzare bias che possono emergere nei modelli generici.

2.3.4 Svantaggi del Fine-Tuning

Catastrophic Forgetting:

Il fine-tuning può portare al fenomeno del catastrophic forgetting, dove il modello dimentica alcune delle conoscenze pre-addestrate mentre si adatta al nuovo compito.

Overfitting:

Il fine-tuning può portare a una specializzazione eccessiva del modello per il compito specifico, riducendo la sua capacità di generalizzare a nuovi dati o contesti diversi.

Costi computazionali e di memoria:

Anche con tecniche di fine-tuning efficienti come PEFT, il costo computazionale e la memoria richiesta possono essere notevoli.

2.4 Prompt Engineering

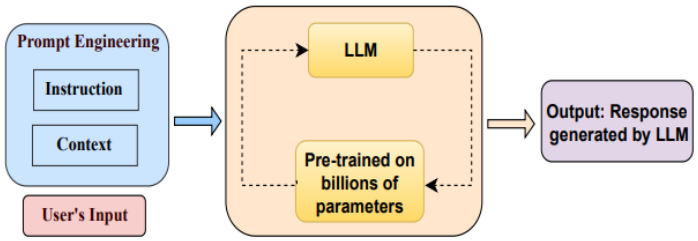


FIGURA 2.7: Prompt Engineering

Il Prompt Engineering è il processo di progettazione e raffinamento di input, o "prompt", per ottenere le risposte desiderate dagli LLM (Prompt Engineering in Large Language Models). I prompt sono cruciali per guidare i LLM nella generazione di output utili e rilevanti. La qualità dei prompt influisce significativamente sull'efficacia degli LLM, rendendo questa tecnica essenziale per migliorare le prestazioni e risparmiare tempo e risorse. Due delle tecniche più famose di Prompt Engineering citate nell'articolo [4] sono:

- **Zero-Shot Prompting**: Questa tecnica si affida a prompt attentamente formulati che guidano il modello verso compiti nuovi, eliminando la necessità dei dati di addestramento. Nello specifico, il modello riceve una descrizione del compito nel prompt e sfrutta la sua conoscenza preesistente per generare risposte al prompt fornito per il nuovo compito. Con questa tecnica non vengono forniti al modello esempi.

- **Few-Shot Prompting**: Con questa tecnica, invece, si forniscono ai modelli alcuni esempi con coppie input-output per indurre la comprensione di un nuovo compito. Anche fornire solo pochi esempi di qualità permette di migliorare le prestazioni del modello.

Quindi, il Prompt Engineering permette ai LLM di generare risposte più accurate e contestuali migliorando la qualità complessiva dell'output. Tuttavia, anche con prompt ottimizzati i LLM possono avere limitazioni intrinseche come la mancanza di contesto aggiornato o la capacità di comprendere sfumature complesse del linguaggio.

2.5 Un confronto delle diverse tecniche

Tabella 2.1: Confronto tra RAG, Fine-Tuning e Prompt Engineering

Caratteristica	**RAG (Retrieval-Augmented Generation)**	**Fine-Tuning**	**Prompt Engineering**
Approccio	Utilizza un motore di ricerca interno per recuperare informazioni rilevanti e un modello generativo per sintetizzare risposte.	Modifica i parametri del modello attraverso un processo di apprendimento supervisionato.	Manipolazione del testo di input senza modificare i pesi del modello.
Scalabilità	Elevata, poiché il corpus di conoscenza può essere esterno e ampliato senza limiti significativi.	Limitata dalla capacità del modello e dalla disponibilità di dati pertinenti.	Molto alta, poiché non richiede risorse computazionali aggiuntive per il training.
Risorse Computazionali	Richiede risorse significative per il recupero e la generazione delle risposte.	Richiede risorse computazionali elevate durante il processo di fine-tuning.	Richiede risorse minime, poiché non implica il riaddestramento del modello.

Continua nella pagina successiva

Caratteristica	**RAG (Retrieval-Augmented Generation)**	**Fine-Tuning**	**Prompt Engineering**
Tempo di Implementazione	Moderato, dipende dalla complessità del sistema di recupero e dall'integrazione con il modello generativo.	Lungo, a causa del tempo necessario per l'addestramento del modello su nuovi dati.	Breve, poiché consiste principalmente nella sperimentazione e ottimizzazione dei prompt.
Flessibilità	Alta, può essere adattato facilmente a nuovi domini aggiungendo o aggiornando il corpus di conoscenza.	Limitata alla disponibilità di dati annotati per il compito specifico.	Alta, può essere rapidamente adattato per diversi tipi di query e obiettivi.
Costo	Potenzialmente elevato, specialmente se si utilizza un ampio corpus di conoscenza esterno.	Elevato, dovuto ai costi di calcolo e alla necessità di set di dati di alta qualità.	Basso, poiché non richiede nuovi dati o risorse computazionali aggiuntive per il training.
Manutenzione	Richiede aggiornamenti regolari del corpus di conoscenza e del motore di ricerca.	Richiede riaddestramenti periodici per mantenere le prestazioni aggiornate.	Richiede monitoraggio e aggiornamento dei prompt per rispondere ai cambiamenti nei dati.

Continua nella pagina successiva

Caratteristica	**RAG (Retrieval-Augmented Generation)**	**Fine-Tuning**	**Prompt Engineering**
Qualità delle Risposte	Alta, con risposte accurate e aggiornate basate sul corpus di conoscenza.	Molto alta, specialmente se il modello è fine-tuned su dati di alta qualità specifici.	Variabile, dipende dalla qualità e precisione dei prompt formulati.
Applicazioni Tipiche	Chatbot avanzati, sistemi di domande e risposte, assistenti virtuali.	Modelli specializzati per domini specifici, assistenti virtuali dedicati, analisi dei sentimenti.	Risoluzione di problemi specifici, generazione di contenuti creativi, assistenza nel coding.
Dipendenza dal Modello di Base	Dipende meno dal modello di base grazie al recupero esterno.	Fortemente dipendente dal modello di base e dalla sua capacità di apprendere dal nuovo set di dati.	Fortemente dipendente dal modello di base e dalla sua comprensione dei prompt.

2.6 Problematiche aperte

Nonostante l'adozione delle tecniche menzionate, permangono diverse problematiche aperte riguardanti i LLM. Tra le problematiche citate nell'articolo [3] troviamo:

2.6.1 Knowledge shortcut

Il "knowledge shortcut" si riferisce al fenomeno per cui i modelli di linguaggio di grandi dimensioni (LLM) come GPT tendono a fare affidamento su correlazioni spurie nei dati di training invece di comprendere realmente la conoscenza fattuale. Questi modelli spesso dipendono eccessivamente da statistiche di co-occorrenza, vicinanza posizionale e conteggio dei documenti rilevanti presenti nei dati di training. Questo può introdurre un bias verso correlazioni spurie, che può portare a "allucinazioni" se il bias riflette informazioni fattualmente errate. Per esempio un modello alla domanda "qual è la capitale del Canada?" potrebbe rispondere erroneamente "Toronto" a cause delle frequenti co-occorrenze dei termini "Canada" e "Toronto" nei documenti di addestramento, senza cogliere la conoscenza fattuale sulla capitale del Canada.

2.6.2 Riconoscimento dei Limiti della Conoscenza

I LLM faticano a riconoscere i propri limiti di conoscenza, il che porta alla produzione di informazioni false con sicurezza. Molti studi stanno esplorando i confini della conoscenza dei LLM e la loro capacità di identificare domande a cui non possono rispondere in modo affidabile.

2.6.3 Bilanciamento tra Creatività e Fattualità

Un'altra sfida è bilanciare la creatività dei LLM con la loro accuratezza fattuale. Mentre la riduzione delle allucinazioni è una priorità, è importante non compromettere la capacità dei modelli di generare contenuti creativi utili in contesti come la narrazione e il brainstorming.

2.6.4 Conoscenze Obsolete

I LLM non possono aggiornare automaticamente le loro conoscenze dopo l'addestramento, il che porta a risposte basate su informazioni obsolete. Questa limitazione è particolarmente problematica in un mondo in rapida evoluzione, dove le informazioni cambiano costantemente.

2.6.5 Gestione di scenari di ragionamento complessi

Anche quando i LLM possiedono le conoscenze necessarie, possono avere difficoltà nel produrre risultati accurati in scenari che richiedono al modello deduzioni

logiche o ragionamenti complessi. Anche con l'utilizzo della RAG (Retrieval-Augmented-Generation) un modello può avere difficoltà nel generare risposte accurate anche se ha accesso ai documento contenenti le risposte corrette. Questo avviene a causa dell'inadeguatezza nell'utilizzare efficacemente quella conoscenza, e quindi, a causa di una scarsa capacità di ragionamento.

Queste problematiche evidenziano le attuali limitazioni dei LLM e delineano le aree chiave per la ricerca futura al fine di migliorare l'affidabilità e l'accuratezza di questi modelli.

2.7 Sistemi basati su LLM nel contesto della Pubblica Amministrazione e dei governi

Prima di analizzare gli attuali sistemi basati su LLM nel contesto degli enti pubblici è necessario capire cosa sono gli Open Government Data e perchè sono così importanti. Come citato nell'articolo [5] gli Open Government Data sono dati e informazioni prodotte da enti pubblici che possono essere liberamente utilizzate e riutilizzate da chiunque. I principi fondamentali degli Open Government Data (OGD) includono:

- **Completezza**: Tutti i dati pubblici devono essere resi disponibili.
- **Primarietà**: I dati devono essere raccolti alla fonte con il massimo livello di granularità possibile.
- **Tempestività**: I dati devono essere resi disponibili rapidamente per preservarne il valore.
- **Accessibilità**: I dati devono essere accessibili al maggior numero di utenti per il maggior numero di scopi.
- **Processabilità da macchine**: I dati devono essere strutturati in modo da permettere l'elaborazione automatizzata.
- **Non discriminazione**: I dati devono essere disponibili a chiunque.
- **Non proprietarietà**: I dati devono essere disponibili in un formato su cui nessuna entità ha il controllo esclusivo.
- **Libertà di licenza**: I dati non devono essere soggetti a regolamentazioni di copyright, brevetti, marchi o segreti commerciali.

I sistemi basati su LLM che utilizzano gli OGD (Open Government Data) possono essere usati per analizzare i bilanci pubblici e monitorare la spesa governativa promuovendo la trasparenza e la responsabilità dei governi. Inoltre, gli Open Government Data possono essere utilizzati dai LLM per supportare le decisioni in settori come la sanità, l'istruzione e la gestione urbana. I modelli di linguaggio di grandi dimensioni (LLM) a scopo generale possono fornire risposte a diverse domande, spesso questi modelli devono essere adattati per completare compiti specifici o per comprendere domini differenti che potrebbero non essere stati inclusi nei dati di addestramento. I modelli di intelligenza artificiale generativa possono essere usati anche in specifici domini utilizzando il fine-tuning o le architetture RAG. Innanzitutto, il fine-tuning comporta l'addestramento di uno o più aspetti di un modello preaddestrato su specifici set di dati aperti. Le architetture RAG implicano l'impostazione di un sistema di recupero per grandi set di dati e la loro integrazione nel modello di intelligenza artificiale generativa. L'idea è di indirizzare il LLM a dare priorità ai dati di dominio del sistema RAG rispetto al dataset generico su cui è stato addestrato. Le architetture RAG richiedono un volume maggiore di dati e capacità computazionale rispetto al fine-tuning e sono più adatte per compiti in cui l'informazione contestuale o fattuale è cruciale. I dati devono essere altamente accurati, aggiornati e chiaramente etichettati. Le architetture RAG sono più lente a produrre output, ma possono fornire preziose informazioni contestuali. Per entrambi gli approcci, i dati utilizzati devono essere pertinenti al compito mirato e rappresentativi dei problemi che si stanno cercando di risolvere. I dati possono essere tabulari o non strutturati. Come citato nella ricerca di *Hanna Chafetz, Sampriti Saxena e Stefaan Verhulst* [1] tra i sistemi esistenti basati su LLM nel contesto della pubblica amministrazione, dei governi e in generale delle entità pubbliche abbiamo:

2.7.1 Sistemi basati su LLaMandement

LLaMandement è un modello di linguaggio di grandi dimensioni (LLM) sviluppato e ottimizzato su task specifici con fine-tuning dal governo francese, con l'obiettivo di supportare gli agenti amministrativi nell'analisi e nella redazione di sintesi delle proposte di legge elaborate in Parlamento per altri ministeri e dipartimenti. Questo progetto è stato realizzato con la collaborazione della Direzione Generale delle Finanze Pubbliche, della Delegazione per la Trasformazione Digitale, della Direzione della Legislazione Fiscale e di altri enti, sotto l'egida del French Digital Republic Act, il quale promuove progetti pubblici di intelligenza artificiale aperti per impostazione predefinita. Il team ha scelto

di ottimizzare il modello preaddestrato LLaMA 70B, riconosciuto per la sua capacità di elaborare e analizzare documenti legali complessi in francese e di adattarsi rapidamente a nuove informazioni. Il processo di ottimizzazione ha incluso l'aggiunta di parametri al modello tramite tecniche di adattamento a basso rango (note anche come LORA). Sono stati utilizzati dati provenienti da SIGNALE, una piattaforma impiegata nel processo legislativo del governo francese, per ottimizzare il modello, attingendo a informazioni di vari ministeri, tra cui il Ministero della Transizione Ecologica e della Coesione Territoriale, il Ministero della Cultura e altri. Il team ha effettuato una revisione alla cieca, durante la quale dieci esperti hanno valutato la qualità dei memorandum sviluppati dal modello e confrontato queste valutazioni con i benchmark di modelli precedenti e con quelli redatti manualmente. Dai risultati è emerso che i memorandum generati da LLaMandement si sono classificati quasi al livello di quelli scritti da persone. Inoltre, LLaMandement è stato testato per verificare eventuali bias relativi a "genere, etnia e ideologia politica". I risultati hanno dimostrato che l'output del modello era "neutrale su varie dimensioni demografiche e ideologiche".

2.7.2 CensusGPT

CensusGPT è uno sistema avanzato di intelligenza artificiale generativa progettato per rendere più accessibili e comprensibili i dati demografici degli Stati Uniti. Sfruttando la potenza dei modelli di linguaggio di grandi dimensioni (LLM), come i Generative Pretrained Transformers (GPT), CensusGPT permette agli utenti di interrogare i dati censuari in modo semplice e intuitivo. CensusGPT è stato sviluppato per facilitare l'accesso e l'analisi dei dati dell'American Community Survey (ACS) del 2021, una delle principali fonti di dati demografici, sociali ed economici negli Stati Uniti. Questo strumento serve una vasta gamma di utenti, dai ricercatori ai decisori politici, dalle organizzazioni no-profit ai cittadini interessati, permettendo loro di ottenere risposte dettagliate e specifiche su vari aspetti della popolazione statunitense. Esso si contraddistingue per:

- **Accessibilità**: Una delle caratteristiche principali di CensusGPT è la sua interfaccia user-friendly, che consente agli utenti di porre domande in linguaggio naturale e ricevere risposte strutturate in modo chiaro e comprensibile.
- **Precisione e Aggiornamento**: CensusGPT utilizza dati aggiornati e accurati dell'American Community Survey, garantendo che le informazioni

fornite siano sempre rilevanti e precise.

- **Versatilità**: Oltre a rispondere a domande generali sui dati demografici, CensusGPT può elaborare richieste specifiche e complesse, fornendo analisi dettagliate e visualizzazioni come tabelle e grafici.
- **Integrazione dei Dati**: Il modello è in grado di combinare dati provenienti da diverse fonti, migliorando la profondità e la rilevanza delle risposte fornite.

Inoltre, CensusGPT è utilizzato in vari contesti per una serie di applicazioni pratiche:

- **Analisi Demografica**: I ricercatori possono utilizzare CensusGPT per ottenere informazioni dettagliate sulla composizione della popolazione in diverse aree geografiche, analizzando parametri come età, reddito, istruzione e occupazione.
- **Pianificazione Urbana**: I pianificatori urbani possono interrogare CensusGPT per comprendere meglio le caratteristiche demografiche dei quartieri, aiutandoli a prendere decisioni informate su sviluppo e servizi pubblici.
- **Politiche Pubbliche**: I decisori politici possono utilizzare lo strumento per analizzare l'impatto delle politiche pubbliche su diverse comunità, basandosi su dati precisi e aggiornati.
- **Servizi Sociali**: Le organizzazioni no-profit e i fornitori di servizi sociali possono utilizzare CensusGPT per identificare le aree con maggiore necessità di intervento, basandosi su dati economici e sociali.
- **Educazione e Sensibilizzazione**: Le scuole e le organizzazioni educative possono utilizzare CensusGPT per creare materiali didattici e campagne di sensibilizzazione basate su dati demografici accurati.

2.7.3 Wobby

Wobby è una sistema avanzato di intelligenza artificiale generativa progettato per facilitare l'accesso e l'analisi dei dati governativi. La piattaforma è stata creata per democratizzare l'accesso ai dati, ospitando dataset di organizzazioni come Statbel (l'ufficio statistico nazionale del Belgio), Statistics Netherlands ed Eurostat, oltre a dati di organizzazioni intergovernative come la Banca Mondiale. Wobby consente agli utenti di interagire con dati complessi attraverso

un'interfaccia intuitiva che accetta query in linguaggio naturale e genera risposte strutturate accompagnate da visualizzazioni grafiche. Questo approccio rende i dati accessibili non solo a ricercatori e decisori politici, ma anche a cittadini senza competenze tecniche avanzate. Una delle caratteristiche distintive di Wobby è la sua capacità di rendere i dati governativi aperti facilmente accessibili e interpretabili. La piattaforma si concentra sull'affidabilità dei dati, utilizzando dataset provenienti da fonti ufficiali e con API ben strutturate, garantendo così la qualità e l'affidabilità delle informazioni fornite. Wobby viene utilizzato in vari contesti. I ricercatori lo impiegano per accedere a dati statistici dettagliati da diverse nazioni e organizzazioni intergovernative, facilitando studi comparativi e analisi approfondite. I decisori politici sfruttano la piattaforma per ottenere rapidamente dati rilevanti per la formulazione di politiche basate su evidenze. Inoltre, le organizzazioni utilizzano Wobby per generare visualizzazioni grafiche di dati complessi, migliorando la comunicazione dei risultati a un pubblico più ampio. Anche le istituzioni educative e le organizzazioni no-profit trovano in Wobby uno strumento prezioso per creare materiali didattici e campagne di sensibilizzazione basate su dati concreti e accurati. Con il suo approccio innovativo e le sue capacità avanzate, Wobby contribuisce a una maggiore trasparenza e all'adozione dei dati aperti nei processi decisionali e di ricerca.

2.7.4 NEPAccess

NEPAccess è un progetto sviluppato da un team di ricercatori dell'Università dell'Arizona, con l'obiettivo di rendere operativa la National Environmental Policy Act (NEPA) del 1969. Questa legge promuove il coinvolgimento dei cittadini nelle decisioni governative e utilizza la ricerca per migliorare i risultati delle politiche relative al benessere sociale e ambientale. NEPA raccoglie e pubblica dati sulle condizioni ambientali in tutto il paese e sui progetti finanziati dal governo federale. NEPAccess mira a sbloccare l'accesso a questi dati utilizzando un modello di intelligenza artificiale generativa. Questo modello supporta gli utenti nella ricerca dei dati e delle recensioni NEPA, che altrimenti sarebbero difficili da trovare. Attualmente, il modello si concentra principalmente sul recupero dei dati NEPA non strutturati tramite prompt di ricerca testuale, senza ancora avere capacità analitiche avanzate. La piattaforma NEPAccess è stata progettata per migliorare l'accessibilità dei dati ambientali e facilitarne l'utilizzo da parte di cittadini, ricercatori e responsabili politici. La prossima versione di NEPAccess prevede l'inclusione di funzionalità per aiutare gli utenti a scrivere valutazioni di impatto ambientale, offrire nuove possibilità di analisi dei dati e

altre caratteristiche utili. Grazie all'uso di modelli AI generativi, NEPAccess consente un'interazione più semplice e intuitiva con i dati ambientali, contribuendo così a una maggiore partecipazione pubblica e a decisioni politiche più informate basate su dati accurati e aggiornati.

2.7.5 Parla

PARLA è un sistema di intelligenza artificiale sviluppato da CityLab Berlin per migliorare l'accesso ai dati dell'amministrazione pubblica della città. Questo strumento è progettato per facilitare l'interazione tra i cittadini e l'amministrazione comunale, permettendo agli utenti di porre domande in linguaggio naturale e ottenere risposte dettagliate e utili. L'obiettivo di PARLA è rendere più accessibili i dati pubblici amministrativi, rispondendo a richieste di informazioni attraverso un sistema di recupero e generazione automatizzata. In questo modo, gli utenti possono ottenere risposte rapide e pertinenti, migliorando la trasparenza e la partecipazione civica. Una delle caratteristiche distintive di PARLA è l'accessibilità dei documenti: esso accede a oltre 10.000 documenti di dominio pubblico. Questo vasto database garantisce che gli utenti possano trovare informazioni rilevanti su una vasta gamma di argomenti. Inoltre, l'interfaccia di PARLA è progettata per rispondere a semplici prompt in linguaggio naturale, rendendo l'interazione con il sistema intuitiva e facile da usare per chiunque. Per mitigare i rischi di errori e garantire la qualità delle risposte, PARLA fornisce riferimenti alle fonti dei dati utilizzati. Questo approccio aiuta a migliorare la trasparenza e la fiducia nelle informazioni fornite dal sistema. PARLA viene utilizzato in diversi contesti per una serie di applicazioni pratiche. I cittadini possono utilizzarlo per ottenere informazioni dettagliate sui servizi pubblici, sulle politiche locali e sui piani di sviluppo urbano, facilitando una maggiore partecipazione alle decisioni della comunità e una migliore comprensione delle operazioni amministrative. Gli amministratori pubblici possono sfruttare PARLA per recuperare rapidamente documenti e dati necessari per il loro lavoro quotidiano, migliorando l'efficienza e riducendo i tempi di ricerca manuale. Anche le istituzioni educative e le organizzazioni no-profit possono utilizzare PARLA per creare materiali didattici e campagne di sensibilizzazione basate su dati concreti e aggiornati, supportando l'educazione civica e l'informazione pubblica. PARLA rappresenta un passo significativo verso la modernizzazione dell'accesso ai dati pubblici e l'aumento della trasparenza amministrativa. Con il suo approccio innovativo e le sue capacità avanzate, contribuisce a una maggiore partecipazione pubblica e a decisioni politiche più informate basate su dati accurati e aggiornati.

Capitolo 3

Metodologia

Per superare i limiti dei LLM e supportare un recupero efficiente di informazioni contestualmente rilevanti e aggiornate sui bandi regionali ho sviluppato un sistema basato su RAG, precedentemente menzionato nell'introduzione. Di seguito, sono illustrate le scelte chiave prese per l'implementazione del sistema RAG evidenziando le sue caratteristiche principali, e gli aspetti che contraddistinguono il sistema dalle soluzioni già esistenti.

3.1 Framework

LLamaIndex è stato scelto come framework per potenziare la capacità del LLM LLaMAntino, grazie alla sua straordinaria efficacia in applicazioni che richiedono ricerca testuale e risposte di alta qualità. È particolarmente adatto per sistemi di domande e risposte (QA), chatbot e assistenti virtuali. LLamaIndex ottimizza il recupero delle informazioni convertendo i documenti in vettori embeddings, migliorando velocità e precisione delle ricerche. Integra dati esterni aggiornati ampliando la conoscenza del modello e permettendo risposte basate su informazioni recenti e specifiche. La sua architettura efficiente include componenti per acquisire, indicizzare e recuperare dati rapidamente.

3.2 Componenti principali di LLamaIndex

- **Data Connectors**: Permettono di acquisire dati di diversi formati da diverse sorgenti e inserirli in oggetti Document (testo e metadati).
- **Indice**: L'indice è una struttura dati che recupera velocemente informazioni rilevanti dal database vettoriale in base alla query dell'utente. Esso lavora dividendo i documenti in porzioni di testo chiamate "Nodi", per costruire un indice su questi nodi (anche chiamati "chunks"). Una volta

costruito, l'indice può essere usato per effettuare delle query alla collezione di documenti. LlamaIndex offre una grande varietà di indici differenti, in seguito vedremo nel dettaglio l'indice scelto.

- **Retriever**: È responsabile del recupero dei nodi rilevanti data la query dell'utente. LlamaIndex permette la creazione di diversi tipi di Retriever in base alle necessità dell'applicazione che si sta sviluppando.

- **Response Synthesizer**: Si occupa della generazione della risposta passando al LLM la query dell'utente, il contesto recuperato e il Prompt Template.

- **Query Engine**: È una generica interfaccia di alto livello che prende in input una query dell'utente in linguaggio naturale e restituisce una risposta. Si interfaccia con il Retriever e il Response Synthesizer per recuperare il contesto relativo alla query e generare la risposta.

- **Chat Engine**: È un interfaccia di alto livello utilizzata per ottenere una risposta ad una determinata query. A differenza del Query Engine, però, quest'ultimo è usato per avere delle conversazioni vere e proprie poichè tiene traccia della cronologia delle conversazioni. Quindi, grazie al Chat Engine, un'applicazione basata su RAG può rispondere alle domande dell'utente tenendo in mente il contesto delle domande e delle risposte precedenti.

- **Vector Store**: Il database vettoriale in cui vengono memorizzati i nodi pronti ad essere recuperati e forniti al LLM come contesto per generare la risposta contestuale.

3.3 Architettura e funzionamento applicazione

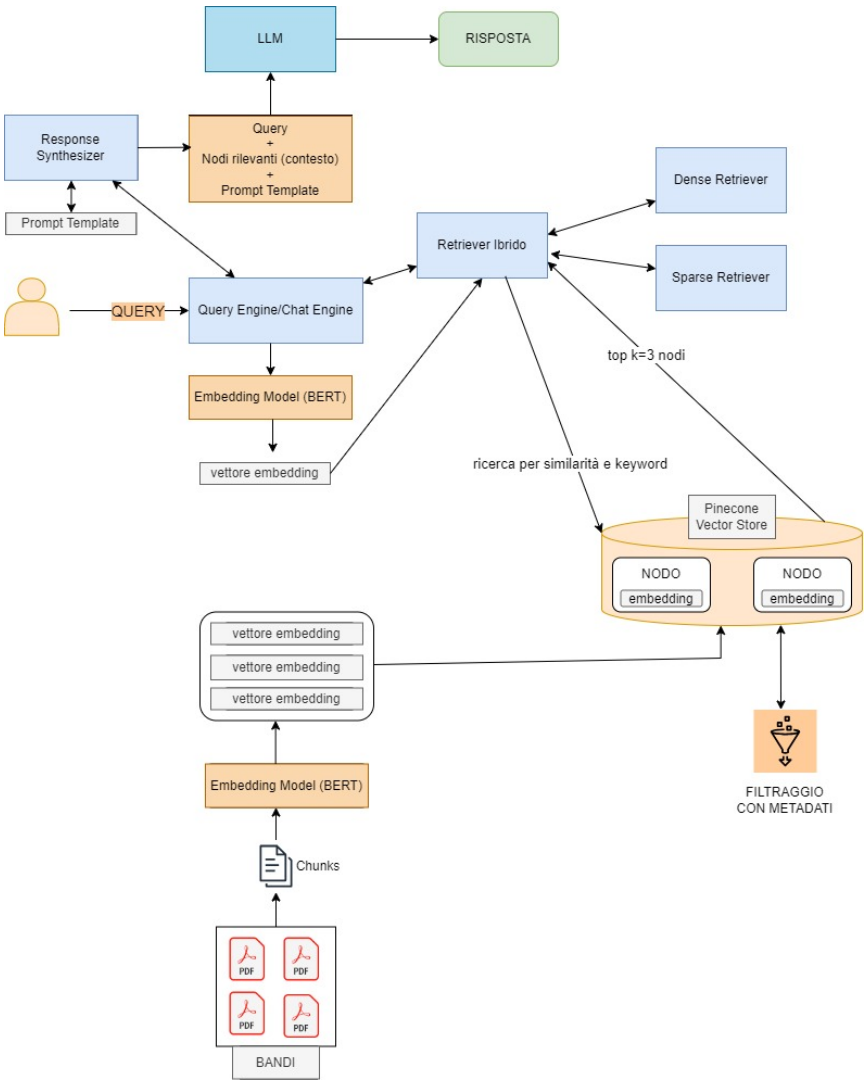


FIGURA 3.1: Architettura RAG.

Il diagramma presentato illustra l'architettura dell'applicazione sviluppata per fornire risposte accurate, aggiornate e contestuali sui bandi della regione Puglia. Innanzitutto, per lo sviluppo del sistema basato su RAG, è fondamentale memorizzare il contenuto dei bandi in un formato adeguato per la ricerca delle porzioni di testo rilevanti alla generazione della risposta finale. I documenti dei bandi vengono suddivisi in "chunks", o nodi (porzioni di testo), che sono successivamente convertiti in vettori di embedding tramite il modello BERT. Questo processo trasforma le porzioni di testo in rappresentazioni numeriche dense, capaci di catturare la semantica del testo. I nodi, con i rispettivi embedding, vengono indicizzati nel Vector Store, il database vettoriale utilizzato per il recupero del contesto necessario al LLM per generare risposte contestuali

alle query. L'applicazione offre all'utente la possibilità di scegliere il tipo di conversazione con il chatbot. Se viene selezionata la modalità "STANDARD", il sistema utilizzerà il Query Engine; se viene scelta la modalità "CHAT", il sistema utilizzerà il Chat Engine. Quando l'utente inserisce una query, questa viene passata al Query Engine o al Chat Engine, a seconda della modalità scelta. La query dell'utente è quindi convertita in un vettore di embedding tramite BERT e passata al Retriever. Per migliorare significativamente le performance del sistema RAG, è stato implementato un Retriever Ibrido, capace di combinare i vantaggi di uno Sparse Retriever e quelli di un Dense Retriever. Il Retriever Ibrido viene definito grazie all'utilizzo dell'indice ibrido di *Pinecone*, il Vector Store utilizzato per memorizzare i nodi e i relativi vettori. Attraverso la ricerca ibrida vengono recuperati i tre nodi più rilevanti. I nodi recuperati vengono successivamente forniti al Response Synthesizer, che passa al LLM la query iniziale dell'utente, i nodi contenenti il contesto rilevante e il Prompt Template per la generazione della risposta finale. Inoltre, il Response Synthesizer utilizza come parametri uno o più Prompt Template e il response mode. Infine, il LLM, utilizzando il contesto fornito, genera una risposta dettagliata che viene presentata all'utente. Questa sofisticata architettura garantisce agli utenti risposte precise e contestualmente appropriate, rendendo l'applicazione altamente efficace per compiti complessi di recupero delle informazioni. Ciò che distingue il sistema sviluppato rispetto alle soluzioni esistenti è, innanzitutto, la possibilità di visualizzare i nodi utilizzati per generare la risposta, offrendo trasparenza e comprensibilità nel processo di risposta. Inoltre, l'uso di Prompt Template personalizzati ad hoc per il recupero di informazioni sui bandi regionali ha significativamente migliorato l'accuratezza, la rilevanza e l'efficacia delle risposte fornite dall'applicazione. Questa personalizzazione consente di adattare le risposte alle specifiche esigenze della Regione Puglia, garantendo che le informazioni recuperate siano pertinenti e precise. Un'altra innovazione del sistema è la possibilità per l'utente di scegliere la modalità di risposta utilizzata dal Response Synthesizer. Questa opzione permette di influenzare direttamente il modo in cui il LLM genera le risposte, offrendo maggiore controllo e flessibilità agli utenti. Infine, l'applicazione permette agli utenti di selezionare la collezione di documenti da cui recuperare le informazioni desiderate. Questa funzionalità è particolarmente utile per la Regione Puglia, che può così focalizzarsi sui bandi più rilevanti per le proprie esigenze specifiche. In sintesi, queste caratteristiche innovative rendono il sistema non solo più efficace, ma anche più versatile e user-friendly rispetto alle soluzioni esistenti.

Capitolo 4

Implementazione

4.1 Configurazione del LLM per il sistema RAG

```python
def setLLM():
    # Define the quantization configuration
    quantization_config = BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_compute_dtype=torch.float16,
        bnb_4bit_quant_type="nf4",
        bnb_4bit_use_double_quant=True,
    )

    # Define the generate_kwargs
    generate_kwargs = {
        "do_sample": True,
        "min_length": 50,  # Lunghezza minima della risposta generata
        "no_repeat_ngram_size": 5,  # Evita la ripetizione di n-grammi
        "temperature": 0.3,
        "top_p": 0.95,
        "top_k": 10,
    }

    # Define the prompt template
    prompt_template = PromptTemplate("<s> [INST] {query_str} [/INST] ")

    # Load the HuggingFaceLLM with specified configurations
    llm = HuggingFaceLLM(
        model_name="swap-uniba/LLaMAntino-2-chat-7b-hf-UltraChat-ITA",
        tokenizer_name="swap-uniba/LLaMAntino-2-chat-7b-hf-UltraChat-ITA",
        query_wrapper_prompt=prompt_template,
        context_window=3900,
        max_new_tokens=512,
        generate_kwargs=generate_kwargs,
        model_kwargs={"quantization_config": quantization_config},
        # tokenizer_kwargs={"token": hf_token},
        device_map="auto", # Automatically allocate the model to GPU if available
    )

    return llm
```

FIGURA 4.1: Configurazione parametri LLM

Per la configurazione del LLM utilizzato nel sistema RAG sviluppato, sono stati scelti specifici parametri per ottimizzare le performance e l'efficienza della risposta. Il modello selezionato, "*LLaMAntino-2-chat-7b-hf-UltraChat-ITA*", è stato configurato con una quantizzazione a 4 bit mediante la libreria *BitsAndBytesConfig*. La quantizzazione a 4 bit, abilitata tramite il parametro "*loadIn4bit=True*", riduce significativamente l'uso di memoria, permettendo l'esecuzione del modello su hardware meno potente senza compromettere la qualità delle risposte. La scelta del tipo di quantizzazione 'nf4' (normalized floating-point 4) e l'utilizzo della doppia quantizzazione ("*bnb4BitUseDoubleQuant=True*") assicurano una precisione numerica migliorata durante le operazioni di calcolo,

pur mantenendo i benefici della quantizzazione. Per la generazione delle risposte, i parametri all'interno di "*generateKwargs*" sono stati selezionati con attenzione per garantire risposte pertinenti e fluide. Il parametro "*minLength=50*" stabilisce una lunghezza minima per le risposte generate, assicurando completezza e coerenza. L'opzione *"earlyStopping=True"* permette di fermare la generazione quando viene raggiunto un risultato soddisfacente, ottimizzando i tempi di risposta. I parametri *"topP=0.95"* e *"topK=10"* controllano la generazione del testo limitando la probabilità cumulativa e il numero di token considerati, rispettivamente, per mantenere un equilibrio tra diversità e plausibilità. La *"temperature=0.3"* regola la casualità del testo generato, producendo risposte meno variegate e più focalizzate. Ho deciso di utilizzare una temperatura molto bassa a seguito di molte sperimentazioni per ridurre il numero di parole inesistenti o non idonee al contesto generate dal LLM. Infine, il parametro "*noRepeatNgramSize=5*" previene la ripetizione di n-grammi di cinque token, evitando così ripetizioni all'interno del testo generato e migliorando la qualità generale della risposta. Questa configurazione avanzata assicura che il sistema RAG sia efficiente e capace di fornire risposte precise e contestuali sui bandi regionali, massimizzando l'utilizzo delle risorse disponibili e garantendo una qualità elevata delle risposte generate.

4.2 Gradio

Per la creazione dell'interfaccia si è utilizzato Gradio, una libreria open-source di Python che facilita la creazione di interfacce web per applicazioni di machine learning e data science. La sua capacità di integrare HTML, CSS e JavaScript consente una personalizzazione avanzata dell'interfaccia utente, permettendo di creare un'applicazione interattiva. Questa integrazione è particolarmente rilevante per il mio progetto, poiché consente di collegare in modo efficace la logica applicativa sviluppata in Python con un'interfaccia utente interattiva. Grazie a Gradio, ho potuto sviluppare un'interfaccia che rende l'applicazione accessibile e facile da usare, pur mantenendo la capacità di eseguire funzionalità complesse necessarie per il recupero e la presentazione delle informazioni sui bandi regionali.

4.3 Caricamento dei documenti

Per il caricamento dei documenti si è utilizzata la libreria *SimpleDirectoryReader*, la quale semplifica il processo di lettura e indicizzazione dei documenti

presenti in una directory. Consente di caricare documenti da una directory con poche righe di codice. La classe è in grado di gestire diversi formati di documenti come file di testo (.txt), file PDF, file Word (.docx), e altri formati comuni. Questo permette di indicizzare una vasta gamma di tipi di documenti senza necessità di conversioni manuali. Quindi, la libreria *SimpleDirectoryReader* estrae automaticamente il contenuto dei documenti, trasformandolo in un formato strutturato che può essere facilmente indicizzato.

Ottimizzazione dei chunks

Una volta caricati i documenti, questi ultimi vengono divisi in *chunks* attraverso la funzione *SentenceSplitter* con i parametri *chunkSize=975* e *chunkOverlap=200*. A seguito di numerose sperimentazioni e attente valutazioni si è scelto di dividere i documenti in *chunks* di massimo 975 caratteri poichè un chunk di questa dimensione è sufficientemente grande da contenere informazioni dettagliate, comprese intere frasi o paragrafi, mantenendo comunque una buona velocità di elaborazione. Questo è cruciale per assicurare che le risposte generate siano complete e pertinenti. Inoltre, chunks di questa dimensione ottimizzano l'efficienza del processo di recupero, riducendo il numero totale di chunks da gestire e indicizzare. Questo migliora le performance del sistema in termini di velocità e utilizzo delle risorse. Il parametro chunkOverlap, invece, specifica il numero di token di sovrapposizione tra chunks adiacenti. Una sovrapposizione di 200 caratteri garantisce che le informazioni rilevanti non siano divise tra due chunks, la sovrapposzione aiuta a catturare informazioni che potrebbero essere al confine tra due chunks, riducendo il rischio di perdere dettagli cruciali che potrebbero essere essenziali per una risposta accurata e contestualmente appropriata.

4.4 Pinecone

Come Vector Store si è utilizzato Pinecone. Pinecone è una piattaforma innovativa che fornisce un servizio di database vettoriale progettato per facilitare la gestione e la ricerca di dati ad alta dimensione. Sviluppato per rispondere alle esigenze di intelligenza artificiale e machine learning, Pinecone consente di gestire e interrogare dati vettoriali in modo efficiente e scalabile. In particolare, Pinecone permette di creare, aggiornare e interrogare spazi vettoriali in tempo reale. Questo è cruciale per applicazioni che richiedono la gestione di grandi volumi di dati non strutturati, come immagini, testi e altri tipi di dati

che possono essere rappresentati come vettori numerici. La piattaforma utilizza algoritmi avanzati per garantire che le operazioni di ricerca e recupero dei dati siano rapide e precise, anche quando si lavora con dataset di dimensioni significative. Un aspetto distintivo di Pinecone è la sua capacità di integrare facilmente con altre tecnologie e servizi di machine learning, rendendola una scelta ideale per aziende e sviluppatori che necessitano di una soluzione robusta e flessibile per la gestione dei dati vettoriali. Pinecone supporta l'analisi in tempo reale e offre funzionalità come la ricerca di similarità, che è particolarmente utile in applicazioni come il riconoscimento di immagini, la raccomandazione di prodotti, e la comprensione del linguaggio naturale. Inoltre, la piattaforma è progettata per essere altamente scalabile, garantendo prestazioni costanti anche quando il volume di dati cresce esponenzialmente. Questo la rende adatta sia per piccole imprese che per grandi organizzazioni che necessitano di elaborare enormi quantità di dati in modo rapido ed efficiente. In sintesi, Pinecone rappresenta una soluzione avanzata per la gestione di dati vettoriali, offrendo una combinazione di velocità, precisione e scalabilità che la rende un asset prezioso per qualsiasi progetto di intelligenza artificiale e machine learning. Si è fatto uso di due indici creati e salvati sul cloud di Pinecone, "*indexbandisistemapuglia*" e "*indexbandi*" in modo tale da avere una grande scalabilità e velocità nel recupero. Infatti, grazie alla memorizzazione degli indici su cloud possiamo scalare facilmente il sistema aumentando il numero di bandi da indicizzare senza preoccuparci di acquisire nuove risorse hardware o di avere un degrado delle prestazioni nel recupero. L'indice *indexbandisistemapuglia* è stato usato per memorizzare i nodi con i relativi vettori per quanto riguarda la collezione dei bandi della piattaforma "Sistema Puglia". L'indice *indexbandi*, invece, è stato usato per memorizzare i nodi con i relativi vettori per quanto riguarda l'altra collezione di bandi. La dimensionalità dei vettori memorizzati nei due indici è 768, ciò vuol dire che i vettori hanno 768 componenti per catturare la semantica di una porzione di testo. La ricerca ibrida in Pinecone combina la ricerca semantica e quella basata su parole chiave, utilizzando vettori densi e sparsi. Questa combinazione migliora la rilevanza dei risultati, specialmente per query che non rientrano nel dominio semantico. La ricerca vettoriale semantica supera significativamente le performance dei metodi tradizionali di ricerca quando i modelli embedding sono stati addestrati su dati del dominio di riferimento. Questo cambia quando si utilizzano questi modelli per compiti su altri domini. Se abbiamo molti dati su un dominio specifico, possiamo perfezionare un modello embedding per creare vettori densi e ottenere ottime prestazioni. Il problema sorge quando non abbiamo dati sufficienti. In questo scenario, un

modello embedding preaddestrato potrebbe avere prestazioni inferiori a BM25. In questa situazione o si utilizza un grande dataset per perfezionare il modello embedding oppure si utilizza la ricerca ibrida. In questo caso, si è fatto uso della ricerca ibrida.

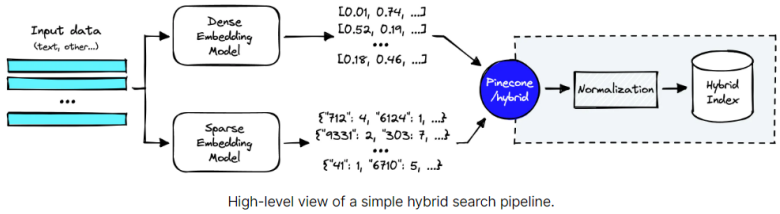


FIGURA 4.2: Costruzione indice ibrido

Pinecone permette la creazione di un indice ibrido, nonchè uno sparse-dense index. Senza l'utilizzo di un indice ibrido per effettuare una ricerca ibrida è necessario effettuare il recupero con due indici differenti, dove un indice viene usato per il recupero basato su similarità semantica, l'altro indice, invece, per il recupero basato su keyword. Dopodichè si combinano i risultati dei due retriever con un processo di re-ranking in cui si ordinano i nodi per rilevanza. Quindi, la ricerca ibrida di Pinecone ci permette di evitare di combinare i risultati di due differenti ricerche. Per quanto riguarda l'indicizzazione dei dati, nel nostro caso dei documenti, Pinecone usa un embedding model per creare vettori densi e un embedding model per creare vettori sparsi, successivamente, questi vettori vengono combinati in unica rappresentazione vettoriale sia densa che sparsa, in modo da permettero un recupero ibrido. Infine, questi vettori densi-sparsi sono memorizzati nell'indice ibrido pronti per essere recuperati.

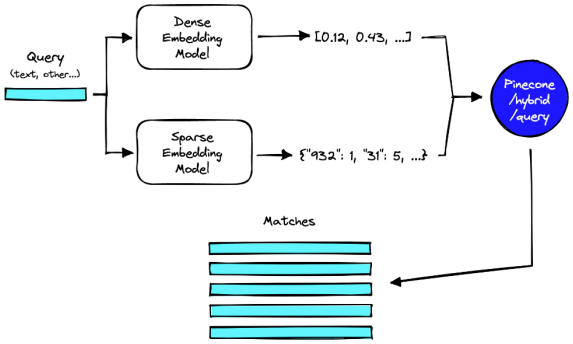


FIGURA 4.3: Query su indice ibrido

Per quanto riguarda il processo di retrieval, le query, nel momento in cui l'utente effettua una domanda, la query viene trasformata in un dense-sparse vector, e utilizzato per la ricerca dei vettori simili nel database vettoriale, i k

vettori più simili vengono recuperati. La ricerca ibrida nel Vector Store combina i vantaggi della ricerca semantica e della ricerca tradizionale.

4.5 Creazione del Retriever Ibrido

Il retriever ibrido viene costruito grazie all'indice ibrido caricato dinamicamente dal cloud, in base alla collezione selezionata viene caricato il giusto indice, grazie al quale si costruisce il retriever. Pinecone permette di specificare il parametro *alpha* per dare più o meno peso alla ricerca basata su similarità semantica oppure alla ricerca basata su keyword. In questo contesto, si è fatto uso di *alpha=0.5* per rendere la ricerca perfettamente ibrida per la collezione dei bandi di sistema puglia. Mentre, per l'altra collezione si è scelto *alpha=0.4* per dar maggior peso alla ricerca basata su keyword a seguito di sperimentazioni.

4.6 Response mode e Prompt Template

Come già accennato in precedenza, l'utente ha la possibilità di scegliere il parametro *response mode* in base alle proprie esigenze.

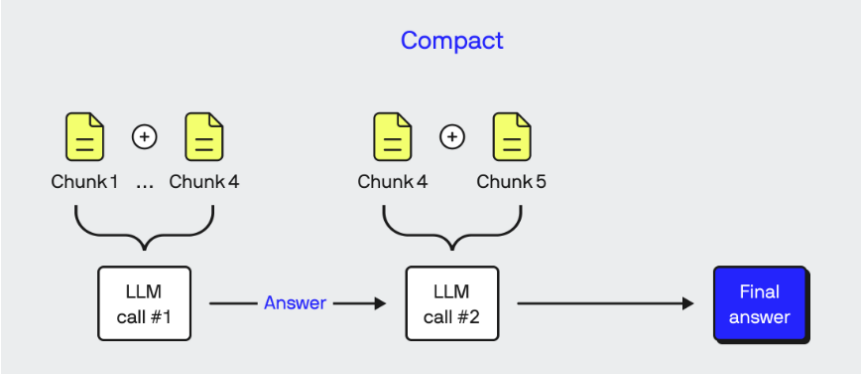


FIGURA 4.4: Response Mode "Compact"

Nella modalità di risposta *compact*, vengono concatenati quanti più *chunk* possibile in una sola chiamata al modello di linguaggio (LLM) per ridurre il numero di chiamate necessarie. La prima chiamata al LLM utilizza il prompt *textQaTemplate*. La risposta ottenuta dalla prima chiamata viene fornita come contesto, insieme ai nuovi *chunk*, per ottenere una nuova risposta. Mentre per la prima chiamata si utilizza il prompt *textQaTemplate*, per le chiamate successive si utilizza il prompt *refineTemplate* per raffinare o aggiornare le risposte iterativamente con i nuovi *chunk*.

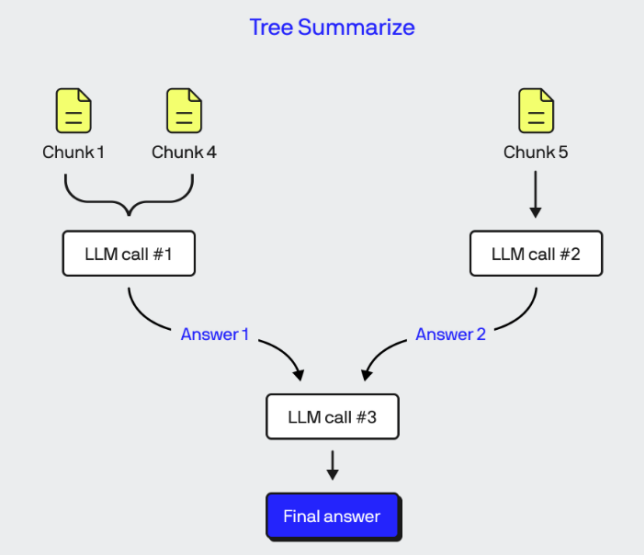


FIGURA 4.5: Response Mode "Tree Summarize"

La modalità di risposta *TreeSummarize*, invece, generalmente considerata la più efficiente, effettua una serie di chiamate al LLM concatenando i *chunk* in modo da massimizzare il limite di token in input al modello di linguaggio. Le risposte iniziali fornite dal LLM vengono poi passate come contesto in una nuova chiamata per derivare una risposta finale. Per la modalità *TreeSummarize* si utilizza esclusivamente il *textQaTemplate*.

4.7 Filtraggio con metadati

Il filtraggio con metadati è una tecnica per filtrare i documenti (o nodi) sulla base del valore di metadati, quindi sulla base di informazioni non presenti nel documento. Nel sistema RAG sviluppato si è fatto uso di questo tipo di filtraggio per selezionare i nodi in base al nome del bando interessato. Nello specifico, ogni nodo memorizzato nel Vector Store ha nei metadati il nome del bando relativo. Ogni volta che viene effettuata una richiesta al sistema si controlla se sono presenti nella query una o più parole chiave associabili al nome di bandi. Se presenti, si effettua un filtraggio inziale dei nodi sulla base del nome del bando presente nei metadati. Altrimenti, si effettua il recupero normale per mezzo del Retriever ibrido senza un filtraggio iniziale dei nodi. Se, per esempio, nella query è presente una parola chiave come "nidi" verrano filtrati tutti i nodi aventi nei metadati il nome del bando "Nuove Iniziative d'Impresa". Invece, se presente una parola chiave come "INNOAID", verranno filtrati i nodi aventi nei metadati il nome del bando "INNOAID-RIAPERTURA". Una volta effettuato

il filtraggio iniziale dei nodi usando i metadati, il Retriever recupererà dal Vector Store i 3 nodi con la maggior similarità alla query iniziale.

4.8 Streaming

Infine, è stato utilizzato il parametro 'Streaming=True' nella creazione del *Response Synthesizer* per attivare la modalità di streaming delle risposte. Questa configurazione risulta cruciale per migliorare l'interazione utente, poiché consente la visualizzazione della risposta in tempo reale sull'interfaccia utente sviluppata con Gradio. L'abilitazione dello streaming riduce significativamente la latenza percepita durante l'elaborazione delle query, offrendo un'esperienza utente più fluida e reattiva. Questo approccio ottimizza non solo l'efficienza del sistema, ma incrementa anche la soddisfazione dell'utente, fornendo risposte incrementali e immediate mentre il modello continua a processare e generare il testo completo.

Capitolo 5

Sperimentazione

Il sistema sviluppato consente agli utenti di reperire rapidamente informazioni sui bandi regionali. Ad esempio, è possibile utilizzare il sistema per scoprire gli obiettivi, i finanziamenti, i requisiti, la procedura di selezione e le scadenze di un bando specifico. Il sistema, una volta ricevuta la domanda, utilizzerà i bandi della regione Puglia, raccolti in una collezione di documenti, per generare una risposta informativa, accurata e contestualmente rilevante.

5.1 Caratteristiche principali del sistema:

- **Scelta della collezione:** Gli utenti possono selezionare quale collezione di bandi interrogare, in base alle proprie esigenze informative.
- **Trasparenza delle fonti:** L'interfaccia mostra le porzioni di testo dei bandi utilizzate dal sistema per generare le risposte, permettendo agli utenti di verificare l'attendibilità delle informazioni fornite.
- **Download dei documenti:** È possibile scaricare il file del bando regionale recuperato, offrendo un accesso diretto ai documenti originali.

5.2 Modalità di utilizzo:

- **STANDARD**: Questa modalità permette agli utenti di porre domande indipendenti tra loro, con risposte che non influenzano le successive.
- **CHATBOT**: Questa modalità consente di avere conversazioni con il sistema sulle informazioni dei bandi regionali. Ogni risposta generata dal chatbot tiene conto dello storico della chat, comprese le domande e le risposte precedenti.

In sintesi, il sistema offre una soluzione efficiente e affidabile per ottenere informazioni dettagliate sui bandi regionali, con modalità di interrogazione flessibili e trasparenti che soddisfano diverse esigenze informative.

5.3 Documenti indicizzati

I documenti indicizzati e usati per il recupero del contesto sono bandi di due differenti collezioni. Per quanto riguarda la collezione dei bandi caricati sul portale Sistema Puglia, qui di seguito riportiamo i nomi dei bandi indicizzati con il corrispettivo numero di token:

- Aiuti ai programmi di internazionalizzazione delle Piccole e Medie Imprese: 2457 token
- Apprendistato Professionalizzante: 542 token
- Avviso Multimisura - Garanzia Giovani II Fase: 1227 token
- Avviso Multimisura POC: 1154 token
- Scheda Accreditamento Organismi Formativi: 1096 token
- Scheda Accreditamento Servizi per il Lavoro: 725 token
- Scheda Avviso "Punti Cardinali" punti di orientamento per la formazione e il lavoro: 663 token
- Scheda Avviso PNRR - Impianti idrogeno rinnovabile: 1019 token
- Scheda Avviso Pubblico "Giardiniere d'arte per giardini e parchi storici": 1067 token
- Scheda Avviso Pubblico Diploma Professionale 2022: 2348 token
- Scheda Avviso Pubblico IFTS 2023: 988 token
- Scheda Avviso Pubblico OF 2023-2024: 1447 token
- Scheda Avviso Tecnonidi - Aiuti alle piccole imprese innovative: 673 token
- Scheda di sintesi Avviso INNOAID - RIAPERTURA: 764 token
- Scheda Efficientamento Energetico Edifici Pubblici: 570 token
- Scheda Garanzia di occupabilità dei lavoratori - GOL: 380 token

- Scheda MicroPrestito della Regione Puglia - edizione 2021: 753 token
- Scheda NIDI - Nuove iniziative d'impresa: Strumento di ingegneria finanziaria: 512 token
- Scheda Pass Laureati 2023: 1476 token
- Scheda RED 2020: 1155 token
- Scheda Titolo VI - Aiuti per la tutela dell'ambiente: 2459 token
- Scheda Piani Formativi Aziendali: 518 token

La collezione dei bandi di Sistema Puglia memorizza 22 documenti con un peso totale di 3.94 MB. L'altra collezione, invece, memorizza ben 5703 documenti contenenti informazioni sui bandi regionali, con una dimensione totale di 250 MB. I nodi e i vettori vengono memorizzati insieme ai metadati nell'indice del database vettoriale in base alla collezione di appartenenza. L'indice "*indexbandisistemapuglia*" contiene 96 vettori, mentre l'indice "*indexbandi*" contiene 116.000 vettori. In totale i due indici occupano 0.94 GB di memoria. Poichè l'indice *indexbandisistemapuglia* memorizza 96 vettori per 22 documenti, sono stati creati 4.36 vettori in media per ogni documento. Invece, nell'indice "*indexbandi*" sono stati memorizzati 116.498 vettori per i 5703 documenti, perciò sono stati generati 20.42 vettori in media per ogni documento.

5.4 Tempi di recupero nodi

Il Retriever del sistema è configurato per recuperare 3 nodi ogniqualvolta viene posta una domanda al sistema RAG. Tuttavia, per domande specifiche relative ad un bando dal breve contenuto come il bando "Scheda Garanzia di occupabilità dei lavoratori" il sistema potrebbe recuperare meno di tre nodi dal momento che la dimensione di un nodo è di ben 975 token. Effettuando alcuni test su diverse query, sono emersi i seguenti tempi di recupero:

- Data la query "Parlami del bando NIDI" il sistema ha recuperato i tre nodi relativi al bando NIDI (Nuove Iniziative d'Impresa) in 0.677497386932373 secondi.
- Data la query "Confronta il bando NIDI e TECNONIDI" il sistema ha recuperato i tre nodi relativi ai due bandi in 0.6250035762786865 secondi.

- Data la query "Descrivi nel dettaglio obiettivi e beneficiari del bando Giardiniere d'arte" il sistema ha recuperato i tre nodi relativi al bando "Scheda Avviso Pubblico Giardiniere d'arte per giardini e parchi storici" in appena 0.13485121726989746 secondi.

- Data la query "A chi è rivolto il bando Laureati?" il sistema ha recuperato i tre nodi relativi al bando "Scheda Pass Laureati 2023" in 0.6238908767700195 secondi.

- Data la query "Puoi sintetizzarmi il bando Punti Cardinali?" il sistema ha recuperati i tre nodi relativi al bando "Scheda Avviso Punti Cardinali punti di orientamento per la formazione e il lavoro" in 0.629082202911377 secondi.

- Data la query "Quali sono gli obiettivi del bando GOL?" il sistema ha recuperato il nodo relativo al bando "Scheda Garanzia di occupabilità dei lavoratori" in 0.7270689010620117 secondi.

Facendo una media dei tempi di recupero sopra riportati otteniamo una media di 0.56 secondi per il recupero dei nodi data una query in input al sistema.

5.5 Tempi di risposta

- Data la query "Parlami del bando NIDI" il sistema ha fornito una risposta completa in 35.75824952125549 secondi.

- Data la query "Confronta il bando NIDI e TECNONIDI" il sistema ha fornito una risposta completa in 37.78096675872803 secondi.

- Data la query "Descrivi nel dettaglio obiettivi e beneficiari del bando Giardiniere d'arte" il sistema ha fornito una risposta completa in 36.434473514556885 secondi.

- Data la query "A chi è rivolto il bando Laureati?" il sistema ha fornito una risposta completa in 37.98905920982361 secondi.

- Data la query "Puoi sintetizzarmi il bando Punti Cardinali?" il sistema ha fornito una risposta completa in 37.1066780090332 secondi.

- Data la query "Quali sono gli obiettivi del bando GOL?" il sistema ha fornito una risposta completa in 37.578052043914795 secondi.

Facendo una media dei tempi di risposta sopra riportati otteniamo una media di 37.1 secondi per la generazione di una risposta completa. Per ridurre la latenza percepita dall'utente i token della risposta vengono generati progressivamente attraverso le streaming della risposta.

5.6 Risultati Sperimentazione

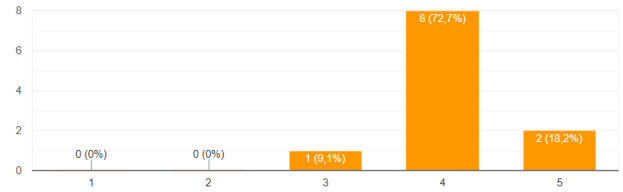
Per la sperimentazione il sistema è stato testato da dipendenti della Regione Puglia, i quali dopo averlo utilizzato per determinati task prestabiliti hanno compilato un questionario. Il questionario in questione è stato creato per raccogliere i risultati e comprendere i punti di forza e di debolezza del sistema. Nello specifico, all'interno del questionario è stata inserita una breve descrizione del sistema ed un link per un tutorial all'uso. Quindi, il sistema è stato testato dagli utilizzatori sui seguenti tre task:

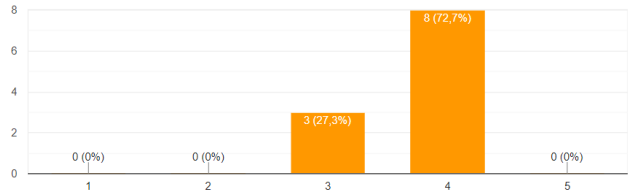
1. Ritrova obiettivi e beneficiari del bando Nuove Iniziative d'impresa (NIDI) nella collezione "BANDI SISTEMA PUGLIA".
2. Ritrova una descrizione accurata del programma Garanzia Occupabilità Lavoratori (GOL) nella collezione "BANDI SISTEMA PUGLIA".
3. Ritrova i beneficiari del bando "Giardiniere d'arte per giardini e parchi storici" nella collezione "BANDI SISTEMA PUGLIA".

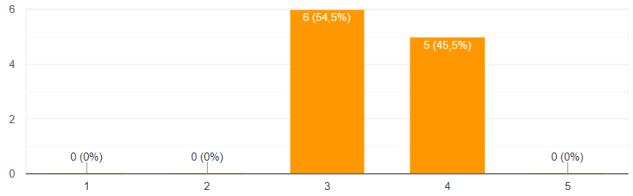
Per ogni task gli utilizzatori hanno valutato in una scala da 1 a 5 prima la correttezza e la completezza della risposta fornita dal sistema. Successivamente, per una valutazione generale del sistema gli utilizzatori hanno risposto a diverse domande sulla medesima scala da 1 a 5. Ogni valore della scala ha il seguente significato:

- Per niente
- Poco
- Moderatamente
- Molto
- Completamente

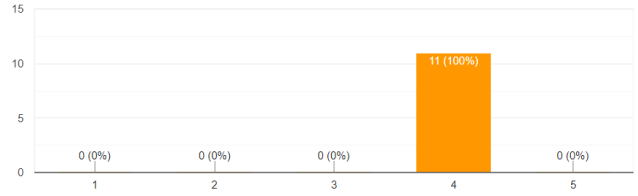
Per il primo task si sono ottenuti i seguenti risultati:

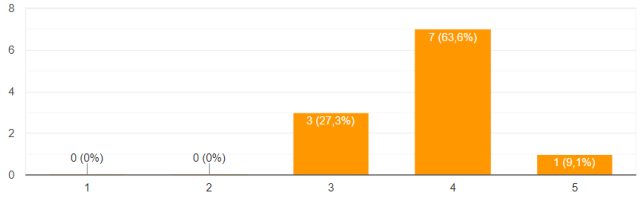


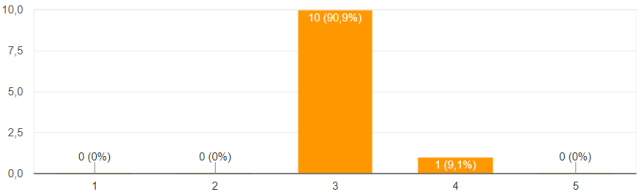




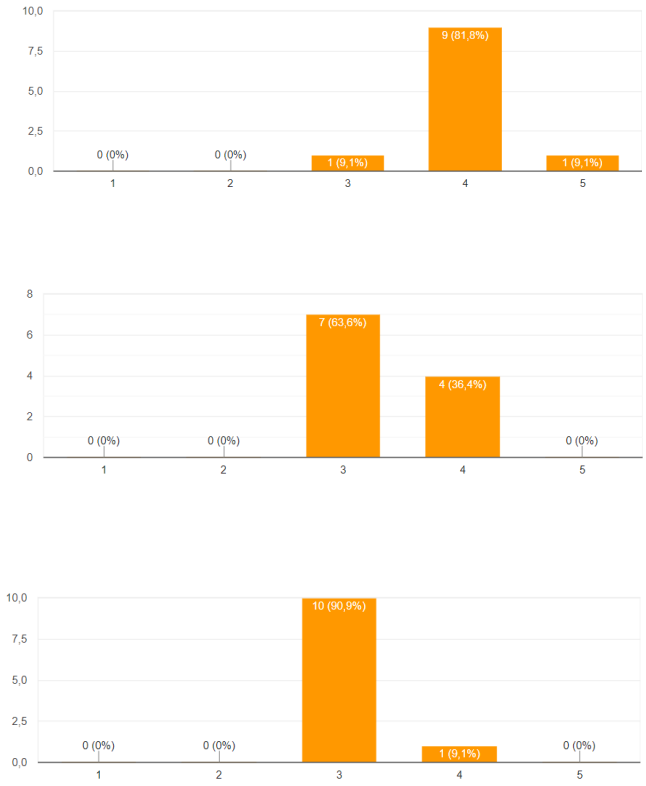
Per il secondo task si sono ottenuti i seguenti risultati:





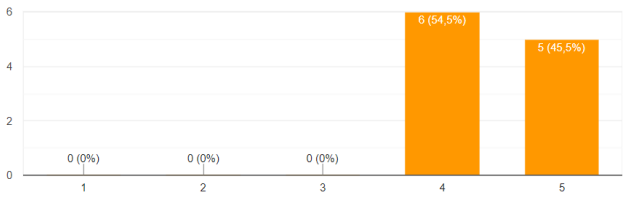


Per il terzo task si sono ottenuti i seguenti risultati:

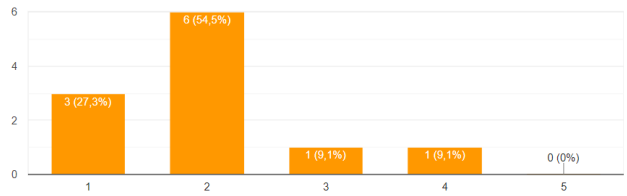


Analizzando i risultati del questionario relativo ai tre task, si rileva che mediamente il 63.3% degli utenti ha valutato le risposte del sistema come molto corrette, il 6% le ha giudicate completamente corrette, mentre la restante parte ha ritenuto le risposte moderatamente corrette. In media, il 39.4% dei dipendenti regionali ha valutato le risposte del sistema come moderatamente complete, il 57.5% le ha considerate molto complete, mentre il 3.1% restante degli utenti ha ritenuto che le risposte fossero del tutto complete. Per quanto riguarda la correttezza grammaticale delle risposte, il 54.5% degli utenti ha valutato le risposte del sistema come moderatamente corrette dal punto di vista grammaticale, mentre il rimanente 42.4% le ha giudicate molto corrette grammaticalmente. Quindi, possiamo dire che le risposte fornite dal sistema sono ritenute complessivamente molto corrette e molto complete. Tuttavia, è emerso il bisogno di perfezionare la correttezza grammaticale delle risposte generate dal sistema. In quanto alla valutazione generale del sistema, gli utilizzatori della Regione hanno espresso il seguente grado di accordo alle elencate affermazioni:

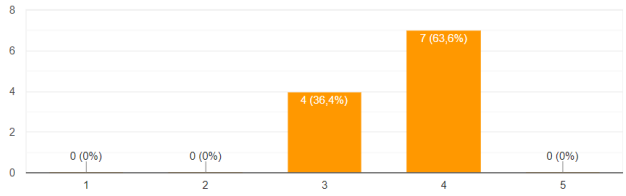
1. Utilizzerei questo sistema frequentemente.



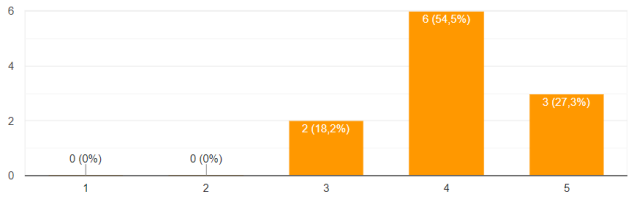
2. Ho trovato il sistema eccessivamente complesso.



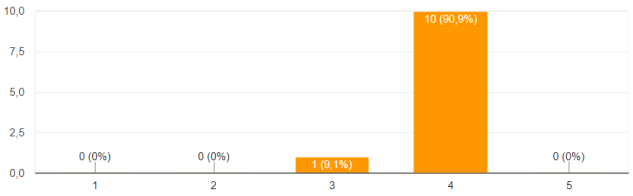
3. Credo che la maggior parte delle persone imparerebbe a usare il sistema molto velocemente.



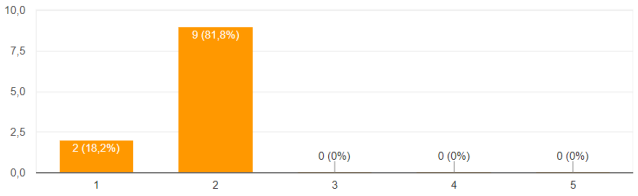
4. Mi sono sentito molto sicuro di me stesso nell'usare il sistema.



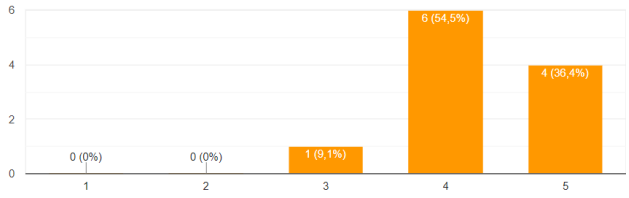
5. Le risposte del sistema sono state utili, informative e appropriate.



6. Il sistema non è stato in grado di capire molte delle mie domande.



7. L'interazione con il sistema è stata realistica e coinvolgente.



Sulla base del feedback degli utenti sull'esperienza generale di utilizzo del sistema possiamo dire che il sistema sviluppato è risultato utile, piacevole, facile da utilizzare, e coinvolgente.

Capitolo 6

Conclusioni

Con questa tesi, abbiamo sviluppato un sistema basato su Retrieval-Augmented Generation (RAG) che permette di recuperare efficacemente informazioni contestuali e rilevanti sui bandi regionali della Puglia. Il sistema consente agli utenti di selezionare la collezione di bandi di interesse e scegliere la modalità di risposta preferita, garantendo un'esperienza personalizzata e flessibile.

In particolare, il sistema offre due modalità di utilizzo in base all'esperienza desiderata:

- **Modalità STANDARD**: In questa modalità, gli utenti possono porre domande indipendenti al sistema, ottenendo risposte accurate e aggiornate non influenzate dal contesto delle conversazioni precedenti.
- **Modalità CHAT**: Questa modalità permette di avere conversazioni sulle informazioni dei bandi regionali, con il sistema che tiene conto della cronologia della chat per generare nuove risposte. Questa funzione migliora la coerenza e la pertinenza delle risposte fornite durante una sessione interattiva prolungata.

I risultati ottenuti dimostrano che il sistema sviluppato è in grado di migliorare l'accesso e la fruizione delle informazioni relative ai bandi regionali, rendendole più facilmente recuperabili e comprensibili per gli utenti. Un ulteriore punto di forza del sistema è la sua capacità di gestire un numero elevato di bandi, infatti, si potrebbero aggiungere nuove collezioni di bandi senza ottenere un degrado delle prestazioni, garantendo così la scalabilità necessaria per ampliamenti futuri.

Tuttavia, ci sono ancora margini di miglioramento. In particolare, il sistema può essere ottimizzato per:

- Raggiungere una migliore correttezza grammaticale nelle risposte generate.

- Recuperare velocemente un numero maggiore di nodi dal Vector Store senza usare informazioni irrilevanti per la generazione della risposta.
- Aumentare la flessibilità nella comprensione delle domande, adattandosi meglio alle diverse formulazioni e contesti di utilizzo.

Inoltre, il sistema potrebbe essere migliorato per permettere agli utenti stessi di caricare nuovi bandi, che verrebbero indicizzati dal sistema in tempo reale. Questo miglioramento aumenterebbe l'autonomia degli utenti e la flessibilità di aggiornamento del database vettoriale del sistema RAG. Un'altra possibile evoluzione del sistema potrebbe essere quella di permettere al sistema di raggruppare i bandi caricati dagli utenti in collezioni di bandi semanticamente simili, facilitando così la gestione e la ricerca delle informazioni.

Questi sviluppi futuri potranno ulteriormente rafforzare l'efficacia del sistema RAG sviluppato, rendendolo uno strumento ancora più potente e affidabile per il recupero di informazioni sui bandi regionali della Puglia.

Bibliografia

[1] Sampriti Saxena Hannah Chafetz e Stefaan G.Verhulst. «A Fourth Wave of Open Data? Exploring the Spectrum of Scenarios for Open Data and Generative AI». In: (2024).

[2] Kai Zhang Shiwei Tong Qi Liu Hao Yu Aoran Gan e Zhaofeng Liu. «Evaluation of Retrieval-Augmented Generation: A Survey». In: (2024).

[3] Weitao Ma1 Weihong Zhong1 Zhangyin Feng1 Haotian Wang1 Qianglong Chen2 Weihua Peng2 Xiaocheng Feng1 Bing Qin1 Ting Liu1 Lei Huang1 Weijiang Yu2. «A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions». In: (2023).

[4] Sriparna Saha Vinija Jain Samrat Mondal e Aman Chadha Pranab Sahoo Ayush Kumar Singh. «A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications». In: (2024).

[5] Barbara Ubaldi. «Open Government Data: Towards Empirical Analysis of Open Government Data Initiatives». In: (2023).

[6] Pavlos Vougiouklis Nikos Papasarantopoulos Jeff Z. Pan Wenyu Huang Mirella Lapata. «Retrieval Augmented Generation with Rich Answer Encoding». In: (2023).

[7] Wenhao Yu. «Retrieval-augmented Generation across Heterogeneous Knowledge». In: (2022).

[8] Fandong Meng Yafu Li Jie Zhou Yue Zhang Yun Luo Zhen Yang. «An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning». In: (2024).

[9] Yinyang Liu Jeff Zhang Sai Qian Zhang Zeyu Han Chao Gao. «Parameter Efficient Fine-Tuning for Large Models: A Comprehensive Survey». In: (2024).

[10] Wentao Zhang. «Retrieval-Augmented Generation for AI-Generated Content: A Survey». In: (2024).

Ringraziamenti

Desidero esprimere la mia più sincera e profonda gratitudine a tutti coloro che hanno contribuito alla realizzazione di questa tesi e al completamento del mio percorso accademico.

In primo luogo, ringrazio i miei relatori, Pasquale Lops e Pierpaolo Basile, per la loro guida, per la loro pazienza e per i preziosi consigli forniti durante tutto il percorso di ricerca. La loro esperienza e disponibilità sono state fondamentali per il completamento di questo lavoro.

Un ringraziamento speciale va a mia sorella, la mia guida durante questo percorso, la persona che più di tutti mi ha sostenuto nei momenti difficili. Senza il suo costante supporto e la sua incrollabile fiducia in me, non sarei mai riuscito a superare tutte le sfide incontrate lungo il cammino.

Un grande ringraziamento va ai miei genitori, il cui supporto è stato un fondamentale punto di riferimento nel mio percorso accademico. Un ringraziamento speciale va anche alle mie nonne, agli zii, alle zie e ai cugini, che hanno contribuito con il loro affetto e incoraggiamento.

Ringrazio di cuore Nicolò per i preziosi consigli e le utili indicazioni fornite durante tutto il percorso di studi. La sua saggezza e il suo supporto sono stati di inestimabile importanza.

Desidero esprimere, inoltre, la mia profonda gratitudine a tutti i miei amici per il loro sostegno e la loro costante presenza durante questo percorso. Un ringraziamento in particolare va a Biagio, Donato, Emanuele, Francesco e Nicola, i quali hanno reso questi anni speciali con la loro compagnia, regalandomi momenti di spensieratezza e felicità indimenticabili. Senza di loro, questo cammino sarebbe stato decisamente più faticoso.

Concludo esprimendo il mio sincero ringraziamento a tutti coloro che, in un modo o nell'altro, hanno contribuito a questo importante traguardo della mia vita. Ogni gesto di supporto, ogni parola di incoraggiamento e ogni atto di gentilezza hanno avuto un ruolo fondamentale nel rendere possibile questo successo, e per questo sarò eternamente grato.