Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Informatica

A.a. 2025/2026

Sessione di laurea Marzo 2026

Agentic AI RAG-based nei workflow documentali a supporto dei processi amministrativi digitali

EOS Reply Case Study

Relatore:

Andrea Bottino

Candidato:

Davide Proglio

Sommario

La digitalizzazione e ottimizzazione dei processi amministrativi è un aspetto chiave nell'evoluzione delle aziende. EOS Reply si è occupata di realizzare, per un proprio cliente, un applicativo basato su un workflow collaborativo che definisce l'insieme strutturato delle fasi operative per la gestione, redazione e pubblicazione degli atti amministrativi. Questo processo presenta un'elevata complessità dovuta alla sua natura multiattore e multifase, vincoli normativi stringenti e prassi eterogenee. Sebbene l'adozione del sistema abbia efficientato significativamente il processo amministrativo, ha anche rappresentato un cambiamento rilevante per operatori con competenze digitali variabili, generando in alcuni casi disorientamento, errori operativi e quindi minor produttività. Un'analisi dei ticket di assistenza ha evidenziato un tempo medio di risoluzione di circa 7 ore lavorative, con oltre il 50% delle richieste risolvibili tramite consultazione della documentazione.
Il presente progetto di tesi mira a realizzare un assistente conversazionale in grado di supportare in tempo reale gli operatori del sistema documentale, fornendo loro informazioni affidabili per guidarli nel corretto svolgimento del workflow amministrativo, potenziandone l'operatività e rendendo il processo più accessibile.
L'assistente sviluppato si basa sulle capacità cognitive dei Large Language Model (LLM) e sulla Retrieval-Augmented Generation (RAG) per generare risposte fondate sulla documentazione ufficiale del sistema documentale, storicizzata in una knowledge base vettoriale. Il sistema è stato integrato nell'infrastruttura preesistente come servizio tramite API e distribuito in ambiente containerizzato, fornendo agli utenti finali un'interfaccia dedicata all'interno dell'applicativo originale. Nel corso della sperimentazione sono stati sviluppati e confrontati vari prototipi differenziati per scelte architetturali. La valutazione è stata condotta mediante un test set reference-based di 30 istanze costruito con un esperto di dominio. Il componente di retrieval è stato valutato con metriche di precision, recall e rank-based (MRR, MAP), al fine di misurare la capacità del sistema di estrarre e ordinare i documenti rilevanti dalla knowledge base. Il modulo di generazione è stato invece analizzato mediante metriche tradizionali (ROUGE), semantiche (BERTScore) e LLM-as-a-Judge (RAGAS), unitamente alla latenza di risposta, per valutare pertinenza, accuratezza e fedeltà delle risposte.
I risultati hanno permesso di identificare il prototipo ottimale con tempi di risposta compatibili con scenari real-time (5.7 secondi), buone capacità di retrieval e promettenti performance di coerenza, correttezza, e in particolare di aderenza alla documentazione (faithfulness 0.847), dimostrando equilibrio tra qualità, affidabilità ed efficienza.
Il progetto fornisce una metodologia per la realizzazione e valutazione di assistenti

conversazionali basati su LLM e RAG in contesti ad elevata complessità procedurale, dalla costruzione della knowledge base all'integrazione sistemica, con evidenze empiriche sui trade-off progettuali.
Nel complesso il lavoro di tesi rappresenta un tentativo concreto di applicazione di agenti intelligenti come strumenti di supporto e potenziamento operativo, rendendo processi complessi più accessibili e riducendo il carico di assistenza. Nonostante i risultati promettenti, restano margini di miglioramento sulla limitatezza del test set, l'aggiornamento della knowledge base e l'ottimizzazione delle performance.

Indice

Elenco delle tabelle

Elenco delle figure

Glossario

BPM

Business Process Management

BPMS

Business Process Management System

LLM

Large Language Model

MRAG

Multimodal Retrieval-Augmented Generation

RAG

Retrieval-Augmented Generation

Capitolo 1

Introduzione

1.1 Contesto Applicativo

La digitalizzazione e l'ottimizzazione dei processi di business costituiscono un pilastro strategico per l'efficientamento delle organizzazioni pubbliche e private. Un processo aziendale può essere definito come un insieme di attività svolte in modo coordinato in un ambiente organizzativo e tecnologico, che nel loro insieme realizzano un obiettivo di business [1]. Tra questi rientrano i processi amministrativi che consistono in procedure operative per la gestione di aspetti amministrativi, contabili o documentali di un'organizzazione.
La trasformazione digitale di questi processi rappresenta un aspetto d'interesse per le organizzazioni in quanto consente di gestirne la complessità in maniera tracciata e controllata, coordinando persone, sistemi e dati e migliorando le prestazioni aziendali sul piano operativo ed economico.
Il Business Process Management (BPM) è la disciplina che, integrando prospettive informatiche e gestionali, si occupa di identificare e analizzare i processi aziendali, formalizzandoli in modelli di processo che definiscono le attività e i vincoli di esecuzione tra esse, fungendo da schema di riferimento per le rispettive istanze di processo [1]. Alla componente analitica si affianca quella tecnologica che consente di attualizzare i modelli progettati: i Business Process Management System (BPMS) sono sistemi software generici guidati da rappresentazioni esplicite dei processi, che coordinano l'esecuzione delle istanze distribuendo le attività tra i soggetti coinvolti secondo ruoli e responsabilità predefiniti [1].
Negli ultimi anni si è assistito a una rapida evoluzione nel campo dell'intelligenza artificiale [2]. In particolare, la diffusione dei Large Language Models (LLM), con le loro abilità emergenti, ha posto le basi per la realizzazione di agenti intelligenti, in grado di svolgere compiti complessi e articolati [3]. Inoltre, l'introduzione di nuove soluzioni architetturali come la Retrieval-Augmented Generation (RAG) [4] hanno

consentito il superamento di alcune limitazioni caratteristiche dei grandi modelli linguistici, rendendo possibile la generazione di contenuti fondati su documentazione ufficiale e specialistica, estendendo le conoscenze implicite degli LLM con fonti esterne aggiornate. L'avanzamento di questo tipo di tecnologie è stato accompagnato dall'introduzione di numerosi frameworks per la realizzazione di agenti AI [5], che astraggono e semplificano la creazione di sistemi basati su LLM, aprendo la strada alla loro applicazione in contesti operativi reali. Tale evoluzione offre nuove opportunità anche nel campo della gestione dei processi aziendali per efficientare ulteriormente i processi digitali dal punto di vista operativo ed economico [6].
Reply [7] è un gruppo specializzato nella progettazione e realizzazione di soluzioni digitali, strutturato come rete di aziende altamente specializzate. Il gruppo affianca i principali operatori dei settori Telco & Media, Industria e Servizi, Banche e Assicurazioni e Pubblica Amministrazione, supportandoli nella definizione e nello sviluppo di modelli di business abilitati da tecnologie come intelligenza artificiale, cloud computing e Internet of Things. I servizi offerti spaziano dalla consulenza alla system integration, fino ai servizi digitali. Il presente progetto di tesi è stato svolto internamente a Eos Reply, società del gruppo focalizzata sulla trasformazione e digitalizzazione dei processi nell'area Finance & Accounting e amministrativa. Attraverso la combinazione di competenze funzionali, tecnologiche e di project management, Eos Reply affianca i propri clienti nell'ottimizzazione dei processi amministrativi, permettendo loro di concentrarsi sulle attività core grazie a soluzioni ad alto valore aggiunto.

1.2 Identificazione del Problema

Eos Reply si è occupata, per conto di un proprio cliente, della modellazione e dell'attualizzazione tramite BPMS di un workflow collaborativo che definisce l'insieme delle operazioni necessarie alla gestione, redazione e pubblicazione di atti amministrativi. Il processo amministrativo in questione non è una semplice sequenza lineare: si tratta di un flusso estremamente articolato, che coinvolge numerosi attori e attività eterogenee, quali la scrittura collaborativa dei testi, l'attivazione di processi di registrazione contabile che generano allegati ufficiali, molteplici livelli di approvazione, operazioni di firma digitale, rilascio di visti da parte di uffici competenti, e infine la pubblicazione verso l'esterno. A complicare ulteriormente il quadro contribuiscono vincoli normativi stringenti, responsabilità distribuite e pratiche operative non sempre uniformi. La modellazione del workflow amministrativo e l'attuazione dell'applicativo software che ne digitalizza l'esecuzione hanno permesso di efficientare significativamente dal punto di vista operativo ed economico il processo che prima veniva gestito in maniera non strutturata. Tuttavia, questa evoluzione ha anche rappresentato un cambiamento rilevante per gli operatori

coinvolti, caratterizzati da frequenze di utilizzo differenti e competenze digitali eterogenee. Il disorientamento che ne è conseguito si è tradotto in errori operativi, con un impatto negativo sulla produttività complessiva e un aumento del carico di lavoro in termini di assistenza.
Attraverso un'analisi preliminare condotta sui ticket di assistenza relativi all'applicativo software, raccolti tra il 12 maggio e il 19 settembre 2025 e classificati manualmente, è emersa la distribuzione di casistiche presentata in Figura 1.1.

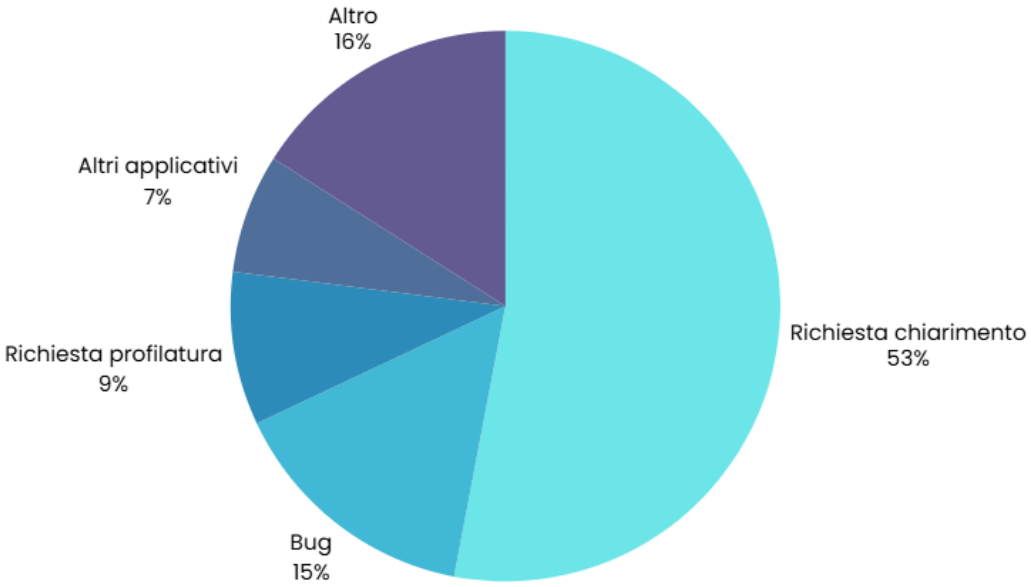

Figura 1.1: Distribuzione tickets di supporto

Delle 89 richieste di assistenza esaminate, il 53% riguarda richieste di chiarimento relative alla sequenza di operazioni che compongono il modello di processo e alle funzionalità fornite dell'applicativo per l'esecuzione di quest'ultimo. Questo tipo di problematiche risultano facilmente risolvibili attraverso la consultazione della documentazione ufficiale del sistema nella quale sono riportate in maniera dettagliata tutte le fasi del processo amministrativo, le responsabilità coinvolte, le casistiche verificabili e le procedure operative da seguire. Attualmente, le richieste di supporto vengono gestite in modo capillare attraverso assistenza telefonica, a cui segue la chiusura del ticket corrispondente. Dall'indagine svolta è emerso un tempo medio di risoluzione di 6.98 ore lavorative, un intervallo di tempo durante il quale gli operatori del sistema subiscono inevitabili rallentamenti nelle loro mansioni quotidiane.

1.3 Obiettivi del Progetto

La sperimentazione condotta in questo progetto di tesi mira a intervenire sulle criticità appena descritte, sviluppando e integrando nell'applicativo amministrativo un agente conversazionale basato su LLM e RAG, capace di supportare in tempo reale gli operatori del sistema, guidandoli nel corretto svolgimento del processo amministrativo attraverso informazioni affidabili e pertinenti rispetto alle esigenze espresse, ricavate dalla documentazione ufficiale del sistema. Tale soluzione permette di potenziare l'operatività e la produttività degli utenti, riducendone la dipendenza dai tempi di assistenza, abbattere la probabilità che si verifichino errori procedurali in un contesto nel quale gli atti amministrativi acquisiscono valore legale, rendere un processo complesso e articolato più accessibile a un'utenza con competenze digitali diversificate e di alleggerire il carico di assistenza erogato all'organizzazione cliente.

1.4 Struttura della Tesi

Il progetto di tesi si articola in cinque capitoli, organizzati secondo una progressione logica che muove dai concetti teorici fondamentali alla comprensione della soluzione proposta, fino all'analisi dei risultati ottenuti. Di seguito viene illustrata brevemente la strutturazione in capitoli del progetto e i contenuti trattati:

- **Introduzione**: presenta il contesto applicativo in cui si colloca il progetto di tesi, presentando le motivazioni alla base della sperimentazione e gli obiettivi perseguiti. Il capitolo si conclude con la descrizione della struttura e dell'organizzazione dei contenuti dell'elaborato.

- **Stato dell'Arte e Background**: analizza il panorama tecnologico di riferimento, partendo dal campo applicativo del BPM per poi concentrarsi su LLM e agenti AI. Vengono presentate le principali soluzioni architetturali per la realizzazione di sistemi agentici, con particolare attenzione alla RAG, sulla quale si basa la soluzione proposta. Il capitolo si conclude con la descrizione di alcuni frameworks per la realizzazione di agenti intelligenti.

- **Progettazione, Sviluppo e Integrazione**: illustra il percorso che ha portato alla realizzazione dell'assistente conversazionale, partendo dall'analisi dei requisiti e dalle scelte progettuali iniziali. Vengono descritti nel dettaglio il sistema RAG, con le relative pipeline di ingestion e retrieval, e i due prototipi di agente sviluppati. Il capitolo si chiude con una panoramica dell'infrastruttura adottata e dell'interfaccia utente progettata per l'integrazione nell'applicativo di BPMS preesistente.

- **Risultati Sperimentali**: descrive la metodologia di valutazione adottata e le metriche impiegate per misurare le performance dell'agente conversazionale. Vengono presentati e confrontati i risultati ottenuti per ciascun prototipo realizzato, evidenziando trade-off progettuali e individuando la configurazione ottimale per il caso d'uso di interesse.
- **Conclusioni**: riassume i principali contributi della sperimentazione, evidenziando i risultati significativi raggiunti e le limitazioni ancora presenti. Vengono inoltre, discusse possibili direzioni future per migliorare il sistema realizzato e superare le criticità emerse.

Capitolo 2

Stato dell'Arte e Background

2.1 Business Process Management: Fondamenti ed Evoluzione nell'Era dell'AI

Questa sezione introduce i fondamenti del Business Process Management (BPM), analizzando il ruolo dei processi di business nelle organizzazioni e il ciclo di vita che ne guida progettazione, esecuzione e miglioramento continuo. Viene inoltre discusso il funzionamento dei Business Process Management System e l'evoluzione del BPM nell'era dell'intelligenza artificiale, con particolare attenzione ai nuovi paradigmi conversazionali e ai sistemi BPM aumentati dall'AI.

2.1.1 I Processi di Business e il Business Process Management

I processi di business rappresentano il motore operativo delle imprese, attraverso i quali viene prodotto valore per l'organizzazione. Questi possono essere categorizzati in due tipologie [8]: processi primari e di supporto. I processi primari trasformano risorse, informazioni e materie prime in servizi o prodotti destinati ai clienti, come avviene ad esempio nei processi di produzione, vendita o consegna. I processi di supporto hanno invece come scopo abilitare il corretto funzionamento dei processi primari, comprendendo procedure amministrative, di approvvigionamento o HR. Nel settore privato i processi amministrativi rientrano tipicamente in questa seconda categoria, occupandosi di supportare il business principale dell'organizzazione attraverso attività come la gestione delle fatture, l'archiviazione documentale o l'elaborazione delle buste paga. Nel settore pubblico, invece, i processi amministrativi

assumono il ruolo di processi primari, in quanto il procedimento amministrativo costituisce esso stesso il servizio erogato ai cittadini o alle imprese.
Come definito nel libro di Weske sul BPM un processo aziendale consiste in un insieme di attività svolte in coordinazione in ambiente organizzato e tecnico, che realizzano collettivamente un obiettivo aziendale. Solitamente questi processi vengono svolti all'interno di una singola organizzazione, ma possono anche interagire con processi di imprese esterne per conseguire il proprio obiettivo.
Il Business Process Management (BPM) è l'ambito scientifico che include concetti, metodi e tecniche finalizzate alla progettazione, gestione, configurazione, attuazione e analisi dei processi di business [1]. L'adozione del BPM, unita alla rappresentazione esplicita dei processi attraverso modelli, produce numerosi benefici per le organizzazioni [9]. Sul piano operativo, consente di identificare e ridurre le inefficienze nei flussi di lavoro, come i periodi di inattività e le attività ridondanti, abbattendo i costi e migliorando la qualità dell'esecuzione. Sul piano strategico, accresce l'agilità aziendale, abilitando la capacità di adattare rapidamente i propri processi alla dinamicità del mercato. I modelli di processo favoriscono inoltre la trasparenza e la comunicazione tra le diverse funzioni aziendali, fungendo da linguaggio comune tra esperti di business e tecnici. Infine, un repository strutturato di modelli costituisce un patrimonio di conoscenza organizzativa, utile per la governance, la tracciabilità e il miglioramento continuo delle prestazioni.

2.1.2 Il Ciclo di Vita del Processo e i Business Process Management System

Dal punto di vista applicativo il BPM fornisce un framework strutturato per la progettazione, realizzazione e miglioramento continuo dei modelli di processo, detto ciclo di vita del processo [10, 11]. La prima fase è quella di design e di analisi, nella quale vengono esaminate le operazioni aziendali al fine di individuare i processi di business e formalizzarli in modelli di processo. Un modello di processo costituisce una descrizione statica delle attività e delle relazioni che intercorrono tra di esse [1]. Per la rappresentazione di questi modelli esiste uno standard adottato dalla comunità scientifica e industriale: il Business Process Model and Notation (BPMN) [12], una notazione grafica standardizzata che consente di descrivere i processi di business in modo preciso e comprensibile sia per profili tecnici che manageriali. Le attività di processo sono unità di lavoro necessarie al corretto svolgimento del processo e si distinguono in due tipologie: manuali, completate dagli operatori coinvolti nel processo, e automatiche, eseguite senza alcun intervento umano. Le relazioni tra le attività sono rappresentate attraverso archi di connessione che ne definiscono la sequenza di esecuzione, seguendo schemi ricorrenti noti come control flow patterns [13], che descrivono le modalità con cui le attività possono essere orchestrate: in sequenza, in parallelo, in modo condizionale o iterativo. Oltre a

modellare la sequenza di operazioni da svolgere, in questa fase viene anche definito chi deve eseguire ciascuna attività, associando ad esse operatori con ruoli e responsabilità ben definiti.
La fase successiva del ciclo di vita è quella di configurazione, che consiste nell'implementare concretamente il flusso di attività modellato in precedenza. Questa implementazione può avvenire attraverso un insieme di procedure e policy operative, ma i benefici più significativi si ottengono adottando un sistema software dedicato alla coordinazione delle attività e alla loro assegnazione ai ruoli previsti in fase di modellazione. Questi applicativi prendono il nome di Business Process Management System (BPMS) e permettono di mettere in atto il processo di business modellato. L'architettura di un BPMS si articola in un insieme di componenti interconnesse che coprono l'intero ciclo di gestione del processo [14]: il sottosistema di modellazione, il repository dei modelli, il motore di processo, i fornitori di servizi e l'interfaccia utente.
Il sottosistema di modellazione fornisce gli strumenti necessari per definire i modelli di processo, specificandone le attività, la struttura e l'ambiente tecnico di esecuzione. I modelli prodotti vengono archiviati nel repository dei modelli, il componente che storicizza il know-how operativo aziendale rendendolo riutilizzabile. Il motore di processo rappresenta il cuore del sistema: è responsabile di istanziare i processi relativi a casi operativi reali e di coordinare l'esecuzione delle attività, determinando in ogni momento quali operazioni devono essere svolte e da chi. Il motore di processo viene avviato dall'ambiente di business process sulla base dei modelli di processo presenti nel repository. Le attività che compongono il processo sono svolte operativamente dai fornitori di servizi, un'altra componente fondamentale del sistema, che possono essere applicazioni software nel caso di attività automatiche, oppure persone nel caso di attività manuali, e che in entrambi i casi interagiscono tramite interfacce standardizzate. I partecipanti umani interagiscono con il sistema attraverso l'interfaccia utente, che presenta loro le attività pronte per l'esecuzione sotto forma di liste di lavoro, consentendo ai knowledge workers di operare in modo guidato e coerente con il modello di processo.
In seguito alla configurazione si avvia la fase di attuazione, nella quale vengono avviate le istanze di processo relative a casi specifici dell'operatività aziendale, che seguono il modello definito in fase di design al fine di perseguire gli obiettivi di business. Questa fase prevede il controllo della corretta orchestrazione delle operazioni e il monitoraggio dello stato delle istanze in esecuzione, oltre alla raccolta di log di sistema che costituiranno la base per la fase successiva.
L'ultima fase del ciclo di vita è quella di valutazione, nella quale i log raccolti durante l'attuazione del BPMS vengono analizzati per valutare la qualità dei modelli di processo e l'adeguatezza della configurazione implementata, individuando inefficienze e possibili margini di miglioramento. Le informazioni ottenute alimentano direttamente una nuova fase di design e analisi, creando un ciclo iterativo in cui il

processo viene corretto e migliorato in modo incrementale.

2.1.3 L'Evoluzione del BPM nell'Era dell'Intelligenza Artificiale

L'avvento dell'intelligenza artificiale sta ridefinendo le logiche fondamentali del Business Process Management, introducendo nuovi paradigmi che vanno ben oltre il semplice impiego di nuovi strumenti tecnologici.
Nell'articolo pubblicato da Rosemann et al. vengono presentate alcune limitazioni del BPM tradizionale nel contesto attuale, individuando tre spostamenti di paradigma (drift), determinati dalla diffusione dell'intelligenza artificiale che caratterizzeranno il BPM di prossima generazione.
La prima generazione di BPM, pur avendo aumentato significativamente la produttività e l'efficienza dei processi di business aziendali, presenta alcune criticità.
La prima è quella che viene definita come la "corsa verso lo zero": una volta raggiunta un'esecuzione impeccabile, il processo BPM diventa un semplice fattore igienico, cessando di rappresentare un valore distintivo per le organizzazioni e perdendo il proprio ruolo come fattore di differenziazione competitiva.
La seconda criticità riguarda la rigida logica transazionale che caratterizza i BPM tradizionali, i quali interpretano i processi come sequenze discrete di transazioni modellate a priori per essere replicate in maniera uniforme. Questa visione statica è limitante in quanto non consente al modello di processo di adattarsi autonomamente a eventi imprevisti o variazioni contestuali.
È proprio in risposta a queste criticità che l'intelligenza artificiale sta abilitando tre spostamenti di paradigma che ridefiniscono le basi del BPM di prossima generazione.
L'avvento dell'AI generativa sta introducendo una modalità alternativa e complementare alla logica transazionale del BPM: quella conversazionale. Per BPM conversazionale si intende un sistema caratterizzato dalla capacità di interagire in linguaggio naturale con le informazioni di processo e con i dati contestuali correlati. Questo cambiamento di paradigma consente agli utenti finali di operare su processi anche estremamente articolati e complessi senza dover padroneggiare la struttura tecnica delle transazioni che li compongono, sostituendo interfacce rigide con conversazioni fluide e adattive.
Il secondo drift riguarda il passaggio da automazione di processo ad autonomizzazione. Quest'ultima va oltre la semplice esecuzione di regole predefinite dagli esseri umani: le macchine acquisiscono la capacità di prendere decisioni autonome e di modificare i processi stessi in base a obiettivi e vincoli definiti, in risposta a dati e eventi contestuali. A differenza dei processi automatizzati, i processi autonomi non sono completamente predicibili, essendo per natura sensibili al contesto e dinamicamente adattativi.

Il terzo drift riguarda il passaggio dalla semplificazione alla sofisticazione. Il BPM tradizionale ha storicamente puntato a rendere i processi più snelli, standardizzati e privi di attività non a valore aggiunto. Tuttavia, concentrarsi esclusivamente sulla semplificazione significa sottoutilizzare le potenzialità delle tecnologie emergenti. La sofisticazione consiste nell'impiego dell'intelligenza artificiale per creare esperienze e output che prima erano semplicemente impossibili, superando le aspettative del cliente invece di limitarsi a ridurre costi e tempi.

2.1.4 Sistemi Conversazionali per il BPM Aumentato dall'AI

Dal punto di vista operativo, in conformità con i drift di paradigma descritti nella sezione precedente, la ricerca sta esplorando concretamente l'integrazione di tecnologie AI nei sistemi BPM tradizionali. In questa direzione, Casciani et al. delineano la visione degli AI-augmented BPM Systems, una nuova generazione di BPMS che espande il ciclo di vita BPM in due direzioni: da un lato potenziando con l'AI le fasi tradizionali di modellazione, analisi, esecuzione e monitoraggio, dall'altro incorporando nuove funzionalità realizzabili esclusivamente grazie a questa tecnologia, quali l'adattamento autonomo, la spiegazione delle decisioni e il miglioramento continuo. La caratteristica centrale di questi sistemi è quella di essere conversationally actionable, ovvero in grado di interagire proattivamente con gli utenti in linguaggio naturale riguardo alle azioni, agli obiettivi e all'avanzamento del processo, superando il carico cognitivo imposto dalle tradizionali interfacce transazionali.
La survey condotta da Casciani et al. identifica quattro aree operative in cui i sistemi conversazionali si integrano con il BPM.
La Descriptive Process Analytics è l'area orientata alla descrizione e comprensione dei processi nella loro configurazione attuale. La Predictive Process Analytics si focalizza invece sulla previsione dell'andamento futuro del processo sulla base dei dati storici. A seguire si ha la Prescriptive Process Optimization, dedicata alla generazione di raccomandazioni operative per migliorarne l'esecuzione. Infine, l'Augmented Process Execution realizza concretamente il drift verso l'autonomizzazione, supportando o sostituendo l'operatore umano nell'esecuzione del processo.
Nonostante il sistema proposto in questo progetto di tesi non rientri nella categoria degli ABPMS, in quanto non interviene proattivamente sull'esecuzione del processo né produce alcuna forma di autonomizzazione, rappresenta un'integrazione concreta di un sistema conversazionale all'interno di un BPMS. L'obiettivo di rendere il processo più accessibile agli utenti finali tramite interazioni in linguaggio naturale fondate su documentazione ufficiale lo riconduce all'area della Descriptive Process Analytics. In particolare, il sistema risponde alle sfide aperte identificate da Casciani et al. relative allo sviluppo di interfacce conversazionali per la comprensione

dei processi da parte di utenti non tecnici. A livello architetturale, il sistema adotta la Retrieval-Augmented Generation (RAG), tecnica ibrida identificata nella survey come particolarmente promettente per ancorare le risposte degli LLM a fonti documentali ufficiali e specialistiche, superando i limiti degli approcci puramente generativi in termini di affidabilità e aderenza al dominio.

2.2 I Large Language Models

Questa sezione presenta i Large Language Models (LLM), illustrandone l'evoluzione storica nel campo del Natural Language Processing e le principali capacità che ne caratterizzano il funzionamento. Vengono inoltre analizzate le abilità emergenti che hanno ampliato le potenzialità applicative di questi modelli, l'estensione al paradigma multimodale e le principali limitazioni e sfide aperte che ne orientano lo sviluppo futuro.

2.2.1 Large Language Models: Background ed Evoluzione

I Large Language Models (LLM) rappresentano una delle innovazioni più rivoluzionarie degli ultimi anni nel campo dell'intelligenza artificiale e in particolare del Natural Language Processing (NLP), con ripercussioni trasversali in molteplici ambiti dal punto di vista tecnologico grazie alle loro straordinarie capacità di comprensione, generazione e utilizzo del linguaggio naturale [2].
Il Language Modeling (LM) è il processo attraverso cui un modello impara a comprendere e generare testo predicendo sequenze di parole. Come illustrato nel survey di Zhao et al., questo campo di studio è caratterizzato da una progressione storica articolata in quattro fasi, ciascuna contraddistinta da un diverso livello di capacità nel risolvere compiti di NLP.
La prima generazione risale agli anni '90, con la nascita degli Statistical Language Models (SLM). Questi modelli si fondano su metodi di apprendimento statistico e costruiscono la predizione delle parole sulla base dell'ipotesi di Markov, secondo la quale la probabilità di uno stato dipende esclusivamente da un numero finito di stati precedenti. Per questa ragione vengono detti modelli n-gram, in quanto considerano solo le ultime n-1 parole per la previsione della parola successiva. I loro limiti principali erano la difficoltà nel catturare dipendenze a lungo raggio e la scarsa capacità di generalizzazione di fronte a combinazioni linguistiche non presenti nei dati di addestramento.
A partire dal 2013, con la diffusione di architetture neurali come le Recurrent Neural Networks (RNN) e le Long Short-Term Memory (LSTM), emerse la seconda generazione: i Neural Language Models (NLM). La loro innovazione più significativa fu l'introduzione delle distributed word representations, rappresentazioni vettoriali continue in cui parole semanticamente simili risultano vicine nello spazio

di rappresentazione. Tuttavia, tali rappresentazioni erano ancora statiche: a ogni parola veniva associato un unico vettore indipendentemente dal contesto in cui appariva, il che impediva di catturare fenomeni come la polisemia e le variazioni semantiche dipendenti dal contesto.
La terza generazione prese forma a partire dal 2018, grazie all'introduzione dell'architettura Transformer e del meccanismo di attenzione [16]. Nacquero così i Pretrained Language Models (PLM), caratterizzati da un nuovo paradigma di addestramento in due fasi: un pretraining su grandi corpora testuali auto-supervisionati, finalizzato all'acquisizione di capacità generali di modellazione del linguaggio, seguito da un fine-tuning su task specifici. La novità più rilevante di questa generazione fu l'introduzione delle context-aware representations: a differenza delle rappresentazioni statiche, il vettore associato a una parola varia in funzione del contesto in cui essa appare, consentendo la gestione della polisemia e delle dipendenze a lungo raggio che limitavano i sistemi precedenti.
L'ultima fase evolutiva, avviata intorno al 2020, è rappresentata dai Large Language Models. La differenza fondamentale rispetto alle generazioni precedenti risiede nella scala di questi modelli, che contano decine se non centinaia di miliardi di parametri. Tale incremento è stato guidato dalle scaling laws [17, 18], che hanno mostrato empiricamente come l'aumento simultaneo di parametri, dati e potenza computazionale produca un miglioramento continuo delle prestazioni. A differenza dei modelli precedenti, pensati per assistere in compiti specifici, gli LLM si configurano come risolutori di compiti di carattere generale, capaci di adattarsi a una gamma vastissima di applicazioni del mondo reale a partire da un singolo modello. È proprio questa versatilità a rappresentare uno degli elementi più dirompenti della rivoluzione introdotta da questa tecnologia.
La popolarità degli LLM è esplosa con il lancio di ChatGPT nel 2022, che ha catalizzato l'attenzione pubblica verso queste tecnologie e stimolato un'ulteriore accelerazione della ricerca [2].

2.2.2 Le Abilità Emergenti degli LLM

L'aumento di scala che caratterizza gli LLM, oltre a produrre un significativo miglioramento delle prestazioni, ha determinato la comparsa improvvisa di alcune capacità del tutto assenti nei modelli di dimensioni più ridotte, note come abilità emergenti [19]. Queste capacità sono fondamentali per consentire agli LLM di affrontare problemi del mondo reale che le generazioni precedenti di modelli non erano in grado di risolvere. Le principali abilità emergenti individuate in letteratura sono tre [2, 19]: in-context learning, instruction following e step-by-step reasoning.
L'in-context learning (ICL) è la capacità di un LLM di riconoscere e risolvere un nuovo compito ricevendo esclusivamente una descrizione del task e/o alcuni esempi dimostrativi all'interno del prompt, ovvero il testo fornito in input al modello per

guidarne la generazione. A differenza del fine-tuning, l'ICL non richiede alcun aggiornamento dei pesi del modello: l'apprendimento avviene interamente durante la fase di inferenza, sulla base del contesto disponibile. Un prompt tipico per l'ICL include una descrizione del compito da svolgere e una serie di esempi composti da coppie input-output, modalità nota come few-shot prompting. L'ICL può operare secondo due meccanismi distinti: il modello riconosce il compito dagli esempi e attinge alle conoscenze acquisite durante il pre-addestramento per risolverlo (task recognition), oppure apprende compiti nuovi o mai visti basandosi esclusivamente sulle informazioni fornite nelle dimostrazioni (task learning).
L'instruction following è la capacità di un LLM di eseguire correttamente compiti mai visti in precedenza basandosi esclusivamente su istruzioni in linguaggio naturale, senza necessità di esempi dimostrativi espliciti. Rispetto all'ICL che spesso richiede esempi dimostrativi, questa abilità consente una generalizzazione più diretta e flessibile a nuovi task, migliorando significativamente la versatilità del modello. Il modello riesce a interpretare l'obiettivo descritto nell'istruzione e a conformare il proprio output ai requisiti espressi.
Infine, lo step-by-step reasoning è la capacità che consente agli LLM di affrontare compiti complessi scomponendoli in una sequenza di passaggi logici intermedi prima di giungere alla risposta finale. Mentre i modelli di scala ridotta faticano nella risoluzione di problemi che richiedono molteplici fasi di ragionamento o calcolo, i modelli di grandi dimensioni riescono a costruire un percorso articolato verso la soluzione. Questa abilità viene tipicamente attivata attraverso il Chain-of-Thought prompting, una tecnica che integra nel prompt una serie di passaggi intermedi esemplificativi che guidano il modello nel ragionamento.
In conclusione, le abilità emergenti non rappresentano un semplice miglioramento quantitativo delle prestazioni, ma una trasformazione qualitativa che ha ridefinito il ruolo dei modelli linguistici nell'intelligenza artificiale, abilitando capacità di ragionamento e generalizzazione che le generazioni precedenti non potevano nemmeno approssimare.

2.2.3 I Multimodal Large Language Models

Gli LLM, grazie all'aumento di scala e all'emergere di capacità non esplicitamente supervisionate durante l'addestramento, hanno raggiunto prestazioni eccellenti in numerosi task di Natural Language Processing. Tuttavia, come analizzato nella survey di Yin et al., tali modelli operano esclusivamente su dati testuali e non dispongono di alcuna capacità percettiva diretta del contenuto visivo. Questa limitazione impediva di sfruttare pienamente le loro avanzate capacità di ragionamento, apprendimento e generalizzazione in contesti che richiedevano l'integrazione di informazioni provenienti da modalità diverse dal testo. Parallelamente, i Large Vision Models (LVM) hanno mostrato notevoli capacità nell'elaborazione e comprensione

di contenuti visivi, ma si sono rivelati generalmente meno efficaci nei compiti di ragionamento astratto e instruction-following rispetto agli LLM di larga scala. Questa complementarità funzionale ha favorito la nascita dei Multimodal Large Language Models (MLLM), una nuova categoria di modelli che integra competenze linguistiche e percettive all'interno di un unico sistema, come presentato nella survey di Yin et al. Con il progressivo sviluppo del settore, tali sistemi hanno esteso le proprie capacità oltre la modalità visiva, includendo anche audio e video.
Dal punto di vista architetturale, gli MLLM si compongono tipicamente di tre macro-componenti principali [20]: un encoder di modalità, un LLM pre-addestrato e un connettore di modalità che funge da interfaccia tra i due.
L'encoder di modalità costituisce il modulo deputato all'acquisizione e all'elaborazione dei segnali grezzi in ingresso (immagini, video o audio). Il suo compito è trasformare tali segnali in rappresentazioni vettoriali compatte, capaci di preservarne il contenuto semantico e le caratteristiche rilevanti. Questo processo di estrazione delle feature consente di mappare dati eterogenei in uno spazio numerico strutturato, adatto alle successive fasi di elaborazione.
Il connettore di modalità svolge invece la funzione di meccanismo di proiezione e allineamento tra lo spazio di rappresentazione prodotto dall'encoder e quello comprensibile dall'LLM. Poiché gli LLM operano nativamente su sequenze testuali, il connettore ha il compito di ridurre il divario tra linguaggio naturale e le altre modalità, traducendo o proiettando le rappresentazioni multimodali in una forma compatibile con il modello linguistico. In questo modo, segnali provenienti da diverse modalità vengono allineati in uno spazio condiviso che consente un'elaborazione integrata ed efficiente.
Infine, l'LLM rappresenta il nucleo cognitivo del sistema ed è responsabile delle operazioni di comprensione, integrazione e ragionamento sulle informazioni precedentemente elaborate e allineate. Grazie al massiccio pre-addestramento su corpora di larga scala e alle proprietà emergenti osservate con l'aumento della dimensione del modello, l'LLM mette a disposizione dell'intero sistema multimodale una ricca conoscenza del mondo e solide capacità di generalizzazione.
Analogamente a quanto osservato per gli LLM puramente testuali, anche gli MLLM hanno mostrato comportamenti non esplicitamente programmati durante l'addestramento [20]. Tali capacità emergenti sono in larga parte attribuibili alla scala del modello linguistico integrato e a specifiche strategie di addestramento, tra cui il multimodal instruction tuning [21], che incentiva il sistema a seguire istruzioni eterogenee e a generalizzare il proprio comportamento a scenari multimodali non visti in precedenza. Tra gli esempi di abilità osservate figurano la generazione di codice per la creazione di pagine web a partire dall'interpretazione del layout visivo di un'immagine, la comprensione semantica di meme complessi e lo svolgimento di compiti di ragionamento matematico direttamente su contenuti visivi, senza ricorrere esplicitamente a pipeline OCR tradizionali.

2.2.4 Limitazioni e Direzioni Future

Nonostante le loro straordinarie capacità, gli LLM presentano ancora oggi diverse limitazioni e sfide aperte che la ricerca scientifica sta cercando di affrontare, come ampiamente discusso nella survey di Naveed et al..
Dal punto di vista tecnico, una delle problematiche più rilevanti è la tendenza a generare allucinazioni, ossia contenuti plausibili e convincenti ma fattualmente errati o privi di fondamento. A ciò si aggiungono limitazioni nel ragionamento logico e nella pianificazione, dovute al fatto che gli LLM non applicano un vero e proprio ragionamento come gli esseri umani, ma si limitano a completare testo in modo statisticamente plausibile. Gli LLM faticano inoltre a mantenere la coerenza in conversazioni lunghe o nell'analisi di documenti estesi, tendendo a dimenticare le informazioni fornite in precedenza. Infine, l'overfitting rappresenta un'ulteriore criticità tecnica, in quanto la tendenza a memorizzare pattern specifici del training set può compromettere la capacità del modello di generalizzare efficacemente di fronte a situazioni inedite.
Sul fronte delle risorse e dei costi, sia l'addestramento che l'inferenza richiedono risorse computazionali ingenti, con investimenti economici molto significativi. A questo si associa un elevato consumo energetico, con ricadute ambientali non trascurabili in termini di emissioni di CO_2. L'enorme numero di parametri di questi modelli li rende inoltre inadatti all'esecuzione in tempo reale su dispositivi con risorse limitate, come smartphone o sistemi embedded, ostacolandone l'impiego in applicazioni che richiedono risposte immediate o operano in ambienti con connettività ridotta.
In termini di sicurezza e affidabilità, gli LLM risultano vulnerabili a diverse tipologie di attacchi informartici, che possono aggirarne i meccanismi di protezione. A ciò si sommano rischi per la privacy, derivanti dalla tendenza dei modelli a memorizzare dati sensibili presenti nel training set, potenzialmente estraibili da utenti malintenzionati.
Sul piano etico e sociale, gli LLM tendono a ereditare e amplificare i bias presenti nei dati di addestramento, con il rischio di produrre risposte discriminatorie verso determinati gruppi sociali, etnici o di genere. L'elevato costo di sviluppo concentra inoltre la ricerca nelle mani di poche grandi organizzazioni. La scarsa rappresentazione delle lingue diverse dall'inglese nei dataset penalizza infine strutturalmente gli utenti non anglofoni, limitando l'accessibilità globale di questi sistemi.
Tra le ulteriori limitazioni, la conoscenza degli LLM è cristallizzata al momento del training e tende inevitabilmente all'obsolescenza. Tecniche come la Retrieval-Augmented Generation (RAG) [23] mitigano parzialmente questo problema, ma introducono una maggiore complessità architetturale. La natura black-box di questi modelli rende infine difficile comprendere e spiegare il processo decisionale sottostante, limitandone l'adozione in contesti ad alta responsabilità dove la trasparenza

è imprescindibile.
Per affrontare queste sfide, la ricerca scientifica sta esplorando diverse direzioni future. Sul fronte dell'efficienza, si lavora allo sviluppo di architetture più leggere e tecniche per ridurre costi computazionali e impatto ambientale. Sul fronte della sicurezza, si investe nella progettazione di meccanismi di difesa più robusti. Ulteriori direzioni riguardano soluzioni per consentire ai modelli di aggiornarsi nel tempo senza perdere le conoscenze pregresse, il miglioramento dell'interpretabilità per rendere i modelli più trasparenti e adottabili in contesti critici, e lo sviluppo di dataset multilingue più equilibrati. Infine, la definizione di quadri normativi ed etici condivisi appare sempre più necessaria per garantire uno sviluppo responsabile e affidabile di questi sistemi.

2.3 Gli Agenti AI

Questa sezione introduce il concetto di agenti AI, sistemi intelligenti progettati per percepire l'ambiente, ragionare e intraprendere azioni al fine di raggiungere obiettivi specifici. Dopo aver illustrato il ruolo dei Large Language Models come nucleo cognitivo degli agenti moderni, vengono analizzate la struttura funzionale degli agenti LLM-based e le principali architetture agentiche proposte in letteratura, con particolare attenzione ai modelli ad agente singolo e multi-agente.

2.3.1 Dagli LLM agli Agenti AI

Le grandi potenzialità dimostrate dagli LLM, hanno portato alla nascita di un ambito di ricerca estremamente promettente, orientato alla realizzazione di sistemi intelligenti in grado di gestire problemi complessi: gli agenti AI.
Come analizzato da Xi et al., un agente AI può essere definito come un'entità artificiale progettata per percepire l'ambiente, prendere decisioni e intraprendere azioni per raggiungere obiettivi specifici. In questo contesto, gli LLM sono oggi considerati il nucleo cognitivo privilegiato per la costruzione di agenti AI, in quanto rappresentano modelli versatili e generalisti, in grado di integrare comprensione, ragionamento e generazione del linguaggio naturale all'interno di un'unica architettura.
Tale centralità è riconducibile in larga misura alle abilità emergenti che caratterizzano questi modelli. In particolare, l'instruction following permette di comprendere ed eseguire compiti inediti formulati in linguaggio naturale. Lo step-by-step reasoning consente di scomporre problemi complessi in sequenze strutturate di passaggi logici e azioni intermedie. Infine, l'in-context learning abilita un adattamento dinamico al contesto, consentendo al modello di modificare il proprio comportamento sulla base di esempi o feedback immediati, senza necessità di un nuovo addestramento. Queste capacità, integrate con le avanzate abilità di comprensione e generazione

del linguaggio naturale e con l'elevata capacità di generalizzazione cross-task, costituiscono la condizione cognitiva abilitante per l'implementazione delle proprietà tipicamente associate agli agenti AI. Secondo Xi et al., tali proprietà includono autonomia, reattività, proattività e abilità sociale. In primo luogo, gli agenti AI presentano un certo grado di autonomia operativa, intesa come capacità di eseguire azioni e perseguire obiettivi senza un intervento umano continuo, pur in assenza di intenzionalità intrinseca. In secondo luogo, essi sono caratterizzati da reattività, ossia dalla capacità di rispondere ai cambiamenti e agli stimoli provenienti dall'ambiente. A ciò si affianca la proattività, che implica la possibilità di adottare comportamenti orientati a uno scopo, pianificando azioni e prendendo iniziative funzionali al raggiungimento degli obiettivi prefissati. Infine, l'abilità sociale si riferisce alla capacità dell'agente di interagire con altri sistemi basati su LLM o con esseri umani attraverso il linguaggio naturale, abilitando dinamiche di cooperazione, coordinamento o competizione.
È tuttavia necessario sottolineare che un LLM, considerato isolatamente, si limita a trasformare un input testuale in un output testuale sulla base delle conoscenze apprese durante l'addestramento. In questa configurazione, il modello non dispone di una percezione diretta dell'ambiente, né di capacità autonome di azione o di gestione persistente dello stato. Un agente AI, al contrario, emerge quando il modello linguistico viene integrato all'interno di un'architettura più ampia, composta da componenti aggiuntivi che ne estendono significativamente le funzionalità, superando le limitazioni di un LLM puramente generativo.

2.3.2 Struttura Funzionale degli Agenti LLM-based

Nello studio condotto da Xi et al. viene proposto un quadro concettuale degli agenti LLM-based articolato in tre moduli principali: percezione, cervello e azione. Insieme, questi tre componenti delineano le capacità fondamentali che abilitano il comportamento agentico all'interno dei sistemi basati su LLM, operando in sequenza secondo un ciclo continuo di percezione, elaborazione e risposta.
Il modulo di percezione costituisce l'apparato sensoriale del sistema che consente all'agente di ricevere dati provenienti dall'ambiente e di trasformarli in rappresentazioni interpretabili dal modulo cognitivo. Questo componente estende lo spazio percettivo dell'agente oltre il solo testo, abilitandolo a elaborare input multimodali. Sul piano pratico, il modulo può gestire tre tipologie di dati: le informazioni testuali, composte da istruzioni in linguaggio naturale, le informazioni visive, elaborate tramite image captioning o fusione modale tra encoder visivo e modello linguistico, e infine le informazioni sonore, gestite convertendole in spettrogrammi bidimensionali trattabili come immagini. In prospettiva futura, il modulo potrebbe integrarsi con ulteriori sensori fisici, come GPS e Lidar, per una percezione ancora più ricca dell'ambiente circostante. Una volta raccolte e convertite, tutte queste

informazioni vengono trasmesse al cervello del sistema, dove alimentano i processi di ragionamento e pianificazione che guidano le azioni dell'agente.
Il cervello rappresenta il nucleo cognitivo dell'agente che si occupa di coordinare le sue funzioni principali di ragionamento, pianificazione e guida delle azioni. In seguito alla ricezione di un input da parte del modulo di percezione, il cervello integra le informazioni dalla memoria, si impegna in attività di ragionamento e pianificazione, determina l'azione da intraprendere e contatta il modulo d'azione per l'esecuzione effettiva. Questo processo è iterativo e permette all'agente di ricevere feedback e interagire dinamicamente con l'ambiente.
Il componente centrale di questo modulo è l'LLM, che fornisce all'agente la conoscenza necessaria per poter prendere decisioni informate. Essa deriva dal processo di addestramento del modello e può essere distinta in due tipologie: conoscenza comune e conoscenza di dominio. La conoscenza comune è costituita da fatti e informazioni generali che compongono l'intelligenza di base del modello e conferiscono a quest'ultimo le capacità di ragionamento a livello generale. La conoscenza di dominio invece è composta da informazioni specifiche di determinati settori, cruciali affinché l'agente possa operare in determinati campi applicativi.
Un altro elemento fondamentale del cervello dell'agente è la memoria, che permette la registrazione e l'archiviazione di osservazioni, azioni e interazioni utili allo svolgimento delle proprie attività. L'agente può avere a disposizione due tipologie di memoria: a breve termine e a lungo termine. La memoria a breve termine permette all'agente di mantenere il contesto corrente, storicizzando le informazioni recenti per il processo decisionale. Essa è intrinsecamente vincolata dalla finestra di contesto dell'LLM sottostante. Per quanto riguarda invece la memoria a lungo termine, essa consente di memorizzare in maniera persistente esperienze passate e conoscenze riutilizzabili per compiti futuri.
Infine, il modulo di azione è responsabile della traduzione delle decisioni prodotte dal motore di ragionamento e pianificazione dell'agente in operazioni concrete. Questa capacità è resa possibile attraverso l'uso di strumenti esterni, che ampliano significativamente le funzionalità dell'agente. Tali strumenti possono includere sistemi di information retrieval, motori di ricerca, interpreti di codice, API applicative, fino ad arrivare a dispositivi robotici attraverso i quali è possibile eseguire azioni nel mondo reale. Quando il processo di ragionamento determina un corso d'azione, il modulo di azione seleziona e invoca lo strumento appropriato per realizzare l'operazione. In questo modo, l'agente supera le limitazioni intrinseche di un LLM puro, che di per sé produce soltanto testo. Un esempio emblematico è l'esecuzione di operazioni matematiche: un LLM tradizionale genera semplicemente l'output statisticamente più probabile per un'espressione come $2 + 2$, senza garantire correttezza computazionale. Un agente, invece, riconosce attraverso il proprio processo di ragionamento la necessità di utilizzare uno strumento adatto, come una calcolatrice, invoca tale strumento ed elabora il risultato effettivo, restituendolo poi al cervello.

Questo approccio consente all'agente di ottenere output affidabili anche in compiti in cui un LLM puro sarebbe inaffidabile o impreciso.

2.3.3 Architetture Agentiche e Design Patterns

Dal punto di vista realizzativo, come presentato all'interno della survey di Masterman et al., gli agenti AI si possono distinguere in due grandi categorie architetturali: le architetture ad agente singolo (Single Agent Architectures) e le architetture multi-agente (Multi-Agent Architectures).

Architetture ad Agente Singolo

Le architetture ad agente singolo sono caratterizzate dalla presenza di un unico nucleo cognitivo, responsabile di ragionamento, pianificazione e gestione degli strumenti necessari al completamento dei compiti impartiti. In questo genere di soluzione, all'agente viene assegnato un ruolo specifico tramite un system prompt, ossia una descrizione testuale che rende il sistema consapevole del suo scopo, del comportamento da adottare e di come utilizzare efficacemente le funzioni a sua disposizione.
Questo tipo di architettura è particolarmente adatto per problemi semplici e ben definiti, risolvibili mediante un insieme limitato e circoscritto di strumenti, senza la necessità di feedback da parte di altri agenti o di intervento umano. Il principale vantaggio di questa soluzione è la sua semplicità di implementazione. Tuttavia, il successo dell'agente dipende dalle capacità del nucleo cognitivo di pianificazione, autovalutazione e gestione del contesto, in assenza delle quali il sistema rischia di eseguire operazioni ripetitive o di rimanere in loop senza completare il compito.
In letteratura sono stati proposti diversi design patterns per la realizzazione di architetture ad agente singolo. Il pattern ReAct [25] integra le capacità di ragionamento e azione dell'agente tramite un ciclo iterativo composto da tre fasi principali. Per cominciare l'agente genera una traccia di ragionamento testuale utile a scomporre il problema e mantenere traccia dei progressi. Dopodichè, sulla base di questo ragionamento, esegue un'azione concreta attraverso gli strumenti disponibili e infine riceve un feedback che viene aggiunto al contesto per guidare il passo esecutivo successivo. RAISE [26] estende questo framework introducendo un sistema di doppia memoria ispirato alla cognizione umana. La prima componente, detta Scratchpad, opera come memoria a breve termine registrando ragionamenti e osservazioni nel corso dell'esecuzione. La seconda, detta Examples, costituisce invece una memoria a lungo termine che archivia coppie query-risposta recuperabili in base alla pertinenza con il contesto corrente. Il pattern Reflexion [27] adotta un approccio differente, basato sul Verbal Reinforcement Learning. Un modulo di auto-riflessione valuta la qualità delle azioni prodotte dall'agente e genera un

feedback linguistico mirato che viene reintegrato nel contesto, consentendo al sistema di apprendere dai propri errori senza necessità di aggiornare i parametri del modello.
Nonostante l'efficacia di questi pattern, le architetture ad agente singolo mostrano limiti quando applicate a problemi particolarmente complessi o che richiedono competenze eterogenee. Un singolo agente potrebbe affrontare compiti articolati, ma la sua capacità di scomporre problemi, pianificare e raffinare le azioni può risultare insufficiente, portando a inefficienze o comportamenti subottimali come, ad esempio, ripetizioni o loop improduttivi. Questo genere di soluzioni risulta inoltre meno adatta quando è utile introdurre prospettive diverse, specializzazioni distinte o una divisione strutturata del lavoro.

Architetture Multi Agente

Per affrontare alcune delle limitazioni pratiche degli agenti singoli nei sistemi LLM-based, è stata introdotta l'architettura Multi-Agent. In questo tipo di sistemi, due o più agenti LLM-based interagiscono attivamente per la risoluzione di un compito complesso. Ciascun agente può essere basato sullo stesso LLM o su modelli differenti, e può avere accesso a strumenti condivisi o a strumenti specifici. Tipicamente, ogni agente è configurato con un proprio system prompt, che definisce il ruolo e il comportamento da assumere, oltre a descrivere i compiti assegnati e gli strumenti a disposizione. Questa struttura permette di distribuire il lavoro in maniera intelligente e specialistica, migliorare la parallelizzazione dei compiti e aumentare la robustezza e l'efficacia complessiva del sistema.
Nello studio presentato da Masterman et al. viene introdotta un'ulteriore categorizzazione interna ai sistemi Multi-Agent basata sulla struttura di interazione tra agenti: architetture verticali e architetture orizzontali. Le architetture verticali sono caratterizzate dalla presenza di un agente leader che coordina un insieme di agenti subordinati, ciascuno dei quali svolge compiti specifici seguendo le direttive del leader, che mantiene una visione d'insieme del sistema. Questo tipo di architettura è particolarmente adatta a compiti in cui è necessario un chiaro isolamento delle responsabilità, ad esempio nella gestione e chiamata degli strumenti a disposizione degli agenti. Al contrario, nelle architetture orizzontali tutti gli agenti sono considerati pari e collaborano tra loro, scambiandosi informazioni e coordinando le proprie azioni in modo distribuito. Questo approccio è generalmente preferito per compiti in cui la collaborazione, il feedback e la discussione di gruppo sono fondamentali per il successo complessivo, sebbene richieda protocolli di comunicazione più sofisticati e meccanismi di consenso per prevenire conflitti o situazioni di stallo.
Nonostante i vantaggi offerti, le architetture Multi-Agent introducono una serie di sfide che è opportuno considerare. Sul piano implementativo, questi sistemi sono intrinsecamente più complessi da realizzare, richiedendo una gestione robusta

delle conversazioni e, spesso, una struttura di coordinamento ben definita. Dal punto di vista comportamentale, gli agenti possono perdersi in scambi di messaggi superflui, distraendo il sistema dall'obiettivo principale e riducendone l'efficienza. Un rischio ulteriore è rappresentato dalla tendenza degli agenti a farsi influenzare dai feedback degli altri membri del gruppo anche quando questi sono errati, con il rischio di produrre piani d'azione scorretti. Si aggiungono poi problemi legati alla condivisione delle informazioni: nelle architetture verticali il leader potrebbe omettere dettagli critici ai subordinati, mentre in quelle orizzontali l'eccesso di informazioni condivise può generare confusione. Infine, in assenza di un filtraggio intelligente dei messaggi, il costo in termini di risorse e tempo per gestire gli scambi tra agenti può diventare significativo.

2.4 La Retrieval-Augmented Generation

Questa sezione presenta il paradigma della Retrieval-Augmented Generation (RAG), un'architettura progettata per estendere le capacità dei Large Language Models integrando fonti di conoscenza esterne durante il processo di generazione delle risposte. Dopo aver discusso le limitazioni degli LLM che hanno motivato lo sviluppo di questo approccio, vengono analizzati l'architettura e il funzionamento dei sistemi RAG, le principali criticità ancora aperte e le più recenti evoluzioni verso soluzioni multimodali.

2.4.1 Dalle Limitazioni degli LLM alla Retrieval-Augmented Generation

Gli LLM hanno dimostrato grandi potenzialità grazie alla loro capacità di immagazzinare una notevole quantità di conoscenza direttamente nei loro parametri, attraverso il processo di addestramento. Tuttavia, questo approccio presenta limitazioni rilevanti [4]. In primo luogo, questi modelli non possono espandere facilmente la propria conoscenza interna che risulta limitata al momento del loro addestramento, infatti qualsiasi aggiornamento richiede un riaddestramento o un fine-tuning con nuovi dati, un processo estremamente costoso. A ciò si aggiunge la difficoltà di tracciare la provenienza delle informazioni generate, rendendo complessa la verifica della correttezza delle affermazioni prodotte. Gli LLM sono inoltre soggetti al fenomeno delle allucinazioni, ovvero la tendenza a generare contenuti plausibili ma fattualmente errati. Infine, questi modelli generici spesso non dispongono di conoscenze specialistiche legate a particolari domini di interesse, e creare modelli settoriali richiede ingenti risorse sia per quanto riguarda la raccolta dati sia per l'addestramento.
Per superare tali limitazioni, Lewis et al. hanno introdotto la Retrieval-Augmented

Generation (RAG), un'architettura avanzata progettata per migliorare le capacità degli LLM collegandoli a basi di conoscenza esterne. Attraverso la RAG è possibile fornire al modello informazioni pertinenti e fattuali recuperate da un datastore esterno al momento della generazione, mitigando il fenomeno delle allucinazioni e garantendo risposte fondate su materiale informativo verificabile. Questo approccio consente inoltre di mantenere la conoscenza del sistema costantemente aggiornata semplicemente intervenendo sul database esterno, senza necessità di riaddestramento del modello. Infine, costruendo una base di conoscenza settoriale mirata, è possibile trasformare un LLM generico in uno strumento specializzato per uno specifico dominio applicativo, con costi notevolmente inferiori rispetto allo sviluppo di un modello dedicato.

2.4.2 Archiettura e Funzionamento RAG

Nello studio condotto da Wu et al. vengono presentati i moduli fondamentali che compongono un sistema RAG: il modulo di recupero, il modulo di fusione e il modulo di generazione.

Modulo di Retrieval

Il retriever è il modulo responsabile della ricerca e del recupero delle informazioni più pertinenti e rilevanti da una base di conoscenza esterna rispetto alle richieste presentate dall'utente al sistema. La sua efficacia è determinante per la qualità complessiva del sistema, e il suo funzionamento si articola in due fasi principali: la costruzione e l'interrogazione.
La fase di costruzione del retriever ha lo scopo di creare la knowledge base vettoriale dalla quale verranno recuperate le informazioni rilevanti da impiegare durante la generazione. Questo processo, noto come ingestion pipeline, si compone di diversi passaggi eseguiti in maniera sequenziale. La procedura comincia con il pre-processamento del materiale informativo, solitamente documenti testuali, con l'obiettivo di migliorarne la qualità e semplificare le operazioni successive. Segue la fase di chunking, in cui i documenti vengono suddivisi in frammenti di dimensione più contenuta, detti chunk. Il chunking può essere effettuato secondo diverse modalità operative. La più semplice prevede una suddivisione sequenziale basata su una dimensione predefinita, veloce da eseguire ma rischiosa dal punto di vista della coerenza semantica, poiché può troncare frasi o concetti. In alternativa, è possibile suddividere il testo in corrispondenza di marcatori semantici come punti o interruzioni di riga, preservando così l'integrità delle frasi e dei paragrafi. Infine, la strategia più sofisticata segmenta i documenti seguendone la struttura e il contenuto, preservandone l'unità concettuale e producendo segmenti tematicamente coesi. Tuttavia, questa soluzione richiede una conoscenza preventiva dell'organizzazione

del testo e risulta meno applicabile a documenti non strutturati.
A valle del chunking si colloca la fase di encoding, in cui ciascun chunk viene trasformato in una rappresentazione vettoriale numerica, detta embedding, che ne cattura il significato semantico. Esistono due principali metodi di codifica che differiscono per la densità dei vettori prodotti: sparse encoding e dense encoding. La tecnica di codifica sparsa crea vettori ad alta dimensionalità con molti valori nulli. È una tecnica computazionalmente efficiente, ma in grado di catturare una semantica meno profonda. La codifica densa, al contrario, genera vettori con valori prevalentemente non nulli mediante reti neurali profonde, catturando informazioni linguistiche e semantiche più ricche. È importante sottolineare che documenti eccessivamente lunghi rispetto alla dimensionalità dei vettori prodotti possono penalizzare le performance del sistema: il vettore di embedding, infatti, potrebbe non disporre di capacità rappresentativa sufficiente a codificare con precisione tutto il contenuto semantico.
Una volta ottenuti gli embedding, si procede con la fase di indexing, che consiste nell'organizzare i vettori all'interno di un database vettoriale secondo algoritmi di indicizzazione appositamente progettati per ottimizzare le procedure di ricerca. Questo passaggio conclude la pipeline di costruzione della fonte di conoscenza alla base del sistema di retrieval. Il database vettoriale viene quindi memorizzato in maniera persistente come una raccolta di coppie chiave-valore, dove le chiavi sono gli identificatori unici degli embeddings e i valori corrispondono ai dati originali.
Una volta costruita questa fonte di conoscenza esterna, essa può essere interrogata per individuare ed estrarre le informazioni pertinenti rispetto alle query ricevute dal sistema. Questo processo, noto come retrieval pipeline, si articola anch'esso in più fasi. Il meccanismo comincia con la query in ingresso che viene trasformata in un embedding vettoriale, utilizzando lo stesso modello di codifica impiegato per i chunks durante l'ingestion. L'embedding così ottenuto viene confrontato con quelli presenti nel database attraverso una metrica di similarità, come la cosine similarity o la distanza euclidea, al fine di recuperare i k vettori più simili e i relativi segmenti documentali. A questo punto, la pipeline può prevedere ulteriori passaggi opzionali di filtraggio e reranking. Il filtraggio consente di restringere l'insieme dei risultati recuperati selezionando quelli più pertinenti, mentre il reranking permette di riordinare i risultati rimanenti secondo metriche di rilevanza più sofisticate rispetto alla sola similarità vettoriale. Queste operazioni contribuiscono a migliorare la qualità delle informazioni trasmesse al modulo di generazione.

Modulo di Fusione

Come riportato da Wu et al., il modulo di fusione all'interno di un'architettura RAG ha lo scopo di definire le modalità con cui le informazioni recuperate dal database esterno tramite il meccanismo di retrieval vengono integrate nel processo

di generazione della risposta.
L'approccio più diretto è la fusione basata sulla query (query-based fusion), che può essere realizzata secondo due modalità operative. La prima consiste nel concatenare i contenuti recuperati dal modulo di retrieval direttamente alla query dell'utente, formando un unico prompt da fornire al modello generativo. Questa tecnica è particolarmente adatta per modelli black-box accessibili solo tramite API. La seconda modalità opera invece a livello vettoriale, concatenando gli embedding delle informazioni recuperate con quelli della query per costruire una rappresentazione unificata da utilizzare come input per il generatore.
Oltre alla fusione a livello di query, esistono tecniche più avanzate che intervengono direttamente durante il processo di inferenza del modello generativo. La logits-based fusion agisce nella fase finale di output del modello, intervenendo sui valori grezzi prodotti dall'ultimo strato del generatore, detti logit, prima che questi vengano trasformati in una distribuzione di probabilità. Le informazioni recuperate vengono incorporate in questa fase modificando tali valori, influenzando così direttamente la selezione dei token generati. La latent fusion, invece, integra le informazioni estratte direttamente negli stati nascosti del generatore, intervenendo quindi in una fase più profonda del processo di inferenza.

Modulo di Generazione

Il modulo di generazione all'interno di un'architettura RAG ha lo scopo fondamentale di produrre la risposta finale o effettuare predizioni basandosi sull'input dell'utente e sulle informazioni rilevanti estratte tramite il meccanismo di retrieval. Nella maggior parte dei casi il modello generativo adottato è un LLM, che riceve le informazioni recuperate attraverso la tecnica di query-based fusion. Tuttavia, come riportato da Wu et al., esistono anche modelli appositamente progettati per supportare tecniche di fusione più profonde, come la logits-based fusion o la latent fusion, che richiedono accesso diretto agli stati interni del modello.

2.4.3 Limitazioni e Direzioni Future della RAG Tradizionale

Nonostante la Retrieval-Augmented Generation sia nata per risolvere i limiti intrinsechi degli LLM potenziandone accuratezza, conoscenza e affidabilità, essa, come descritto da Wu et al., presenta a sua volta diverse limitazioni tecniche e sfide operative che ne condizionano l'efficacia.
Una prima limitazione riguarda la qualità del meccanismo di retrieval, poiché la qualità delle risposte finali generate dal sistema dipende strettamente dalla pertinenza e dalla rilevanza dei contenuti recuperati dalla knowledge base esterna. Nel caso in cui i contenuti recuperati risultino contraddittori o privi delle informazioni

necessarie per la produzione della risposta, il sistema può comunque generare allucinazioni o risposte incoerenti. Influisce inoltre sulla qualità del retrieval il trade-off tra le dimensioni di chunks e vettori embedding, necessario garantire la rappresentatività di quest'ultimi. Stabilire il compromesso ottimale tra questi due elementi è un processo spesso soggettivo che richiede sforzi manuali considerevoli. A ciò si aggiunge il fatto che gli algoritmi di indicizzazione e ricerca applicati al database vettoriale sacrificano frequentemente la precisione del recupero in favore di una riduzione dei tempi di calcolo.
Le tecniche di fusione utilizzate per integrare i dati recuperati nel processo generativo presentano ciascuna limitazioni intrinseche. Il metodo più diffuso, la query-based fusion, comporta un aumento significativo della lunghezza dell'input a causa della concatenazione dei contenuti tramite prompt, determinando elevati costi computazionali e complicazioni legate al rispetto della context window dell'LLM, che può portare al troncamento delle informazioni. La fusione logits-based invece integra le informazioni recuperate solo al termine della fase di inferenza, il che può risultare insufficiente per compiti di ragionamento complesso che richiedono un'interazione profonda tra query e dati. Infine, la latent fusion agisce sugli stati nascosti del modello generativo, rendendo difficile comprendere come le informazioni vengano effettivamente utilizzate nel processo di generazione. Questo approccio richiede inoltre modifiche strutturali profonde all'LLM e un addestramento specifico, aumentando significativamente la complessità implementativa del sistema.
Le limitazioni descritte rappresentano le principali sfide aperte nella ricerca sulla RAG e definiscono le direzioni lungo cui il campo si sta attivamente evolvendo, con l'obiettivo di migliorare la qualità del retrieval, ridurre i costi computazionali, rendere le tecniche di fusione più interpretabili ed efficienti e ottimizzare congiuntamente tutti i componenti del sistema. Una delle direzioni più promettenti è inoltre rappresentata dalla RAG multimodale, che mira a superare i limiti delle rappresentazioni puramente testuali integrando immagini, audio e video per arricchire la qualità del recupero e creare interazioni più naturali con l'utente.

2.4.4 La Multimodal Retrieval-Augmented Generation

L'MRAG (Multimodal Retrieval-Augmented Generation) viene presentata all'interno della survey di Mei et al. come un'evoluzione significativa del sistema RAG tradizionale, progettata per integrare dati multimodali come testo, immagini, audio e video sia nel processo di recupero informativo che in quello di generazione delle risposte. Mentre i sistemi RAG classici si affidano quasi esclusivamente a dati testuali, l'MRAG ne estende le capacità sfruttando le ricche informazioni contestuali disponibili in altri formati, consentendo di generare risposte più complete oltre che fattualmente fondate. A differenza della RAG classica, questi sistemi non si limitano a recuperare contenuti testuali simili alla query dell'utente, ma

individuano e integrano conoscenze pertinenti provenienti da modalità eterogenee, gestendo attivamente le relazioni tra di esse. La generazione delle risposte è affidata a modelli linguistici multimodali di grandi dimensioni, detti MLLM, capaci di elaborare e sintetizzare informazioni derivanti da più tipologie di dati.
Mei et al. categorizzano lo sviluppo dell'MRAG in tre fasi evolutive distinte.

MRAG 1.0

L'MRAG 1.0, definito "pseudo-MRAG", rappresenta la fase iniziale dello sviluppo dei sistemi di generazione aumentata dal recupero multimodale. Questa architettura è un'estensione diretta del paradigma RAG tradizionale, progettata per supportare dati multimodali pur mantenendo una struttura operativa basata quasi esclusivamente sul testo. La caratteristica distintiva di questa soluzione è che i dati non testuali (immagini, video, tabelle) vengono convertiti in didascalie testuali (captions) tramite modelli specializzati che ne descrivono il contenuto. Sia i frammenti di testo originali che le didascalie generate vengono trasformati in embeddings e memorizzati all'interno di un database vettoriale. Il modulo di recupero elabora le query dell'utente, in questa versione limitate al solo formato testuale, estraendo dalla knowledge base i frammenti e le didascalie più rilevanti. Nella fase di fusione, query e conoscenza recuperata vengono sintetizzate in un unico prompt, sulla base del quale un LLM tradizionale genererà una risposta esclusivamente testuale.
Nonostante il successo iniziale, questa architettura presenta limiti significativi che hanno portato allo sviluppo delle versioni successive. La conversione da immagine a testo comporta una perdita inevitabile di particolari visivi, che una didascalia non è in grado di catturare fedelmente. La necessità di impiegare modelli distinti per ciascuna modalità aumenta inoltre la complessità del sistema e i costi computazionali. A ciò si aggiunge il fatto che la frammentazione dei concetti durante il chunking e la dipendenza dalla qualità delle didascalie limitano la precisione del recupero, con impatti significativi sulla qualità delle risposte generate dal sistema.

MRAG 2.0

Mentre la prima generazione convertiva forzatamente contenuti multimodali in equivalenti testuali, l'MRAG 2.0 integra le capacità degli MLLM per gestire i dati nella loro forma originale. In fase di costruzione della knowledge base, invece di ricorrere a modelli specializzati distinti per ciascuna modalità, viene impiegato un unico MLLM unificato per estrarre descrizioni e metadati dai documenti. A differenza dell'MRAG 1.0, i dati multimodali originali vengono conservati all'interno della base di conoscenza e non solo sostituiti dalle rispettive descrizioni testuali. Grazie all'uso degli MLLM, il sistema è inoltre in grado di elaborare input da parte dell'utente che includano immagini o altri media, abilitando una ricerca cross-modale attraverso cui, ad esempio, una query testuale può recuperare direttamente

un'immagine pertinente dal database vettoriale. Il modulo di generazione produce la risposta finale a partire da un unico prompt coerente che integra la query dell'utente e i risultati multimodali originali del retrieval, minimizzando la perdita di informazioni rispetto all'approccio precedente.
Come riportato da Mei et al., questo approccio migliora significativamente le prestazioni nei compiti di Question Answering complessi, in cui informazioni visive e testuali sono strettamente correlate. Nonostante i progressi rispetto alla generazione precedente, questa architettura introduce nuove sfide. Le capacità di recupero multimodale rimangono ancora inferiori a quelle del recupero puramente testuale e l'integrazione di input multimodali complessi può talvolta ridurre l'accuratezza nella gestione delle componenti testuali della query.

MRAG 3.0

L'MRAG 3.0 rappresenta l'evoluzione più recente e completa dei sistemi di generazione aumentata dal recupero multimodale, introducendo innovazioni strutturali che completano il framework in modo end-to-end. Mentre le versioni precedenti si concentravano sull'input multimodale o sulla costruzione della base di conoscenza, l'MRAG 3.0 estende le capacità del sistema alla produzione di output multimodali e all'ottimizzazione dinamica della ricerca.
Sul fronte della costruzione della knowledge base, per superare la perdita di informazioni tipica delle versioni precedenti, l'MRAG 3.0 cattura e conserva screenshot delle intere pagine dei documenti, preservando relazioni strutturali, layout e contesti visivi complessi che andrebbero persi con il chunking tradizionale. Tramite un MLLM fine-tuned, questi screenshot vengono vettorializzati e indicizzati, consentendo al sistema di recuperare direttamente la rappresentazione visiva della pagina più pertinente alla query dell'utente.
Una delle innovazioni più significative di questo sistema è l'introduzione di un modulo di Multimodal Search Planning, progettato per gestire query complesse in modo intelligente. Questo componente determina dinamicamente se sia necessario ricorrere a conoscenza esterna e, in caso affermativo, secondo quale modalità: solo testo, solo immagini o entrambe. Qualora la ricerca esterna sia ritenuta necessaria, il modulo provvede inoltre a riformulare la query per ottimizzare il recupero delle informazioni rilevanti. A differenza delle versioni precedenti, che producevano risposte esclusivamente testuali, l'MRAG 3.0 è in grado di generare output che integrano testo, immagini e video. Questo avviene secondo due modalità. Nella prima, detta Native MLLM-Based Output, la generazione multimodale è interamente guidata dal modello, che produce l'output in un unico passaggio. Nella seconda, detta Augmented Multimodal Output, il sistema genera dapprima una risposta testuale e la arricchisce successivamente individuando le posizioni ottimali in cui inserire elementi visivi, recuperati da basi di conoscenza esterne o dal web.

Grazie a queste innovazioni, l'MRAG 3.0 unifica in un unico framework integrato scenari precedentemente separati, come il Visual Question Answering, il RAG tradizionale e la generazione multimodale, a scapito però di una complessità implementativa significativamente maggiore rispetto alle versioni precedenti.

2.5 Agentic Frameworks

Questa sezione presenta gli agentic frameworks, strumenti software progettati per facilitare lo sviluppo e l'orchestrazione di sistemi basati su agenti AI e Large Language Models. Vengono analizzati in particolare LangChain e LangGraph, due tecnologie ampiamente utilizzate per la costruzione di applicazioni LLM-based e di sistemi agentici complessi, che costituiscono inoltre la base tecnologica della soluzione sviluppata in questo progetto di tesi.

2.5.1 Panoramica sugli Agentic Frameworks

Lo sviluppo di agenti intelligenti basati su modelli linguistici di grandi dimensioni ha stimolato la comparsa di numerosi framework, progettati per facilitare la costruzione e l'orchestrazione di sistemi agentici complessi [5]. Tali strumenti forniscono agli sviluppatori una serie di astrazioni e componenti predefiniti utili a orchestrare strumenti e memoria, definire sistemi di comunicazione e collaborazione e garantire misure di sicurezza e affidabilità, abbattendo il tempo di implementazione e la complessità tecnica che richiederebbe sviluppare ogni componente da zero. Secondo l'analisi condotta da Derouiche et al., tra i framework più diffusi figurano CrewAI, LangGraph, AutoGen, Semantic Kernel e altri. Di particolare interesse per questo progetto di tesi sono LangChain, framework general-purpose per la costruzione di applicazioni basate su LLM, e LangGraph, un framework complementare specificamente progettato per la realizzazione di sistemi agentici con struttura a grafo, che costituiscono le fondamenta tecnologiche del sistema sviluppato nella parte sperimentale.

2.5.2 LangChain

LangChain [29] è un framework open-source progettato per facilitare lo sviluppo di applicazioni intelligenti basate su modelli linguistici di grandi dimensioni. Il suo obiettivo principale è rendere componibili, modulari e riutilizzabili tutti i componenti necessari alla costruzione di sistemi basati su LLM. L'ecosistema è organizzato attorno a una serie di componenti fondamentali, progettati per interagire tra di loro all'interno di pipeline di elaborazione:

- Model: fornisce un insieme di astrazioni per interagire con LLM, modelli di embedding e modelli multimodali, standardizzando le API dei diversi provider e favorendo così l'intercambiabilità e la portabilità del sistema.
- Prompt: gestisce template dinamici per strutturare gli input da inviare al modello, rendendoli riutilizzabili e parametrici.
- Chain: componente che rappresenta sequenze riutilizzabili di operazioni che combinano i vari componenti fondamentali del framework. Questo approcio è stato poi deprecato e sostituito con il LangChain Expression Language (LCEL) [30] che offre maggiore flessibilità nella costruzione delle pipeline d'esecuzione.
- Tool: funzioni invocabili dall'LLM per estendere le sue capacità oltre la generazione testuale, come chiamate API, interrogazioni di database o esecuzione di calcoli.
- Memory: componente che si occupa della gestione dello stato e del contesto del sistema. Si hanno due tipologie di memoria: la short-term memory che mantiene le informazioni della sessione corrente e la long-term memory che garantisce la persistenza dei dati tra più sessioni.
- Agent: moduli che implementano un loop decisionale guidato da LLM, in cui il modello può scegliere autonomamente quali tool utilizzare, quali azioni compiere e come iterare fino al raggiungimento dell'obiettivo.

Accanto a questi componenti fondamentali, LangChain integra componenti avanzati dedicati alla RAG, che consentono di costruire pipeline di ingestion e di retrieval da integrare all'interno di sistemi agentici per aumentarne accuratezza e affidabilità:

- Document Loader: si occupa del caricamento e della trasformazione dei dati provenienti da fonti esterne in oggetti di tipo Document con struttura standardizzata.
- Text Splitter: suddivide i documenti in segmenti ottimizzati per l'encoding e il retrieval.
- Vector Stores: astrazioni che consentono di collegare al sistema un database vettoriale esterno.
- Retriever: permette di ricercare e recuperare documenti rilevanti da fonti di dati esterne come database vettoriali o siti web online.
- Output Parser: trasforma output grezzi prodotti dagli LLM in strutture dati formali, come JSON o oggetti tipizzati.

La maggior parte dei componenti LangChain presentati implementano l'interfaccia unificata dei Runnables, che li rende unità eseguibili e combinabili all'intetno di pipeline d'esecuzione dove l'output dell'eleborazione di un componente diventa l'input per il successivo. La composizione di queste sequenze Runnables è resa possibile dal LangChain Expression Language (LCEL) [30], un linguaggio dichiarativo che consente di orchestrare sequenze lineari o parallele, gestire branching, fallback e streaming.
LangChain rappresenta dunque un framework flessibile e modulare, con un ampio supporto per integrazioni con API esterne, database e vector store e una documentazione matura sostenuta da una comunità ampia e attiva. Tuttavia, per orchestrazioni agentiche avanzate, come workflow o sistemi multi-agente complessi, è necessario ricorrere a LangGraph [31], il framework complementare specificamente progettato per questo scopo. Complessivamente, LangChain si conferma uno strumento potente per applicazioni LLM modulari e RAG-centriche, pur richiedendo attenzione nella progettazione di sistemi agentici complessi.

2.5.3 LangGraph

LangGraph [31] è un framework di orchestrazione a basso livello per costruire, gestire e distribuire agenti AI stateful e a lunga esecuzione. A differenza di LangChain, che offre architetture preconfigurate ad alto livello, LangGraph garantisce un controllo granulare sulla logica dell'agente, consentendo di modellarne il comportamento esecutivo attraverso una rappresentazione a grafo. Il framework ruota attorno ad alcuni componenti fondamentali:

- Graph: struttura centrale di LangGraph che definisce il flusso logico delle operazioni svolte dal sistema agentico sotto forma di grafo diretto, potenzialmente ciclico. È composto da nodi (nodes), che rappresentano unità eseguibili, ed archi (edges) che li collegano definendo la sequenza di esecuzione.
- State: struttura dati condivisa tra tutti i nodi del grafo, che mantiene il contesto corrente dell'intera applicazione. Lo stato può contenere informazioni temporanee (in-thread memory) o persistenti (cross-thread memory) e viene aggiornato progressivamente dai nodi durante l'esecuzione del grafo.
- Nodes: unità eseguibili che ricevono in input lo stato corrente del grafo e lo aggiornano in base al risultato della propria esecuzione. I nodi possono performare chiamate a LLM, esecuzione di tool o logiche custom.
- Edges: collegamenti tra i nodi del grafo che definiscono il flusso di esecuzione del sistema agentico. Possono essere archi sequenziali, condizionali, o ciclici.
- Checkpointer: componente che salva lo stato del grafo in punti intermedi dell'esecuzione, preservandolo tra esecuzioni differenti e consentendo di riprendere

il flusso da dove era stato interrotto. È fondamentale per scenari long-running e per garantire resilienza in caso di guasti o interruzioni.

Grazie a questa architettura, LangGraph offre una serie di funzionalità che lo distinguono da soluzioni che operano a livelli d'astrazione più elevati. La modellazione del flusso di esecuzione attraverso una rappresentazione a grafo consente innanzitutto di realizzare workflow agentici e sistemi multi-agente complessi e articolati, con logiche completamente personalizzabili. A ciò si affianca la gestione dello stato condiviso, che consente a ciascun nodo di leggere e aggiornare dinamicamente il contesto corrente durante l'esecuzione, garantendo coerenza in conversazioni multi-turno e workflow complessi e supportando la persistenza sia locale sia esterna tra sessioni distinte. La continuità tra sessioni è ulteriormente rafforzata dal meccanismo di checkpointing, che preserva lo stato del grafo in punti intermedi dell'esecuzione e consente di riprendere il flusso in caso di interruzioni, rendendo il sistema resiliente e adatto a scenari long-running. LangGraph supporta inoltre il paradigma Human-in-the-Loop (HITL), che consente l'intervento umano all'interno del flusso di esecuzione. I nodi del grafo possono essere infatti configurati per mettersi in pausa e attendere input o approvazioni da un operatore, il quale può modificare lo stato, accettare o rifiutare azioni pianificate prima che il grafo prosegua. Questo meccanismo è fondamentale in domini dove la supervisione umana è necessaria per garantire sicurezza e affidabilità. Infine, dal punto di vista dell'ecosistema, LangGraph si integra nativamente con LangChain, potendo sfruttare tutti i suoi componenti precostruiti. Questa complementarità consente di combinare la potenza di orchestrazione di LangGraph con la ricchezza di integrazioni offerta da LangChain. LangGraph è inoltre integrato nativamente con LangSmith [32], strumento di osservabilità che permette di tracciare le operazioni eseguite, visualizzare le transizioni di stato e monitorare metriche dettagliate in tempo reale.
In conclusione, LangGraph si conferma un framework potente e flessibile per la costruzione di agenti e applicazioni stateful basate su LLM. I suoi principali punti di forza risiedono nella capacità di modellare flussi complessi tramite orchestrazione graph-based, nella gestione dello stato sia temporaneo sia persistente, nel checkpointing per la resilienza e la continuità tra sessioni, e nel supporto nativo a meccanismi Human-in-the-Loop. Tuttavia, l'elevata flessibilità comporta una maggiore complessità progettuale: la definizione di grafi articolati, la gestione dello stato e l'implementazione di meccanismi di persistenza richiedono competenze più avanzate rispetto all'uso degli agenti standard di LangChain. Il framework presenta pertanto una curva di apprendimento più ripida e può risultare sovradimensionato per applicazioni semplici o prototipali.

Capitolo 3

Progettazione, Sviluppo e Integrazione

3.1 Analisi dei Requisiti

La sperimentazione condotta all'interno di questo progetto di tesi, mira alla realizzazione di un agente conversazionale da integrare all'interno di un applicativo di BPM per la gestione e la redazione di atti amministrativi fornito da EOS Reply a uno dei propri clienti. Il sistema si propone di supportare operativamente gli utenti coinvolti nell'esecuzione del modello di processo attualizzato, che si presenta come un flusso estremamente articolato e complesso.
A tal proposito, i requisiti fondamentali che l'assistente conversazionale deve soddisfare sono tre: tempestività, pertinenza e affidabilità. Il sistema deve innanzitutto fornire risposte in tempo reale, così da risolvere prontamente incertezze e problematiche che potrebbero generare errori operativi o ridurre la produttività degli operatori. Le risposte generate devono inoltre essere pertinenti e rilevanti rispetto alle richieste degli utenti, fornendo un aiuto concreto e non una fonte di ulteriore confusione. Infine, i contenuti forniti devono essere affidabili, ovvero corretti dal punto di vista contenutistico e coerenti con le procedure del sistema.
Questi requisiti assumono un'importanza particolare per due ragioni strettamente legate alla natura del contesto applicativo. In primo luogo, indicazioni errate o poco pertinenti possono rallentare il lavoro degli operatori, causando ritardi nella pubblicazione degli atti amministrativi e producendo l'effetto opposto a quello che l'assistente dovrebbe generare. In secondo luogo, gli atti amministrativi lavorati dal BPMS acquisiscono valore legale nel corso del flusso verso la pubblicazione, attraversando diversi step di verifica che ne garantiscono la conformità giuridica e tecnica. In seguito ad alcuni di questi step, i documenti vengono sottoposti a firma digitale e non possono più essere modificati. Ogni errore nelle operazioni può

quindi non solo generare ritardi, ma rendere estremamente difficile la correzione del documento, richiedendo spesso l'intervento diretto di assistenza o, nei casi più gravi, l'archiviazione definitiva dell'atto e la sua ricreazione dall'inizio.

3.2 Scelte Progettuali Preliminari

Questa sezione descrive le principali scelte progettuali preliminari adottate prima dell'avvio della fase sperimentale, necessarie per soddisfare i requisiti di qualità del sistema e per guidare l'implementazione dell'agente conversazionale. In particolare, vengono illustrate le decisioni relative alla selezione dell'architettura di Retrieval-Augmented Generation, del database vettoriale, dei modelli di intelligenza artificiale e del framework agentico utilizzato per la realizzazione del sistema conversazionale. Tali scelte costituiscono le fondamenta tecnologiche dell'assistente sviluppato e influenzano direttamente le sue prestazioni, affidabilità e capacità di integrazione con l'applicativo di BPM.

3.2.1 Selezione dell'Architettura RAG

Dall'analisi dei ticket di supporto dell'applicativo di BPM amministrativo di EOS Reply presentata in Figura 1.1 è emerso che più del 50% delle richieste di assistenza risultava risolvibile attraverso la consultazione della documentazione ufficiale del sistema. Questa fonte copre le funzionalità del sistema e le operazioni previste dal modello di processo attualizzato, racchiudendo quindi al suo interno le informazioni di cui gli operatori necessitano durante l'utilizzo del BPMS.
Questo scenario si presta particolarmente all'applicazione di un meccanismo di Retrieval-Augmented Generation. La scelta della RAG come soluzione architetturale per la generazione delle risposte è motivata dalle limitazioni intrinseche degli LLM: l'ampliamento specialistico della conoscenza parametrica è estremamente oneroso e la tendenza a produrre allucinazioni compromette l'affidabilità dei contenuti generati. Queste criticità rendono i modelli linguistici, presi da soli, insufficienti per soddisfare i requisiti del dominio applicativo. La RAG permette di superare questi ostacoli grazie all'ingestion della documentazione ufficiale dell'applicativo di BPM all'interno di una knowledge base vettoriale, dalla quale è possibile recuperare dinamicamente i frammenti semanticamente più rilevanti rispetto alle richieste dell'utente e utilizzarli come contesto per la generazione delle risposte. La pertinenza delle risposte è garantita dal meccanismo di retrieval, che seleziona i contenuti sulla base della similarità semantica con la query dell'utente, assicurando che il modello disponga delle informazioni più rilevanti rispetto alla necessità espressa. L'affidabilità è invece garantita dal fatto che i contenuti generati non derivano dalla conoscenza parametrica del modello, ma sono direttamente ancorati alla documentazione ufficiale dell'applicativo, che costituisce la fonte autorevole di

riferimento per le procedure e le funzionalità del sistema.
La documentazione del BPMS consiste in un documento non strutturato contenente un numero significativo di immagini raffiguranti schermate esplicative dell'applicativo, utili a guidare visivamente gli utenti nell'utilizzo dell'interfaccia dell'applicativo di BPM e nell'esecuzione delle operazioni previste dal modello di processo che attualizza. I contenuti visivi sono quindi parte integrante del valore informativo della documentazione e ignorarli rappresenterebbe una perdita significativa in termini di qualità nelle risposte generate. Questa considerazione motiva la scelta di estendere l'architettura RAG verso un sistema di Multimodal RAG, in modo che l'agente conversazionale possa sfruttare non solo i contenuti testuali della documentazione ufficiale, ma anche quelli visivi, fornendo risposte più complete che permettono all'utente di orientarsi sull'applicativo.

3.2.2 Selezione del Database Vettoriale

Alla base dell'architettura RAG vi è una knowledge base vettoriale, costruita attraverso il processo di ingestion documentale delle fonti di riferimento. Un database vettoriale è un sistema di archiviazione specializzato nella memorizzazione e nel recupero di embedding, in grado di catturare il significato semantico dei contenuti codificati. A differenza dei database tradizionali, che recuperano dati tramite corrispondenze esatte, un database vettoriale esegue ricerche per similarità: dato un vettore di query, il sistema individua i vettori più simili utilizzando metriche come la cosine similarity o la distanza euclidea.
La diffusione degli LLM e l'introduzione della RAG, hanno portato alla nascita di numerosi database vettoriali. Nello studio condotto da Filipovska et al. quattro di essi, ChromaDB, Qdrant, FAISS e Pinecone, sono stati comparati dal punto di vista prestazionale su un task di Question Answering (QA) paragonabile a quello relativo alla documentazione del BPMS amministrativo trattato all'interno di questo progetto di tesi. In particolare, l'analisi di Filipovska et al. si è occupata di confrontare le prestazioni dei database vettoriali analizzati all'interno di un'architettura RAG standard. Il sistema è stato testato su un dataset di dominio specifico composto da 100 policy di privacy, con l'obiettivo di rispondere a 45 domande binarie relative al al Regolamento Generale sulla Protezione dei Dati (GDPR). I risultati hanno evidenziato che ChromaDB e Qdrant offrono prestazioni superiori e comparabili tra loro, risultando circa 30 volte più veloci di FAISS e 15 volte più veloci di Pinecone, mantenendo tempi di esecuzione costantemente inferiori a 0.05s anche all'aumentare della lunghezza dei documenti, mostrando un'ottima scalabilità. Inoltre, i due database hanno dimostrato una consistenza equivalente nel recupero dei chunk rilevanti.
La scelta del database vettoriale da impiegare in questa sperimentazione per la realizzazione del sistema di RAG utilizzato dall'agente conversazionale è ricaduta

su ChromaDB [34] per una serie di ragioni complementari. Oltre alle prestazioni evidenziate da Filipovska et al., ChromaDB è open-source e può essere eseguito completamente in locale, senza dipendere da infrastrutture cloud esterne, garantendo inoltre maggiore controllo sui dati. Infine, ChromaDB supporta nativamente l'associazione di metadati agli embedding, una funzionalità essenziale per la realizzazione di sistemi di MRAG 2.0, in cui le immagini estratte dai documenti originali vengono archiviate come metadati associati ai rispettivi chunk testuali per poter essere utilizzate in fase generativa.

3.2.3 Selezione dei Modelli di Intelligenza Artificiale

L'architettura RAG e i sistemi agentici basano il loro funzionamento su modelli di intelligenza artificiale, in particolare gli LLM. Per questo motivo è stato necessario selezionare un LLM da impiegare sia nella pipeline di ingestion documentale sia nella realizzazione dell'agente conversazionale.
Al giorno d'oggi esiste un vastissimo panorama di LLM, che si distinguono principalmente per modalità di accesso, capacità multimodali, dimensione della context window, performance e costi operativi. Una prima distinzione rilevante riguarda i modelli proprietari, accessibili tramite API fornite dal vendor con un costo a consumo, e i modelli open-source, eseguibili localmente. I modelli proprietari sono generalmente molto performanti e garantiscono tempi di risposta ridotti, beneficiando dell'infrastruttura hardware del provider, ma sono completamente black-box nel loro funzionamento. I modelli open-source, al contrario, garantiscono pieno controllo sui dati e assenza di costi di licenza, ma richiedono infrastrutture hardware costose per ottenere tempi di inferenza adeguati, soprattutto in scenari real-time come quello considerato in questa tesi. Proprio quest'ultimo aspetto ha determinato la scelta di impiegare un modello proprietario a consumo.
Oltre a questa distinzione di natura economica, la scelta del modello dipende da requisiti funzionali specifici del sistema. La documentazione del BPMS contiene un elevato numero di immagini esplicative, essenziali per guidare l'utente nell'utilizzo dell'interfaccia dell'applicativo e nello svolgimento delle attività operative. Per questo motivo è stato necessario selezionare un modello in grado di elaborare non solo contenuti testuali ma anche visivi. Questa capacità multimodale è indispensabile per la realizzazione di meccanismi di MRAG. Durante la pipeline di ingestion, è necessario un MLLM per generare didascalie descrittive delle immagini presenti nella documentazione, producendo embedding da indicizzare che tengano conto anche del contenuto visivo. In versioni MRAG avanzate un MLLM è necessario anche durante la fase di generazione, in modo da poter generare risposte elaborando non solo i frammenti testuali estratti dal meccanismo di retrieval, ma anche le immagini associate.
Un ulteriore criterio di selezione riguarda la context window, ovvero il numero

massimo di token che il modello è in grado di elaborare in una singola esecuzione. Tale limite include sia i token di input (prompt, istruzioni e contesto) sia i token di output generati dal modello. Questo parametro è particolarmente rilevante nel contesto di questa sperimentazione poiché, adottando un modello proprietario, l'unico meccanismo di fusione utilizzabile è la query-based fusion, che prevede la concatenazione dei documenti recuperati dal retrieval direttamente nel prompt inviato al modello generativo. Superare il limite della context window comporta il troncamento dell'input o dell'output con un conseguente deterioramento della qualità delle risposte, oppure la restituzione di un messaggio d'errore da parte dell'LLM. È stato quindi necessario selezionare un modello con una context window estesa, in grado di gestire contesti lunghi senza perdita di informazioni.
Infine, un aspetto determinante è la scala del modello. Modelli con un numero elevato di parametri offrono performance generalmente superiori ma tempi di inferenza più lunghi, mentre modelli di scala ridotta garantiscono maggiore tempestività a fronte di capacità leggermente inferiori. Visto l'obiettivo di realizzare un assistente conversazionale in grado di supportare l'utenza in tempo reale è stato selezionato un modello di scala ridotta cercando pur sempre di mantenere un buon livello di intelligenza necessario a garantire la qualità delle risposte generate.
In conclusione sulla base delle necessità progettuali e sulle considerazioni presentate, la scelta è ricaduta su GPT-5 nano [35] di OpenAI, un LLM proprietario di piccola scala progettato per garantire inferenza rapida e costi contenuti, ideale per soddisfare la necessità di tempestività di risposta del caso d'uso. Questo modello possiede una context window di 400k token, sufficiente a gestire conversazioni estese e incorporare nel prompt attraverso la query-based fusion i documenti ottenuti tramite il meccanismo di RAG. GPT-5 nano è inoltre multimodale, caratteristica essenziale per la costruzione della knowledge base vettoriale a partire dalla documentazione del BPMS ricca di immagini.
Oltre agli LLM, il meccanismo di RAG richiede un modello di embedding per trasformare i chunks documentali in rappresentazioni vettoriali. Come per gli LLM anche in questo caso esistono moltissimi modelli di questo genere. L'aspetto più importante nella selezione di un modello di embedding è il trade off tra la dimensione dei chunks da codificare e la dimensionalità del vettore risultante. Questo compromesso è difficile da individuare a livello sperimentale, ma è molto rilevante per realizzazione di un meccanismo di recupero performante. Infatti, se la dimensionalità del vettore prodotto risulta insufficiente rispetto alla quantità di contenuto da codificare, il vettore perde di rappresentatività semantica, compromettendo il meccanismo di retrieval basato su similarity search e quindi la qualità delle risposte generate dal sistema. Per valutare l'impatto di questo parametro sul modulo di retrieval, si è scelto di utilizzare due modelli con caratteristiche diverse: text-embedding-3-small di OpenAI [36], con una dimensionalità di 1536 elementi [37], e il modello open-source nomic-embed-text di Nomic attraverso Ollama [38, 39],

con una dimensionalità più ridotta pari a 768 elementi [40].

3.2.4 Selezione del Framework per la Realizzazione di Sistemi Agentici

Oltre alle scelte progettuali preliminari presentate finora, è stato necessario selezionare un framework per la realizzazione di sistemi agentici. Negli ultimi anni, infatti, la diffusione degli LLM ha favorito la nascita di numerosi strumenti dedicati allo sviluppo di agenti basati su tali modelli [5]. Per la realizzazione dell'assistente conversazionale da integrare all'interno del BPMS degli atti amministrativi è stato selezionato il framework LangGraph [31]. La scelta è stata guidata sia dai requisiti specifici del contesto applicativo sia dalle funzionalità offerte dal framework.
LangGraph adotta un approccio di orchestrazione graph-based, che consente di rappresentare il flusso logico di esecuzione dell'agente attraverso una struttura a grafo. Tale paradigma permette di modellare in modo esplicito il processo attraverso cui l'agente elabora le richieste dell'utente e genera le risposte. Attraverso meccanismi come branching, looping e nodi di fallback, il framework consente di rappresentare percorsi di esecuzione complessi e di gestire in maniera appropriata anche i casi in cui l'assistente non sia in grado di fornire una risposta attendibile. L'affidabilità e la controllabilità del comportamento dell'agente sono garantite proprio dalla struttura del grafo di esecuzione, che impone un percorso operativo ben definito. Il grafo consente inoltre di tracciare in modo dettagliato il processo di generazione delle risposte. Ciò può avvenire sia tramite meccanismi di logging sia attraverso LangSmith [32], il sistema di osservabilità messo a disposizione dall'ecosistema LangChain. Questo permette di monitorare e analizzare il comportamento dell'agente durante l'esecuzione, facilitando attività di debugging, valutazione e miglioramento del sistema.
Un ulteriore elemento rilevante è rappresentato dall'integrazione di LangGraph all'interno dell'ecosistema LangChain. LangGraph supporta infatti le principali astrazioni ad alto livello offerte da LangChain, incluse quelle necessarie per implementare meccanismi di RAG. Tali meccanismi risultano fondamentali nel contesto del progetto, in cui l'assistente deve generare risposte basate sulla documentazione relativa al BPMS degli atti amministrativi. L'integrazione con LangChain garantisce inoltre un'elevata compatibilità con numerosi strumenti e servizi esterni, mettendo a disposizione oltre mille integrazioni per lo sviluppo di applicazioni basate su LLM [41]. Tra le integrazioni più rilevanti per questa sperimentazione vi sono provider di modelli linguistici e di embeddings, come OpenAI per l'utilizzo di modelli proprietari e Ollama per l'esecuzione di modelli open source, database vettoriali come ChromaDB per l'implementazione dei meccanismi di RAG e framework per la creazione di API web, come FastAPI [42], utilizzati per l'esposizione delle funzionalità dell'agente.

Un altro vantaggio significativo di LangGraph riguarda la gestione dello stato dell'applicazione. Al grafo di esecuzione è infatti associata una struttura dati che registra l'evoluzione della conversazione man mano che l'utente interagisce con il sistema. Questo meccanismo consente all'assistente di gestire efficacemente richieste di follow-up, sfruttando le informazioni derivanti dalle interazioni precedenti. Inoltre, le conversazioni possono essere memorizzate in maniera persistente, rendendo possibile sia la consultazione delle interazioni passate sia il riutilizzo del contesto conversazionale per rispondere a nuove richieste. La persistenza dello stato, unita al meccanismo di checkpointing, consente inoltre all'utente di riprendere conversazioni precedentemente interrotte.
Il framework supporta anche lo streaming delle risposte, permettendo di restituire i token generati dal modello linguistico progressivamente, senza attendere il completamento dell'intero processo di esecuzione del grafo. Lo streaming può riguardare anche lo stato di avanzamento dell'esecuzione, consentendo di aggiornare l'utente in tempo reale sull'evoluzione delle operazioni in corso. Questa funzionalità contribuisce a migliorare l'esperienza utente, riducendo la percezione dei tempi di attesa.
Alla luce delle caratteristiche descritte, LangGraph rappresenta una soluzione particolarmente adatta per la realizzazione dell'assistente conversazionale previsto dal progetto. Il framework consente infatti di implementare agenti controllabili, osservabili e affidabili, mantenendo al contempo un'elevata flessibilità nello sviluppo dell'applicazione. Inoltre, l'integrazione con l'ecosistema LangChain mette a disposizione un ampio insieme di strumenti e astrazioni che facilitano la realizzazione di applicazioni basate su LLM complesse e strutturate.

3.3 Il Sistema di Retrieval-Augmented Generation

Come discusso nelle sezioni precedenti, al fine di garantire che l'assistente conversazionale integrato nell'applicativo di BPM amministrativo sia in grado di generare risposte affidabili a supporto degli utenti del sistema, è stata adottata una soluzione basata su un sistema di RAG.
Tale soluzione architetturale prevede due meccanismi fondamentali: l'ingestion pipeline e la retrieval pipeline. La prima riguarda il processo di costruzione della knowledge base vettoriale a partire dalla documentazione ufficiale del BPMS, mentre la seconda costituisce il meccanismo che consente di individuare e recuperare le informazioni più rilevanti rispetto alle query formulate dagli utenti del sistema.
La documentazione su cui si basa il meccanismo di RAG consiste in un file in formato PDF caratterizzato da un layout non strutturato. Ciò significa che il documento non presenta un formato predeterminato e che i contenuti non sono

organizzati secondo una struttura facilmente interpretabile in modo automatico, rendendo più complesso il processo di suddivisione del testo in chunk. Oltre al contenuto testuale, il documento include un numero significativo di immagini illustrative delle funzionalità offerte dal sistema di gestione e lavorazione degli atti amministrativi, nonché alcune tabelle. Questi contenuti multimodali rivestono un ruolo rilevante, poiché consentono all'assistente di supportare gli utenti del BPMS nell'utilizzo dell'interfaccia del sistema e nell'esecuzione delle attività operative previste dal modello di processo. Per sfruttare il valore informativo di tali contenuti visivi, nel corso della sperimentazione sono stati sviluppati e confrontati due diversi sistemi di MRAG. Il primo è basato su un'architettura MRAG 2.0, che integra i contenuti visivi sia nella fase di ingestion sia in quella di generazione delle risposte. Il secondo, invece, si basa su un'architettura MRAG 1.0, nella quale le immagini vengono convertite in descrizioni testuali utilizzate successivamente dal modello generativo. Il confronto tra i due approcci ha l'obiettivo di valutare l'impatto della gestione multimodale sulle performance complessive dell'assistente conversazionale, al fine di identificare la più performante.

3.3.1 Ingestion Pipeline

Precedentemente al processo di ingestion documentale per la realizzazione della knowledge base vettoriale alla base del sistema di RAG è stato eseguito un preprocessing manuale sulla documentazione del BPMS da elaborare, finalizzato a rimuovere elementi superflui che avrebbero potuto ostacolare la creazione del database vettoriale o comprometterne la qualità. In particolare, sono stati eliminati la numerazione grafica delle pagine, le intestazioni e i piè di pagina, elementi privi di contenuto informativo che avrebbero introdotto rumore nel processo di chunking. A seguito di questa fase preparatoria sono state eseguite due pipeline di ingestion differenti, in conformità ai paradigmi MRAG 1.0 e MRAG 2.0 introdotti nella Sezione 2.4.4.

Ingestion Pipeline MRAG 2.0

Di seguito, in Figura 3.1, viene illustrata la pipeline di ingestion MRAG 2.0.
Il primo step della pipeline consiste nel partitioning della documentazione del BPMS. Questa fase consiste nell'estrazione dei contenuti elementari che compongono il documento di partenza e la loro conversione in un formato manipolabile per le elaborazioni successive. A tale scopo è stata impiegata la libreria open-source Unstructured [43] che fornisce un toolkit progettato per semplificare l'ingestion e il pre-processing di dati in formati eterogenei, inclusi documenti basati su testo come PDF, con un focus sull'ottimizzazione dei flussi di lavoro per gli LLM.
Per la fase di partitioning, Unstructured mette a disposizione diverse astrazioni

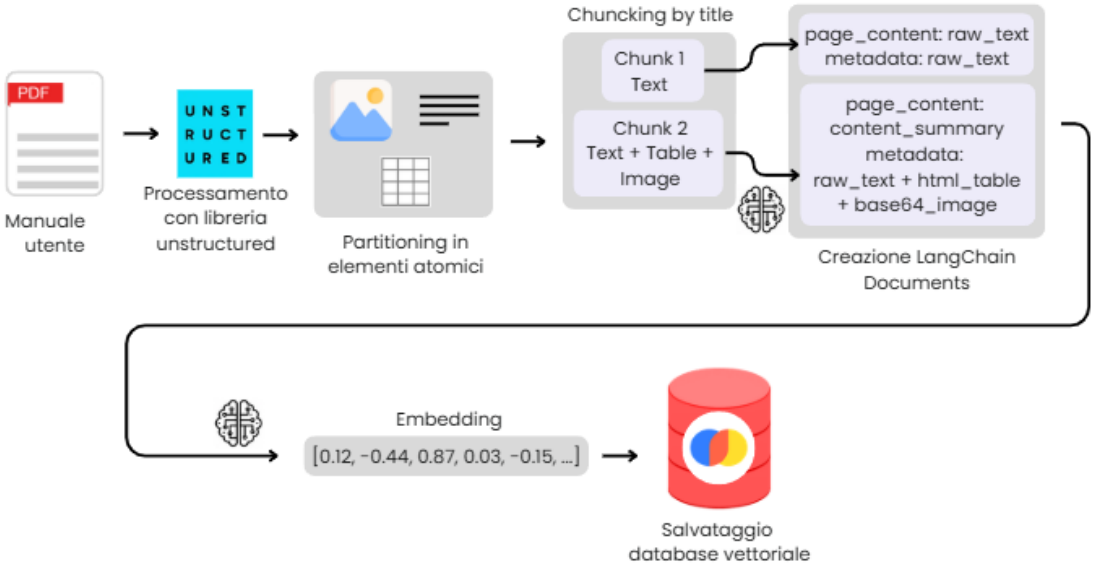

Figura 3.1: Ingestion Pipeline MRAG 2.0

specifiche per i differenti formati documentali supportati, che consentono di estrarre automaticamente contenuti strutturati a partire da documenti grezzi non strutturati.
La libreria consente l'estrazione dei contenuti element-wise, che permette di suddividere il documento in elementi di tipologia differente a seconda della struttura del documento stesso, come ad esempio *Title* per i titoli, *NarrativeText* per i paragrafi testuali e così via. Nel caso della documentazione del BPMS, è stata utilizzata la funzione *partition_pdf* [44]. Questa funzione utilizza un modello di layout detection per analizzare la struttura del documento e mappare gli elementi che lo compongono nei tipi semantici definiti da Unstructured. Inoltre, impiega un parser rule-based per l'estrazione del testo digitale. La combinazione di questi due meccanismi consente di ottenere un'estrazione ad alta qualità, preservando al contempo le informazioni relative all'organizzazione strutturale dei contenuti. La funzione utilizzata permette inoltre di estrarre le tabelle in formato html, preservandone fedelmente la struttura senza dover ricorrere a meccanismi di OCR più approssimativi, e le immagini in formato base64, supportato da LangChain per l'elaborazione di immagini con modelli linguistici multimodali.
La scelta di Unstructured per il partitioning è supportata dai risultati presentati nello studio di Adhikari and Agarwal, nel quale sono stati confrontati dieci strumenti di parsing PDF su sei categorie documentali. Per la categoria "manuals", assimilabile al documento trattato di questa sperimentazione, Unstructured ha

mostrato prestazioni di estrazione particolarmente elevate: un'accuratezza parola-per-parola molto alta (F1 Score: 0.9843), un'ottima conservazione dell'ordine e dell'identificazione delle parole (BLEU Score: 0.8913) e una solida qualità complessiva di local alignment (0.8835).
Al termine della fase di partitioning vengono estrapolati i seguenti componenti elementari:

- Title, NarrativeText, Text, ListItem, FigureCaption: in formato testuale.
- Image: in formato base64.
- Table: in formato html.

Questa rappresentazione strutturata è un prerequisito fondamentale per la fase successiva: il chunking. La fase di chunking è una delle più critiche per ottenere un sistema RAG capace di recuperare contenuti rilevanti in modo efficace e preciso. Essa consiste nell'organizzare i componenti elementari estratti dalla fase precedente in porzioni più ampie e coerenti dal punto di vista concettuale. Questo processo di fatto permette di suddividere il documento originale in sezioni tematicamente omogenee. Questa necessità nasce dal fatto che processare l'intero manuale utente con un LLM non sarebbe possibile a causa del limite imposto dalla context window del modello generativo e al fenomeno del "lost-in-the-middle" [46], per cui le informazioni intermedie di testi estesi tendono ad essere trascurate, mentre il modello tende a focalizzarsi maggiormente sulle informazioni iniziali e finali. Segmentare i documenti risulta essenziale anche per garantire l'efficacia del processo di retrieval. Questo aspetto è dovuto a due motivazioni principali. In primo luogo, i modelli di embedding utilizzati per l'indicizzazione semantica dei contenuti nella knowledge base vettoriale oltre ad avere una propria context window, generano vettori di dimensione fissa. Di conseguenza se il contenuto da codificare è troppo lungo, il vettore risultante perde capacità rappresentativa, compromettendo la qualità della similarity search. In secondo luogo, i chunks devono essere semanticamente coerenti, costituendo idealmente una singola unità concettuale. Questa coerenza determina un retrieval più preciso, permettendo al sistema di recuperare parti del documento più specifiche e ben delimitate, invece di sezioni tematicamente eterogenee.
In questo caso è stata adottata una strategia di chunking semantico, con l'obiettivo di determinare i punti di suddivisione ottimali in modo da preservare la coerenza concettuale dei segmenti e quindi migliorare la precisione del processo di retrieval. Per il processo di chunking è stato impiegato il metodo *chunk_by_title* [47] della libreria Unstructured, che raggruppa i componenti elementari risultanti dal processo di partitioning sulla base della struttura della documentazione del BPMS. Le sezioni in questo caso vengono identificate dagli elementi di tipo *Title*, consentendo così di unire in un unico chunk le informazioni appartenenti alla stessa porzione di documento.

In seguito alla fase di chunking, gli insiemi di elementi raggruppati vengono convertiti in LangChain Document e preparati per la fase embedding. Per i chunks contenenti contenuti multimodali viene generata una sintesi dei contenuti capace di integrare testo, immagini e tabelle in un'unica descrizione ricercabile in fase di retrieval. Per questa operazione è stato impiegato il modello multimodale GPT-5 nano e un prompt apposito. Questa sintesi costituirà l'elemento da codificare come embedding, usato per indicizzare il corrispettivo Document all'interno del database vettoriale. Nonostante ciò, i contenuti originali del chunk non vengono persi: essi sono inclusi come metadati all'interno del rispettivo LangChain Document. Nell'eventualità in cui il chunk contenga esclusivamente testo, non viene prodotta alcuna sintesi, ma viene codificato il suo contenuto originale.
La struttura dei documenti creati è la seguente:

```
Document(
    page_content='...',
    metadata={
        'original_content': {
            "raw_text": "...",
            "tables_html": [...],
            "images_base64": [...]
        },
    }
)
```

Dove il campo *page_content* contiene la sintesi ricercabile dei contenuti multimodali del chunk d'origine oppure, nel caso il chunk sia composto da soli elementi testuali, il testo originale di quest'ultimo. Il campo *metadata* preserva invece tutti i contenuti originali del corrispettivo chunk.
In seguito della creazione dei LangChain Document, si ha la fase di embedding in cui il campo *page_content* viene codificato attraverso il modello text-embedding-3-small [36] di OpenAI e utilizzato per indicizzare i segmenti di documentazione corrispondenti all'interno un database Chroma persistente.

Ingestion Pipeline MRAG 1.0

In Figura 3.2 viene illustrata la pipeline di ingestion MRAG 1.0.
Anche in questa soluzione, la fase di partitioning è stata eseguita utilizzando la libreria Unstructured, con l'obiettivo di estrarre i componenti elementari del documento e renderli manipolabili.
La fase di chunking, come nella pipeline precedente, segue un approccio di segmentazione semantica che mira a raggruppare nello stesso chunk gli elementi appartenenti

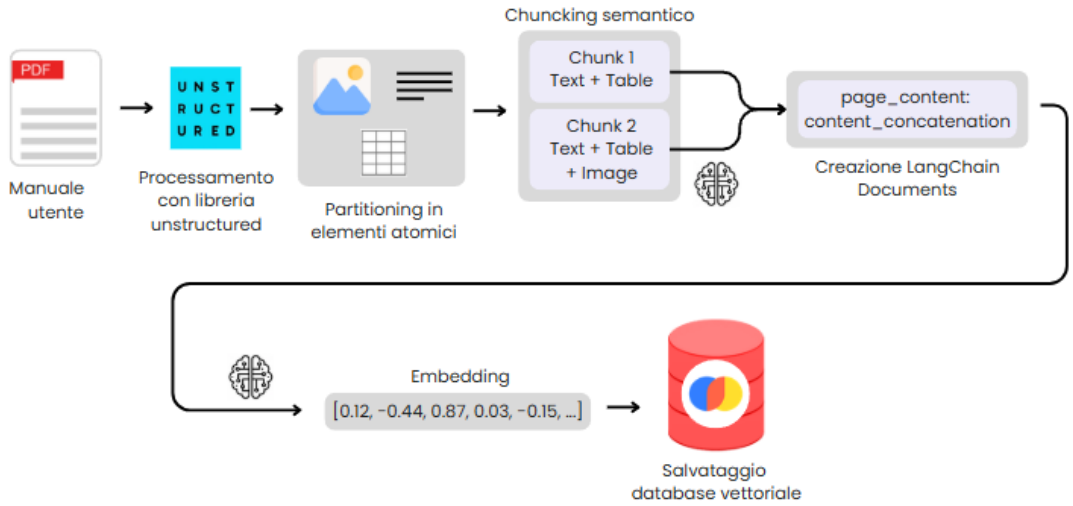

Figura 3.2: Ingestion Pipeline MRAG 1.0

alla medesima sezione della documentazione. In questo caso, tuttavia, è stato adottato un metodo di chunking custom. Tutti gli elementi estratti dal partitioning vengono processati sequenzialmente e uniti in segmenti distinti sulla base degli elementi di tipo *Title*, che identificano i paragrafi del documento. I contenuti testuali vengono concatenati in ordine, le tabelle vengono incorporate in formato html, mentre per le immagini viene utilizzato il modello GPT-5 nano, al quale viene fornito un prompt dedicato per generare una descrizione visiva corrispondente, da inserire in continuità con i contenuti già processati del segmento. Il risultato di questo processo è un insieme di chunks interamente discorsivi, in cui testi, tabelle e immagini sono fuse in un'unica narrazione continua.
In questa pipeline, i LangChain Document vengono costruiti utilizzando esclusivamente il contenuto discorsivo dei chunks prodotti nella fase precedente, senza includere alcun metadato aggiuntivo. La struttura che ne risulta è la seguente:

```
Document(
    page_content = '...',
)
```

Dove il campo *page_content* contiene la concatenazione discorsiva del segmento di origine.
Una volta creati i Document, il contenuto della proprietà *page_content* viene vettorizzato tramite il modello text-embedding-3-small e utilizzato per l'indicizzazione del documento all'interno di un database Chroma, come nella prima pipeline. Parallelamente, in questo caso, è stato utilizzato anche il modello nomic-embed-text per generare un secondo database (corrispondente al primo come contenuti), in

modo da poter valutare l'impatto di modelli di embedding differenti sulla qualità del meccanismo di recupero.

3.3.2 Retrieval Pipeline

Oltre alla pipeline di ingestion documentale, i sistemi RAG sono caratterizzati da un secondo meccanismo fondamentale: la pipeline di retrieval. La pipeline di retrieval è il sistema che consente all'agente di accedere alla knowledge base vettoriale realizzata attraverso l'ingestion pipeline, con l'obiettivo di individuare ed estrapolare informazioni relevanti necessarie per rispondere in maniera affidabile e pertinente alle richieste dell'utente.
Il meccanismo di retrieval che è stato sviluppato è lo stesso per entrambe le soluzioni di ingestion presentate nella sezione precedente. Ciò che differisce è la struttura dei LangChain Document recuperati dalla knowledge base, e quindi i dati a disposizione dell'agente per la generazione delle risposte.
Per la pipeline MRAG 2.0 i Document indicizzati all'interno del database vettoriale contengono una sintesi dei contenuti originali nel campo *page_content* e i dati originali del segmento corrispondente come metadati. Per quanto riguarda invece la pipeline MRAG 1.0, i Document includono esclusivamente il campo *page_content*, con testo, tabelle e descrizioni visive delle immagini concatenati in forma discorsiva. Le Figure 3.3 e 3.4 riportano gli schemi implementativi delle due pipeline di retrieval corrispondenti.

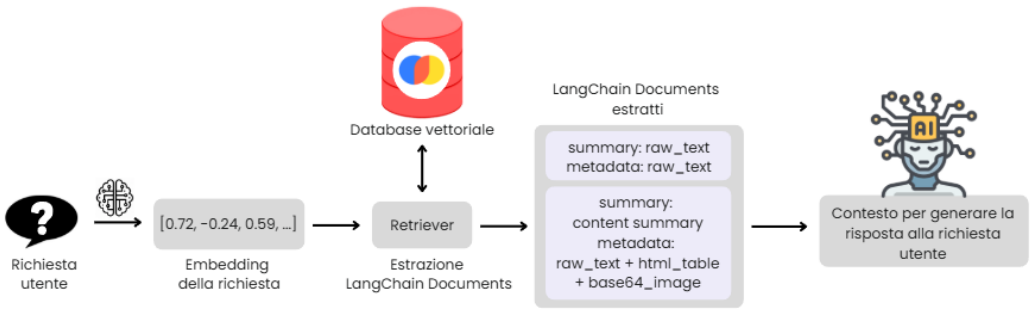

Figura 3.3: Retrieval Pipeline MRAG 2.0

Il funzionamento delle due procedure di retrieval è sostanzialmente identico. In entrambi i casi, il processo inizia con l'encoding della richiesta dell'utente, operazione che consiste nella trasformazione della query testuale in un vettore numerico. Questo vettore può essere confrontato con quelli associati ai documenti indicizzati nella knowledge base, permettendo di misurare la somiglianza semantica tra la domanda dell'utente e i contenuti disponibili. Attraverso tale confronto, il sistema

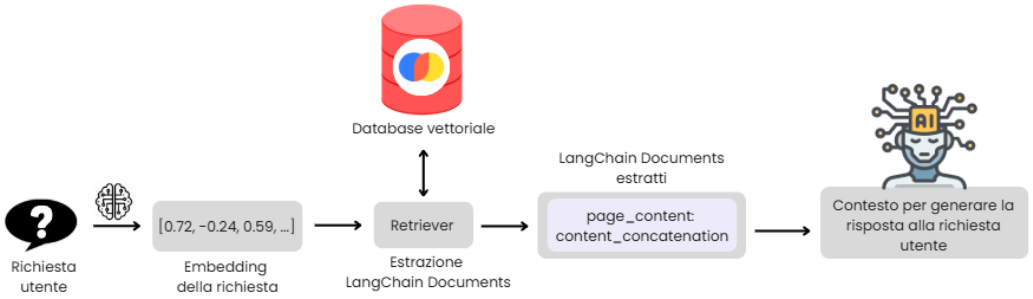

Figura 3.4: Retrieval Pipeline MRAG 1.0

individua i segmenti di documentazione più rilevanti dal punto di vista semantico, che verranno successivamente utilizzati come contesto informativo per la generazione della risposta. Le pipeline di retrieval sono state configurate per estrarre, per ogni query dell'utente, i tre frammenti di documentazione semanticamente più simili. Da un lato, questa scelta consente di fornire all'agente un contesto informativo sufficientemente ampio per gestire anche richieste complesse in scenari applicativi reali. Dall'altro, evita di introdurre un numero eccessivo di segmenti nel contesto, aspetto particolarmente rilevante considerando che i contenuti recuperati vengono successivamente integrati in un prompt tramite un meccanismo di fusione tra query e documenti recuperati.
Affinché il meccanismo di retrieval funzioni correttamente, è necessario eseguire l'encoding della query utilizzando lo stesso modello di embedding impiegato durante la fase di ingestion. Questo requisito nasce dal fatto che modelli di embedding diversi producono spazi vettoriali differenti. Di conseguenza due vettori generati da modelli distinti non sono direttamente confrontabili tramite similarity search, poiché le direzioni e le distanze nello spazio vettoriale non hanno lo stesso significato. Per questa ragione, a seconda del modello utilizzato per la costruzione del database vettoriale, la query dell'utente viene codificata tramite text-embedding-3-small oppure nomic-embed-text.
La similarità tra il vettore della query e quelli dei documenti viene calcolata tramite cosine similarity, una metrica particolarmente efficace per questo tipo di operazioni [48]. La cosine similarity misura l'allineamento direzionale tra due vettori ignorandone la magnitudine, valutando quindi il grado di somiglianza semantica tra i contenuti della query e quelli dei documenti. Questa proprietà risulta particolarmente vantaggiosa nel contesto del Question Answering, dove le query degli utenti sono tipicamente molto più brevi rispetto ai chunks estratti dalla knowledge base. Poichè la magnitudine di un vettore dipende dalla quantità di contenuto codificato, query e documenti producono naturalmente vettori di grandezze molto

diverse. Metriche basate sulla distanza assoluta, come la distanza euclidea, risulterebbero penalizzanti in questo scenario, in quanto verrebbero influenzate da questa asimmetria indipendentemente dalla reale affinità semantica tra i contenuti. La cosine similarity supera questo limite concentrandosi esclusivamente sulla direzione dei vettori, che codifica il significato semantico, garantendo così un confronto equo tra testi di lunghezza diversa. In termini geometrici, questa metrica confronta l'angolo tra il vettore della query e quello dei documenti, permettendo di identificare con precisione i segmenti più rilevanti da fornire all'agente per la generazione della risposta. Di seguito viene riportata la formulazione matematica della cosine similarity:

$$\cos(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \cdot \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{3.1}$$

Dove:

- A_i e B_i sono rispettivamente le componenti dei vettori A e B all'indice i.
- n è la dimensionalità dei vettori, ossia il loro numero di componenti.
- Il numeratore rappresenta il prodotto scalare dei vettori A e B.
- I denominatori rappresentano rispettivamente le magnitudini dei vettori A e B.

Grazie all'integrazione con Chroma, supportata dall'ecosistema LangChain, è stato possibile implementare un retriever che astrae le operazioni di accesso, ricerca ed estrazione dei documenti più pertinenti dalla knowledge base vettoriale. I documenti recuperati tramite questo componente vengono utilizzati dall'agente come contesto informativo, costituendo la base di conoscenza necessaria per generare risposte fondate sui contenuti della documentazione ufficiale del BPMS.
Nel caso della pipeline di retrieval MRAG 2.0, l'agente può utilizzare per la generazione delle risposte l'intero contenuto originale della documentazione, memorizzato nella proprietà *metadata* del Document recuperato. Per quanto riguarda invece la pipeline MRAG 1.0, l'agente genera le risposte utilizzando il contenuto presente nella proprietà *page_content* del Document estratto, che corrisponde alla concatenazione testuale degli elementi appartenenti al segmento semantico.

3.4 L'Assistente Conversazionale

Dopo l'implementazione del meccanismo di Multimodal RAG e della knowledge base vettoriale per la realizzazione di un sistema in grado di generare risposte pertinenti e affidabili, si è passati allo sviluppo dell'agente conversazionale in cui

integrare il modulo di recupero.
Nel corso della sperimentazione sono stati implementati due prototipi dell'assistente AI, che condividono la struttura generale del flusso d'esecuzione ma si differenziano nel meccanismo di filtraggio dei documenti estratti in fase di retrieval, con l'obiettivo di individuare la configurazione più performante per il caso d'uso analizzato.
Le soluzioni presentate rientrano nella categoria dei sistemi single-agent, in quanto pur essendo composte da molteplici nodi che prevedono l'utilizzo di un LLM, il comportamento complessivo rimane unitario e coerente. Questa scelta architetturale ad alto livello è stata adottata perchè le architetture multi-agent risultano particolarmente adatte a scenari caratterizzati da task eterogenei, competenze differenziate o requisiti di pianificazione complessi.
Per il caso d'uso d'interesse relativo alla realizzazione di un agente in grado di supportare l'utenza dell'applicativo di BPM, l'impiego di un sistema multi-agente sarebbe risultato poco appropriato. Tali sistemi, infatti, si compongono di entità distinte con finalità e strategie proprie, un presupposto che per il contesto applicativo avrebbe rappresentato una forzatura e introdotto un livello di complessità non necessario legato alla gestione di meccanismi di coordinamento e cooperazione tra agenti. L'adozione di un agente singolo riduce l'overhead computazionale e la complessità del sistema, facilitandone al contempo il controllo.
I prototipi implementati adottano un workflow agentico realizzato tramite l'orchestrazione graph-based di LangGraph. In termini generali, ciascun agente riceve un input, lo interpreta attraverso decisioni interne e produce un output selezionando le azioni più appropriate tra quelle previste dal flusso logico. Le soluzioni si distinguono per la strategia di filtraggio dei documenti estratti in fase di retrieval: il primo prototipo adotta un nodo di grading LLM-based, mentre il secondo incorpora il filtraggio direttamente nel nodo di retrieval tramite una soglia di similarità coseno.

3.4.1 Prototipo 1

Il primo prototipo di assistente conversazionale presenta il grafo d'esecuzione riportato in Figura 3.5.
Il primo nodo che viene eseguito dal sistema alla ricezione di una query da parte dell'utente è il $message\_classifier$. Questo primo step esecutivo ha il compito di analizzare la richiesta in ingresso e classificarla in modo da selezionare il flusso d'esecuzione più adatto a gestirla correttamente. All'attivazione del nodo viene aggiornato lo stato centralizzato del sistema. Nel caso in cui il messaggio ricevuto in input dia avvio a una nuova conversazione vengono inizializzate le variabili di stato, altrimenti vengono resettate tutte le variabili relative a interazioni precedenti, preservando soltanto la lista dei messaggi già presenti nella conversazione.
Successivamente nel nodo avviene la costruzione di un prompt di classificazione composto da un messaggio di sistema e dal messaggio umano contenente la richiesta

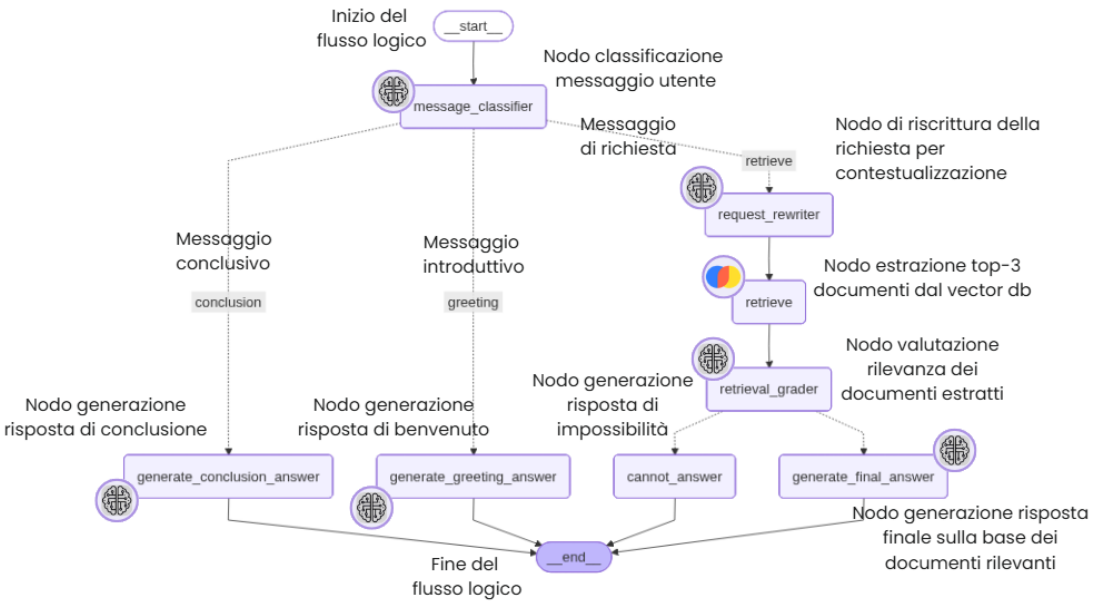

Figura 3.5: Assistente conversazionale prototipo 1

dell'utente. Il messaggio di sistema istruisce il modello a comportarsi come un classificatore rigido, che ha il compito di etichettare in maniera esclusiva il messaggio in ingresso secondo tre possibili casistiche:

- Greeting: se il messaggio consiste in un semplice un saluto o un'introduzione alla conversazione senza alcuna richiesta particolare.
- Conclusion: se l'utente sta chiudendo la conversazione o dichiara di essere soddisfatto rispetto alle proprie necessità.
- Retrieve: in tutti gli altri casi, ossia quando c'è una richiesta effettiva da elaborare.

Il nodo inizializza poi un modello LLM configurato per fornire un output strutturato secondo la classificazione appena presentata. Il prompt viene quindi fornito al modello generativo e il risultato, ovvero la classificazione del messaggio, viene memorizzato nello stato globale del grafo, in modo da poter essere impiegato attraverso un'apposita funzione di routing per stabilire il prossimo step del processo.
Nel caso in cui il $message\_classifier$ classifichi il messaggio come greeting, il workflow d'esecuzione procede con il nodo $generate\_greeting\_answer$, responsabile della generazione di una risposta di benvenuto introduttiva alla conversazione. Per fare ciò viene definito un messaggio di sistema che istruisce il modello a rispondere in maniera cordiale al messaggio fornito dall'utente, esprimendo disponibilità nel

supporto alla risoluzione di eventuali necessità relative alle funzionalità del BPMS amministrativo e al modello di processo attualizzato. Il nodo costruisce poi un prompt di generazione che combina il messaggio di sistema con il messaggio dell'utente e invoca un LLM per la generazione della risposta introduttiva da fornire all'utente. La risposta prodotta aggiorna lo stato conversazionale del sistema, registrando un messaggio generato in modo da tracciare l'evoluzione dell'interazione. A questo punto la risposta viene restituita all'utente e il flusso esecutivo si interrompe nell'attesa di nuovi messaggi.
Nel caso in cui invece il nodo $message\_classifier$ etichetti l'input dell'utente come conclusion, viene eseguito il nodo $generate\_conclusion\_answer$, delegato alla generazione di un messaggio conclusivo coerente con quello introdotto nel sistema dall'utente. Come nel caso precedente si ha la creazione di un prompt dato dall'unione di un messaggio di sistema per generare una risposta che sottolinei la disponibilità a rimanere di supporto per eventuali dubbi futuri riguardo l'applicativo e il messaggio utente. Analogamente a quanto avveniva per i messaggi di greeting, la risposta generata dall'invocazione del modello generativo viene registrata nello stato globale del grafo d'esecuzione e fornita all'utente concludendo l'elaborazione.
Infine, nel caso in cui il messaggio venga interpretato dal $message\_classifier$ come una vera richiesta di supporto, l'esecuzione procede con il nodo $request\_rewriter$, che si occupa di riformulare la richiesta utente rispetto alla conversazione pregressa in modo che diventi chiara e comprensibile anche se letta fuori dal contesto, preservandone il senso originale. A tale scopo il nodo di rewriting costruisce un prompt composto da più elementi da fornire a un LLM per la riformulazione. In primo luogo, un messaggio di sistema che istruisce il modello a riscrivere la domanda in modo da essere comprensibile autonomamente dal contesto e rendendola più chiara, pur mantenendo il significato originale. In secondo luogo, lo storico della conversazione in modo che il modello generativo possa tenere conto del contesto che ha portato alla richiesta nella sua riscrittura. Infine, viene inclusa la richiesta dell'utente da rielaborare. Una volta costruito il prompt, esso viene fornito a un LLM che produce una versione ottimizzata della richiesta e la registra nello stato globale del sistema. Nel caso in cui invece non vi siano messaggi nello storico della conversazione, il nodo semplicemente copia la richiesta originale nello stato senza apportare alcuna rielaborazione.
Questo meccanismo di riformulazione è fondamentale per gestire richieste di follow-up da parte dell'utente, ossia quelle richieste semanticamente legate a messaggi scambiati precedentemente all'interno della conversazione. Senza una visione dei messaggi scambiati in precedenza, l'agente non sarebbe in grado di interpretare correttamente la richiesta, determinando così risposte più imprecise o addirittura impossibilità di risposta. Inoltre, il rewriting permette di rendere le richieste fornite in input al sistema più precise e chiare facilitando in questo modo il processo d'estrazione di informazioni rilevanti dalla knowledge base vettoriale, e aumentando

infine la qualità delle risposte generate dal sistema.
In seguito al nodo di riscrittura, il flusso esecutivo procede con il nodo di retrieve, il cui scopo consiste nell'individuare e nell'estrarre dalla knowledge base vettoriale i contenuti più rilevanti rispetto alla necessità dell'utente in modo da poter usare le informazioni recuperate per la generazione della risposta finale. A tal proposito, il nodo può impiegare sia il meccanismo di retrieval MRAG 1.0 che quello MRAG 2.0 presentati nella Sezione 3.3.2. I due meccanismi come discusso in precedenza determinano come vengono forniti i contenuti multimodali della documentazione del BPMS amministrativo per la generazione delle risposte finali: nel primo caso descrizioni visive totalmente testuali, nel secondo caso immagini vere e proprie. I documenti estratti attraverso il nodo di retrieval vengono memorizzati nello stato globale dell'agente, in modo da poter essere impiegati negli step d'esecuzione successivi.
Al fine di aumentare il livello di affidabilità e correttezza delle risposte prodotte dall'assistente conversazionale, a seguito del nodo di retrieve il grafo d'esecuzione procede con il nodo *retrieval_grader*. Questo nodo ha il compito di valutare la rilevanza dei LangChain Document estratti nella fase di retrieval rispetto alla richiesta dell'utente, selezionando quali impiegare per la generazione della risposta finale e quali scartare. In questo step viene definito un messaggio di sistema che istruisce il modello linguistico a comportarsi come un valutatore, classificando i risultati di retrieval come rilevanti o irrilevanti. Il nodo crea per ciascun elemento estratto dal retriever un prompt che combina direttive di sistema, query utente e il contenuto da valutare. Questo prompt viene poi fornito a un LLM configurato per la restituzione di un output strutturato che stabilisce se scartare il segmento estratto o tenerlo in considerazione. L'introduzione di questo nodo mira a migliorare la qualità delle risposte generate dall'agente. Il retriever, infatti, è configurato per restituire sempre tre risultati, tuttavia, non è garantito che ciascuno di essi sia effettivamente pertinente rispetto alla richiesta dell'utente. Il nodo di grading consente quindi di filtrare i documenti non rilevanti, riducendo il contesto utilizzato nella fase di generazione della risposta finale ed evitando che informazioni non pertinenti possano comprometterne la qualità. Al termine della fase di valutazione, i Document giudicati come significativi vengono memorizzati nello stato globale del sistema in modo da essere impiegati nelle ultime fasi previste dal grafo d'esecuzione.
Nel caso in cui a seguito del grading dei risultati di retrieval non vi siano contenuti rilevanti, attraverso una funzione di routing, il flusso d'esecuzione prosegue con il nodo *cannot_answer*. Questo step viene eseguito quando il sistema non dispone di informazioni sufficienti per rispondere in maniera affidabile alla richiesta dell'utente. Tale condizione può verificarsi quando il retriever non risulta in grado di restituire risultati rilevanti o quando la richiesta è estranea al dominio applicativo del sistema. Dal punto di vista pratico questo passaggio permette all'assistente di restituire un messaggio predefinito all'utente che lo informa di non poter rispondere alla

richiesta presentata per mancanza di informazioni o perchè potrebbe non riguardare il BPMS amministrativo, invitando l'utilizzatore a riformulare o chiarire la propria necessità. Senza un meccanismo di fallback come questo, l'agente rischierebbe di generare risposte imprecise o fuorvianti, compromettendo l'affidabilità complessiva del sistema di QA. Inoltre, questo meccanismo permette di bloccare i tentativi di utilizzo improprio dell'assistente AI, impedendo la generazione di risposte per richieste al di fuori del dominio di riferimento. Come per tutti i nodi anche in questo caso la risposta di impossibilità viene aggiunta allo storico conversazionale.
A seguito del processo di filtraggio dei contenuti estratti, qualora almeno uno di essi venga valutato come rilevante, la funzione di routing del nodo *retrieval_grader* indirizza il flusso verso il nodo *generate_final_answer*. Quest'ultimo è responsabile della generazione della risposta finale da presentare all'utente, utilizzando le informazioni valutate come rilevanti. Il nodo *generate_final_answer* costituisce l'ultimo stadio del grafo d'esecuzione, in cui confluiscono e vengono integrate tutte le elaborazioni svolte nei passaggi precedenti, con l'obiettivo di produrre una risposta completa, coerente e allineata alla richiesta dell'utente. Anche in questo step viene costruito un prompt di generazione specifico, combinando: la richiesta dell'utente riformulata, le informazioni validate provenienti dalla documentazione del BPMS amministrativo su cui basare la risposta e le istruzioni operative da seguire per svolgere la generazione.
A seconda del meccanismo di retrieval utilizzato dal prototipo tra quelli presentati nella Sezione 3.3.2, varia la natura dei contenuti integrati all'interno del prompt. Nel caso della soluzione MRAG 2.0, il prompt include esclusivamente i contenuti originali del chunk associato all'oggetto Document estratto, presenti nel campo *metadata* di quest'ultimo. In particolare, vengono integrati il testo e le tabelle come html, mentre le immagini vengono trasferite al modello come base64, ma decodificate e interpretate come immagini vere e proprie. Nel caso della soluzione MRAG 1.0, invece, il prompt contiene la concatenazione testuale dei contenuti del chunk di riferimento, presente nel campo *page_content* del Document estratto. In questa variante, eventuali immagini sono gestite come descrizioni testuali durante il processo di ingestion. Una volta costruito il prompt, il nodo utilizza un LLM per generare la risposta finale. L'output prodotto viene inserito nello stato globale dell'assistente conversazionale e restituito all'utente, completando così il flusso di esecuzione.

3.4.2 Prototipo 2

Oltre alla prima soluzione di assistente AI appena presentata, è stato sviluppato un secondo prototipo con l'obiettivo di ridurre la latenza complessiva del sistema. In Figura 3.6 è riportato il grafo di esecuzione di questa variante.

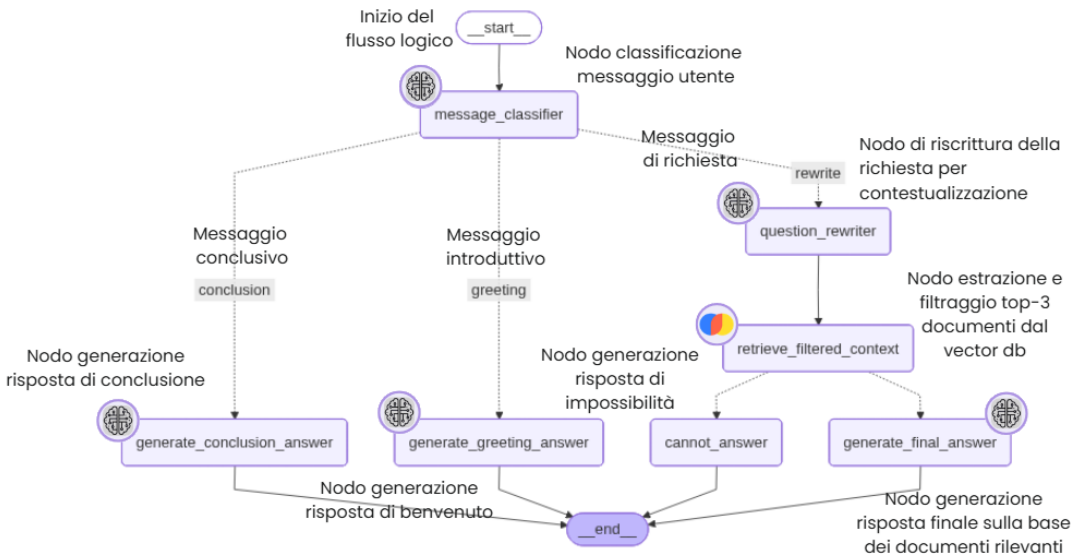

Figura 3.6: Assistente conversazionale prototipo 2

Questa variante del sistema conversazionale mantiene lo stesso funzionamento di base della soluzione precedente, ma si differenzia per il meccanismo di filtraggio utilizzato per valutare i contenuti recuperati dalla knowledge base vettoriale attraverso il nodo di retrieve.
Il nodo *retrieval_grader* che nel prototipo 1 applicava un filtraggio dei contenuti irrilevanti basato su LLM viene rimosso. In questa seconda configurazione, il meccanismo di filtraggio dei contenuti viene incorporato direttamente all'interno del nodo di retrieval. Il nodo *retrieve_filtered_context* esegue l'estrazione dei documenti attraverso similarity search, analogamente alla soluzione precedente. Tuttavia, in questa variante, i documenti recuperati vengono filtrati direttamente sulla base del punteggio di similarità associato a ciascuna istanza. È stata definita una soglia di accettazione pari a 0.5, al di sotto della quale il documento viene escluso dall'insieme dei contenuti rilevanti e, conseguentemente, trascurato nella fase di generazione della risposta finale. Questa scelta progettuale consente di ridurre il tempo di risposta dell'agente, eliminando la necessità di una chiamata aggiuntiva a un LLM e rendendo superfluo il nodo *retrieval_grader*. Tale approccio risulta vantaggioso anche dal punto di vista economico, in contesti in cui si adottano modelli a consumo, come nel caso di GPT-5-nano di OpenAI.

3.5 Integrazione e Infrastruttura

In seguito alla fase di prototipazione, la sperimentazione si è focalizzata sull'integrazione dell'assistente AI con l'applicativo di BPM preesistente, con l'obiettivo di renderne possibile l'utilizzo in un contesto operativo reale. A tal fine è stato sviluppato un layer di API, realizzato tramite il framework FastAPI [42], che consente ai sistemi esterni di comunicare con l'agente e di accedere alle sue funzionalità.
FastAPI è un framework moderno per lo sviluppo di API web in Python, progettato per essere semplice da utilizzare, ma allo stesso tempo estremamente performante. Grazie alla sua architettura basata su Starlette e Pydantic, FastAPI permette di creare API veloci, con validazione automatica dei dati. Inoltre, il framework consente di generare in automatico una documentazione interattiva delle API, accessibile via browser. Queste caratteristiche rendono FastAPI particolarmente adatto a progetti legati all'intelligenza artificiale dove sono necessarie API rapide, ben documentate e affidabili.
L'endpoint principale esposto dall'agente è POST /query, che permette all'utente finale di interrogare il sistema. Ogni richiesta include il messaggio dell'utente e un identificativo di conversazione, utile per gestire il contesto conversazionale, sessioni parallele e la tracciabilità delle interazioni.
Per semplificare la distribuzione e garantire l'isolamento dall'ambiente di sviluppo, il progetto è stato containerizzato utilizzando Docker [49], creando un'immagine eseguibile definita attraverso Dockerfile. Docker permette di rendere il sistema agentico portabile, coerente, isolato e facilmente scalabile, racchiudendo codice, dipendenze e configurazioni in un'immagine che può essere eseguita come container, semplificando sia lo sviluppo sia il deployment in ambienti diversi.
L'integrazione dell'assistente AI con l'applicativo di BPM è stata realizzata mediante la riconfigurazione di un cluster Kubernetes [50] preesistente, gestito tramite minikube. Il cluster è ospitato su una macchina appartenente alla stessa rete della macchina d'esecuzione del BPMS. Kubernetes è una piattaforma open source progettata per l'orchestrazione e la gestione automatizzata di container applicativi in ambienti distribuiti. Essa consente di definire, distribuire e scalare applicazioni containerizzate su un insieme eterogeneo di macchine fisiche o virtuali, garantendo al contempo alta disponibilità, tolleranza ai guasti e bilanciamento del carico. La riconfigurazione del cluster ha permesso il deployment del container relativo all'immagine dell'agente all'interno di un Pod del cluster, rendendolo così raggiungibile per la macchina del BPMS.
Lo schema architetturale del sistema risultante è raffigurato nella Figura 3.7.
L'interfaccia dell'applicativo è accessibile agli utenti autenticati tramite browser web, digitandone l'indirizzo. Attraverso questa interfaccia, l'utente finale può contattare l'assistente AI inviando una richiesta HTTPS di tipo POST per ottenere supporto in tempo reale. Le richieste vengono inizialmente gestite dal backend

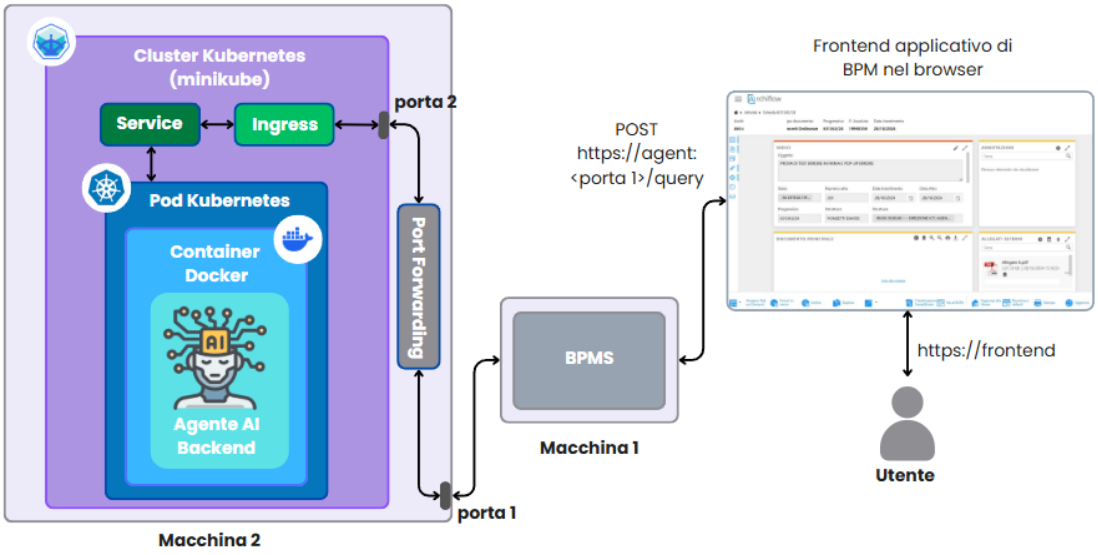

Figura 3.7: Architettura infrastrutturale

dell'applicativo di BPM in esecuzione sulla macchina 1, che si occupa di elaborarle e inoltrarle verso la macchina 2, dove è ospitato il cluster Kubernetes contenente il Pod dell'agente AI. Per rendere il servizio accessibile dall'esterno del cluster, viene utilizzato un componente di port forwarding, che collega una porta esposta sulla macchina 2 con la porta interna del cluster su cui è esposto il servizio dell'agente AI. In questo modo, le richieste provenienti dalla macchina 1 possono raggiungere correttamente il cluster Kubernetes. All'interno del cluster, un Ingress Controller gestisce il routing delle richieste HTTP. L'Ingress analizza l'URL della richiesta e la indirizza verso il Kubernetes Service appropriato. Il Service, a sua volta, inoltra il traffico al Pod che esegue il container Docker contenente l'assistente AI. Grazie a questo meccanismo, le richieste degli utenti vengono elaborate dal backend sviluppato con FastAPI in esecuzione nel container dell'agente AI. Le risposte generate vengono quindi restituite al backend dell'applicazione amministrativa e infine mostrate all'utente tramite l'interfaccia web.

3.6 Interfaccia Utente

Per rendere utilizzabile l'assistente conversazionale agli utenti finali dell'applicativo amministrativo è stato necessario modificare il modulo frontend del sistema, realizzando un'interfaccia dedicata alla conversazione con l'assistente di supporto, riportata in Figura 3.8.

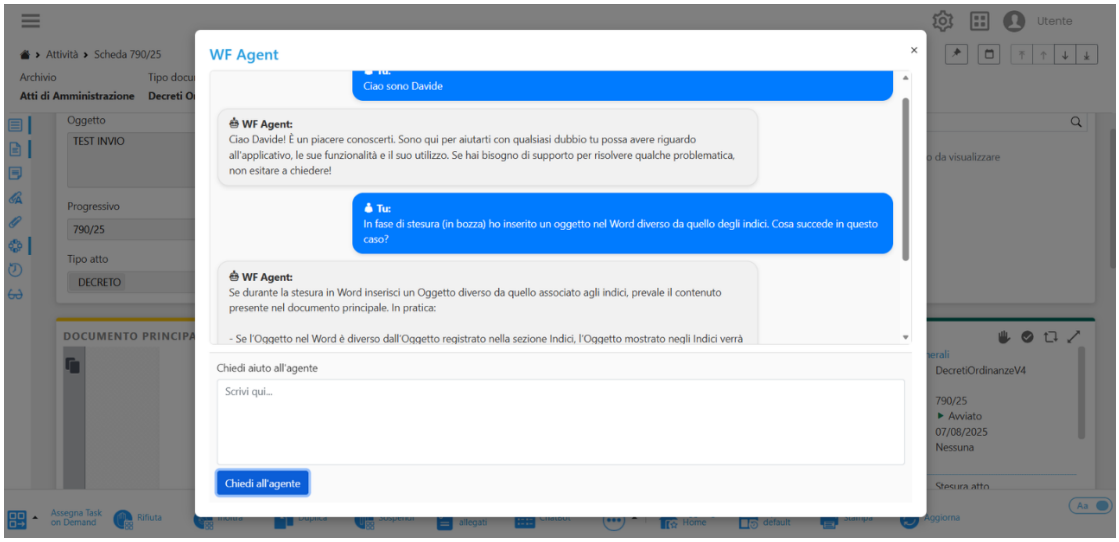

Figura 3.8: Interfaccia utente

L'interfaccia si apre tramite la selezione di un apposito pulsante, apparendo come finestra modale. All'utente è consentito digitare la propria richiesta nell'apposito campo di testo e inviarla premendo il pulsante "Chiedi all'agente". Il messaggio viene visualizzato nella finestra di dialogo sopra il campo di scrittura, permettendo di seguire facilmente lo svolgersi dell'interazione. La richiesta dell'utente viene elaborata dall'agente AI tramite l'esecuzione del relativo grafo logico, e la risposta generata viene restituita all'interno della stessa finestra conversazionale. I messaggi scambiati tra utente e assistente vengono visualizzati con colori differenti in modo da poter esser distinti rapidamente, mentre la finestra di dialogo consente di scorrere l'intera conversazione, mantenendo traccia dell'evoluzione della conversazione.
L'interfaccia si apre tramite la selezione di un apposito pulsante, apparendo come una finestra modale. L'utente può digitare la propria richiesta nell'apposito campo di testo e inviarla premendo il pulsante "Chiedi all'agente". Il messaggio viene visualizzato nella finestra di dialogo sopra il campo di scrittura, permettendo di seguire facilmente l'interazione.
La richiesta dell'utente viene elaborata dall'agente AI tramite l'esecuzione del relativo grafo logico, e la risposta generata viene restituita all'interno della stessa finestra conversazionale. I messaggi scambiati tra utente e assistente sono visualizzati con colori differenti per distinguerli rapidamente, mentre la finestra di dialogo consente di scorrere l'intera conversazione, mantenendo traccia dell'evoluzione del dialogo in modo chiaro e intuitivo.

Capitolo 4

Risultati Sperimentali

4.1 Valutazione di un Assistente Conversazionale basato su Retrieval-Augmented Generation

I sistemi di Question Answering (QA) rappresentano un'area di forte interesse nell'ambito del Natural Language Processing (NLP), in quanto rendono possibile lo svolgimento di interazioni più naturali tra esseri umani e macchine [51]. Tuttavia la valutazione di questi sistemi, si rivela un compito particolarmente complesso, principalmente a causa della natura intrinseca del linguaggio naturale, che è ricco di ambiguità lessicali, semantiche e sintattiche, oltre che di sinonimi. Ciò consente di esprimere lo stesso concetto in una molteplicità quasi illimitata di modi. Inoltre, l'output di un sistema di QA consiste in un oggetto testuale strutturato (una frase o un paragrafo) e non un semplice valore numerico, per questo motivo la valutazione della sua qualità, in termini di fluidità, correttezza e coerenza rappresenta una sfida tecnica significativa.
Nel corso del tempo sono state proposte molte metriche di valutazione, che lo studio presentato da Farea et al. classifica in due categorie principali: HCES (Human-Centric Evaluation Scores) e AES (Automatic Evaluation Scores).
Le HCES rappresentano la metodologia considerata più affidabile, poiché si basano su valutazioni dirette effettuate da esperti o specialisti del dominio. In questo approccio, gli output del sistema di QA vengono giudicati secondo criteri predefiniti, come adeguatezza, coerenza e fluidità del testo. Le valutazioni HCES possono inoltre assumere due forme principali. Nel caso della valutazione assoluta, i valutatori assegnano un punteggio su una scala agli output del sistema di QA sulla base di criteri prestabiliti. I punteggi vengono poi mediati per ottenere una misura complessiva. Nella valutazione relativa, invece, i valutatori non attribuiscono punteggi numerici. Essi ordinano più output in base alla loro qualità complessiva. Lo score finale viene quindi ricavato a partire dal ranking prodotto dai valutatori.

Nonostante la sua affidabilità, l'approccio HCES presenta alcuni limiti pratici. È infatti costoso, richiede tempo e competenze specialistiche, risultando quindi inadatto a task su larga scala o a scenari dinamici. Inoltre, questa tipologia di valutazioni può risentire di bias soggettivi e di una certa variabilità tra i diversi valutatori.
Le AES rappresentano metodi di valutazione automatici che sono generalmente più economici e scalabili. Non richiedono l'intervento diretto di esperti e risultano quindi particolarmente adatti a contesti con grandi quantità di dati o con comportamenti dinamici dei sistemi. In molti casi, i punteggi ottenuti con queste metriche mostrano una buona correlazione con i giudizi umani. Le AES possono essere ulteriormente suddivise in due categorie principali. La Context-Free Evaluation non considera il contesto in cui l'output è stato generato. Valuta esclusivamente la corrispondenza tra output e riferimento, cioè il ground truth, utilizzando ad esempio sequenze di parole, n-gram o altre unità testuali. Al contrario, la Context-Based Evaluation tiene conto del contesto specifico in cui l'output viene prodotto. Può considerare elementi come l'intera conversazione, la trama di un testo o l'intento della domanda. Questo permette una valutazione più completa e contestualizzata.
Il sistema proposto in questo progetto di tesi non consiste in un tradizionale sistema di QA, ma bensì in un assistente conversazionale che utilizza un meccanismo di RAG per rispondere alle richieste dell'utente. La valutazione di sistemi ibridi di questo tipo presenta sfide particolari, dovute alla loro architettura complessa, che integra sia componenti di recupero informativo sia moduli di generazione del testo. Per analizzare un sistema basato su RAG è quindi necessario valutare le prestazioni dei moduli recupero e di generazione di cui si compone e, allo stesso tempo, le performance complessive del sistema. A tal proposito lo studio condotto da Yu et al. introduce il framework Auepora (A Unified Evaluation Process of RAG) che identifica obiettivi di valutazione specifici per i componenti di retrieval e di generazione, oltre a requisiti critici per il sistema nel suo insieme.

4.1.1 Valutazione del Modulo di Retrieval

Il componente di retrieval rappresenta un elemento fondamentale nei sistemi RAG, in quanto si occupa di estrarre informazioni rilevanti da fonti esterne che alimenteranno la fase di generazione, influenzandone direttamente la qualità. Le principali sfide di valutazione relative a questo componente riguardano la natura dinamica e la vastità delle basi di conoscenza, che possono spaziare da database strutturati fino all'intero web.
Secondo il framework Auepora, la valutazione del modulo di retrieval deve concentrarsi su due aspetti principali: rilevanza e accuratezza. La rilevanza misura la pertinenza dei documenti estratti rispetto alla query dell'utente. L'accuratezza

invece, valuta la capacità del sistema di distinguere tra documenti realmente rilevanti e quelli meno significativi o irrilevanti.
Per la valutazione del modulo di retrieval è disponibile un ampio insieme di metriche, che possono essere suddivise in due categorie principali: le non-rank based metrics e le rank-based metrics.

Metriche Non-Rank Based

Questa tipologia di metriche valuta la pertinenza complessiva degli elementi recuperati dal modulo di retrieval, misurando la capacità del sistema di individuare documenti rilevanti senza tenere conto del ranking con cui vengono presentati nei risultati. Le principali metriche appartenenti a questa categoria sono:

- Accuracy: Rappresenta la proporzione di risultati correttamente classificati, includendo sia i veri positivi sia i veri negativi, rispetto al totale dei casi esaminati. L'accuratezza è definita come segue:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

  Dove TP (True Positives) indica il numero di documenti rilevanti correttamente recuperati, TN (True Negatives) il numero di documenti non rilevanti correttamente scartati, FP (False Positives) il numero di documenti non rilevanti erroneamente recuperati e FN (False Negatives) il numero di documenti rilevanti non recuperati.

- Precision: Misura la frazione di documenti pertinenti tra tutti quelli effettivamente recuperati dal sistema. Essa fornisce un'indicazione della capacità del modulo di retrieval di evitare risultati irrilevanti ed è formalmente definita come il rapporto tra i veri positivi (TP) e la somma di veri positivi e falsi positivi ($TP + FP$):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

  Un valore elevato di Precision indica che la maggior parte dei documenti recuperati è effettivamente rilevante rispetto alla query dell'utente.

- Recall@k: Indica la frazione di documenti rilevanti che vengono recuperati dal sistema rispetto al totale dei documenti pertinenti disponibili, considerando esclusivamente i primi k risultati restituiti. Il richiamo è definito come:

$$\text{Recall@k} = \frac{|RD \cap \text{Top}_{kd}|}{|RD|} \tag{4.3}$$

  Dove RD rappresenta l'insieme dei documenti rilevanti, mentre Top_{kd} indica l'insieme dei primi k documenti recuperati dal sistema.

Questa metrica è particolarmente utile nei contesti in cui solo un numero limitato di documenti viene utilizzato nella fase successiva di generazione.

Sebbene queste metriche siano fondamentali per valutare la capacità di base di un sistema di retrieval nel reperire informazioni pertinenti, esse non sono sufficienti a descrivere completamente l'efficacia del modulo di recupero. In tali sistemi, infatti, l'ordine con cui i documenti vengono recuperati è un fattore determinante, poiché influisce direttamente sulla fase successiva di generazione della risposta per via del fenomeno noto come "lost in the middle" [46]. Per questo motivo, risulta necessario affiancare a tali metriche indicatori che tengano conto del ranking dei risultati.

Metriche Rank-Based

Questa classe di indicatori sono progettati per valutare la qualità di un sistema di retrieval tenendo conto dell'ordine con cui i documenti rilevanti vengono restituiti. A differenza delle metriche non-rank based, queste attribuiscono un'importanza maggiore al posizionamento dei documenti pertinenti nelle prime posizioni della lista dei risultati. Le principali metriche appartenenti a questa categoria sono:

- Mean Reciprocal Rank (MRR): Misura la capacità del sistema di posizionare il primo documento rilevante il più in alto possibile nella lista dei risultati. Il calcolo dell'MRR avviene considerando, per ciascuna query, la posizione del primo documento rilevante recuperato dal sistema. A tale posizione viene associato il suo reciproco, ottenendo un valore compreso tra 0 e 1. Il punteggio finale è ottenuto come media dei reciproci dei ranghi calcolati su tutte le query del dataset di valutazione. Formalmente, l'MRR è definito come:

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \tag{4.4}$$

  dove $|Q|$ è il numero totale di query, mentre $rank_i$ rappresenta la posizione del primo documento rilevante per la i-esima query.
  Il valore dell'MRR è compreso tra 0 e 1, con valori più elevati che indicano una maggiore efficacia del sistema nel fornire rapidamente documenti pertinenti.

- Mean Average Precision (MAP): Misura l'efficacia complessiva di un sistema di retrieval nel restituire tutti i documenti rilevanti per un insieme di query, tenendo conto sia della loro presenza sia della loro posizione nella lista dei risultati.
  La MAP viene calcolata come media dell'Average Precision (AP) di tutte le query del dataset di valutazione. L'AP per una singola query è definita come:

$$\text{AP} = \frac{1}{|RD|} \sum_{k=1}^{n} P(k) \cdot \text{rel}(k) \tag{4.5}$$

dove $|RD|$ è il numero totale di documenti rilevanti per la query, n è il numero di documenti restituiti dal sistema, $P(k)$ la precisione calcolata considerando solo i primi k documenti e $rel(k)$ una funzione indicatrice che vale 1 se il documento in posizione k è rilevante, 0 altrimenti.
Di conseguenza l'MAP risulta:

$$\text{MAP} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \text{AP}_i \tag{4.6}$$

dove $|Q|$ indica il numero di query valutate.
La MAP, compresa tra 0 e 1, riflette l'efficacia del sistema di retrieval nel restituire i documenti rilevanti in posizioni elevate. Valori prossimi a 1 indicano un recupero accurato e ordinato, essenziale per la generazione di risposte coerenti in un assistente RAG, mentre valori prossimi a 0 evidenziano un recupero inefficace o un posizionamento inadeguato dei documenti rilevanti.

Le principali metriche appartenenti a questa categoria sono:

4.1.2 Valutazione del Modulo di Generazione

Per quanto riguarda il componente di generazione dell'agente conversazionale, l'obiettivo principale consiste nella produzione di risposte coerenti, accurate e contestualmente appropriate, a partire dalle informazioni fornite dal modulo di retrieval. La valutazione di tale componente riveste un ruolo centrale, poiché incide direttamente sulla qualità complessiva dell'interazione e sulla percezione di affidabilità del sistema da parte dell'utente.
Il framework Auepora [52] struttura la valutazione del modulo di generazione secondo tre dimensioni fondamentali: relevance (pertinenza), faithfulness (fedeltà) e correctness (correttezza). La relevance misura quanto la risposta generata sia effettivamente allineata all'intento informativo e al contenuto della query dell'utente. In altre parole, valuta se le informazioni fornite soddisfano la richiesta originale e rispondono correttamente al quesito formulato, senza includere elementi irrilevanti o fuorvianti. La faithfulness misura quanto la risposta generata sia coerente e supportata dalle informazioni presenti nel contesto a disposizione. In altre parole, valuta se l'output riflette correttamente i dati forniti dalle fonti, evitando affermazioni non supportate o inesatte (allucinazioni), e garantendo che il contenuto prodotto dal sistema sia affidabile rispetto alle evidenze disponibili. La correttezza misura l'accuratezza fattuale e l'appropriatezza della risposta generata confrontandola con una o più risposte di riferimento. In altre parole, valuta se le informazioni fornite sono corrette, coerenti con i dati attesi e appropriate rispetto al contesto della query, assicurando che il sistema non introduca errori o imprecisioni.
Per la valutazione di tali aspetti in letteratura sono state proposte diverse metodologie.

Metriche Tradizionali

Questa tipologia di metriche si basano sulla sovrapposizione testuale tra la risposta generata e una risposta di riferimento. Tali indicatori misurano il grado di similarità tra i due testi considerando la corrispondenza di parole o sequenze di parole. Tra gli indicatori più diffusi di questa classe vi sono:

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [53]: ROUGE è un insieme di metriche automatiche progettato per valutare la qualità dei testi generati in maniera automatica, confrontandoli con un testo di riferimento prodotto da un essere umano. Il principio alla base di ROUGE consiste nel misurare la sovrapposizione di unità testuali (parole o sequenze di parole) tra la risposta generata e il riferimento.
  La valutazione delle metriche ROUGE viene effettuata attraverso Precisione, Recall e F1-score.
  Esistono svariate varianti di ROUGE, tra le principali si trovano:

  – ROUGE-N: misura il grado di sovrapposizione lessicale locale tra il testo generato e il riferimento, considerando sequenze contigue di n token. In particolare, ROUGE-1 valuta la presenza delle singole parole, mentre ROUGE-2 analizza la corrispondenza di coppie di parole consecutive, fornendo un'indicazione della correttezza terminologica e, in parte, della struttura sintattica dell'output.
  Le misure di Precisione, Recall e F1-score relative a ROUGE-N sono definite come segue:

$$\text{Recall} = \frac{\sum_{gram_n \in R} \min\left(\text{Count}_n(gram_n \in C), \text{Count}_n(gram_n \in R)\right)}{\sum_{gram_n \in R} \text{Count}_n(gram_n \in R)} \tag{4.7}$$

$$\text{Precision} = \frac{\sum_{gram_n \in R} \min\left(\text{Count}_n(gram_n \in C), \text{Count}_n(gram_n \in R)\right)}{\sum_{gram_n \in C} \text{Count}_n(gram_n \in C)} \tag{4.8}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.9}$$

  dove C rappresenta la risposta generata, R la risposta di riferimento e $Count_n(gram)$ il numero di occorrenze di un n-gram all'interno di un

testo.
Il recall indica quanto del contenuto presente nel testo di riferimento è stato correttamente recuperato nella risposta generata, mentre la precision esprime quanto del contenuto della risposta generata è effettivamente supportato dai riferimenti. L'F1-score combina entrambe le misure in un unico valore, fornendo un compromesso tra completezza e accuratezza.

  - ROUGE-L: misura la somiglianza strutturale globale tra i testi attraverso la LCS (Longest Common Subsequence), ossia la sequenza più lunga di parole condivise che compaiono nello stesso ordine relativo. Questa metrica cattura la coerenza dell'organizzazione del contenuto, risultando meno sensibile a variazioni superficiali nella formulazione.
  - ROUGE-W: estende ROUGE-L introducendo una funzione di pesatura che privilegia le sottosequenze più continue e compatte, penalizzando quelle frammentate. In questo modo, la metrica enfatizza la fluidità e la coesione del testo generato.
  - ROUGE-S, ROUGE-SU: si basa sugli skip-bigram, ovvero coppie di parole che mantengono lo stesso ordine relativo nei due testi pur non essendo necessariamente consecutive. Questa formulazione consente di catturare relazioni lessicali a distanza, risultando meno sensibile a variazioni locali nella struttura della frase. La variante ROUGE-SU estende tale approccio includendo anche gli unigrammi, aumentando la robustezza della metrica e riducendo il rischio di ottenere punteggi nulli, in particolare nel caso di testi di breve lunghezza.

- BLEU (Bilingual Evaluation Understudy) [54]: BLEU è una metrica di valutazione automatica originariamente proposta per la valutazione dei sistemi di traduzione automatica, con l'obiettivo di fornire una misura rapida, economica e indipendente dalla lingua della qualità dei testi generati. Il principio alla base di BLEU è che un output prodotto automaticamente sia tanto migliore quanto più risulta simile a una o più produzioni umane di riferimento. Sebbene concepita per la traduzione, la metrica è stata ampiamente adottata anche in altri ambiti del Natural Language Processing, tra cui il Question Answering.
  Il calcolo del punteggio BLEU si fonda sul confronto tra una risposta candidata e uno o più testi di riferimento attraverso tre componenti fondamentali.
  In primo luogo, BLEU valuta la corrispondenza di n-grammi, misurando la sovrapposizione tra le sequenze di parole presenti nella risposta generata e quelle contenute nei riferimenti. Gli n-grammi di ordine crescente consentono di valutare progressivamente sia l'adeguatezza del contenuto informativo sia la correttezza sintattica e la fluidità del testo generato.
  In secondo luogo, BLEU introduce il concetto di precisione n-gram modificata,

progettato per evitare che la metrica favorisca risposte che ripetono eccessivamente termini corretti. A tal fine, il conteggio di ciascun n-gramma nel testo candidato viene limitato al numero massimo di occorrenze osservato nei testi di riferimento (clipping). La precisione modificata per n-grammi di ordine n è definita come:

$$p_n = \frac{\sum_{\text{ngram} \in C} \min\left(\text{count}_C(\text{ngram}), \max_{R \in \mathcal{R}} \text{count}_R(\text{ngram})\right)}{\sum_{\text{ngram} \in C} \text{count}_C(\text{ngram})} \tag{4.10}$$

dove C rappresenta la risposta candidata generata dal sistema, R l'insieme delle risposte di riferimento, $count_C(ngram)$ il numero di occorrenze dell'n-gramma nel candidato e $count_R(ngram)$ il numero di occorrenze dell'n-gramma in un singolo riferimento. Il termine di clipping (min) impedisce che un n-gramma venga conteggiato più volte di quanto avvenga nei riferimenti.
Infine, BLEU incorpora una Brevity Penalty (BP) per compensare il fatto che una misura basata esclusivamente sulla precisione tenderebbe a favorire risposte eccessivamente brevi. La penalità riduce il punteggio quando la lunghezza della risposta candidata è significativamente inferiore a quella dei riferimenti ed è definita come:

$$BP = \begin{cases} 1 & \text{se } |C| > |R^*| \\ \exp\left(1 - \frac{|R^*|}{|C|}\right) & \text{se } |C| \leq |R^*| \end{cases} \tag{4.11}$$

dove $|C|$ indica la lunghezza della risposta candidata in token e $|R^*|$ la lunghezza del riferimento la cui dimensione è più vicina a quella del candidato.
Il punteggio BLEU finale è calcolato come la media geometrica delle precisioni modificate degli n-grammi, pesata tramite la penalità di brevità:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{4.12}$$

dove N è l'ordine massimo degli n-grammi considerati, w_n sono pesi non negativi tali che $\sum_{n=1}^{N} w_n = 1$, p_n rappresenta la precisione n-gram modificata di ordine n e BP la brevity penalty.
Il valore del punteggio BLEU è compreso tra 0 e 1, con valori più elevati che indicano una maggiore similarità tra la risposta generata e le risposte umane di riferimento.

Il principale limite della categoria di metriche tradizionali risiede nella loro natura prevalentemente lessicale, che non consente di catturare in modo adeguato la similarità semantica tra i testi. Di conseguenza, risposte che veicolano lo stesso contenuto informativo ma utilizzano formulazioni lessicali differenti possono ottenere punteggi ridotti, pur risultando semanticamente equivalenti.

Metriche Semantiche

Le metriche semantiche rappresentano un'evoluzione rispetto agli approcci di valutazione tradizionali, in quanto mirano a misurare la qualità di un testo generato in termini di somiglianza di significato rispetto a un testo di riferimento, piuttosto che basarsi esclusivamente sulla corrispondenza lessicale.
A tal fine, queste metriche sfruttano embedding contestuali ottenuti da modelli transformer pre-addestrati, che consentono di rappresentare le parole come vettori dipendenti dal contesto in cui compaiono. La valutazione non si limita quindi alla ricerca di token identici, ma si fonda sul calcolo della similarità tra rappresentazioni vettoriali, permettendo di cogliere relazioni semantiche profonde, come sinonimia e parafrasi, e di fornire una stima più aderente al giudizio umano sulla qualità del testo.
La metrica storicamente più conosciuta relativa a questa categoria di metriche è il BERTScore [55]. Si tratta di una metrica di valutazione automatica per sistemi di generazione del linguaggio naturale che misura la similarità tra una risposta generata e una o più risposte di riferimento sfruttando le rappresentazioni contestuali prodotte da modelli di tipo BERT.
In linea con l'approccio delle metriche semantiche, BERTScore valuta la similarità tra i testi a livello di significato, consentendo di catturare parafrasi e relazioni semantiche complesse e mostrando una maggiore correlazione con il giudizio umano.
Il calcolo di BERTScore si basa sul confronto tra le rappresentazioni vettoriali dei token della risposta candidata e di quelli del testo di riferimento. In una prima fase, le frasi vengono tokenizzate in word pieces e trasformate in embedding contestuali mediante un modello BERT pre-addestrato. Successivamente, viene calcolata la similarità coseno tra ogni coppia di token appartenenti ai due testi.
Dati $C = \{c_1, c_2, ..., c_n\}$ l'insieme dei token della risposta candidata e $R = \{r_1, r_2, ..., r_n\}$ l'insieme dei token della risposta di riferimento, la similarità coseno tra due token è definita come:

$$\text{sim}(c_i, r_j) = \frac{E(c_i) \cdot E(r_j)}{\|E(c_i)\| \|E(r_j)\|} \tag{4.13}$$

dove $E(c_i)$ e $E(r_j)$ rappresentano i rispettivi embedding contestuali per i token considerati.
A partire da tali valori di similarità, viene effettuato un greedy matching che associa a ciascun token della risposta candidata il token del riferimento con il massimo grado di similarità semantica:

$$match(c_i) = \max_j sim(c_i, r_j) \tag{4.14}$$

Un procedimento analogo viene applicato simmetricamente ai token del riferimento rispetto a quelli del candidato.

Sulla base di questi accoppiamenti, vengono quindi definite le misure di Precision, Recall e F1-Score, dove la precisione misura quanto i contenuti della risposta generata siano semanticamente pertinenti rispetto al riferimento, il recall valuta la copertura del contenuto del riferimento da parte della risposta generata e l'F1-Score ne fornisce una sintesi bilanciata.

$$\text{Precision} = \frac{1}{m}\sum_{i=1}^{m}\text{match}(c_i) \tag{4.15}$$

$$\text{Recall} = \frac{1}{n}\sum_{j=1}^{n}\text{match}(r_j) \tag{4.16}$$

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.17}$$

Metriche LLM-as-a-Judge

LLM-as-a-Judge è un approccio di valutazione emerso più recentemente che prevede l'impiego di Large Language Models come valutatori automatici della qualità dei testi generati. Questo paradigma consente di analizzare aspetti complessi dell'output, difficilmente catturabili da metriche puramente numeriche, quali la coerenza, la pertinenza e la fluidità linguistica. Un elemento distintivo dell'approccio è la capacità di operare anche in assenza di una risposta di riferimento esplicita, poiché il modello è in grado di impiegare la propria conoscenza parametrica. Inoltre, il giudizio espresso può essere parzialmente allineato alle preferenze umane, incorporando criteri qualitativi che riflettono le aspettative di un utente finale.
La metodologia si basa sull'istruzione del modello tramite prompt di sistema accuratamente progettati, con l'obiettivo di standardizzare il processo di valutazione e definire linee guida esplicite. Oltre all'utilizzo in questa modalità zero-shot, è possibile adottare configurazioni few-shot, fornendo esempi di valutazioni corrette, oppure ricorrere ad approcci più avanzati che prevedono il fine-tuning del modello su annotazioni umane, migliorando ulteriormente l'aderenza ai criteri di giudizio umani.
Un ulteriore vantaggio dell'approccio LLM-as-a-Judge risiede nella possibilità di condurre analisi multi-target, particolarmente rilevanti nel contesto dei sistemi Retrieval-Augmented Generation (RAG). In questo scenario, il modello può valutare simultaneamente la pertinenza tra query e risposta, la fedeltà dell'output rispetto ai documenti recuperati e, quando disponibile, la correttezza fattuale mediante il confronto con una risposta di riferimento. Questa capacità di ragionare su più dimensioni di qualità rende la valutazione più completa e contestualizzata.
Nonostante i numerosi vantaggi, l'approccio LLM-as-a-Judge presenta alcune limitazioni. L'allineamento con il giudizio umano non è sempre garantito e mancano

standard di valutazione universalmente condivisi. Inoltre, l'impiego di modelli di grandi dimensioni comporta un costo computazionale significativo, che può rappresentare un fattore critico in scenari di valutazione estesa.
Questo genere di approccio è stato implementato in framework come RAGAS (Retrieval-Augmented Generation Assessment) [56, 57], che sfruttano LLM come giudici per calcolare metriche di qualità, fornendo strumenti automatici per il monitoraggio delle prestazioni dei sistemi RAG su larga scala. RAGAS si concentra su tre dimensioni fondamentali, valutate attraverso un LLM: faithfulness, relevance e correctness.

- La Faithfulness [58] misura il grado in cui la risposta generata è effettivamente supportata dal contesto recuperato dal meccanismo di retrieval. L'obiettivo della metrica è verificare che le informazioni presenti nella risposta derivino realmente dai documenti forniti al modello, riducendo il rischio di allucinazioni. Operativamente, RAGAS scompone la risposta generata in singole affermazioni atomiche. Successivamente, un LLM viene utilizzato per valutare se ciascuna affermazione sia supportata dalle informazioni presenti nei documenti recuperati. Il punteggio di faithfulness viene quindi calcolato come il rapporto tra il numero di affermazioni supportate dal contesto e il numero totale di affermazioni presenti nella risposta, secondo la seguente formula:

$$\text{Faithfulness} = \frac{\text{Numero di affermazioni supportate}}{\text{Numero totale di affermazioni}} \tag{4.18}$$

  Il valore risultante è normalizzato nell'intervallo [0,1], dove valori più elevati indicano che una maggiore proporzione delle affermazioni presenti nella risposta è effettivamente supportata dai documenti recuperati, suggerendo quindi una maggiore affidabilità dell'output generato.

- La Relevance [59] misura quanto la risposta generata sia pertinente rispetto alla query originale dell'utente, penalizzando contenuti ridondanti o non direttamente utili alla risoluzione della richiesta. In RAGAS, questa metrica viene calcolata utilizzando un LLM che, a partire dalla risposta generata, produce un insieme di possibili domande che potrebbero essere soddisfatte da tale risposta. La pertinenza viene quindi stimata confrontando gli embedding delle domande generate con l'embedding della query originale tramite cosine similarity. Il punteggio finale è calcolato come media delle similarità ottenute, secondo la seguente formula:

$$\text{Relevance} = \frac{1}{N}\sum_{i=1}^{N}\frac{E_{g_i}\cdot E_o}{\|E_{g_i}\|\|E_o\|} \tag{4.19}$$

  Dove N rappresenta il numero di domande generate dall'LLM a partire dalla risposta prodotta, E_{g_i} indica l'embedding della i-esima domanda generata,

mentre E_o rappresenta l'embedding della query originale dell'utente. Il termine $(E_{g_i} \cdot E_o)/(\|E_{g_i}\|\|E_o\|)$ corrisponde alla cosine similarity tra i due embedding. Valori più elevati della metrica indicano una maggiore vicinanza semantica tra le domande derivate dalla risposta e la query originale, suggerendo quindi una maggiore pertinenza della risposta rispetto alla richiesta dell'utente.

- La Correctness [60] valuta l'accuratezza fattuale e l'allineamento semantico tra la risposta generata e una risposta di riferimento. In RAGAS, il punteggio viene calcolato considerando due componenti principali: la similarità fattuale e la similarità semantica. La similarità fattuale misura quanto le affermazioni presenti nella risposta generata coincidano con quelle della risposta di riferimento. Questa valutazione viene effettuata tramite un LLM, che confronta le affermazioni estratte dalla risposta generata con quelle di riferimento, identificando i fatti corretti (True Positives, TP), gli errori (False Positives, FP) e le omissioni (False Negatives, FN). Il punteggio fattuale è calcolato tramite una variante dell'F1 score:

$$\text{Correctness} = \frac{|TP|}{|TP| + 0.5 \times (|FP| + |FN|)} \tag{4.20}$$

  La similarità semantica, invece, misura quanto il significato complessivo della risposta generata sia coerente con quello del riferimento. Viene calcolata confrontando gli embeddings dei due testi mediante cosine similarity. La combinazione di entrambe le componenti, tramite una media pesata, produce un punteggio normalizzato tra 0 e 1, in cui valori più alti indicano una maggiore correttezza.

Dal punto di vista applicativo, RAGAS è facilmente integrabile nei flussi di lavoro degli sviluppatori grazie a un'API Python. Inoltre, supporta l'integrazione con framework ampiamente diffusi per la costruzione di sistemi RAG e agentici, come LangChain, rendendolo uno strumento pratico per il monitoraggio e il miglioramento delle prestazioni di tali sistemi.

Metriche Operative

Oltre alla valutazione delle componenti principali di recupero e generazione, un sistema basato su RAG deve soddisfare una serie di requisiti aggiuntivi, fondamentali per garantirne l'effettiva applicabilità in contesti reali e in scenari d'uso interattivi. Tali requisiti riguardano non solo la qualità informativa delle risposte, ma anche aspetti operativi e comportamentali del sistema nel suo complesso.
Un primo aspetto rilevante è rappresentato dalla latenza, che misura il tempo medio impiegato dal sistema per produrre una risposta a una singola query, includendo sia la fase di recupero dei documenti sia quella di generazione del testo. La latenza

costituisce un fattore critico per l'esperienza dell'utente finale, in particolare in applicazioni conversazionali e sistemi di assistenza in tempo reale, dove tempi di risposta elevati possono compromettere l'usabilità del sistema.
Un ulteriore requisito riguarda la robustezza al rumore, intesa come la capacità del sistema di gestire efficacemente informazioni irrilevanti, ridondanti o potenzialmente fuorvianti presenti nei documenti recuperati.
Infine, un aspetto cruciale per l'affidabilità di un assistente conversazionale è il rifiuto negativo, ovvero la capacità del sistema di astenersi dal fornire una risposta quando le informazioni disponibili risultano insufficienti, incomplete o eccessivamente ambigue. In tali situazioni, un comportamento prudente, che segnali l'impossibilità di rispondere in modo accurato, è preferibile rispetto alla generazione di risposte speculative o potenzialmente errate, contribuendo a ridurre il rischio di allucinazioni e a rafforzare la fiducia dell'utente nel sistema.

4.2 Metodologia di Valutazione e Test Set

L'assistente conversazionale sviluppato nel presente progetto di tesi è stato valutato secondo le linee guida del framework Auepora [52], che prevede la valutazione dei due moduli fondamentali di un sistema RAG: il modulo di retrieval, responsabile del recupero dei documenti rilevanti dalla knowledge base vettoriale, e il modulo di generazione, responsabile della produzione della risposta finale.
Nel sistema sviluppato, entrambi i moduli sono integrati all'interno dell'agente conversazionale. Questo implica che, mentre il meccanismo di retrieval può essere valutato come componente indipendente confrontando direttamente i documenti recuperati con quelli attesi, la valutazione del modulo di generazione coincide necessariamente con la valutazione dell'agente nel suo complesso. Per questa ragione, la valutazione è stata articolata in due fasi distinte. In una prima fase viene analizzato il comportamento del meccanismo di retrieval nelle diverse pipeline e configurazioni considerate. Successivamente, vengono valutate le prestazioni complessive dell'assistente conversazionale nei suoi differenti prototipi, utilizzando le metriche di valutazione definite nella Sezione 4.1. Per le metriche LLM-as-a-Judge viene utilizzato come modelli per la valutazione GPT-4o-mini [61] e il modello di embedding text-embedding-3-small di OpenAI.

4.2.1 Metodologia di Valutazione del Meccanismo di Retrieval

Il meccanismo di recupero delle informazioni è stato valutato in isolamento, confrontando direttamente gli elementi di documentazione restituiti dalle diverse pipeline

con quelli considerati rilevanti nel test set, indipendentemente dal comportamento dell'assistente conversazionale.
Il componente di retrieval è stato valutato mediante le metriche non rank-based di Precision e Recall@3, impiegate per misurare l'efficacia del sistema nell'individuare segmenti di documentazione rilevanti rispetto alle necessità esplicitate dell'utente nella richiesta fornita all'assistente conversazionale. Il calcolo del richiamo è stato eseguito tre elementi in modo da rispettare la configurazione del meccanismo di recupero, che restituisce i tre risultati semanticamente più vicini alla query utente, e per allinearsi al limite superiore di tre riferimenti rilevanti per le query di test del dataset di valutazione. Data la coincidenza tra il valore k=3 impostato per il calcolo della metrica e il numero fisso di risultati recuperati dal componente di retrieval, la metrica verrà indicata semplicemente come Recall.
A queste si affiancano le metriche rank-based di Mean Reciprocal Rank (MRR) e Mean Average Precision (MAP), che permettono di considerare anche il posizionamento degli elementi rilevanti all'interno della lista dei risultati restituiti dal meccanismo di retrieval. Infine, è stata misurata la latenza di estrazione, al fine di valutare il tempo medio necessario al sistema per completare il processo di recupero dei contenuti rilevanti.

4.2.2 Metodologia di Valutazione dell'Assistente Conversazionale

La valutazione dell'assistente conversazionale ha riguardato i prototipi nel loro complesso, analizzando sia la qualità delle risposte generate sia le prestazioni del nodo di filtraggio dei contenuti recuperati.
Per valutare la qualità delle risposte è stata adottata la metrica tradizionale ROUGE, che misura la sovrapposizione lessicale tra la risposta generata e una risposta di riferimento. A questa si affianca BERTScore, che consente di cogliere il grado di similarità semantica tra i testi confrontati, superando i limiti delle metriche basate esclusivamente sulla sovrapposizione lessicale. Sono state inoltre impiegate le metriche RAGAS di correctness, faithfulness e relevance, calcolate secondo il paradigma LLM-as-a-Judge, con l'obiettivo di ottenere una valutazione più approfondita e qualitativa delle risposte generate, tenendo conto della correttezza fattuale, della fedeltà rispetto ai documenti recuperati e della pertinenza rispetto alla query dell'utente.
Per valutare il nodo di filtraggio, le metriche Precision e Recall sono state ricalcolate confrontando i documenti effettivamente utilizzati dall'assistente per generare la risposta con quelli considerati rilevanti nel test set, evidenziando la capacità del sistema di selezionare le informazioni pertinenti e di scartare quelle non rilevanti.
Infine, è stata misurata la latenza di risposta, calcolando il tempo medio impiegato

dall'assistente per generare una risposta a partire dalla richiesta dell'utente.

4.2.3 Test Set di Valutazione

Al fine di testare il sistema sviluppato e di calcolare le metriche di valutazione selezionate, è stato costruito un test set dedicato, realizzato con il supporto di un esperto di dominio che si occupa quotidianamente dell'assistenza agli utenti dell'applicativo di BPM tramite la gestione dei ticket di supporto. Il dataset è composto da 30 istanze di test, ciascuna progettata per simulare richieste realistiche formulate dagli utenti finali e per consentire una valutazione completa delle diverse componenti del sistema. Le istanze di test presentano la seguente struttura:

- id: Identificatore dell'istanza di test, utilizzato per attività tracciamento e analisi degli errori.
- query: Domanda formulata in linguaggio naturale che riproduce una richiesta reale da parte dell'utente. Questo campo viene utilizzato come input sia per la valutazione del meccanismo di retrieval che per quella dell'assistente conversazionale.
- expected_answer: Risposta di riferimento, redatta manualmente con il supporto dell'esperto di dominio sulla base dei contenuti presenti nel manuale utente ufficiale, in modo da garantire correttezza e completezza informativa. Viene utilizzata per il calcolo delle metriche che prevedono il confronto con una ground truth, ovvero ROUGE, BERTScore e RAGAS correctness.
- relevant_docs: Lista degli identificativi dei segmenti di documentazione considerati rilevanti per rispondere correttamente alla query. Questo campo rappresenta il riferimento del processo di retrieval ed è impiegato per il calcolo di Precision, Recall@3, MRR e MAP, sia nella valutazione del meccanismo di retrieval in isolamento sia nella valutazione del nodo di filtraggio dell'assistente.
- category: Indica se la query posta all'agente è coperta dalla documentazione dell'applicativo (in_scope) oppure se rappresenta una richiesta al di fuori del dominio di competenza (out_of_scope). Questo campo viene impiegato per testare la capacità dell'assistente di gestire richieste fuori contesto.

Il calcolo delle metriche di valutazione è stato automatizzato mediante lo sviluppo di due script Python separati, dedicati rispettivamente alla valutazione del modulo di retrieval e alla valutazione dell'assistente conversazionale. Questa impostazione consente di esaminare da un lato le prestazioni del meccanismo di retrieval come componente indipendente e, dall'altro, il comportamento del sistema completo in

condizioni di utilizzo realistiche.
Gli script sono stati progettati per operare direttamente sul test set, eseguendo in modo sistematico l'intero processo di valutazione per ciascuna istanza del dataset. Questo approccio ha reso possibile la costruzione di un benchmark personalizzato, specifico per il dominio applicativo considerato, sul quale misurare in maniera coerente e riproducibile le prestazioni dell'assistente conversazionale.
La disponibilità di un benchmark automatizzato consente inoltre di monitorare nel tempo l'evoluzione del sistema, permettendo il confronto diretto tra diverse versioni dell'assistente e tra differenti configurazioni, e risulta particolarmente utile per quantificare i benefici di futuri miglioramenti e individuare tempestivamente eventuali regressioni nelle prestazioni.

4.3 Risultati Raccolti

I prototipi di assistente conversazionale presentati nella Sezione 3.4 sono stati valutati utilizzando il test set appositamente costruito. La valutazione è stata condotta applicando la metodologia e le metriche descritte nella sezione precedente ed è articolata in due parti: prima vengono analizzati i risultati del meccanismo di retrieval nelle diverse configurazioni, poi quelli dell'assistente conversazionale nei suoi diversi prototipi.

4.3.1 Risultati del Componente di Retrieval

La valutazione del meccanismo di retrieval ha analizzato le pipeline di recupero informativo multimodale presentate nella Sezione 3.3.2 in tutte le loro configurazioni. Le due pipeline differiscono principalmente per la struttura e il contenuto dei LangChain Document estratti dalle rispettive knowledge base vettoriali, costruite durante la fase di ingestion della documentazione del sistema di BPM amministrativo. Nella pipeline MRAG 2.0, gli oggetti estratti includono come metadati i contenuti originali del segmento di documentazione corrispondente, comprese le immagini in formato base64. Nella pipeline MRAG 1.0, invece, gli elementi recuperati consistono in una concatenazione testuale dei contenuti originali, con le immagini rappresentate tramite descrizioni visive integrate nel testo. È importante ricordare che la pipeline MRAG 1.0 è stata realizzata in due varianti: una basata sul modello di embedding proprietario text-embedding-3-small di OpenAI e una basata sul modello open-source nomic-embed-text. I risultati registrati per le diverse configurazioni sono riportati nella Tabella 4.1.

Pipeline di Retrieval	Precision	Recall	MRR	MAP	Latenza d'estrazione [s]
MRAG 2.0 text-embedding-3-small	0.333	0.887	**0.869**	0.833	0.493
MRAG 1.0 text-embedding-3-small	**0.346**	**0.938**	0.864	**0.843**	0.622
MRAG 1.0 nomic-embed-text	0.148	0.401	0.426	0.383	**0.314**

Tabella 4.1: Risultati retrieval

La Tabella 4.1 riporta i risultati delle metriche calcolate per valutare le prestazioni delle diverse configurazioni di retrieval implementate. Le pipeline basate su text-embedding-3-small hanno dimostrato performance paragonabili e complessivamente elevate. La pipeline MRAG 1.0 con text-embedding-3-small ha ottenuto i valori migliori di Recall (0.938), Precision (0.346) e MAP (0.843), mentre la pipeline MRAG 2.0 ha registrato l'MRR più elevato (0.869). La pipeline MRAG 1.0 con nomic-embed-text ha invece ottenuto valori nettamente inferiori su tutte le metriche di accuratezza (precision: 0.148, recall: 0.401, MRR: 0.426, MAP: 0.383), compensando parzialmente con la latenza di estrazione più contenuta (0.314 secondi) rispetto alle varianti basate su text-embedding-3-small (0.493 e 0.622 secondi). Le differenze di latenza tra le tre configurazioni rimangono comunque nell'ordine dei decimi di secondo.

4.3.2 Risultati dell'Assistente Conversazionale

La valutazione dell'assistente conversazionale ha analizzato le prestazioni dei due prototipi presentati nella Sezione 3.4.
I due prototipi condividono lo stesso funzionamento di base ma si distinguono per il meccanismo utilizzato per filtrare i contenuti recuperati dal nodo di retrieval. Il prototipo 1 impiega un LLM per individuare ed eliminare i segmenti di documentazione ritenuti irrilevanti rispetto alla query dell'utente. Il prototipo 2 adotta invece un criterio basato su una soglia di accettazione predefinita, escludendo i segmenti il cui punteggio di similarità semantica risulta inferiore al valore stabilito. È inoltre opportuno ricordare che il prototipo 1 è stato valutato in due configurazioni distinte, in cui viene abbinato rispettivamente alla pipeline MRAG 1.0 e alla pipeline MRAG 2.0, entrambe basate sull'utilizzo del modello di embedding text-embedding-3-small. Le prestazioni medie ottenute dai vari sistemi sul test set di valutazione per le metriche selezionate sono riportate di seguito:

Metriche tradizionali

Prototipo	Pipeline di Retrieval	ROUGE-1 F1	ROUGE-2 F1	ROUGE-L F1
P1	MRAG 2.0	0.351	0.190	0.239
P1	MRAG 1.0	**0.447**	0.236	**0.316**
P2	MRAG 1.0	0.425	**0.239**	0.310

Tabella 4.2: Risultati metriche tradizionali

La Tabella 4.2 riporta i risultati delle metriche tradizionali ROUGE calcolate per valutare la sovrapposizione lessicale tra le risposte generate dai diversi prototipi e i riferimenti del test set. Per quanto riguarda ROUGE-1 F1, che misura la sovrapposizione di singole parole, il prototipo P1-MRAG 1.0 ha ottenuto il punteggio più elevato (0.447), seguito dal prototipo P2-MRAG 1.0 (0.425) e dal prototipo P1-MRAG 2.0 (0.351). Per ROUGE-2 F1, che valuta la sovrapposizione di bigrammi, il prototipo P2-MRAG 1.0 ha registrato il valore massimo (0.239), seguito dal prototipo P1-MRAG 1.0 (0.236) e dal prototipo P1-MRAG 2.0 (0.190). Infine, per ROUGE-L F1, che considera la più lunga sottosequenza comune, il prototipo P1-MRAG 1.0 ha ottenuto le performance migliori (0.316), seguito dal prototipo P2-MRAG 1.0 (0.310) e dal sistema P1-MRAG 2.0 (0.239). Nel complesso, i valori di F1 risultano ridotti per tutte le configurazioni, indicando una limitata sovrapposizione lessicale tra le risposte generate e i riferimenti di valutazione.

Metriche Semantiche

Prototipo	Pipeline di Retrieval	BERTScore Precision	BERTScore Recall	BERTScore F1
P1	MRAG 2.0	0.701	0.777	0.737
P1	MRAG 1.0	**0.752**	**0.782**	**0.765**
P2	MRAG 1.0	0.747	0.775	0.760

Tabella 4.3: Risultati metriche semantiche

La Tabella 4.3 riporta i risultati delle metriche semantiche BERTScore calcolate per valutare la similarità semantica tra le risposte generate dai diversi prototipi e i riferimenti del test set. Per quanto riguarda la BERTScore Precision, il prototipo P1-MRAG 1.0 ha ottenuto il valore più elevato (0.752), seguito dal prototipo P2-MRAG 1.0 (0.747) e dal sistema P1-MRAG 2.0 (0.701). La BERTScore Recall vede nuovamente il prototipo P1-MRAG 1.0 registrare il punteggio massimo (0.782), seguito dal prototipo P1-MRAG 2.0 (0.777) e dalla variante P2-MRAG 1.0 (0.775).

Per quanto concerne la BERTScore F1, la soluzione P1-MRAG 1.0 ha ottenuto le performance migliori (0.765), seguita dal prototipo P2-MRAG 1.0 (0.760) e dal prototipo P1-MRAG 2.0 (0.737). Nel complesso, tutti i prototipi hanno dimostrato valori elevati su tutte le metriche semantiche, con punteggi superiori a 0.7, indicando una buona equivalenza semantica tra le risposte generate e i riferimenti di valutazione.

Metriche LLM-as-a-Judge

Prototipo	Pipeline di Retrieval	RAGAS Relevance	RAGAS Correctness	RAGAS Faithfulness
P1	MRAG 2.0	0.621	0.626	**0.869**
P1	MRAG 1.0	**0.691**	**0.649**	0.847
P2	MRAG 1.0	0.672	0.614	0.785

Tabella 4.4: Risultati metriche RAGAS

La Tabella 4.4 riporta i risultati delle metriche RAGAS calcolate per valutare la qualità delle risposte generate dai diversi prototipi di agente conversazionale. Per quanto riguarda la RAGAS Relevance, il prototipo P1-MRAG 1.0 ha ottenuto il punteggio più elevato (0.691), seguito dal prototipo P2-MRAG 1.0 (0.672) e dal prototipo P1-MRAG 2.0 (0.621). La RAGAS Correctness vede il sistema P1-MRAG 1.0 registrare il valore massimo (0.649), mentre le due configurazioni rimanenti mostrano performance paragonabili (0.626 e 0.614). Per la RAGAS Faithfulness, la configurazione P1-MRAG 2.0 ha ottenuto il punteggio migliore (0.869), seguita dalla versione P1-MRAG 1.0 (0.847) e dal prototipo P2-MRAG 1.0 (0.785). Nel complesso, tutti i prototipi hanno dimostrato performance complessivamente buone su tutte e tre le dimensioni valutative.

Metriche Aggiuntive

Prototipo	Pipeline di Retrieval	Precision	Recall	Latenza di risposta [s]
P1	MRAG 2.0	0.561	**0.911**	42.448
P1	MRAG 1.0	**0.767**	0.861	5.739
P2	MRAG 1.0	0.683	0.778	**5.029**

Tabella 4.5: Risultati prestazioni nodo di filtraggio e latenza di risposta

La Tabella 4.5 riporta i risultati delle metriche aggiuntive calcolate per valutare le prestazioni del nodo di filtraggio e la latenza di risposta dei diversi prototipi di

agente conversazionale. Per quanto riguarda le metriche di filtraggio, il prototipo P1-MRAG 2.0 ha registrato il valore di recall più elevato (0.911), mentre la configurazione P1-MRAG 1.0 ha ottenuto la precision massima (0.767). Il prototipo P2-MRAG 1.0 ha mostrato valori intermedi sia per precision (0.683) che per recall (0.778). Per quanto riguarda la latenza di risposta, emerge un divario significativo tra il sistema P1-MRAG 2.0, che ha registrato un tempo medio di 42.448 secondi, e i prototipi basati sul meccanismo MRAG 1.0, che hanno evidenziato latenze notevolmente inferiori, rispettivamente 5.739 secondi per il prototipo 1 e 5.029 secondi per il prototipo 2.

4.4 Discussione dei risultati

In questa sezione vengono analizzati i risultati raccolti nel corso della valutazione sperimentale. L'analisi è organizzata in due sottosezioni principali. Nella prima vengono esaminate le prestazioni del meccanismo di retrieval nelle diverse pipeline e configurazioni considerate. La seconda è invece dedicata alla valutazione delle prestazioni complessive dell'assistente conversazionale nei suoi diversi prototipi. Per ciascuna parte vengono discusse le metriche calcolate, con l'obiettivo di interpretare i risultati ottenuti e individuare la configurazione del sistema più efficace.

4.4.1 Analisi dei Risultati del Componente di Retrieval

Il meccanismo di retrieval delle informazioni pertinenti rappresenta un elemento fondamentale per garantire la generazione di risposte accurate e soddisfacenti rispetto alle richieste degli utenti. I documenti estratti dalla knowledge base vettoriale costituiscono infatti il contesto informativo utilizzato dall'LLM integrato nell'agente per produrre la risposta finale. Un meccanismo di retrieval inefficiente comprometterebbe quindi l'intero processo di generazione, portando alla produzione di risposte poco rilevanti, potenzialmente errate o, in alcuni casi, all'impossibilità di fornire una risposta adeguata.
Le pipeline di recupero informativo MRAG 1.0 e 2.0, basate sul modello di embedding text-embedding-3-small di OpenAI, hanno dimostrato prestazioni paragonabili e complessivamente elevate. Entrambe sono caratterizzate da alti valori di recall, rispettivamente 0.887 e 0.938, dimostrando la capacità di estrarre dalla knowledge base vettoriale la quasi totalità dei documenti rilevanti associati alle query dell'utente, mettendo a disposizione dell'agente conversazionale un contesto pressoché completo per la generazione di risposte esaustive. Per quanto riguarda la precision, i valori osservati sono pari a 0.333 per la prima pipeline e 0.346 per la seconda. Sebbene possano apparire contenuti a una prima lettura, poiché indicano che poco più del 30% dei documenti recuperati risulta effettivamente rilevante, tali risultati devono essere interpretati alla luce della configurazione del sistema

e delle caratteristiche del test set. Le pipeline sono infatti state configurate per restituire un numero fisso di tre documenti per ogni query, mentre nell'80% dei casi presenti nel set di valutazione è associato un solo documento rilevante. In questo scenario, anche quando il sistema recupera correttamente l'unico segmento di documentazione pertinente, la presenza degli altri due elementi non rilevanti incide inevitabilmente sul valore della metrica. Nonostante ciò, questa modalità di valutazione risulta comunque significativa, poiché consente di stabilire un valore di riferimento con cui confrontare la precision calcolata successivamente allo step di filtraggio dei contenuti irrilevanti eseguito nelle varie configurazioni dell'assistente conversazionale proposte. Inoltre, in uno scenario applicativo reale, disporre di un numero di contenuti minore potrebbe rivelarsi incapace di gestire la complessità delle richieste presentate al sistema dagli utenti.
Le due pipeline basate sul modello di embedding di OpenAI hanno mostrato risultati simili anche per i valori di MRR, rispettivamente 0.869 e 0.864, dimostrando non solo la capacità di estrarre documenti rilevanti nella loro quasi totalità, ma anche quella di posizionarli nelle prime posizioni della lista restituita dal modulo di retrieval. Questa capacità di ordinamento è ulteriormente confermata dagli elevati valori di MAP, superiori a 0.83 per entrambe le soluzioni. Per quanto riguarda la latenza di estrazione, le due soluzioni hanno registrato valori medi paragonabili di 0.493 secondi per la pipeline 1 e 0.622 secondi per la pipeline 2, indici di un recupero tempestivo indispensabile per contenere i tempi di risposta globali del sistema.
La pipeline di retrieval basata sul modello di embedding nomic-embed-text ha invece dimostrato performance nettamente inferiori rispetto a quelle basate sul modello di OpenAI. Questa variante è stata implementata esclusivamente per la pipeline che non utilizza le immagini del manuale utente. La soluzione open-source ha mostrato scarsa capacità nell'estrarre documenti rilevanti, con un recall di 0.401 e una precision di 0.148, valori più che dimezzati rispetto alla pipeline corrispondente realizzata con text-embedding-3-small. Data la limitata abilità di estrazione, anche le prestazioni di ordinamento sono state inevitabilmente compromesse, con un MRR di 0.426 e un MAP di 0.383. L'unico dato positivo di questa pipeline è rappresentato dalla latenza media di 0.314 secondi, circa 0.3 secondi inferiore rispetto alla soluzione corrispondente basata su OpenAI, sebbene questo vantaggio non sia particolarmente impattante sui tempi di risposta globali del sistema. Il degrado prestazionale può essere attribuito alla dimensionalità ridotta dei vettori di embedding prodotti dal modello, pari a 768 elementi contro i 1536 di text-embedding-3-small. La dimensionalità delle rappresentazioni vettoriali di nomic-embed-text risulta troppo limitata rispetto alla quantità di contenuto testuale da convertire in embedding, determinando una perdita di rappresentatività del vettore rispetto all'informazione di partenza da codificare. Per questa ragione, il meccanismo di estrazione basato su cosine similarity non riesce a identificare con

precisione i documenti associati alla query utente, portando a prestazioni insufficienti. La ridotta dimensionalità del modello open-source offre unicamente il vantaggio marginale di ridurre di qualche decimo di secondo la latenza di estrazione media. Nel complesso, le pipeline di retrieval basate sul modello text-embedding-3-small di OpenAI hanno dimostrato una netta superiorità rispetto alla soluzione sviluppata con il modello di Nomic, evidenziando l'importanza di individuare un equilibrio adeguato tra la dimensionalità dei vettori di embedding e la quantità di contenuto da rappresentare.
Infine, tra le due pipeline basate su OpenAI, quella MRAG 1.0, che concatena i contenuti testuali della documentazione con le descrizioni visive delle immagini, ha mostrato prestazioni leggermente superiori rispetto a quella MRAG 2.0 che preserva le immagini originali come metadati, registrando in particolare un valore di recall più elevato, pari a 0.938 contro 0.887.

4.4.2 Analisi dei Risultati dell'Assistente Conversazionale

L'analisi delle performance dell'assistente conversazionale nel suo complesso coincide con quella del modulo di generazione che il framework Auepora [52] indica come secondo elemento da valutare all'interno di un sistema di RAG. La fase di generazione costituisce il fulcro dell'agente conversazionale proposto in questo progetto di tesi.
L'assistente intelligente, infatti, esegue diverse elaborazioni attraverso l'utilizzo di LLM per rispondere alle richieste degli utenti sulla base dei contenuti estratti dal modulo di retrieval. Risulta quindi fondamentale che il meccanismo di generazione sia in grado di comprendere la query dell'utente e, attraverso l'analisi dei documenti recuperati, produrre una risposta pertinente che contenga le informazioni significative per soddisfare le esigenze del richiedente. Qualora queste capacità di analisi e formulazione risultassero inadeguate, la qualità delle risposte prodotte dall'assistente conversazionale sarebbe insufficiente per le necessità dell'utilizzatore, compromettendone l'applicabilità in contesti reali.

Metriche Tradizionali e Semantiche

Per quanto riguarda le metriche tradizionali basate sulla sovrapposizione di n-grammi, i due prototipi nelle varie configurazioni presentate hanno riportato performance molto simili. I valori di F1 per ROUGE-1, ROUGE-2 e ROUGE-L risultano complessivamente ridotti, indicando una limitata sovrapposizione lessicale tra le risposte generate dal sistema e i riferimenti di test definiti con il supporto di un esperto di dominio. I risultati migliori per questa famiglia di metriche sono stati registrati dal prototipo P1-MRAG 1.0, con un F1 ROUGE-1 di 0.447 e un F1 ROUGE-L di 0.316, mentre il punteggio maggiore di F1 ROUGE-2 è stato ottenuto

dalla soluzione P2-MRAG 1.0. Tuttavia, le metriche ROUGE sono oggi considerate superate per la valutazione di sistemi di question answering poiché misurano esclusivamente la similarità lessicale basata sulla sovrapposizione di elementi testuali, ignorando il significato semantico delle risposte. In un sistema di QA moderno, una stessa domanda può ammettere molteplici risposte corrette formulate diversamente, che ROUGE penalizza nonostante semanticamente equivalenti. Questo limite delle metriche tradizionali emerge chiaramente dai punteggi ottenuti nelle metriche semantiche, dove tutte le varianti di sistema hanno dimostrato buone prestazioni con valori superiori a 0.7 sia per Precision che per Recall di BERTScore. Ciò indica che, nonostante le risposte generate siano lessicalmente differenti rispetto ai riferimenti del test set, esse risultano semanticamente vicine alle risposte di riferimento. Da questo punto di vista, il prototipo P1-MRAG 1.0 ha riportato prestazioni leggermente superiori rispetto agli altri sistemi. È inoltre interessante notare come la configurazione che ha mostrato le performance migliori per le metriche basate su sovrapposizione abbia registrato anche le maggiori prestazioni per quanto riguarda le metriche semantiche, suggerendo comunque l'esistenza di un legame tra equivalenza lessicale e semantica.

Metriche LLM-as-a-Judge

Al fine di condurre un'analisi più approfondita sulla qualità delle risposte, i prototipi sono stati valutati attraverso le metriche RAGAS di Relevance, Correctness e Faithfulness, che consentono di esaminare dimensioni qualitative non catturabili dalle metriche precedenti.
Per quanto riguarda la RAGAS Relevance, le tre configurazioni proposte mostrano performance complessivamente paragonabili, con una leggera superiorità del prototipo P1-MRAG 1.0 che raggiunge un punteggio di 0.691, evidenziando la capacità delle varie soluzioni di rispondere in maniera effettiva alle esigenze espresse dall'utente all'interno della richiesta presentata al sistema. I prototipi che utilizzano il meccanismo di retrieval MRAG 1.0, basato sulla conversione delle immagini in descrizioni testuali durante l'ingestion, dimostrano livelli di relevance superiori. Questo aspetto può essere imputato in parte al fatto che l'utilizzo della pipeline di recupero MRAG 2.0 restituisce al modello generativo un'alternanza di contenuti testuali e visivi per la formulazione delle risposte, richiedendo l'integrazione di informazioni provenienti da modalità diverse. Questa strategia potrebbe introdurre maggiore complessità nell'estrazione delle informazioni pertinenti rispetto alla query utente. Al contrario, nella pipeline MRAG 1.0 tutti i contenuti sono già in forma testuale coesa e uniformemente accessibile al modello generativo, facilitando la produzione di risposte più allineate rispetto alla richiesta formulata nella query.
Un ulteriore fattore che può incidere negativamente sui valori della metrica è legato alla modalità con cui la RAGAS Relevance viene calcolata (4.19). La metrica

genera, a partire dalla risposta prodotta dal sistema, un insieme di possibili domande che tale risposta sarebbe in grado di soddisfare e confronta poi gli embedding delle domande prodotte con l'embedding della query originale. Quando il modello generativo arricchisce la risposta con informazioni aggiuntive, anche se utili o contestualmente pertinenti, l'LLM utilizzato per la valutazione può generare domande che non coincidono perfettamente con la richiesta iniziale dell'utente. Questo fenomeno tende a ridurre la similarità semantica media calcolata dalla metrica e, di conseguenza, il valore complessivo di relevance, pur non indicando necessariamente una minore qualità della risposta dal punto di vista informativo.
Per quanto riguarda la RAGAS Correctness, le tre configurazioni analizzate mostrano performance complessivamente paragonabili, con punteggi tutti superiori a 0.61, dimostrando in generale un buon grado di corrispondenza tra le affermazioni contenute nella risposta generata dall'assistente e quelle presenti nella risposta di riferimento del test set di valutazione. Le maggiori penalizzazioni ai punteggi registrati sono dovute principalmente alla indisponibilità in fase di generazione dei documenti rilevanti per rispondere correttamente alle richieste, un fenomeno dovuto a inefficienze del nodo di filtraggio dei prototipi e in alcune situazioni al meccanismo di retrieval stesso. Questa criticità danneggia il valore complessivo della metrica in particolare la componente di correttezza fattuale che risente della presenza di omissioni nella risposta generata.
Il valore più elevato di Correctness, pari a 0.649, è stato registrato dal prototipo P1-MRAG 1.0, che utilizza un meccanismo di filtraggio dei documenti basato su LLM. Questo risultato può essere attribuito alla maggiore capacità del filtro LLM-based di analizzare semanticamente i segmenti di documentazione recuperati e di identificare quelli effettivamente rilevanti rispetto alla query dell'utente. A differenza del filtraggio basato su una soglia numerica di similarità, il filtro LLM-based è in grado di valutare il contenuto informativo dei documenti in modo contestuale, preservando anche segmenti che, pur presentando valori di similarità non particolarmente elevati, contengono informazioni utili per la formulazione della risposta.
Infine, le performance di Faithfulness mostrano una superiorità del primo prototipo rispetto al secondo, con la variante che utilizza la pipeline di retrieval MRAG 2.0 che raggiunge il punteggio migliore di 0.869. Questo andamento è spiegabile a partire dalle differenze nel meccanismo di filtraggio. Il filtro LLM-based del prototipo 1, valutando la pertinenza dei documenti in modo contestuale, tende a preservare un numero maggiore di segmenti rilevanti, come confermato dai valori di Recall post-filtraggio più elevati (Tabella 4.5). Un contesto informativo più completo in fase generativa riduce la necessità del modello di ricorrere alla propria conoscenza parametrica per colmare le lacune, limitando così il rischio di allucinazioni. Al contrario, il filtro basato su soglia di cosine similarity del prototipo 2 scarta più aggressivamente i documenti con punteggio borderline, impoverendo il contesto

disponibile e spingendo il modello a integrare informazioni non ancorate alla knowledge base, con conseguente calo della fedeltà alle fonti originali. In generale ognuna delle soluzioni proposte ha comunque mostrato punteggi di faithfulness elevati superiori a 0.78, dimostrando che le informazioni restituite dall'assistente conversazionale risultano essere fedeli rispetto ai contenuti del manuale utente forniti in fase di generazione.

Metriche Operative

I vari sistemi realizzati sono stati valutati anche attraverso metriche aggiuntive volte ad analizzare le capacità del meccanismo di filtraggio e la tempestività di risposta dei prototipi, aspetto di particolare rilevanza per il dominio applicativo di riferimento. Il prototipo P1-MRAG 2.0 ha registrato il più alto livello di recall post-filtraggio con un valore di 0.911, a fronte di una precision non particolarmente elevata pari a 0.561. Considerando che la pipeline di retrieval MRAG 2.0 presentava inizialmente un recall di 0.887 e una precision di 0.333, il nodo di filtraggio ha permesso di incrementare significativamente precision e recall. Ciò dimostra che il filtro possiede ottime capacità nel preservare i contenuti rilevanti e discrete qualità nello scartare parzialmente i documenti non utili, sebbene lasci ancora passare numerosi documenti irrilevanti alla fase di generazione.
La seconda variante del prototipo 1, implementata con il meccanismo di recupero MRAG 1.0, ha mostrato il miglior compromesso tra precision e recall: il nodo di filtraggio ha mantenuto elevato il valore di recall, passato da 0.938 a 0.861, incrementando notevolmente la precision da 0.346 a 0.767. In questo caso, il meccanismo di filtraggio si è rivelato particolarmente efficace nel selezionare i contenuti estratti in fase di retrieval, scartando la maggior parte dei documenti irrilevanti e preservando quelli significativi per la generazione, un aspetto che potrebbe aver contribuito a incrementare i valori di relevance e correctness.
Il prototipo 2, che sostituisce il filtro LLM-based con una soglia fissa di cosine similarity pari a 0.5, ha registrato il valore di Recall post-filtraggio più basso tra i sistemi proposti. Nonostante la soglia adottata sia relativamente permissiva, il Recall è sceso da 0.938 a 0.778, a fronte di un incremento della Precision da 0.346 a 0.683. Il calo marcato suggerisce che i punteggi di similarità associati a diversi documenti rilevanti risultino sistematicamente inferiori a 0.5, indicando una limitata capacità rappresentativa dei vettori di embedding per i chunk della pipeline MRAG 1.0. Questo effetto è verosimilmente riconducibile alla lunghezza dei chunk stessi: la concatenazione di testo, descrizioni di tabelle e descrizioni visive delle immagini produce segmenti estesi, la cui codifica in un singolo vettore di dimensione fissa tende a diluire il contenuto semantico, abbassando la similarità con query brevi e specifiche anche in presenza di informazioni rilevanti.
Dai risultati emerge poi una possibile relazione tra alti valori di recall e alti

valori di faithfulness: il fatto di riuscire a fornire la grande maggioranza dei documenti rilevanti contenenti le informazioni utili da includere nella risposta limita probabilmente la tendenza ad allucinare dell'LLM nel tentativo di produrre una risposta alla query utente. La precision sembra invece essere meno critica, in quanto il modello generativo è in grado di non considerare le informazioni irrilevanti nella formulazione della risposta, anche quando queste vengono comunque passate dal nodo di filtraggio.
Per quanto riguarda la latenza di risposta, i valori misurati hanno evidenziato un divario prestazionale significativo tra i sistemi implementati con meccanismo di retrieval MRAG 2.0 e quelli MRAG 1.0. Nel primo caso è stato registrato un tempo di risposta medio superiore a 42 secondi, un valore che rende il sistema inutilizzabile in scenari applicativi real-time come quello dell'assistente conversazionale. Nel secondo caso, i tempi sono risultati significativamente più contenuti con valori poco superiori a 5 secondi, con un leggero vantaggio per il prototipo 2 che, grazie all'utilizzo di un nodo di filtraggio basato su cosine similarity, elimina l'overhead di latenza dovuto a un'ulteriore chiamata all'LLM richiesta dal prototipo 1. Questa differenza sostanziale nei tempi di risposta è imputabile al fatto che l'utilizzo della soluzione MRAG 2.0 obbliga il sistema a elaborare in fase di generazione, oltre ai contenuti testuali, anche le immagini, spesso molteplici per singolo documento estratto, determinando tempi d'esecuzione più elevati e quindi latenze di risposta prolungate. L'utilizzo delle immagini in fase generativa, inoltre, non contribuisce così significativamente alle prestazioni in termini di qualità della risposta rispetto all'utilizzo di informazioni totalmente testuali, come dimostrato dai risultati ottenuti su tutte le altre metriche analizzate.

4.4.3 Identificazione della Soluzione Ottimale

L'analisi complessiva delle metriche di valutazione evidenzia come il prototipo P1-MRAG 1.0 rappresenti la soluzione più equilibrata e performante tra i sistemi proposti, distinguendosi per la capacità di ottenere risultati superiori su pressoché tutte le dimensioni considerate e di offrire il miglior compromesso tra qualità delle risposte ed efficienza operativa.
Dal punto di vista della qualità delle risposte, questo sistema ha ottenuto i valori più elevati di RAGAS Relevance (0.691) e Correctness (0.649), nonché le migliori prestazioni semantiche in termini di BERTScore F1 (0.765), dimostrando una solida capacità di rispondere in modo pertinente e semanticamente accurato alle richieste degli utenti. Per quanto riguarda la Faithfulness (0.847), il sistema non raggiunge il valore massimo assoluto, registrato dalla variante P1-MRAG 2.0 (0.869), tuttavia il divario è contenuto e ampiamente accettabile. Il nodo di filtraggio LLM-based ha dimostrato il miglior equilibrio tra precision (0.767, valore massimo tra tutti i prototipi) e recall (0.861), identificando efficacemente i documenti rilevanti e

scartando la maggior parte di quelli irrilevanti, pur mantenendo un richiamo sufficientemente elevato da limitare il rischio di allucinazioni in fase generativa.
Dal punto di vista operativo, la latenza media di 5.739 secondi rende il sistema pienamente compatibile con scenari applicativi real-time, risultando circa sette volte più rapido rispetto alla variante P1-MRAG 2.0 (42.448 secondi), il cui overhead è imputabile all'elaborazione multimodale delle immagini in fase generativa. Il lieve vantaggio di latenza del prototipo P2-MRAG 1.0 (5.029 secondi) non compensa le sue prestazioni inferiori sulle metriche di qualità della risposta, in particolare per Relevance, Correctness e Faithfulness.
In sintesi, il prototipo P1-MRAG 1.0 emerge come la soluzione ottimale per un deployment operativo dell'assistente conversazionale nel dominio di riferimento, combinando risposte pertinenti e semanticamente accurate, un meccanismo di filtraggio efficace e una latenza compatibile con l'utilizzo in contesti reali.

4.4.4 Analisi qualitativa dei risultati

A complemento dell'analisi quantitativa basata sulle metriche discusse in precedenza, in questa sezione, viene presentata un'analisi qualitativa del sistema identificato come ottimale: il prototipo 1 con pipeline di retrieval MRAG 1.0 basata su text-embedding-3-small. L'analisi condotta consente di osservare e comprendere concretamente la qualità delle risposte prodotte dall'agente conversazionale, offrendo un termine di confronto diretto con i valori numerici calcolati. Le istanze di test riportate di seguito sono state selezionate per illustrare tanto i casi in cui il sistema eccelle, quanto le situazioni in cui emergono criticità che potrebbero richiedere ulteriori interventi.

Esempio 1: Rifiuto di un'attività in carico (QID 4)

Utente: "Ho preso in carico un'attività, ma ho bisogno di cederla, come faccio?"

Sistema: "Per cedere un'attività che hai preso in carico, puoi utilizzare la funzionalità di 'Rifiuta'. In questo modo, l'attività verrà impostata come 'In attesa' e sarà disponibile per altri utenti del tuo ufficio profilati sulla piattaforma. Per farlo, clicca sul pulsante 'Rifiuta' presente nell'interfaccia dell'attività. Una volta rifiutata, l'attività tornerà visibile nella sezione 'Attività in attesa' e potrà essere presa in carico da un altro utente."

Ground thruth: "Per cedere l'attività e renderla disponibile a tutto l'ufficio si ricorre alla funzionalità 'Rifiuta'. L'utente che ha preso in carico l'attività, se nota l'impossibilità a proseguire con la gestione del documento ha modo di cliccare

sul pulsante 'Rifiuta' per rifiutare l'incarico, permettendo agli altri utenti abilitati di prendere in carico l'attività. Il documento risulterà nuovamente visibile nella sezione 'Attività', nel campo 'Attività in attesa'."

Questo caso (Relevance: 0.850, Correctness: 0.966, Faithfulness: 1.000) rappresenta un esempio di risposta eccellente. Confrontando la risposta generata dall'agente conversazionale con quella di riferimento, in accordo con i punteggi elevati di correttezza e rilevanza, si può affermare che il sistema ha saputo fornire informazioni accurate e pertinenti rispetto alla necessità espressa dall'utente nella query. Il punteggio massimo di faithfulness indica inoltre che tutte le informazioni contenute nella risposta risultano essere fondate rispetto ai contenuti dei documenti estratti dalla knowledge base e utilizzati in fase di generazione, confermando l'assenza di allucinazioni.

Esempio 2: Richiesta fuori contesto applicativo (QID 29)

Utente: "Come si posiziona il Milan in serie A?"

Sistema: "Mi spiace la richiesta che hai effettuato potrebbe non riguardare l'applicativo DOCWF, le sue funzionalità e utilizzo. In caso contrario non ho sufficienti informazioni per rispondere in maniera adeguata alla tua richiesta. Ti prego di essere più chiaro, sarò felice di supportarti nella risoluzione di qualsiasi dubbio legato a questo applicativo."

Ground thruth: "Mi spiace la richiesta che hai effettuato potrebbe non riguardare l'applicativo DOCWF, le sue funzionalità e utilizzo. In caso contrario non ho sufficienti informazioni per rispondere in maniera adeguata alla tua richiesta. Ti prego di essere più chiaro, sarò felice di supportarti nella risoluzione di qualsiasi dubbio legato a questo applicativo."

Questo esempio riporta una richiesta fuori contesto da parte dell'utente che viene riconosciuta e gestita correttamente dall'agente, dimostrandone la robustezza in situazioni di rifiuto negativo. Il sistema restituisce una risposta di default configurata appositamente per domande che esulano dal dominio applicativo o per gestire casi in cui il sistema non dispone di informazioni sufficienti per generare una risposta affidabile. Per istanze di questo tipo è stato calcolato esclusivamente lo score di correttezza, che in questo caso assume valore unitario, poiché la risposta generata coincide con quella attesa per domande fuori contesto.

Esempio 3: Gestione allegati con nome errato (QID 24)

Utente: "Ho caricato un allegato al decreto, ma ho usato un nome errato. Posso modificarlo?"

Sistema: "Sì. Puoi correggere l'errore eliminando l'allegato caricato con il nome errato e ricaricandolo successivamente con il nome corretto. La cancellazione degli allegati è possibile fino allo step di approvazione dell'atto; una volta raggiunto tale step (approvazione), non è più possibile modificarli."

Ground thruth: "Nel caso in cui sia stato caricato un allegato con un nome che non segue la convenzione, occorre eliminarlo e ricaricarlo successivamente con il nome corretto. È possibile eliminare gli allegati fino allo step di approvazione."

Questo caso (Relevance: 0.685, Correctness: 0.951, Faithfulness: 1.000) rappresenta un esempio di risposta soddisfacente. Come nel caso precedente, le informazioni restituite dal sistema risultano corrette e ben fondate, in linea con gli elevati valori di correctness e faithfulness. Il punteggio di relevance più contenuto è invece riconducibile alla presenza, nella risposta generata, di informazioni aggiuntive relative alle tempistiche entro cui l'atto amministrativo può essere modificato, un aspetto non esplicitamente richiesto nella query. Questa tendenza del sistema ad arricchire dal punto di vista informativo le risposte da presentare all'utente è risultata piuttosto ricorrente dall'analisi del test set di valutazione. Questo fenomeno ha contribuito ad abbassare lo score globale di relevance e, nelle istanze in cui la risposta di riferimento non include le informazioni addizionali introdotte dal sistema, anche quello di correctness. Tuttavia, come in questo caso specifico, i contenuti extra si rivelano strettamente correlati alla richiesta dell'utente e utili a una comprensione più completa del workflow degli atti amministrativi, configurandosi quindi come un valore aggiunto piuttosto che come una limitazione del sistema.

Esempio 4: Scopo della sezione 'DATI DI PROCESSO' (QID 6)

Utente: "A cosa serve la sezione 'DATI DI PROCESSO' nella dashboard della scheda?"

Sistema: "La sezione 'DATI DI PROCESSO' nella dashboard della scheda serve a fornire all'utente una descrizione dell'attività o dell'azione richiesta, aiutandolo a capire quale compito deve svolgere."

Ground thruth: "La sezione 'DATI DI PROCESSO' include tre elementi utili per l'utente:
1. Una breve spiegazione che indica all'utente quale azione è richiesta o quale attività deve essere svolta.

2. L'elenco degli uffici e degli utenti che hanno accesso in sola lettura alla scheda, con possibilità di consultarla tramite la funzione di ricerca.
3. L'elenco degli istruttori a cui la scheda è stata condivisa con permessi di modifica."

Questo caso (Relevance: 0.942, Correctness: 0.440, Faithfulness: 1.000) risulta critico in quanto si distingue per un valore di correctness notevolmente ridotto. Analizzando la risposta generata, emerge chiaramente come questa sia solo parzialmente corretta, risultando incompleta rispetto al ground truth. La lacuna informativa è riconducibile alla mancata disponibilità, in fase generativa, di tutti i documenti rilevanti associati alla query necessari per produrre una risposta esaustiva. Tale problema può originarsi in due modi distinti: in alcuni casi, come in questo, il modulo di retrieval non riesce a estrarre correttamente tutti i documenti di interesse, in altri, è il nodo di filtraggio a scartare erroneamente documenti pertinenti recuperati. L'incompletezza del contesto documentale rappresenta la criticità più impattante sulla qualità delle risposte, con ripercussioni che si estendono anche a faithfulness e relevance, poiché il modello generativo si trova a produrre una risposta sulla base di un contesto incompleto e potenzialmente fuorviante.

Nel complesso, l'analisi qualitativa delle risposte prodotte dal prototipo ha evidenziato una buona qualità generale del sistema, le cui risposte risultano soddisfacenti nella maggioranza dei casi. Sono emersi tuttavia alcuni limiti che incidono principalmente sulle metriche di relevance e correctness, offrendo quindi prospettive di miglioramento.

Capitolo 5

Conclusioni

5.1 Contributi della Ricerca

Questa sezione raccoglie le considerazioni conclusive del lavoro di ricerca, organizzate attorno a quattro contributi principali: metodologico, analitico, applicativo e pratico.

5.1.1 Contributi Metodologici

Sul piano metodologico, la ricerca documenta l'intero processo di progettazione di un agente conversazionale specializzato nel rispondere a domande relative a documenti aziendali in maniera affidabile, illustrando le scelte progettuali alla base di ciascuna componente del sistema.
In primo luogo, viene descritto il meccanismo di ingestion adottato per costruire una knowledge base vettoriale a partire da un documento multimodale non strutturato, dettagliando le fasi di partitioning, elaborazione e indicizzazione dei contenuti. In secondo luogo, viene presentato il sistema di recupero delle informazioni rilevanti per la generazione di risposte accurate. Infine, viene illustrata la progettazione del grafo di esecuzione dell'agente conversazionale che integra al suo interno i meccanismi di retrieval realizzati.
Un contributo specifico riguarda il confronto tra approcci di Multimodal RAG differenziati per la gestione delle immagini presenti nella documentazione indicizzata. Per il caso d'uso di interesse i risultati sperimentali hanno dimostrato la superiorità dell'approccio MRAG 1.0, che prevede la sintesi di descrizioni testuali delle immagini in fase di ingestion, rispetto all'approccio MRAG 2.0, che utilizza le immagini originali direttamente in fase generativa. Tale superiorità è determinata non solo da prestazioni migliori in termini di qualità del retrieval e della generazione, ma soprattutto dalla capacità di garantire tempi di risposta compatibili con lo scenario d'utilizzo real-time, a differenza della variante multimodale che registra

una latenza media superiore a 42 secondi.
Un ulteriore contributo metodologico riguarda il confronto tra meccanismi alternativi di filtraggio dei documenti recuperati. I risultati ottenuti in termini di correttezza, rilevanza e fedeltà hanno dimostrato la superiorità di un nodo di filtraggio LLM-based rispetto a uno fondato su una soglia di similarità semantica predefinita, evidenziando la maggiore capacità del filtro contestuale di preservare i documenti informativamente rilevanti.

5.1.2 Contributo Analitico

La ricerca presenta un approccio strutturato basato sulle linee guida del framework Auepora [52] per la misurazione delle performance di sistemi agentici RAG-based analoghi a quello proposto. Attraverso un'analisi bibliografica, sono state individuate le metriche più adatte a valutare sia il modulo di retrieval che quello di generazione, consentendo un'analisi completa delle prestazioni lungo dimensioni complementari: sovrapposizione lessicale, similarità semantica, rilevanza, correttezza, fedeltà e latenza. La natura quantitativa delle metriche selezionate permette di confrontare implementazioni differenti e di identificare le scelte progettuali che determinano risultati superiori. Il test set reference-based di 30 istanze, costruito con il supporto di un esperto di dominio, unitamente all'insieme di metriche predisposto, costituisce una baseline replicabile per sviluppi e miglioramenti futuri del sistema.

5.1.3 Contributo Applicativo

Il confronto sperimentale tra le configurazioni implementate ha permesso di identificare il prototipo ottimale per il contesto applicativo d'interesse. Il sistema selezionato, corrispondente al prototipo P1-MRAG 1.0, ha dimostrato buone capacità di retrieval con una precision post-filtraggio di 0.767 e un recall di 0.861, garantendo che la grande maggioranza delle informazioni rilevanti sia disponibile in fase generativa. Dal punto di vista della generazione, il prototipo ha registrato un BERTScore F1 di 0.765, una Relevance di 0.691 e una Correctness di 0.649, dimostrando la capacità di produrre risposte pertinenti e semanticamente accurate. La Faithfulness di 0.847 attesta inoltre l'affidabilità delle informazioni restituite rispetto ai contenuti della documentazione originale, un aspetto di particolare importanza in contesti amministrativi in cui errori operativi posso produrre forti rallentamenti. La latenza media di 5.7 secondi rende infine il sistema compatibile con scenari applicativi real-time. Nel complesso, sulla base dei risultati raccolti, il prototipo realizzato si presenta come un assistente conversazionale in grado di rispondere in maniera affidabile e in tempo ridotti alle richieste degli utenti,

rappresentando un solido punto di partenza per lo sviluppo di un prodotto destinato alla messa in produzione.

5.1.4 Contributo Pratico

Sul piano pratico, il progetto documenta le modalità adottate per l'integrazione dell'agente all'interno di un applicativo preesistente, distribuito come servizio esterno in ambiente containerizzato e accessibile tramite API. La sperimentazione ha così dimostrato la concreta fattibilità dell'impiego di LLM e tecniche RAG come strumenti di supporto operativo in processi di business complessi, con il potenziale di ridurre i carichi di assistenza.

5.2 Limitazioni e Direzioni Future

Nonostante i contributi e i risultati promettenti emersi dalla sperimentazione, il sistema sviluppato presenta alcune limitazioni relative ad aspetti prestazionali, valutativi e funzionali che aprono la strada a futuri miglioramenti, in vista della realizzazione di un prototipo adatto al rilascio in produzione.

5.2.1 Limitazioni Prestazionali

Dal punto di vista delle performance, i valori di Relevance e Correctness registrati per il prototipo ottimale offrono ancora margini di miglioramento significativi.
La rilevanza delle risposte rispetto alle query utente è fortemente influenzata dalla tendenza del sistema ad arricchire le risposte con informazioni aggiuntive rispetto a quelle esplicitamente richieste. Questa evidenza è scaturita dall'analisi dei risultati di RAGAS Relevance congiuntamente a un'osservazione qualitativa delle risposte generate per alcune istanze di test (esempio 3 Sezione 4.4.4) e alla metodologia di calcolo della metrica. Questo comportamento incide negativamente sul punteggio RAGAS Relevance, la cui metodologia di calcolo prevede di estrarre dalla risposta generata un insieme di domande implicite e di misurarne la similarità semantica con la query originale. Una risposta che include contenuti extra porta alla generazione di domande semanticamente distanti dalla richiesta dell'utente, penalizzando il punteggio anche quando le informazioni aggiuntive sono pertinenti e potenzialmente utili. In questo senso, i valori di Relevance registrati rappresentano una sottostima della qualità effettiva delle risposte, poiché la metrica non distingue tra contenuti extra irrilevanti e contenuti extra informativamente validi. Nonostante ciò, miglioramenti futuri possono prevedere di regolare questo comportamento istruendo il modello generativo a rispondere in modo più focalizzato alla richiesta esplicita dell'utente, attraverso tecniche di prompt engineering o few-shot prompting, con l'obiettivo di migliorare sia il punteggio della metrica che la precisione delle risposte

prodotte.
Per quanto riguarda la Correctness, essa subisce un impatto negativo significativo nei casi in cui il meccanismo di retrieval e/o di filtraggio dei documenti estratti non riesca a restituire un contesto completo di tutte le informazioni rilevanti. Questa problematica è emersa dall'analisi delle metriche di RAGAS Correctness registrate, mentre la sua causa è stata individuata come tendenza ricorrente attraverso l'analisi qualitativa di alcune istanze di test (esempio 4 Sezione 4.4.4) e tenendo in considerazione il meccanismo di calcolo della metrica di correttezza. In queste situazioni il sistema produce risposte incomplete che non riescono a soddisfare le richieste dell'utente. Per affrontare questa problematica è necessario garantire che meccanismi di retrieval di filtraggio restituiscano un contesto informativo completo per la fase di generazione. A questo scopo è possibile intervenire su più fronti: dal perfezionamento del meccanismo di ingestion documentale, all'adozione di tecniche di retrieval più sofisticate, fino al perfezionamento delle capacità del nodo di filtraggio. Sviluppi futuri saranno indirizzati in queste direzioni al fine di individuare soluzioni più efficaci.

5.2.2 Limitazioni della Metodologia di Valutazione

Un'ulteriore limitazione che caratterizza il sistema riguarda la limitatezza del test set di valutazione che si riduce a un insieme di 30 istanze, a causa della laboriosità del processo di costruzione manuale che richiede per ciascuna istanza una domanda, una risposta di riferimento e gli identificativi dei documenti rilevanti associati. La dimensione ridotta del test set di valutazione limita la rappresentatività statistica dei risultati e la generalizzabilità delle conclusioni.
Oltre alla dimensione del test set, un'altra limitazione del processo di valutazione riguarda la natura single-hop delle istanze, che non contempla interazioni multi-turno. Poiché un agente conversazionale è per sua natura destinato a gestire dialoghi articolati su più scambi, questa caratteristica del test set lascia aperta la questione di come il sistema si comporti in scenari conversazionali reali, dove il contesto delle interazioni precedenti può influenzare la qualità delle risposte successive.
Una possibile direzione futura per ampliare e diversificare il test set consiste nell'impiegare un LLM per generare automaticamente coppie domanda/risposta a partire dalle fonti di riferimento [62]. Questo approccio ridurrebbe significativamente il costo del processo di costruzione, sebbene richieda comunque una validazione umana parziale.

5.2.3 Limitazioni Funzionali dell'Assistente Conversazionale

Il sistema attuale non dispone di un meccanismo per la persistenza delle conversazioni. Questo aspetto è rilevante per due ragioni: da un lato rappresenterebbe un valore aggiunto per l'utente, che potrebbe riconsultare o riprendere conversazioni passate, dall'altro, l'agente potrebbe attingere allo storico delle interazioni registrate per rispondere più rapidamente a richieste già gestite, riducendo il ricorso al meccanismo di retrieval. LangGraph propone un supporto nativo alla persistenza delle conversazioni cross-thread attraverso il meccanismo checkpointer, che rappresenta una direzione naturale per sviluppi futuri [63].
Una seconda limitazione riguarda la staticità della knowledge base, che allo stato attuale deve essere ricreata manualmente rieseguendo l'intera pipeline di ingestion ogni volta che si rendono necessari aggiornamenti o inserimenti di nuovi documenti. Sviluppi futuri prevedono la realizzazione di apposite API per semplificare e automatizzare la gestione della knowledge base, riducendo l'intervento manuale richiesto. Una limitazione connessa riguarda la copertura della knowledge base, che include attualmente la documentazione relativa a una singola tipologia di utente del sistema di BPM. In prospettiva, si prevede di realizzare knowledge base distinte per le diverse tipologie di operatori coinvolti nel processo amministrativo, configurando l'agente in modo da attingere alla base di conoscenza corrispondente al profilo dell'utente autenticato, con l'obiettivo di migliorare l'accuratezza e la qualità complessiva delle risposte.
Un'ultima limitazione funzionale che caratterizza l'assistente conversazionale riguarda la struttura del grafo d'esecuzione utilizzato per la generazione delle risposte. Allo stato attuale, il nodo di routing in ingresso presenta una logica piuttosto rigida, necessaria per bloccare richieste esterne al dominio tematico del sistema. Sebbene questa rigidità non costituisca un problema sulla base del test set utilizzato, in scenari di utilizzo reale potrebbe rivelarsi insufficiente a gestire l'ampia varietà di richieste che gli utenti possono formulare. Per aumentare l'autonomia e la flessibilità dell'agente sarebbe opportuno ristrutturare il grafo di esecuzione introducendo design patterns che sfruttino le capacità di ragionamento e azione degli LLM stessi [24]. Sviluppi futuri si occuperanno di indagare questa strategia implementativa.

Bibliografia

[1] Mathias Weske. Motivation and definitions. In *Business Process Management: Concepts, Languages, Architectures*, chapter 1.1, pages 3–11. Springer, Berlin, Heidelberg, 2012.

[2] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2):1–124, 2023.

[3] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

[4] Shangyu Wu, Ying Xiong, Yufei Cui, Haolun Wu, Can Chen, Ye Yuan, Lianming Huang, Xue Liu, Tei-Wei Kuo, Nan Guan, et al. Retrieval-augmented generation for natural language processing: A survey. *arXiv preprint arXiv:2407.13193*, 2024.

[5] Hana Derouiche, Zaki Brahmi, and Haithem Mazeni. Agentic ai frameworks: Architectures, protocols, and design challenges. *arXiv preprint arXiv:2508.10146*, 2025.

[6] Michael Rosemann, Jan vom Brocke, Amy Van Looy, and Flavia Santoro. Business process management in the age of ai–three essential drifts. *Information Systems and e-Business Management*, 22(3):415–429, 2024.

[7] Reply S.p.A. Company profile. URL `https://www.reply.com/it/company-profile`. Visitato il 2026-02-07.

[8] Mathias Weske. Motivation and definitions. In *Business Process Management: Concepts, Languages, Architectures*, chapter 2.3, pages 39–43. Springer, Berlin, Heidelberg, 2012.

[9] Mathias Weske. Motivation and definitions. In *Business Process Management: Concepts, Languages, Architectures*, chapter 1.4, page 21. Springer, Berlin, Heidelberg, 2012.

[10] Mathias Weske. Motivation and definitions. In *Business Process Management: Concepts, Languages, Architectures*, chapter 1.2, pages 11–15. Springer, Berlin, Heidelberg, 2012.

[11] Wil MP Van der Aalst. Business process management: a comprehensive survey. *International Scholarly Research Notices*, 2013(1):507984, 2013.

[12] Bruce Silver. *BPMN method and style*, volume 2. Cody-Cassidy Press Aptos, 2011.

[13] Mathias Weske. Business process management architectures. In *Business Process Management: Concepts, Languages, Architectures*, chapter 4.1, pages 126–149. Springer, Berlin, Heidelberg, 2012.

[14] Mathias Weske. Business process management architectures. In *Business Process Management: Concepts, Languages, Architectures*, chapter 3.11, pages 120–123. Springer, Berlin, Heidelberg, 2012.

[15] Angelo Casciani, Mario L Bernardi, Marta Cimitile, and Andrea Marrella. Conversational systems for ai-augmented business process management. In *International conference on research challenges in information science*, pages 183–200. Springer, 2024.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[17] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[18] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

[19] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[20] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403, 2024.

[21] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[22] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72, 2025.

[23] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[24] Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*, 2024.

[25] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

[26] Na Liu, Liangyu Chen, Xiaoyu Tian, Wei Zou, Kaijiang Chen, and Ming Cui. From llm to conversational agent: A memory enhanced architecture with fine-tuning of large language models. *arXiv preprint arXiv:2401.02777*, 2024.

[27] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

[28] Lang Mei, Siyu Mo, Zhihan Yang, and Chong Chen. A survey of multimodal retrieval-augmented generation. *arXiv preprint arXiv:2504.08748*, 2025.

[29] LangChain. Langchain documentation, . URL `https://docs.langchain.com/oss/python/langchain/overview`. Visitato il 2025-11-15.

[30] LangChain. Langchain expression language, 8 2023. URL `https://blog.langchain.com/langchain-expression-language/`. Visitato il 2025-11-15.

[31] LangChain. Langgraph documentation, . URL `https://docs.langchain.com/oss/python/langgraph/overview`. Visitato il 2025-11-15.

[32] LangChain. Langsmith documentation, . URL `https://docs.langchain.com/langsmith/home`. Visitato il 2025-11-15.

[33] Elena Filipovska, Ana Mladenovska, Jovana Dobreva, Dimitar Kitanovski, Goran Mitrov, Petre Lameski, and Eftim Zdravevski. Evaluation of vector databases and llms in rag-based multi-document question answering. In *International Conference on ICT Innovations*, pages 3–18. Springer, 2024.

[34] Chroma. Chroma documentation. URL `https://docs.trychroma.com/docs/overview/introduction`. Visitato il 2025-11-18.

[35] OpenAI. Gpt-5 nano documentation, . URL `https://platform.openai.com/docs/models/gpt-5-nano`. Visitato il 2025-11-19.

[36] OpenAI. text-embedding-3-small model documentation, . URL `https://platform.openai.com/docs/models/text-embedding-3-small`. Visitato il 2025-11-19.

[37] OpenAI. Vector embeddings guide - openai platform documentation, . URL `https://platform.openai.com/docs/guides/embeddings`. Visitato il 2025-11-19.

[38] Ollama. Nomic embed text model. URL `https://ollama.com/library/nomic-embed-text`. Visitato il 2025-11-19.

[39] Ollama. Ollama's documentation, 2026. URL `https://docs.ollama.com/`. Visitato il 2025-11-19.

[40] Nomic AI. Text embedding. URL `https://docs.nomic.ai/api/embeddings-and-retrieval/text-embedding`. Visitato il 2025-11-19.

[41] LangChain. Langchain integrations: All providers, . URL `https://docs.langchain.com/oss/python/integrations/providers/all_providers`. Visitato il 2025-11-20.

[42] FastAPI. Fastapi documentation. URL `https://fastapi.tiangolo.com/`. Visitato il 2025-11-20.

[43] Unstructured. Unstructured documentation, . URL `https://docs.unstructured.io/open-source/introduction/overview`. Visitato il 2025-11-20.

[44] Unstructured. Unstructured partitioning pdf, . URL `https://docs.unstructured.io/open-source/core-functionality/partitioning#partition-pdf`. Visitato il 2025-11-20.

[45] Narayan S Adhikari and Shradha Agarwal. A comparative study of pdf parsing tools across diverse document categories. *arXiv preprint arXiv:2410.09871*, 2024.

[46] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

[47] Unstructured. Unstructured "by-title" chunking strategy, 2024. URL `https://docs.unstructured.io/open-source/core-functionality/chunking#%E2%80%9Dby-title%E2%80%9D-chunking-strategy`. Visitato il 2025-11-22.

[48] Kisung You. Semantics at an angle: When cosine similarity works until it doesn't. *arXiv preprint arXiv:2504.16318*, 2025.

[49] Docker. Docker overview documentation. URL `https://docs.docker.com/get-started/docker-overview/`. Visitato il 2025-11-25.

[50] Kubernetes. Kubernetes concepts documentation. URL `https://kubernetes.io/it/docs/concepts/`. Visitato il 2025-11-25.

[51] Amer Farea, Zhen Yang, Kien Duong, Nadeesha Perera, and Frank Emmert-Streib. Evaluation of question answering systems: complexity of judging a natural language. *ACM Computing Surveys*, 58(1):1–43, 2025.

[52] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. Evaluation of retrieval-augmented generation: A survey. In *CCF Conference on Big Data*, pages 102–120. Springer, 2024.

[53] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[54] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[55] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[56] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th conference of the european chapter of the association for computational linguistics: system demonstrations*, pages 150–158, 2024.

[57] Ragas. Introduction - ragas documentation, 2026. URL `https://docs.ragas.io/en/stable/`. Visitato il 2025-12-20.

[58] Ragas. Faithfulness, . URL `https://docs.ragas.io/en/latest/concepts/metrics/available_metrics/faithfulness/`. Visitato il 2025-12-20.

[59] Ragas. Response relevancy, . URL `https://docs.ragas.io/en/latest/concepts/metrics/available_metrics/answer_relevance/`. Visitato il 2025-12-20.

[60] Ragas. Answer correctness, . URL `https://docs.ragas.io/en/latest/concepts/metrics/available_metrics/answer_correctness/`. Visitato il 2025-12-20.

[61] OpenAI. Gpt-4o mini documentation, . URL `https://developers.openai.com/api/docs/models/gpt-4o-mini`. Visitato il 2025-12-21.

[62] Ragas. Generating a synthetic test set for rag-based question answering with ragas, . URL `https://docs.ragas.io/en/stable/howtos/applications/singlehop_testset_gen/`. Visitato il 2026-1-21.

[63] LangChain. Persistence, langgraph documentation, 2026. URL `https://docs.langchain.com/oss/python/langgraph/persistence`. Visitato il 2026-2-15.