



Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale in Big Data Engineering

*HieRAG: Hierarchical
Retrieval-Augmented Generation
with Multi-Level Caching and
Reranking*

Anno Accademico 2024/2025

Relatore
Ch.mo Prof. Vincenzo Moscato
Ing. Marco Postiglione
Ing. Antonio Romano
Ing. Giuseppe Riccio
Ing. Mariano Barone

Candidato
Pasquale Mosca
matr. M63001532

A mia madre, mio padre e alle mie sorelle,
per il loro amore costante, il sostegno silenzioso e la fiducia che non è
mai mancata.

Ai miei amici di sempre,
per aver condiviso con me ogni tappa di questo percorso, con
leggerezza, ironia e vicinanza autentica.

Ai miei relatori,
per la disponibilità, la guida preziosa e la competenza con cui mi
hanno accompagnato nello sviluppo di questa tesi.

A tutti voi,
grazie di cuore per aver reso possibile questo traguardo.

Purtroppo nella vita, piaccia o non piaccia, esistono le categorie

Massimiliano Allegri

Abstract

I modelli di generazione di testo (Large Language Models, LLM) rappresentano lo stato dell’arte in numerosi compiti di elaborazione del linguaggio naturale, ma presentano limiti strutturali nell’accesso a conoscenze esterne aggiornate. Il paradigma Retrieval-Augmented Generation (RAG) è stato introdotto per superare tale limitazione, integrando la generazione testuale con tecniche di retrieval che permettono di recuperare i documenti da basi di conoscenza esterne. Tuttavia, la fase di retrieval costituisce spesso un collo di bottiglia computazionale, soprattutto in scenari con grandi moli di dati o interrogazioni ripetitive. Questa tesi propone un’architettura innovativa denominata **Hierarchical RAG**, progettata per ottimizzare l’efficienza del retrieval nei sistemi RAG attraverso l’introduzione di una cache multilivello ispirata ai principi delle memorie gerarchiche nei sistemi operativi. Il sistema è strutturato in diversi livelli, ciascuno caratterizzato da una diversa capacità, e integra un motore di fallback per le operazioni di nearest-neighbor search in caso di cache miss. Una logica dinamica di promozione e degradazione dei documenti consente di aggiornare

la cache dinamicamente in base alla natura delle query nel tempo. È inoltre previsto un modulo opzionale di reranking neurale per affinare ulteriormente la selezione dei documenti.

Attraverso una serie di esperimenti condotti su un framework RAG completo, il sistema è stato valutato in termini di accuratezza, latenza, e riuso delle informazioni. I risultati dimostrano che **Hierarchical RAG** consente di ridurre significativamente il numero di interrogazioni dirette al database vettoriale, migliorando al contempo la qualità e la stabilità delle risposte generate. Il sistema proposto si configura quindi come una soluzione efficiente, modulare e scalabile per il retrieval in sistemi generativi basati su conoscenza.

Contents

Abstract	ii
1 Introduzione	1
2 Related work	5
2.1 Tecniche di retrieval	5
2.2 Framework alternativi al retrieval	6
2.3 Limitazioni in letteratura	7
3 Metodologia	9
3.1 Panoramica del sistema	10
3.2 Embedding	10
3.3 Cache gerarchica	12
3.4 Recupero dei documenti e aggiornamento della cache .	13
3.5 Fallback verso il database vettoriale	13
3.6 Reranking	14
4 Sperimentazione	16
4.1 Setup di esperimento	16
4.1.1 Datasets	16

4.1.2	Configurazione Hardware	19
4.1.3	Metriche di valutazione	20
4.1.4	Parametri di training	21
4.2	Risultati	24
4.2.1	Risultati in SQuAD	24
4.2.2	Risultati HotPotQA	75
4.2.3	Risultati PubMedQA	127
4.3	Risultati complessivi	152
5	Conclusioni e sviluppi futuri	163

Chapter 1

Introduzione

Gli LLM (Large Language Models) hanno rivoluzionato il mondo dell'intelligenza artificiale dimostrandosi efficaci nella generazione e comprensione del testo. Tuttavia, questi modelli presentano un limite che è l'impossibilità di accedere a informazioni aggiornate o contestualmente rilevanti che non siano state apprese durante la fase di addestramento. Per superare questo limite, è emerso il framework **Retrieval-Augmented Generation (RAG)** [1] che introduce un modulo che permette di raccogliere documenti rilevanti da una base di conoscenza esterna che consentono al modello LLM di rispondere alle domande poste.

Nel panorama attuale ci sono moltissime soluzioni che propongono sistemi RAG come, ad esempio:

- **Sparse RAG**, basato sui modelli classici BM25. Il retriever BM25 ordina i documenti in base alla frequenza dei termini (TF-

IDF) e alla normalizzazione rispetto alla del documento. Data una query, il BM25 raccoglie i migliori documenti dal database vettoriale e li fornisce al modello LLM come finestra di contesto

- **Dense RAG**, che utilizza embedding per rappresentare sia le domande sia i documenti in uno spazio semantico vettoriale condiviso. Data una query, il dense retrieval seleziona i migliori documenti che si avvicinano semanticamente alla query e infine questi documenti vengono passati al LLM per generare la risposta.

Il primo offre efficienza ma tende a fallire quando il matching semantico tra query e documento è debole; il secondo, invece, è più robusto nella fase di retrieval migliorando la qualità dei documenti raccolti ma presenta una difficoltà computazionale superiore e dipende fortemente dai modelli di embedding utilizzati. Entrambi gli approcci presentano dei limiti ossia tempi di esecuzione elevati e scarsa efficienza.

In risposta a queste limitazioni, recenti lavori hanno proposto il paradigma alternativo del **Cache-Augmented Generation** (CAG) [2], che mira ad eliminare completamente la fase di retrieval in tempo reale. CAG si basa sulla pre-elaborazione dei documenti e memorizzazione del contesto direttamente nella finestra di contesto del modello LLM, sfruttando le capacità dei moderni modelli a lungo contesto. Tale approccio è particolarmente efficace quando la base di conoscenza è di dimensioni

gestibili e può essere interamente precaricata. Tuttavia, CAG presenta due limitazioni principali:

1. non è applicabile quando il corpus è troppo grande per essere gestito in un'unica finestra di contesto
2. non prevede alcun meccanismo di aggiornamento della conoscenza durante l'inferenza.

Questa tesi si colloca esattamente tra efficienza e scalabilità, con l'obiettivo di superare le limitazioni strutturali di RAG e CAG mediante un sistema ibrido che combina caching dinamico e retrieval vettoriale e porta una serie di contributi innovativi:

- Progettazione di una cache gerarchica ispirata alla cache implementata nei calcolatori elettronici in grado di memorizzare i documenti più rilevanti dinamicamente, con meccanismi di promozione e degrado tra livelli.
- Integrazione della cache con un indice vettoriale, che funge da fallback intelligente solo in assenza di documenti pertinenti nei livelli cache, riducendo così il carico computazionale.
- Comparazione con i paradigmi Sparse RAG, Dense RAG e CAG, evidenziando il compromesso ottimale tra latenza, qualità del contesto e scalabilità

- Estensione facoltativa con un reranker per affinare ulteriormente la selezione dei documenti in fase di retrieval

Attraverso una serie di esperimenti, viene dimostrato come il sistema ***Hierarchical RAG*** possa essere una buona alternativa ai paradigmi esistenti, mantenendo un'elevata qualità delle risposte generate e migliorando significativamente i tempi medi di inferenza.

Chapter 2

Related work

2.1 Tecniche di retrieval

Il retrieval rappresenta la prima componente fondamentale in una pipeline RAG. Le tecniche più diffuse si suddividono in approcci sparse e dense. I metodi sparse [11], come BM25, si basano su rappresentazioni testuali tradizionali e utilizzano metriche basate su frequenze dei termini. Questi sistemi sono leggeri dal punto di vista computazionale, ben compresi e altamente interpretabili. Tuttavia, soffrono gravemente nel gestire sinonimi, parafrasi risultando spesso inefficaci quando le domande vengono poste in linguaggio naturale.

Al contrario, i metodi dense [9] adottano modelli neurali per produrre rappresentazioni vettoriali continue di query e documenti. Sistemi come DPR [8] e i più recenti encoder duali hanno mostrato un'elevata efficacia in termini di similarità semantica, specialmente in domini

aperti o complessi. Tali metodi richiedono, tuttavia, la gestione di indici ad alta dimensionalità e motori di nearest-neighbor search come FAISS [3].

Indipendentemente dall’approccio che si decide di applicare, i sistemi di retrieval tradizionali sono progettati in modo tale che ogni query viene trattata come indipendente dalle precedenti. Non viene sfruttata alcuna forma di persistenza o riuso delle informazioni già recuperate, né esistono meccanismi di caching strutturati in grado di alleggerire il carico computazionale in presenza di query ridondanti o simili a quelle precedenti.

2.2 Framework alternativi al retrieval

Recenti studi ha messo in discussione la necessità stessa della fase di retrieval. Con l’avvento di modelli che permettono di avere un lungo contesto, come LLaMA 3.1, è diventato possibile precaricare una enorme quantità di testo direttamente nella finestra di input del modello. Il paradigma di **Cache-Augmented Generation (CAG)** sfrutta questa possibilità cioè tutti i documenti rilevanti vengono caricati preventivamente, e durante l’inferenza il modello utilizza una KV-cache precomputata, evitando il recupero dinamico. Questo approccio elimina completamente la latenza associata al retrieval e semplifica notevolmente l’architettura del sistema riducendo i tempi di esecuzione che si limiteranno semplicemente ai tempi di generazione della risposta

da parte del modello.

CAG però risulta essere inapplicabile sia quando il numero di documenti eccede la finestra di contesto del modello sia quando non si hanno a disposizione risorse hardware elevate per memorizzare nella cache KV tutti i documenti. Inoltre, l'approccio non è dinamico: non è possibile aggiornare, sostituire o riorganizzare i documenti presenti in cache in base alle domande dell'utente. In pratica, CAG è una soluzione efficace ma rigida, non adatta a scenari realistici con conoscenza distribuita e dinamica.

2.3 Limitazioni in letteratura

L'idea di introdurre una cache nei sistemi di NLP non è nuova. Diversi lavori hanno proposto tecniche per il riutilizzo delle risposte sfruttando la cache per evitare di appesantire il calcolo computazionale. Tuttavia, nella letteratura RAG, l'integrazione di un sistema di caching è ancora rara e priva di formalizzazione architettonale. Non esistono, ad oggi, implementazioni che modellano la cache ispirata alle architetture dei calcolatori elettronici né meccanismi di promozione e declassamento dei documenti in base alla rilevanza rispetto alle query.

Da questa analisi emerge che la letteratura attuale affronta in maniera efficace ciascuno dei singoli problemi (retrieval e caching), ma non propone soluzioni integrate che li trattino come parti di un'architettura modulare e interconnessa. Il comportamento dei sistemi attuali resta

inefficiente e privo di ottimizzazione temporale. Le query vengono gestite come eventi indipendenti, senza meccanismi per sfruttare le conoscenze già acquisite.

Questa tesi si propone di colmare questo vuoto progettando un sistema che integri in modo coerente:

- un motore di retrieval vettoriale ad alte prestazioni
- una cache multi-livello
- una logica dinamica di aggiornamento ispirata alla gestione della memoria dei sistemi operativi,
- opzionalmente, un reranker per affinare i risultati selezionati.

Il sistema **Hierarchical RAG** non solo introduce un caching efficiente, ma propone una nuova visione dell’interazione tra retrieval e memoria nei sistemi generativi, con l’obiettivo di ottimizzare le prestazioni senza sacrificare la qualità delle risposte.

Chapter 3

Metodologia

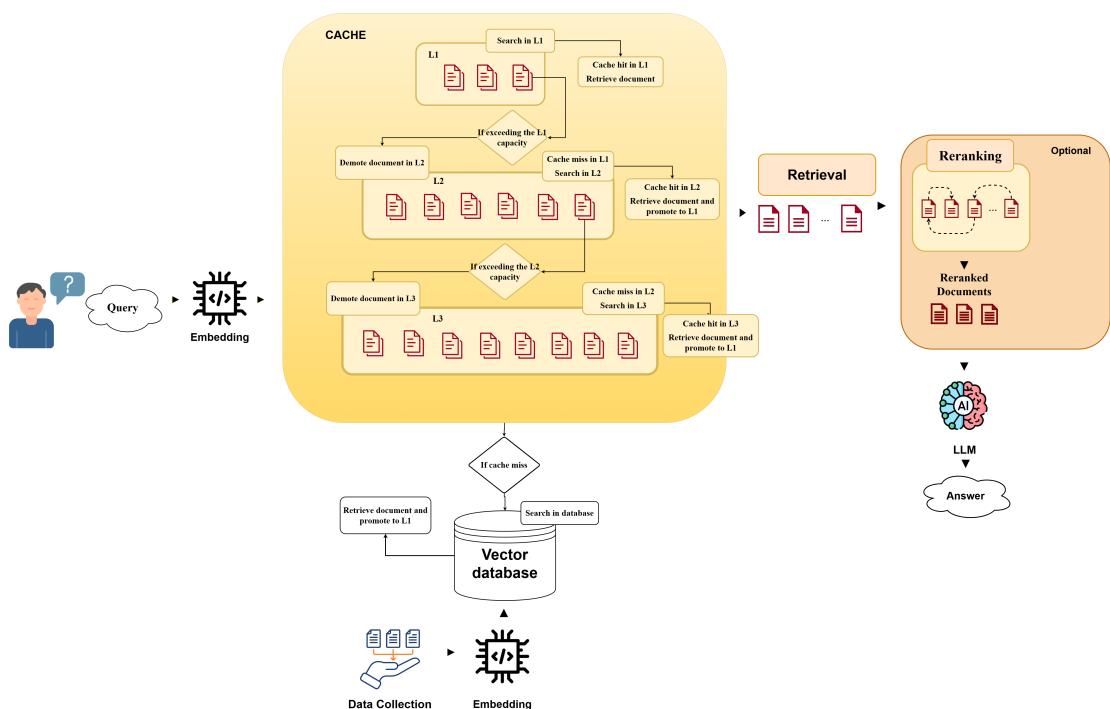


Figure 3.1: HieRAG

3.1 Panoramica del sistema

Il sistema che proponiamo si basa su un’architettura di retrieval augmentation realizzata attraverso cache gerarchica, denominata **Hierarchical RAG**, progettata per migliorare l’efficienza e le operazioni di retrieval in ambienti in cui è presente un elevato numero di documenti. Tale sistema è integrato in un framework di question answering, dove i documenti recuperati vengono utilizzati come contesto per modelli di generazione di testo come gli LLM. Il componente chiave è rappresentato dalla cache a più livelli che sfrutta un indice vettoriale per accelerare l’accesso ai contenuti più rilevanti rispetto alle query poste.

3.2 Embedding

Nel contesto dell’elaborazione del linguaggio naturale (NLP), il testo non può essere direttamente manipolato dai modelli computazionali senza una sua trasformazione in forma numerica. Per questo motivo, ogni documento del corpus viene convertito in una rappresentazione vettoriale densa, denominata *embedding*. Gli embedding sono vettori di dimensione fissa che codificano l’informazione semantica di un testo (frase, paragrafo o intero documento) in uno spazio continuo. A differenza delle rappresentazioni sparse (come one-hot encoding o TF-IDF), gli embedding catturano relazioni semantiche tra parole e frasi: testi con significato simile avranno vettori vicini nello spazio vettoriale.

La generazione degli embedding viene effettuata tramite un modello di *sentence embedding* basato su architetture neurali pre-addestrate. Questi modelli (ad esempio basati su Transformer) sono progettati per proiettare input testuali in uno spazio semantico denso, in cui la distanza tra i vettori riflette il grado di similarità semantica tra i testi originali.

Una volta generati, i vettori vengono normalizzati, ossia proiettati su una sfera unitaria. Questa normalizzazione consente di utilizzare la *distanza euclidea* come approssimazione della *similarità coseno*, che è la misura più comunemente impiegata per confrontare vettori semantici. La similarità coseno è definita come il coseno dell'angolo tra due vettori e assume valori tra -1 e 1 , dove 1 indica una direzione identica (massima similarità), 0 indica ortogonalità (nessuna correlazione), e -1 indica opposizione totale.

I vettori così ottenuti vengono infine salvati in un **database vettoriale**, una struttura ottimizzata per la *similarity search* tra vettori ad alta dimensionalità. A differenza di un database tradizionale, che opera su dati strutturati e consente query esatte, un database vettoriale è progettato per supportare ricerche approssimative (*Approximate Nearest Neighbor Search*, ANN), che restituiscono i vettori più vicini a un dato query vector in termini di distanza (euclidea, coseno o altre metriche).

3.3 Cache gerarchica

Il cuore del sistema, che sarebbe anche la novità di questa tesi, consiste in una cache strutturata su tre livelli gerarchici: **L1**, **L2**, e **L3**, ciascuno dei quali conserva documenti recuperati in precedenti interrogazioni in base alla loro rilevanza. La cache è stata progettata secondo una logica LRU (Least Recently Used) con capienze decrescenti dalla base (L3) al vertice (L1).

- **L1** rappresenta la cache di primo livello, riservata ai contenuti con il punteggio di similarità più elevato.
- **L2** funge da buffer intermedio per elementi precedentemente in L1 ma degradati per mancanza di utilizzo recente.
- **L3** rappresenta l'ultimo livello prima del ricorso all'indice FAISS.

La tecnica LRU consiste nel rimuovere un elemento presente in cache che è stato meno utilizzato in favore di un nuovo documento che viene piazzato nel primo livello di cache. Per cercare i documenti rilevanti la cache consulta prima nel primo livello L1 poi nel livello L2 se non ci sono abbastanza documenti altrimenti consulta il livello L3. In assenza di risultati sufficientemente rilevanti, si attiva il fallback verso il database vettoriale, con conseguente promozione dei documenti trovati al primo livello della cache.

3.4 Recupero dei documenti e aggiornamento della cache

Quando arriva la query, il sistema genera il relativo embedding e confronta tale vettore numerico con quelli dei documenti presenti in ciascun livello della cache, applicando un filtro basato su una soglia di rilevanza fissata. I risultati vengono ordinati secondo il punteggio di similarità e restituiti se soddisfano i criteri di rilevanza. I documenti recuperati vengono quindi promossi a livelli superiori della cache per garantire un accesso più rapido a query future simili. Nel caso in cui si supera il limite massimo di capacità di un livello, gli elementi meno utilizzati sono in fondo al livello di cache e vengono degradati al livello inferiore. Quando si supera la capacità dell'ultimo livello, gli elementi che superano il limite massimo vengono semplicemente tagliati. L'obiettivo di questa logica è quello di preservare la dinamicità e l'efficienza della cache.

3.5 Fallback verso il database vettoriale

Nel caso in cui nessun livello della cache contenga documenti con una similarità sufficiente, il sistema effettua una ricerca diretta nel database vettoriale. L'embedding della query viene utilizzato per identificare i vettori dei documenti più vicini nello spazio semantico. I risultati ottenuti da questa operazione costituiscono la finestra di contesto del

LLM e successivamente verranno salvati nella cache per query future. Durante tutta la fase di retrieval, sia all'interno della cache che nel database vettoriale, viene implementato un controllo di unicità sui testi recuperati per evitare ridondanza nei risultati. In particolare, si escludono documenti duplicati che, pur avendo punteggi elevati, non aggiungono informazione rispetto a quelli già selezionati.

3.6 Reranking

Sebbene il modulo di retrieval fornisca una prima selezione di documenti rilevanti rispetto a una query, questa fase si basa tipicamente su modelli bi-encoder (dense retrievers) che calcolano la similarità in modo indipendente per la query e per i documenti. Tale approccio consente una ricerca efficiente, ma introduce una limitazione strutturale: l'interazione semantica tra query e documento non è modellata congiuntamente, e questo può portare a errori di ranking, specialmente in presenza di ambiguità o contesto隐含的.

Per mitigare questo problema, è stato integrato nel sistema un modulo di **reranking**, che si occupa di ricalcolare la rilevanza tra la query e ciascun documento recuperato utilizzando un modello *cross-encoder*. A differenza dei bi-encoder, i cross-encoder prendono in input la concatenazione diretta della query e del documento, permettendo alla rete neurale di modellare in modo esplicito l'interazione tra i due testi.

Il reranker restituisce uno score di similarità per ciascuna coppia query-documento, in base al quale i documenti vengono riordinati. Vengono selezionati i top- k documenti con punteggio superiore a una soglia di rilevanza τ , i quali verranno inclusi nella finestra di contesto del modello LLM in fase di generazione. Tale operazione migliora sensibilmente la qualità semantica del contesto fornito, riducendo la probabilità che il modello generi risposte basate su documenti poco pertinenti o rumorosi.

È importante notare che il reranking è una fase computazionalmente più onerosa rispetto al retrieval iniziale: essendo un processo *non indicizzabile*, il modello deve essere eseguito per ogni coppia query-documento. Per questo motivo, il reranker viene applicato solo a un sottoinsieme ristretto dei documenti recuperati, solitamente quelli con punteggio più alto secondo il retriever.

L'integrazione del reranker all'interno del sistema HieRAG consente non solo di migliorare la qualità della generazione, ma anche di raffinare dinamicamente il contenuto della cache, favorendo la promozione dei documenti più rilevanti nei livelli superiori.

Chapter 4

Sperimentazione

4.1 Setup di esperimento

4.1.1 Datasets

Per valutare l'efficacia e l'efficienza del nostro metodo proposto, abbiamo condotto una serie di esperimenti su diversi dataset di natura diversa che sono particolarmente famosi. Tra i dataset scelti abbiamo:

- **Stanford Question Answering Dataset (SQuAD)** [7] è un dataset ampiamente utilizzato per la valutazione di sistemi di comprensione del linguaggio naturale in compiti di **machine reading comprehension**. Il dataset comprende articoli tratti da Wikipedia, accompagnati da domande. La caratteristica principale di SQuAD è che, per ogni domanda, la risposta è **una sottostringa esatta (span)** di uno dei paragrafi del contesto

fornito.

- **HotpotQA** [10] è un dataset di **question answering multi-hop**. A differenza di dataset tradizionali come SQuAD, HotpotQA richiede ai modelli di **ragionare su più documenti** per poter rispondere correttamente alle domande. Questo rende il task più complesso e più vicino a scenari reali di ricerca dell’informazione. Il dataset contiene domande, annotate manualmente, basate su contenuti tratti da Wikipedia.
- **PubMedQA** [6] è un dataset di **question answering in ambito biomedico**, introdotto per valutare la capacità dei modelli NLP di comprendere, ragionare e rispondere a domande su letteratura scientifica. Il dataset si basa su articoli tratti da **PubMed**, il principale motore di ricerca per la letteratura medica e biologica.

Per verificare l’impatto dei diversi livelli di lunghezza del testo di riferimento sulla difficoltà nella fase di retrieval, abbiamo creato tre set di test per dataset come HotPotQA e SQuAD, variando la dimensione del testo di riferimento. Ad esempio, nella configurazione del test set di SQuAD, abbiamo creato 3 documenti dal dataset dei documenti SQuAD compattando una parte dei contesti per formare un lungo testo di riferimento. Le coppie domande e risposte associate a questi documenti formano le istanze di test. Stesso ragionamento è

Dataset	Size	Docs	QA pairs
SQuAD	small	3	500
	medium	4	500
	large	7	500
HotPotQA	small	8	500
	medium	16	500
	large	32	500
PubMedQA	—	5	1000

Table 4.1: Test sets per CAG

stato applicato anche al dataset HotPotQA. È stata effettuata questa suddivisione del test set per il modello proposto nel paper CAG mentre con il sistema HieRAG, Sparse RAG e Dense RAG abbiamo lasciato i contesti così come sono forniti per valutare quanto sia efficiente la retrieval di questi documenti in confronto a CAG. Per quanto riguarda il dataset PubMedQA, non è stata fatta nessuna suddivisione del dataset in diverse dimensioni ma abbiamo deciso di sperimentarlo su tutto il dataset quindi con questa scelta il numero di documenti che CAG può usufruire è 5 a causa dei limiti di risorse hardware. Il motivo principale di questa scelta è quella di avere una visione completa del comportamento delle quattro baseline con dataset piccoli (SQuAD), mediopiccoli (HotPotQA) e grandi (PubMedQA).

Dataset	Size	Docs	QA pairs
SQuAD	small	87	500
	medium	115	500
	large	271	500
HotPotQA	small	2500	500
	medium	5000	500
	large	7405	500
PubMedQA	—	212000	1000

Table 4.2: Test sets per HieRAG, Dense RAG e Sparse RAG

4.1.2 Configurazione Hardware

Gli esperimenti sono stati eseguiti sulla piattaforma Google Colab che permette di usufruire gratuitamente la GPU T4 [5]. Per tutti gli esperimenti, abbiamo usato il modello LLaMa 3.1 8B Instruct [4]. Questo modello contiene circa 8 miliardi di parametri, collocandosi in una fascia intermedia tra modelli leggeri e modelli di dimensioni molto grandi (es. 70B). La versione Instruct è ottimizzata per seguire istruzioni in linguaggio naturale, per questo risulta particolarmente adatto ai compiti di question answering. In ambito sperimentale, LLaMA 3.1 8B Instruct è particolarmente utile per valutare architetture retrieval-augmented generation (RAG), grazie alla sua capacità di integrare input con enormi finestra di contesto e generare risposte coerenti e accurate.

4.1.3 Metriche di valutazione

Per valutare l’efficacia del sistema proposto sono state adottate diverse metriche di valutazione che misurano diversi aspetti della qualità del testo generato. Le metriche di valutazione adottate negli esperimenti sono:

- **BLEU** che è una delle più consolidate nell’ambito della valutazione automatica della generazione di testo. Essa calcola la precisione n-gram (tipicamente fino a 4-gram), confrontando i segmenti generati con uno o più testi di riferimento ed è ampiamente utilizzata per valutare modelli di risposta automatica, riassunto e dialogo.
- **ROUGE** è stata pensata specificamente per la valutazione del riassunto automatico, ma trovano applicazione anche nel question answering. A differenza di BLEU, ROUGE enfatizza il **recall**, ossia la quantità di contenuto corretto effettivamente riprodotto nella generazione. In particolare:
 - **ROUGE-1** misura l’overlap di unigrammi (parole singole) tra la generazione e il testo di riferimento
 - **ROUGE-2** estende la valutazione agli n-grammi di lunghezza due (bigrammi)
 - **ROUGE-L** utilizza la lunghezza della sottosequenza comune più lunga per valutare la similarità di struttura tra

le due frasi

- **BERTScore**, invece di confrontare le parole in modo esatto, essa utilizza rappresentazioni dense ottenute tramite modelli preaddestrati come BERT (bert-large-uncased). Le frasi vengono proiettate in uno spazio semantico e confrontate mediante similarità coseno tra token.
- **COMET** è utilizzata nella valutazione di risposte generate, grazie alla sua capacità di stimare la correttezza semantica della risposta generata rispetto a un contesto dato.

4.1.4 Parametri di training

Al fine di valutare l'efficacia del sistema HieRAG in condizioni diverse di utilizzo, è stata condotta un'analisi approfondita sull'impatto di alcuni parametri progettuali fondamentali. L'obiettivo era comprendere in che misura tali configurazioni influenzano la qualità delle risposte generate e l'efficienza del sistema in termini di accesso alla conoscenza, riuso del contesto e riduzione del richiamo del database vettoriale.

Numero di documenti da recuperare

Una prima dimensione di analisi ha riguardato il numero di documenti recuperati per ciascuna query. Sono stati testati diversi valori per il parametro **top-k**, rispettivamente top-1, top-3, top-5 e top-10, al fine

di valutare il compromesso tra qualità del contesto fornito al modello e carico informativo. Come atteso, l'utilizzo di un numero più elevato di documenti tende a migliorare le metriche di accuratezza e completezza, ma introduce un costo maggiore in termini di memoria e possibilità di rumore informativo. I risultati suggeriscono che, in presenza di un sistema di caching efficace, è possibile mantenere buone prestazioni anche con valori moderati di top-k.

Numero di livelli di cache

Una seconda sperimentazione ha riguardato la profondità della gerarchia di cache. Sono stati confrontati due scenari: un'architettura a tre livelli e una versione semplificata con un solo livello di cache. Lo scopo era misurare il contributo effettivo dei livelli inferiori nella gestione della conoscenza e nel recupero incrementale dei documenti. I risultati hanno evidenziato che l'introduzione di più livelli consente una gestione più fine del contesto e una maggiore probabilità di cache hit, soprattutto nel caso di query correlate. In particolare, la struttura a tre livelli si è dimostrata più robusta nella conservazione dei documenti rilevanti e nel riutilizzo efficace della conoscenza accumulata. Da ciò si deduce che, a parità di capacità complessiva, una cache più stratificata risulta preferibile rispetto a una piatta, anche dal punto di vista dell'efficienza.

Capacità dei livelli di cache

Infine, è stata analizzata l'influenza della capacità massima dei livelli di cache, sia nel caso della versione multilivello che in quella a singolo livello. Sono state valutate tre configurazioni principali:

- una cache a tre livelli con capacità elevata per ciascun livello (250 documenti da memorizzare nel livello L1, 500 in L2 e 1000 in L3)
- una cache a tre livelli con capacità limitata (25 documenti da memorizzare in L1, 50 in L2 e 100 in L3)
- una cache a singolo livello con capacità bassa (25 documenti da memorizzare in L1)

Questa analisi ha permesso di osservare come cambia la qualità delle risposte generate man mano che si aumenta o diminuisce la dimensione della cache e soprattutto quantificare i tempi di inferenza dei singoli scenari di utilizzo. Le osservazioni sperimentali indicano che una maggiore capacità per livello si traduce in una minore frequenza di **cache miss** e in un miglior riuso dei documenti precedentemente recuperati, con effetti positivi sia sull'accuratezza delle risposte che sulla riduzione del numero di accessi al database vettoriale.

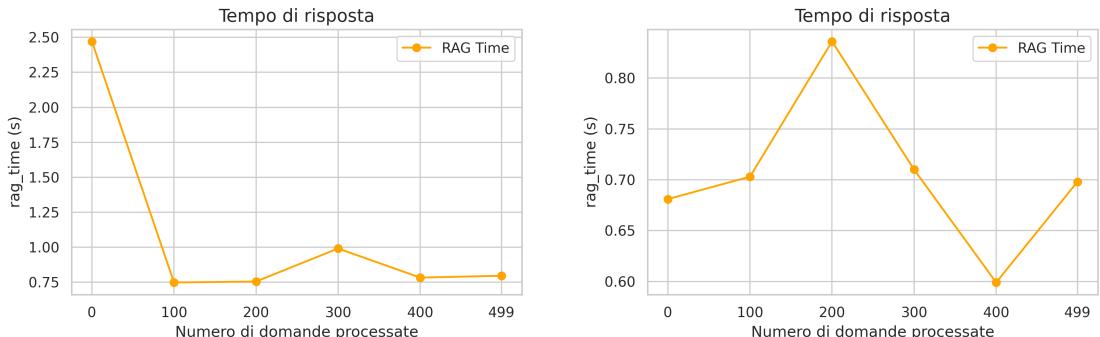
4.2 Risultati

4.2.1 Risultati in SQuAD

Impatto del sistema HieRAG

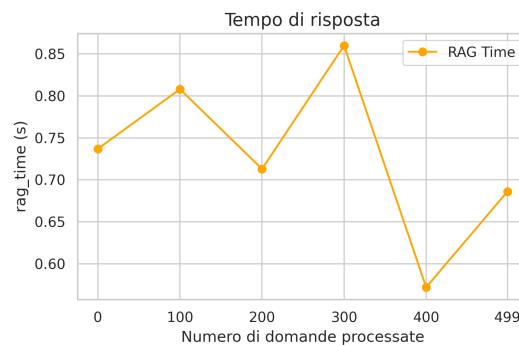
Nel contesto del dataset SQuAD, composto da appena 300 documenti, tutti i sistemi HieRAG raggiungono tempi di risposta molto contenuti, generalmente inferiori al secondo. Questo comportamento riflette un'efficienza strutturale che va ben oltre il semplice meccanismo di retrieval. La chiave risiede nell'impatto della cache in un contesto di dimensioni ridotte: i documenti più rilevanti vengono rapidamente identificati e memorizzati, con una saturazione della cache che avviene già nelle fasi iniziali del processo. Nel sistema con **capacità elevata**, si osserva un tempo iniziale più alto dovuto al setup dei tre livelli, ma la latenza scende rapidamente grazie al progressivo aumento di hit nei livelli superiori. Il sistema con **capacità ridotta** mostra invece un'efficienza ancora più marcata, con tempi stabili e bassi: la ridotta dimensione del corpus favorisce una maggiore probabilità di hit nella cache L1, minimizzando l'accesso ai livelli L2/L3 o al database FAISS. Infine, il sistema **monolivello** beneficia al massimo della compattezza del dominio, riuscendo a saturare completamente L1 in poche interazioni e garantendo così tempi stabili e bassissimi senza necessità di gerarchia. In sintesi, in uno scenario a bassa cardinalità documentale come SQuAD, **la cache non solo è estremamente efficace, ma**

addirittura dominante, rendendo superfluo il ricorso a strategie di retrieval più complesse.



(a) Tempo di risposta di HieRAG a capacità elevata

(b) Tempo di risposta di HieRAG a capacità ridotta



(c) Tempo di risposta di HieRAG monolivello

Figure 4.1: Andamento del tempo di risposta medio dei tre sistemi HieRAG su SQuAD. Il sistema con cache gerarchica completa (a sinistra) presenta una latenza iniziale più elevata ma tende rapidamente alla stabilità. La versione con capacità ridotta (a destra) mostra tempi di risposta ancora più contenuti e regolari. Il sistema monolivello (sotto) raggiunge stabilità in pochissimi step, evidenziando l’efficacia della cache L1 in un corpus di dimensioni ridotte.

I tempi di risposta così bassi sono giustificati dall’analisi dei tempi medi di retrieval per livello di cache che evidenzia in modo netto l’efficacia della cache L1 come unico meccanismo di ottimizzazione. In tutti e tre i sistemi (cache piena, ridotta e monolivello), L1 viene

sfruttata a pieno, mentre L2 e L3 risultano inutilizzate. Questa assenza è coerente con la natura compatta del corpus: il numero di documenti è talmente contenuto da rendere superflua la suddivisione gerarchica. La latenza di accesso alla cache L1 è significativamente inferiore rispetto a FAISS (intorno a 27ms contro oltre 40ms), con un guadagno immediato e costante in termini di performance.

L'impatto della cache, in questo caso, si manifesta più nella semplicità ed efficienza strutturale che nella profondità del meccanismo. A differenza dei dataset più estesi (come PubMedQA o HotPotQA), dove il bilanciamento tra livelli consente di scalare con l'aumento delle domande, su SQuAD è sufficiente un primo livello reattivo per garantire tempi di risposta eccellenti. Questa evidenza sottolinea un punto critico: **la progettazione della cache deve essere proporzionata alla dimensione del corpus**, evitando overhead strutturali non sfruttati. In altri termini, su dataset ristretti, una cache L1 ampia e mirata è preferibile a una struttura multilivello complessa ma ridondante.

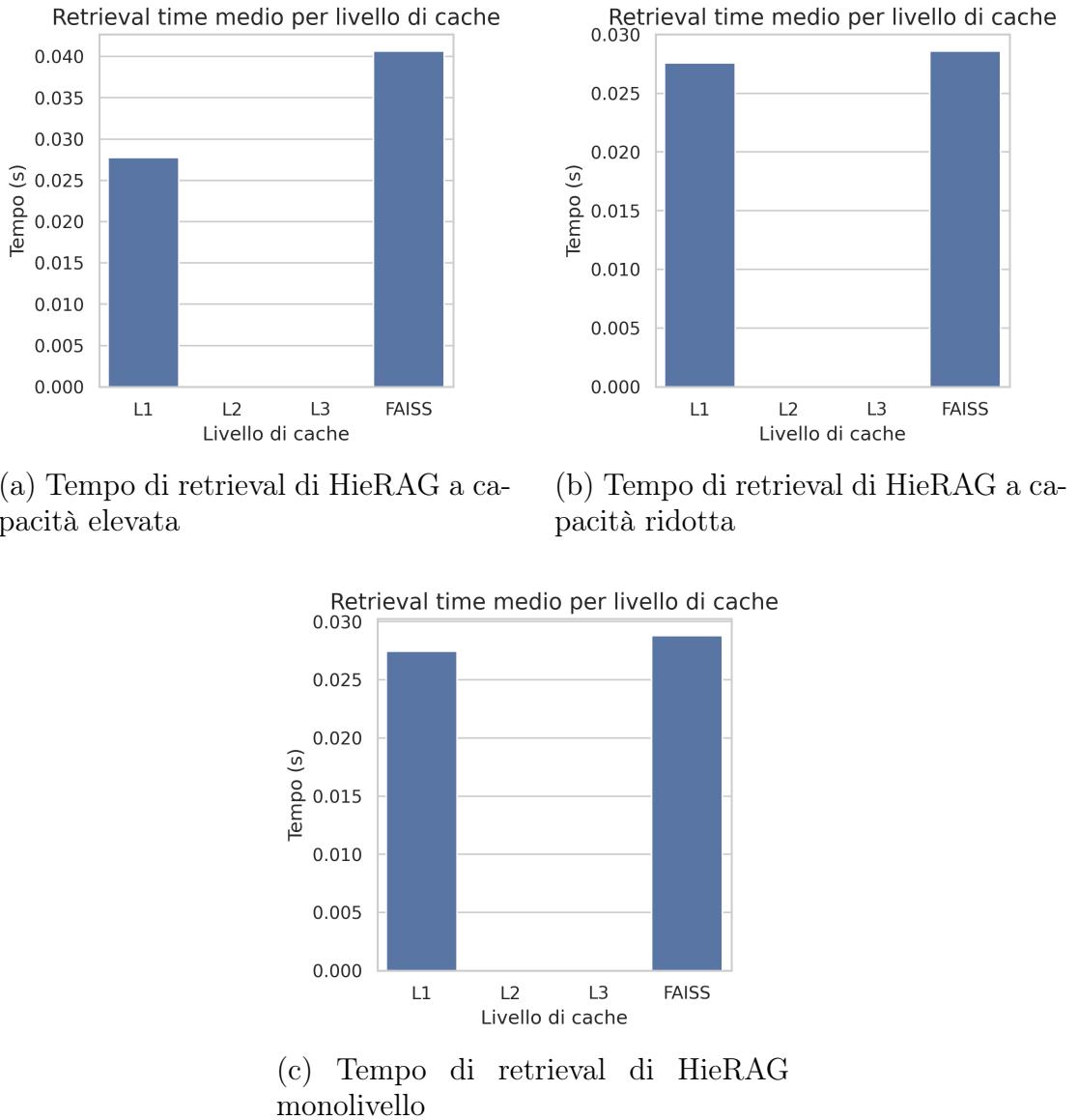


Figure 4.2: Confronto dei tempi medi di retrieval per livello di cache su SQuAD. In tutti i sistemi (cache completa, ridotta e monolivello), solo la cache L1 viene utilizzata, evidenziando la sua efficacia in un contesto con corpus limitato. La latenza di accesso a L1 risulta inferiore rispetto al vector database FAISS, confermando che, in dataset di piccole dimensioni, una struttura di cache semplice e piatta è più efficiente di una gerarchia multilivello

L’analisi della riduzione percentuale dei tempi di retrieval rispetto al database vettoriale, in continuità con i grafici precedenti sui tempi medi, evidenzia le limitazioni strutturali dell’approccio caching su un

corpus di dimensioni contenute come SQuAD. Nella configurazione a capacità elevata, la cache L1 consente un risparmio del tempo di retrieval pari a circa il 32%, grazie alla rapidità di accesso diretto ai chunk frequentemente riutilizzati. Tuttavia, l'assenza di hit nei livelli L2 e L3 (già osservata nei grafici di retrieval time) conferma che la gerarchia cache in questo caso non viene realmente sfruttata, rendendo di fatto inefficaci le componenti più profonde. La versione a capacità ridotta, pur mantenendo tempi competitivi, mostra un beneficio marginale in termini di riduzione (poco sopra il 3%), indicando che la dimensione ridotta della cache non permette di contenere dati riutilizzabili in modo significativo. Anche nel sistema monolivello, il guadagno si mantiene basso (circa 4.6%) perché il contesto è troppo limitato per permettere una reale ottimizzazione: i documenti rilevanti non ricorrono abbastanza frequentemente da massimizzare la cache reuse. In sintesi, mentre la cache migliora leggermente l'efficienza, il beneficio complessivo sul retrieval rimane contenuto e meno determinante rispetto a scenari con corpus di grandi dimensioni.

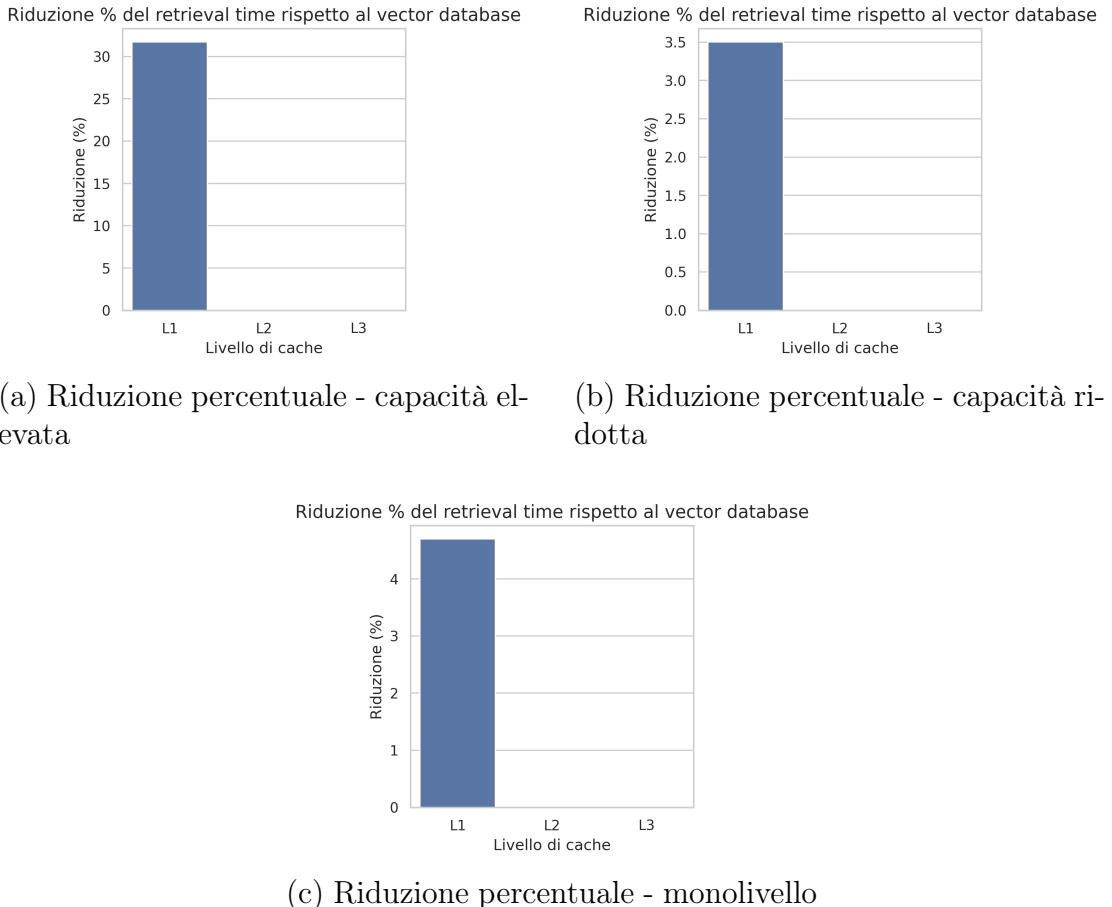


Figure 4.3: Riduzione percentuale dei tempi di retrieval rispetto all’accesso diretto al database vettoriale FAISS per ciascuna configurazione di HieRAG su SQuAD. L’impatto della cache è significativo solo a livello L1, mentre i livelli L2 e L3 risultano trascurabili, a conferma dell’utilizzo limitato della gerarchia in un contesto con numero ridotto di documenti

L’analisi incrociata dei grafici relativi allo stato di riempimento della cache e alla distribuzione dei cache hit nel dataset SQuAD conferma in modo coerente l’efficacia del solo livello L1. In tutti e tre i sistemi (cache gerarchica completa, ridotta e monolivello), i livelli L2 e L3 rimangono praticamente inutilizzati durante l’intera esecuzione. Tale comportamento trova piena spiegazione nei grafici a torta: oltre

il 90% dei retrieval avviene infatti già al primo livello (L1), rendendo superfluo l’accesso ai livelli inferiori e, di conseguenza, il loro popolamento.

Questo fenomeno è strettamente legato alle dimensioni del corpus (300 documenti), che risulta sufficientemente contenuto da poter essere efficacemente “coperto” da una cache L1 ben progettata. La capacità della L1 si dimostra infatti adeguata a garantire un tasso di hit elevato, limitando drasticamente la necessità di consultare il vector database o i livelli L2/L3. Il risultato è una configurazione leggera ed efficiente, in cui l’overhead di gestione dei livelli più profondi può essere evitato senza penalizzazioni sulle performance. In questo scenario, l’adozione di una cache monolivello appare non solo sufficiente, ma addirittura preferibile.

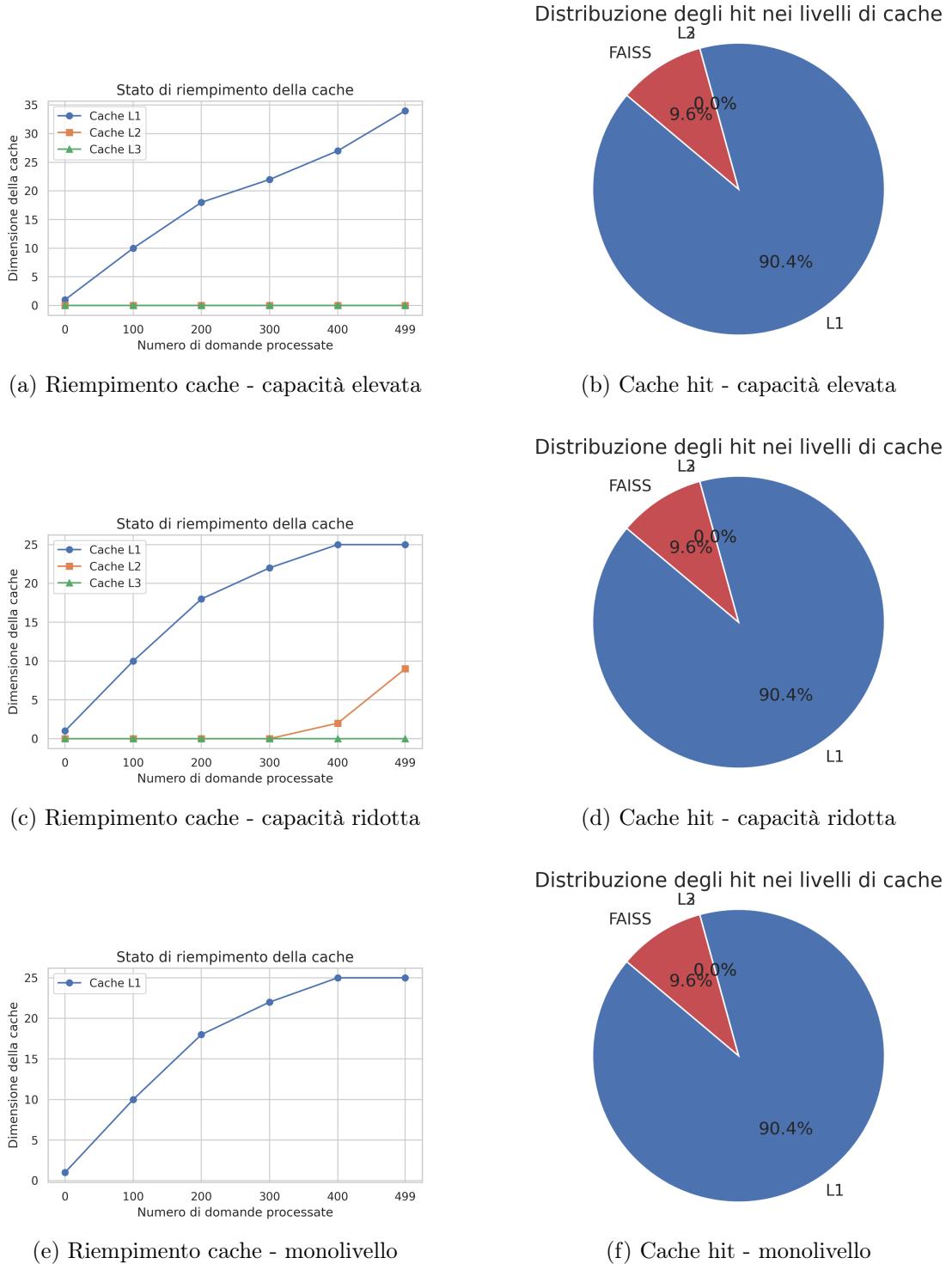


Figure 4.4: Stato di riempimento delle cache (sinistra) e distribuzione dei cache hit (destra) nei tre sistemi HieRAG su SQuAD. In tutti i casi, il livello L1 viene progressivamente popolato ed è sufficiente a soddisfare oltre il 90% delle richieste, rendendo marginale o nullo l'utilizzo dei livelli L2 e L3. Questo comportamento è coerente con la dimensione ridotta del corpus (300 documenti), che consente al solo livello L1 di garantire un'elevata copertura informativa e tempi di retrieval contenuti.

Confronto baselines: SQuAD small

Tempi di inferenza Nel benchmark SQuAD Small, l'apparente superiorità dei sistemi gerarchici in termini di tempi di inferenza si spiega con una condizione favorevole: il **basso carico informativo** richiesto per generare risposte corrette. In questo scenario, la struttura a cache multistrato di HieRAG e HieRAG++ non viene realmente sollecitata, e il sistema opera quasi sempre in regime di **cache hit a livello L1**, evitando costi di fallback sui livelli più profondi. Questo porta a tempi di retrieval estremamente contenuti e stabili ($\sim 0.027\text{s}$), anche al crescere di top-k.

Il reranker in HieRAG++ introduce un overhead minimo in questa fase, ma ciò è in parte attribuibile al **limite strutturale del dataset stesso**: la brevità e semplicità del contesto riducono l'impatto computazionale della fase di scoring, mascherando i costi reali del reranking. È quindi fuorviante valutare l'efficienza di HieRAG++ solo in base a questi numeri: in condizioni più realistiche o con input più lunghi, il trade-off tra qualità e latenza potrebbe diventare più marcato.

Sparse RAG, pur mostrando tempi di retrieval eccellenti grazie al BM25, risente della **mancanza di controllo sul contesto generato**: l'assenza di caching o filtraggio comporta un tempo di generazione che cresce significativamente con top-k, rivelando una pipeline incapace di ottimizzare il carico computazionale. Dense RAG soffre ancora di più questo limite, a causa della maggiore complessità degli embedding

densamente calcolati e dell’assenza di riuso di memoria documentale.

Nel caso di CAG, l’assenza di una fase di selezione documentale porta a un approccio generativo grezzo, che accumula tutto il contesto possibile senza distinzione. Questo compromette pesantemente l’efficienza (1.702s), evidenziando quanto sia penalizzante non avere una logica di retrieval, anche in dataset di dimensioni ridotte.

In sintesi, le buone prestazioni di HieRAG e HieRAG++ sono in parte dovute all’efficacia strutturale della cache, ma in larga misura anche alla semplicità del dataset. Il vero test della scalabilità e dell’efficienza di questi sistemi emerge solo in scenari dove la ridondanza informativa aumenta. Su SQuAD Small, il vantaggio è reale ma parzialmente condizionato dal contesto.

system	capacità	topk	retrieval	generation	RAG time
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.027	0.749	0.776
		3	0.027	0.993	1.020
		5	0.029	1.208	1.237
		10	0.027	1.639	1.666
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.031	0.768	0.799
		3	0.031	1.028	1.056
		5	0.030	1.264	1.294
		10	0.030	1.750	1.780
HieRAG	L1: 25 L2: 50 L3: 100	1	0.029	0.707	0.736
		3	0.028	0.912	0.940
		5	0.028	1.078	1.106
		10	0.029	1.387	1.416
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.027	1.066	1.096
		3	0.027	0.883	0.910
		5	0.027	1.069	1.096
		10	0.027	1.369	1.396
HieRAG	L1: 25	1	0.028	0.728	0.756
		3	0.027	0.953	0.980
		5	0.028	1.169	1.197
		10	0.028	1.572	1.600
HieRAG++	L1: 25	1	0.030	0.724	0.754
		3	0.031	0.951	0.982
		5	0.031	1.180	1.211
		10	0.030	1.607	1.637
Dense RAG	—	1	0.029	0.753	0.782
		3	0.026	0.999	1.025
		5	0.027	1.286	1.313
		10	0.027	1.982	2.009
Sparse RAG	—	1	0.009	0.717	0.726
		3	0.012	1.011	1.023
		5	0.010	1.310	1.320
		10	0.010	2.077	2.087
CAG	—	3	—	1.702	1.702

Table 4.3: Tempi di inferenza

Sistema	Capacità	topk	Retrieval	Generation	RAG Time
HieRAG	L1: 25, L2: 50, L3: 100	1	0.029	0.707	0.736
HieRAG++	L1: 25	1	0.030	0.724	0.754
CAG	—	3	—	1.702	1.702
Dense RAG	—	1	0.029	0.753	0.782
Sparse RAG	—	1	0.009	0.717	0.726

Table 4.4: Migliori configurazioni per ciascun sistema in termini di efficienza su SQuAD small. La tabella riporta i tempi di inferenza più bassi (RAG Time) ottenuti da ciascun sistema, con separazione tra le fasi di retrieval e generazione. Per ogni sistema è indicata la configurazione ottimale in termini di struttura della cache (se presente) e valore di top-k. I valori in grassetto evidenziano i migliori risultati temporali raggiunti

Valutazione delle metriche Nel benchmark **SQuAD Small**, le differenze prestazionali tra i sistemi non sono solo numeriche, ma riflettono scelte architetturali ben precise. Il sistema che emerge con maggiore solidità è **HieRAG++ con cache compatta** (L1: 25, L2: 50, L3: 100). Questo risultato non è casuale: la combinazione di una cache gerarchica leggera, che evita ridondanze inutili, e l'introduzione del **reranker** fa sì che il sistema riesca a selezionare in modo più fine e mirato i documenti più rilevanti. In questo modo, anche con top-k elevati, il contesto fornito alla generazione rimane informativamente denso ma coerente. Questo spiega i valori elevati di **ROUGE-2** e soprattutto il **BERTScore**, che misura la somiglianza semantica: le risposte di HieRAG++ non solo sono lessicalmente corrette, ma risultano anche più vicine, a livello concettuale, alla ground truth.

La variante **HieRAG senza reranker**, pur mantenendo la stessa

struttura di caching, mostra invece una maggiore variabilità. Sebbene in alcune configurazioni ottenga punteggi alti su metriche come BLEU o ROUGE, tende a restare indietro sul piano semantico. Questo accade perché, mancando una fase di ordinamento intelligente dei documenti recuperati, il sistema può includere nel contesto informazioni solo parzialmente pertinenti.

Al contrario, sistemi come **Dense RAG** e **Sparse RAG**, pur beneficiando di top-k più ampi, faticano a competere. Questi approcci, infatti, operano senza caching né reranking: il risultato è un contesto che, pur avendo volume, manca di qualità selettiva. In Dense RAG, la selezione è affidata a una similarità densa, mentre in Sparse RAG si basa su keyword matching. In entrambi i casi, i chunk selezionati non sempre aiutano il modello a costruire una risposta coerente.

Il sistema **CAG** si rivela particolarmente interessante perché, pur accedendo a un numero intermedio di documenti (top-k=3), non riesce a competere con HieRAG++. A differenza degli altri metodi, CAG non adotta una vera fase di retrieval, ma costruisce direttamente un prompt generativo con documenti selezionati a monte. Questo approccio, apparentemente più ricco, si traduce però in un contesto troppo generico e ridondante. Il risultato è una generazione più “ampia” ma meno focalizzata, come dimostrano i punteggi BLEU e ROUGE-2 inferiori.

In definitiva, le performance osservate non sono il frutto del caso,

ma della presenza o meno di **meccanismi di selezione e organizzazione** del contesto. HieRAG++ eccelle perché riesce a bilanciare bene l'**efficienza** (cache compatta), la **rilevanza** (grazie al reranker), e la **coerenza semantica**. Gli altri sistemi, pur competitivi su alcuni fronti, mostrano limiti strutturali: HieRAG standard è preciso ma meno profondo; Dense e Sparse RAG sono stabili ma poco selettivi; CAG ha potenziale ma manca di controllo. Questo scenario evidenzia l'importanza di una pipeline ben progettata, in cui **la qualità non dipende solo dalla quantità di documenti, ma soprattutto da come vengono scelti e ordinati**.

CHAPTER 4. SPERIMENTAZIONE

system	capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.1826	0.4095	0.2748	0.4091	0.6606
		3	0.1444	0.4484	0.2669	0.4456	0.6742
		5	0.1475	0.4890	0.2914	0.4881	0.6857
		10	0.1429	0.5087	0.2997	0.5050	0.6868
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.2923	0.6012	0.4054	0.5971	0.6646
		3	0.2637	0.5923	0.3924	0.5879	0.6700
		5	0.3087	0.6294	0.4150	0.6261	0.6766
		10	0.3155	0.6370	0.4255	0.6337	0.6874
HieRAG	L1: 25 L2: 50 L3: 100	1	0.3866	0.5923	0.4286	0.5912	0.6362
		3	0.1403	0.4500	0.2818	0.4492	0.6555
		5	0.1489	0.4725	0.2872	0.4696	0.6419
		10	0.3165	0.6185	0.4340	0.6153	0.6480
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.2743	0.5621	0.3747	0.5595	0.7530
		3	0.2614	0.5672	0.3841	0.5648	0.7454
		5	0.3000	0.5899	0.4145	0.5855	0.7682
		10	0.3114	0.5906	0.4023	0.5875	0.7625
HieRAG	L1: 25	1	0.1470	0.3428	0.2097	0.3417	0.7415
		3	0.1086	0.4289	0.2472	0.4252	0.7388
		5	0.0776	0.4142	0.2242	0.4101	0.7544
		10	0.0689	0.4409	0.2400	0.4352	0.7547
HieRAG++	L1: 25	1	0.2562	0.5715	0.3743	0.5680	0.7332
		3	0.2394	0.5818	0.3760	0.5783	0.7404
		5	0.2443	0.5904	0.3805	0.5879	0.7351
		10	0.2775	0.6276	0.4063	0.6249	0.7536
Dense RAG	—	1	0.0578	0.3658	0.2112	0.3628	0.6182
		3	0.0617	0.4265	0.2366	0.4243	0.6404
		5	0.0684	0.4426	0.2548	0.4392	0.6356
		10	0.0666	0.4544	0.2517	0.4507	0.6408
Sparse RAG	—	1	0.0730	0.3641	0.1972	0.3622	0.6196
		3	0.0718	0.4196	0.2296	0.4153	0.6395
		5	0.0493	0.4298	0.2359	0.4250	0.6484
		10	0.0588	0.4422	0.2490	0.4395	0.6491
CAG	—	3	0.1361	0.4024	0.2246	0.3992	0.6546

Table 4.5: Risultati sperimentali in SQuAD small

Sistema	Capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG++	L1: 25, L2: 50, L3: 100	5	0.3000	0.5899	0.4145	0.5855	0.7682
HieRAG	L1: 25, L2: 50, L3: 100	10	0.3165	0.6185	0.4340	0.6153	0.6480
CAG	—	3	0.1361	0.4024	0.2246	0.3992	0.6546
Dense RAG	—	5	0.0684	0.4426	0.2548	0.4392	0.6356
Sparse RAG	—	10	0.0588	0.4422	0.2490	0.4395	0.6491

Table 4.6: Migliori configurazioni per ciascun sistema in termini di efficienza su SQuAD-small. La tabella riporta i punteggi massimi ottenuti da ciascun sistema secondo le metriche testuali (BLEU, ROUGE-1/2/L) e semantiche (BERTScore), indicando per ciascuno la configurazione con migliore performance rispetto a struttura della cache (se presente) e valore di top-k. I valori in grassetto evidenziano i risultati migliori per ogni metrica

Test statistici L’analisi statistica è stata condotta tramite **Dunn Test con correzione di Holm**, al fine di valutare la significatività delle differenze nei tempi di inferenza tra HieRAG (in tre configurazioni) e le baseline.

Per la configurazione **a cache estesa**, HieRAG risulta **significativamente più rapido di CAG in tutte le condizioni di top-k** ($p < 0.001$). Le differenze con Dense RAG non sono mai significative, mentre il confronto con Sparse RAG mostra significatività solo per *top-10* ($p = 0.0456$).

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-6} ***$	$10^{-6} ***$	$10^{-4} ***$	$10^{-4} ***$
HieRAG vs Dense RAG	0.50	0.85	0.27	0.83
HieRAG vs Sparse RAG	0.11	0.84	0.57	0.04 *

Table 4.7: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità elevata e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

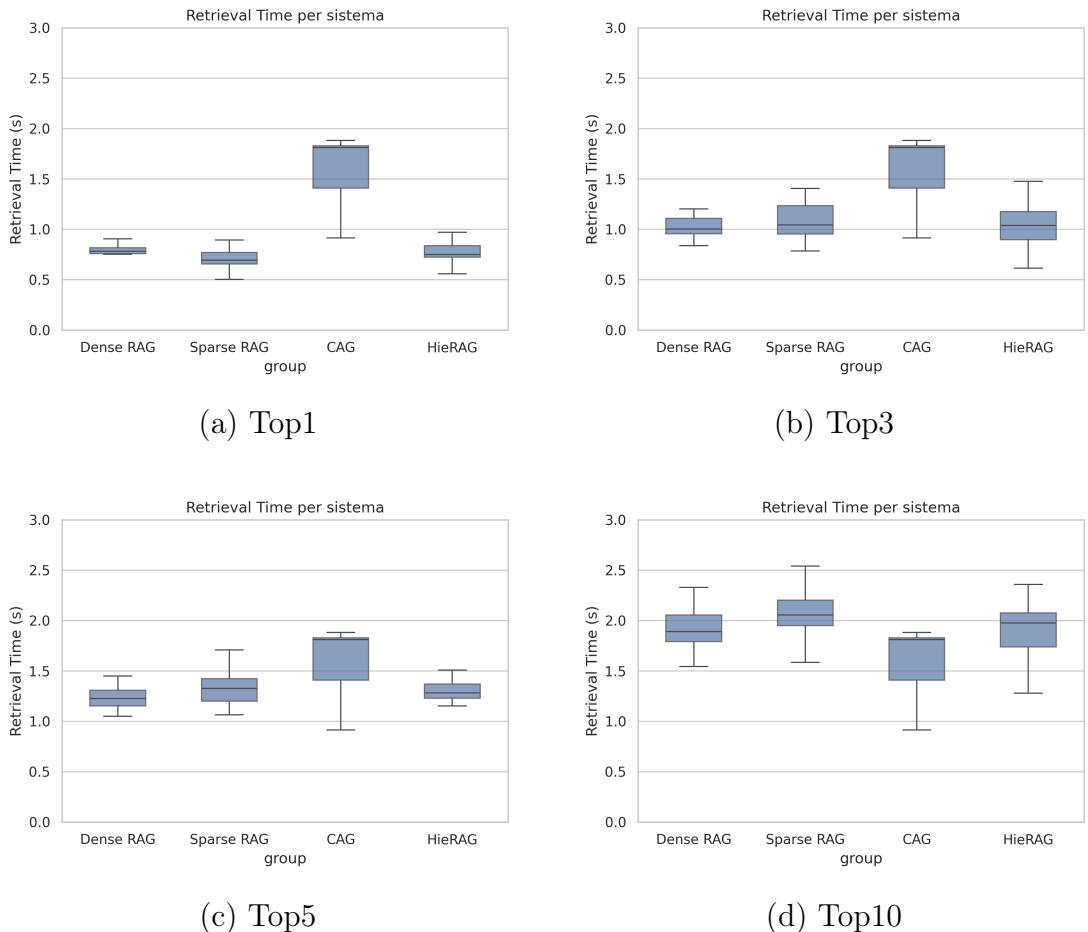


Figure 4.5: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

Con la **cache compatta**, HieRAG mantiene la superiorità su CAG per *top-1* a *top-5*, ma le differenze **si annullano a top-10** ($p = 0.51$). In questo caso, HieRAG è significativamente più efficiente di Dense

RAG solo per *top-10*, e supera Sparse RAG a partire da *top-3*.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-9} ***$	$10^{-11} ***$	$10^{-8} ***$	0.51
HieRAG vs Dense RAG	0.11	0.06	0.25	$10^{-3} ***$
HieRAG vs Sparse RAG	0.62	$10^{-3} ***$	$10^{-2} **$	$10^{-8} ***$

Table 4.8: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità ridotta e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

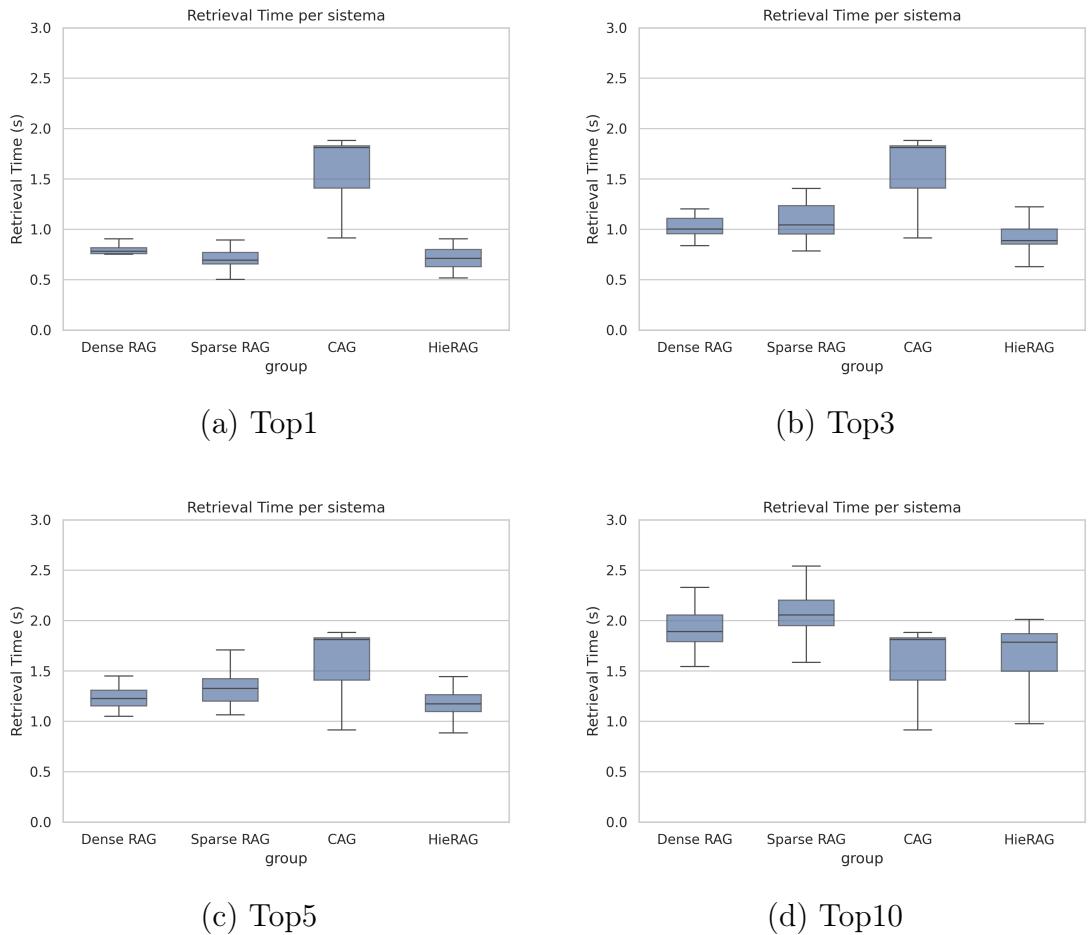


Figure 4.6: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal–Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

Infine, nella configurazione **monolivello**, HieRAG si conferma più efficiente di CAG ($p < 0.01$ per tutti i top-k), ma le differenze con Dense e Sparse RAG non sono significative, tranne per *top-10* nel

confronto con Sparse ($p = 0.01$).

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-9}***$	$10^{-8}***$	$10^{-2}**$	$10^{-2}**$
HieRAG vs Dense RAG	0.09	0.66	0.13	0.62
HieRAG vs Sparse RAG	0.48	0.17	0.95	$10^{-2}**$

Table 4.9: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG monolivello e gli altri sistemi al variare di $top-k$. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

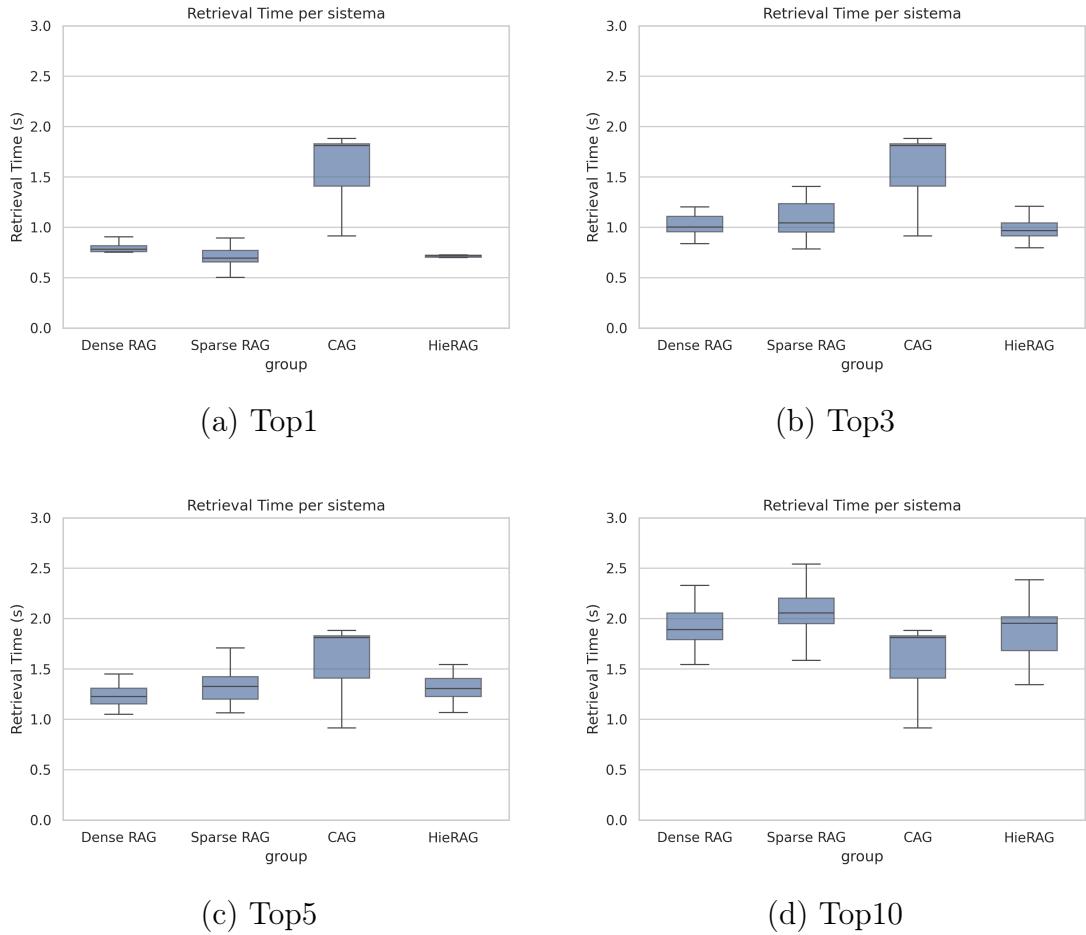


Figure 4.7: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e HieRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta HieRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

Confronto baselines: SQuAD medium

Tempi di inferenza L’analisi dei tempi di inferenza su **SQuAD Medium** mette in luce l’efficienza strutturale del sistema HieRAG, ma allo stesso tempo evidenzia **alcune scelte progettuali che meritano una riflessione più critica**. In apparenza, i risultati sembrano confermare la bontà dell’architettura: HieRAG, anche con $top-k = 10$, riesce a ottenere i tempi di inferenza più bassi tra tutti i sistemi analizzati. Tuttavia, questo comportamento solleva interrogativi. È lecito chiedersi, infatti, se tale efficienza sia davvero merito della gerarchia, o se sia in parte dovuta alla **relativa semplicità del corpus**, che non mette ancora a dura prova la struttura multistrato della cache.

Un esempio emblematico è dato dal fatto che le **configurazioni con cache compatta (L1: 25, L2: 50, L3: 100)** raggiungono prestazioni migliori di quelle con cache estesa. Questo risultato, se da un lato è positivo in termini di ottimizzazione delle risorse, dall’altro **mette in dubbio la reale utilità della profondità della cache in questo scenario**. In un sistema gerarchico, ci si aspetterebbe che una maggiore capacità favorisca la scalabilità. Si potrebbe concludere che la gerarchia, se non bilanciata da un’adeguata complessità del task, **introduca un’overhead strutturale poco giustificato**.

Anche il comportamento di **HieRAG++** rivela criticità. L’aggiunta del reranker, pur teoricamente utile a livello qualitativo, **aumenta sensibilmente il tempo di generazione all’aumentare di top-**

k, senza offrire alcun vantaggio in termini di inferenza. La pipeline, che sulla carta dovrebbe combinare efficienza e selezione fine, in pratica **si appesantisce proporzionalmente alla complessità**. In questa fase sperimentale, la scelta di includerlo indiscriminatamente in tutte le configurazioni rischia di mascherare i reali trade-off tra complessità e beneficio.

Anche le baseline mostrano comportamenti interessanti da leggere in chiave critica. **Sparse RAG**, pur avendo un retrieval velocissimo, **non produce reali vantaggi sulla latenza totale**, dimostrando che ottimizzare solo una componente del sistema è insufficiente. **Dense RAG** si comporta leggermente meglio, ma senza evidenziare una strategia di scaling efficace. Infine, **CAG**, ancora una volta, si conferma una scelta architettonale poco efficiente: l'assenza di qualsiasi forma di selezione porta a un contesto generativo ridondante e dispendioso. Ma proprio per questo, la sua inclusione nel confronto **rischia di funzionare più da strawman che da baseline utile**, abbassando artificialmente il livello della competizione.

In sintesi, l'efficienza mostrata da HieRAG su SQuAD Medium è senza dubbio promettente, ma non va letta in modo acritico. **Le scelte progettuali (profondità della cache, uso sistematico del reranker, selezione delle baseline) vanno valutate con maggiore attenzione**, perché i risultati ottenuti potrebbero riflettere più i limiti del benchmark che i meriti assoluti del sistema. Solo in pre-

senza di dataset più complessi e scenari più realistici sarà possibile determinare se HieRAG rappresenti davvero un’architettura scalabile e robusta, oppure un’ottimizzazione ben riuscita in condizioni favorevoli.

system	capacità	topk	retrieval	generation	RAG time
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.027	0.752	0.779
		3	0.027	0.650	0.677
		5	0.028	0.662	0.690
		10	0.028	0.660	0.688
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.030	0.752	0.782
		3	0.030	1.003	1.033
		5	0.031	1.253	1.284
		10	0.030	1.729	1.759
HieRAG	L1: 25 L2: 50 L3: 100	1	0.028	0.721	0.749
		3	0.028	0.888	0.916
		5	0.028	1.071	1.099
		10	0.030	0.651	0.681
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.029	0.714	0.743
		3	0.027	0.908	0.935
		5	0.027	1.075	1.102
		10	0.027	1.086	1.113
HieRAG	L1: 25	1	0.028	0.727	0.755
		3	0.028	0.937	0.965
		5	0.029	1.145	1.174
		10	0.029	1.594	1.623
HieRAG++	L1: 25	1	0.030	0.718	0.748
		3	0.030	0.926	0.956
		5	0.031	1.160	1.191
		10	0.032	1.604	1.636
Dense RAG	—	1	0.029	0.719	0.748
		3	0.027	0.937	0.964
		5	0.027	1.221	1.248
		10	0.028	1.800	1.828
Sparse RAG	—	1	0.012	0.704	0.716
		3	0.012	0.947	0.959
		5	0.012	1.254	1.266
		10	0.012	1.800	1.812
CAG	—	4	—	2.041	2.041

Table 4.10: Tempi di inferenza in SQuAD medium

Sistema	Capacità	topk	Retrieval	Generation	RAG Time
HieRAG	L1: 25, L2: 50, L3: 100	10	0.030	0.651	0.681
HieRAG++	L1: 25, L2: 50, L3: 100	1	0.029	0.714	0.743
CAG	—	4	—	2.041	2.041
Dense RAG	—	1	0.029	0.719	0.748
Sparse RAG	—	1	0.012	0.704	0.716

Table 4.11: Migliori configurazioni per ciascun sistema in termini di efficienza su SQuAD medium. La tabella riporta i tempi di inferenza più contenuti (RAG Time) osservati per ciascun sistema, suddivisi in tempo di retrieval e di generazione. Ogni configurazione è indicata con il valore di top-k adottato e la struttura di caching, quando presente. I valori in grassetto segnalano i tempi minimi assoluti rilevati nel benchmark

Valutazione delle metriche Su SQuAD Medium, **HieRAG++ con cache compatta e top-k=10** si conferma il sistema più efficace, ottenendo i migliori punteggi su tutte le metriche, tra cui un **BERTScore di 0.7636** e **ROUGE-2 di 0.4183**. Tuttavia, questo risultato, seppur positivo, **rivela una forte dipendenza dal reranker**: la versione HieRAG senza reranking, nella stessa configurazione, ottiene un ROUGE-2 dimezzato, mostrando una selezione documentale molto più debole. Questo evidenzia che l’architettura di caching da sola **non è sufficiente a garantire qualità**, e che il reranker diventa un componente essenziale ma costoso, sia in termini computazionali che progettuali.

Anche la **cache compatta si dimostra più efficace** della versione estesa, suggerendo che una maggiore quantità di documenti non sempre migliora la qualità: al contrario, può introdurre **rumore semantico** che compromette la coerenza della risposta. Tuttavia, questa

osservazione resta limitata a un corpus ancora relativamente semplice: non è chiaro se il vantaggio della compattezza si manterrà su dataset più ricchi o complessi.

Le **metriche utilizzate**, inoltre, presentano dei limiti. ROUGE e BLEU premiano la similarità testuale, ma non valutano la correttezza informativa, mentre BERTScore, pur più sensibile alla semantica, **non distingue tra plausibilità e verità fattuale**. Di conseguenza, l'apparente superiorità di HieRAG++ va interpretata con cautela: le metriche attuali non catturano dimensioni critiche come la specificità, la completezza o la precisione logica della risposta.

Infine, le **baseline** (Dense, Sparse RAG e CAG) si mantengono distanti: migliorano con l'aumento di top-k ma **non riescono a compensare l'assenza di struttura e selezione fine**, confermando che una pipeline efficace richiede un controllo più mirato sul contesto.

system	capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.1549	0.3973	0.2204	0.3968	0.6453
		3	0.1835	0.3536	0.2053	0.3526	0.6642
		5	0.1934	0.3525	0.2081	0.3516	0.6680
		10	0.2267	0.3838	0.2292	0.3820	0.6730
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.2731	0.6715	0.4372	0.6690	0.6440
		3	0.2208	0.6528	0.3988	0.6514	0.6565
		5	0.2014	0.6435	0.4018	0.6416	0.6731
		10	0.2500	0.6768	0.4369	0.6759	0.6636
HieRAG	L1: 25 L2: 50 L3: 100	1	0.1476	0.4066	0.2285	0.4061	0.6274
		3	0.0842	0.4706	0.2439	0.4660	0.6595
		5	0.0928	0.4986	0.2753	0.4973	0.6706
		10	0.1776	0.3443	0.2087	0.3425	0.6839
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.3812	0.5603	0.4086	0.5595	0.7406
		3	0.4025	0.5654	0.4202	0.5652	0.7557
		5	0.3994	0.5654	0.4098	0.5654	0.7488
		10	0.3912	0.5844	0.4183	0.5836	0.7636
HieRAG	L1: 25	1	0.1097	0.3692	0.2127	0.3681	0.7422
		3	0.0948	0.4728	0.2532	0.4699	0.7449
		5	0.0810	0.5084	0.2727	0.5072	0.7490
		10	0.0966	0.5524	0.3168	0.5509	0.7584
HieRAG++	L1: 25	1	0.1818	0.6203	0.3908	0.6171	0.7310
		3	0.1901	0.6359	0.3927	0.6348	0.7404
		5	0.2075	0.6396	0.4016	0.6392	0.7520
		10	0.1877	0.6565	0.4111	0.6538	0.7497
Dense RAG	—	1	0.1018	0.5224	0.2978	0.5218	0.6450
		3	0.0983	0.5801	0.3296	0.5748	0.6795
		5	0.0921	0.5768	0.3264	0.5744	0.6848
		10	0.0876	0.5830	0.3424	0.5822	0.6945
Sparse RAG	—	1	0.0738	0.4619	0.2520	0.4566	0.6783
		3	0.0874	0.5332	0.2954	0.5298	0.6992
		5	0.0918	0.5367	0.3066	0.5350	0.6990
		10	0.0871	0.5618	0.3138	0.5604	0.6908
CAG	—	3	0.1499	0.5069	0.3092	0.5040	0.6870

Table 4.12: Risultati sperimentali dei vari sistemi RAG su **SQuAD Medium**, confrontando diverse configurazioni di cache e valori di *top-k*. Per ciascun sistema sono riportate le principali metriche di valutazione: **BLEU**, **ROUGE-1**, **ROUGE-2**, **ROUGE-L** e **BERTScore**, che misurano rispettivamente la coerenza lessicale e semantica tra la risposta generata e quella attesa. La tabella evidenzia le prestazioni sia per le versioni gerarchiche con o senza reranker (HieRAG / HieRAG++), sia per le baseline Dense RAG, Sparse RAG e CAG. I valori in grassetto rappresentano i migliori risultati all'interno di ciascun blocco di configurazione

Sistema	Capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 250, L2: 500, L3: 1000	10	0.2267	0.3838	0.2292	0.3820	0.6730
HieRAG++	L1: 25, L2: 50, L3: 100	10	0.3912	0.5844	0.4183	0.5836	0.7636
CAG	—	3	0.1499	0.5069	0.3092	0.5040	0.6870
Dense RAG	—	10	0.0876	0.5830	0.3424	0.5822	0.6945
Sparse RAG	—	10	0.0871	0.5618	0.3138	0.5604	0.6908

Table 4.13: Un confronto tra le migliori configurazioni di ciascun sistema sul dataset SQuAD Medium, sulla base delle principali metriche di efficacia: BLEU, ROUGE-1/2/L e BERTScore. Per ogni sistema viene indicata la configurazione ottimale di capacità di cache e valore di top-k. La configurazione HieRAG++ con cache compatta e top-k=10 emerge come la più performante in tutte le metriche, evidenziando il contributo significativo del reranker. Le baseline (Dense RAG, Sparse RAG, CAG) mostrano risultati inferiori, con valori sensibilmente più bassi, soprattutto in BLEU e BERTScore

Test statistici L’analisi statistica riporta i **p-value corretti del Dunn Test** per il confronto tra **HieRAG** (con capacità elevata) e gli altri sistemi (CAG, Dense RAG, Sparse RAG) al variare di *top-k*. In tutte le configurazioni considerate (*Top-1*, *Top-3*, *Top-5*, *Top-10*), i valori risultano **statisticamente significativi** ($p < 0.05$), con livelli di significatività molto elevati nel confronto con **CAG** e **Dense RAG**, e più moderati ma comunque solidi nel caso di **Sparse RAG**.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	10^{-5***}	10^{-22***}	10^{-19***}	10^{-15***}
HieRAG vs Dense RAG	10^{-3***}	10^{-7***}	10^{-7***}	10^{-12***}
HieRAG vs Sparse RAG	10^{-3***}	10^{-5***}	10^{-5***}	10^{-5***}

Table 4.14: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità elevata e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

I risultati statistici confermano con forza la superiorità temporale

di HieRAG rispetto a CAG, con p-value estremamente bassi in tutte le condizioni. Tuttavia, la significatività così marcata rispetto a CAG è poco sorprendente: si tratta di un sistema privo di retrieval, e quindi strutturalmente inefficiente. Il suo utilizzo nel test ha più valore illustrativo che competitivo, e rischia di abbassare artificialmente la soglia del confronto.

Più interessanti sono le differenze con Dense RAG e Sparse RAG, che risultano anch'esse statisticamente significative, anche per Top-1, ma vanno interpretate con cautela. Le dimensioni del corpus, sebbene più ampie rispetto a SQuAD Small, restano contenute, e questo può amplificare differenze marginali nei tempi di retrieval. In altri termini, anche se HieRAG risulta più efficiente, non è detto che lo stesso vantaggio si mantenga su corpus più complessi o reali.

Infine, è importante notare che la significatività statistica non implica automaticamente una rilevanza pratica. Differenze temporali inferiori a 100 ms, pur essendo statisticamente significative, potrebbero non influenzare realmente l'esperienza utente. Questo pone la necessità, per future valutazioni, di affiancare ai test statistici anche un'analisi dell'impatto operativo e della sostenibilità computazionale complessiva.

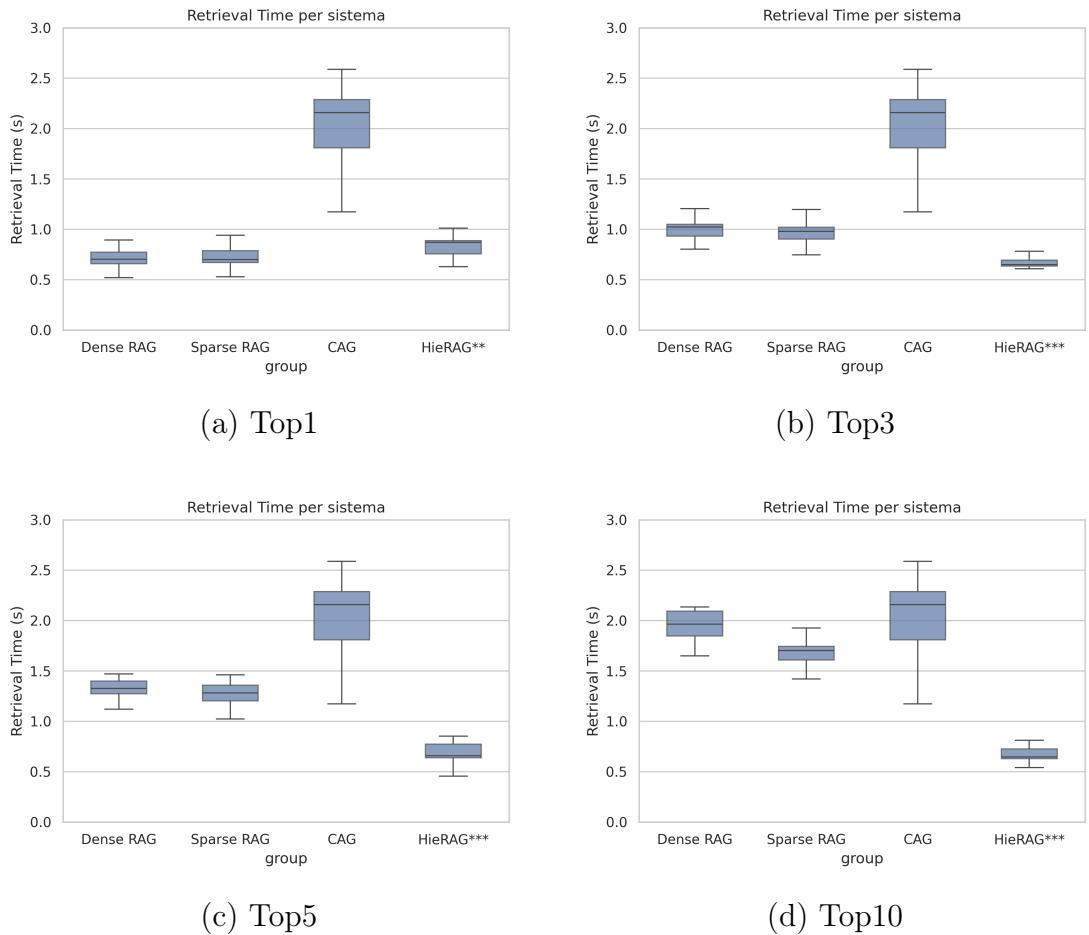


Figure 4.8: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal–Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il p -value < 0.05 , ** indica il p -value < 0.01 e *** indica che il p -value < 0.001). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

L'analisi statistica riporta i risultati del **Dunn Test** tra **HieRAG con cache ridotta** e gli altri sistemi (CAG, Dense RAG, Sparse RAG), considerando quattro configurazioni di *top-k*. I valori di p

mostrano differenze **statisticamente significative** ($p < 0.05$) nella maggior parte dei confronti. In particolare, le differenze con **CAG** sono molto marcate in tutte le soglie, mentre quelle con **Dense RAG** e **Sparse RAG** sono più evidenti per *Top-1*, *Top-5* e *Top-10*.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-6}***$	$10^{-13}***$	$10^{-14}***$	$10^{-15}***$
HieRAG vs Dense RAG	0.02*	0.13	$10^{-4}***$	$10^{-13}***$
HieRAG vs Sparse RAG	0.02	0.47	0.01**	$10^{-5}***$

Table 4.15: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità ridotta e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$

I risultati statistici confermano che **HieRAG con cache ridotta supera significativamente CAG** in tutte le condizioni, come prevedibile dato che CAG non utilizza alcun meccanismo di selezione. Tuttavia, **il confronto perde parzialmente valore comparativo**, trattandosi di un baseline strutturalmente svantaggiata.

Più interessanti sono i risultati con **Dense RAG e Sparse RAG**, che mostrano significatività alternata. Per *Top-1*, le differenze sono marginali ($p \approx 0.02$), mentre per *Top-3* non risultano significative, suggerendo che **la cache ridotta di HieRAG potrebbe non garantire vantaggi consistenti in scenari a bassa profondità**. Le differenze si rafforzano per *Top-5* e *Top-10*, ma vanno interpretate alla luce del fatto che i dataset rimangono gestibili in dimensione.

In sintesi, questi risultati indicano che **la superiorità statistica**

di HieRAG è più netta in scenari con top-k elevati, dove il sistema può sfruttare meglio la sua struttura gerarchica. Tuttavia, il quadro resta parziale: in contesti reali, le differenze osservate potrebbero non tradursi in un impatto operativo rilevante, e il comportamento con corpus più grandi e rumorosi rimane da verificare.

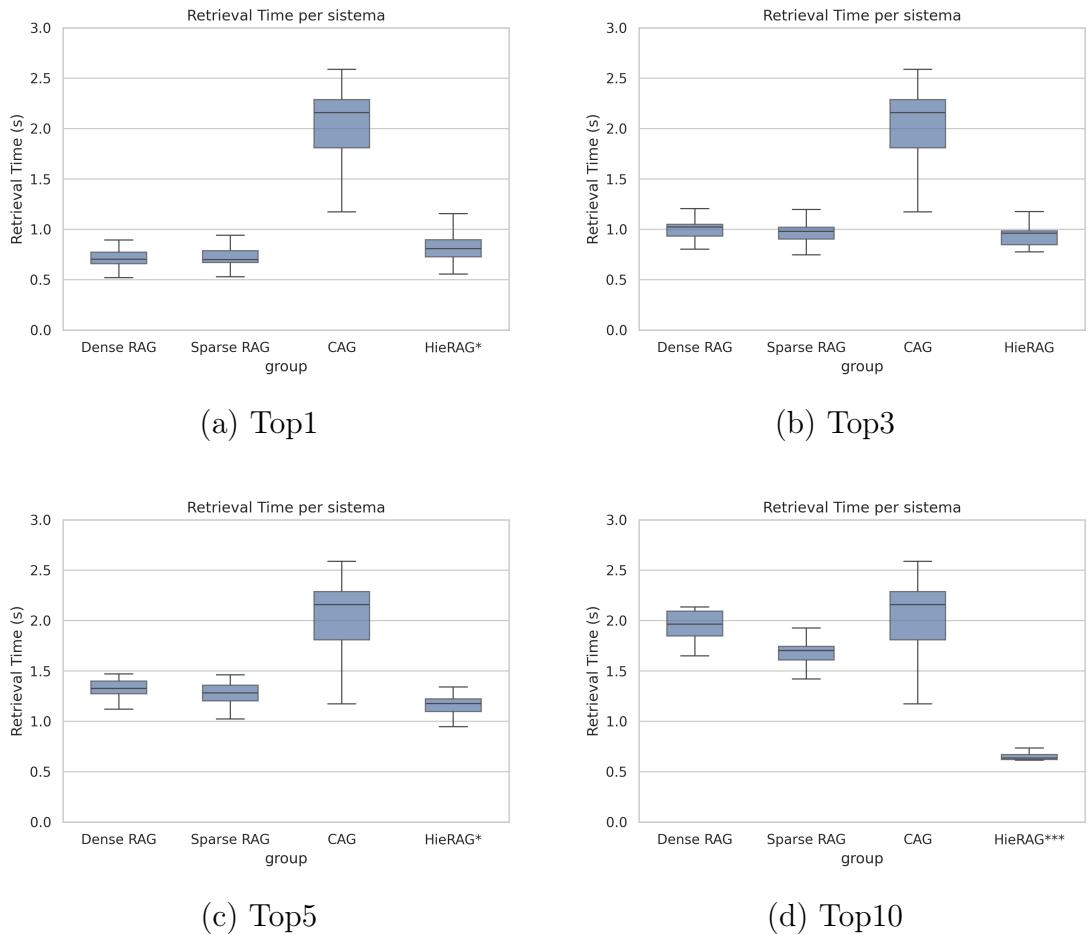


Figure 4.9: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e HieRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea media indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal–Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta HieRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il p -value < 0.05 , ** indica il p -value < 0.01 e *** indica che il p -value < 0.001). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

L'analisi statistica mostra i risultati del **Dunn Test** applicato al sistema **HieRAG con cache monolivello (L1: 25)**, confrontato con CAG, Dense RAG e Sparse RAG su diverse soglie di *top-k*. I

valori di \mathbf{p} indicano che HieRAG differisce in modo **statisticamente significativo** da CAG nelle configurazioni *Top-1*, *Top-3* e *Top-5*, ma non in *Top-10*. Le differenze con Dense RAG non risultano significative in alcuna configurazione, mentre il confronto con Sparse RAG mostra significatività solo per *Top-10*.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-8}***$	$10^{-10}***$	$10^{-8}***$	1
HieRAG vs Dense RAG	0.38	0.92	0.32	1
Sparse RAG vs HieRAG	0.38	0.92	0.83	$10^{-4}***$

Table 4.16: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG monolivello e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

I risultati statistici evidenziano che HieRAG monolivello è significativamente più efficiente di CAG, ma solo fino a top-k=5. La mancanza di significatività a top-10 suggerisce che, in assenza di livelli cache superiori, la scalabilità del sistema si riduce, rendendolo comparabile a un sistema non strutturato.

Il confronto con Dense RAG mostra invece l'assenza totale di differenze significative: ciò evidenzia come la semplice presenza di una cache a un solo livello non conferisca vantaggi concreti in termini temporali, e che i due sistemi si comportano in modo simile.

L'unico confronto significativo con Sparse RAG emerge a top-10, dove HieRAG riesce a mantenere prestazioni più stabili. Tuttavia, anche in questo caso, la differenza rilevata statisticamente potrebbe

avere un impatto pratico minimo, data la vicinanza dei valori osservati.

In sintesi, questi risultati mettono in discussione l'efficacia reale dell'architettura monolivello, che si rivela spesso indistinguibile dalle baseline più semplici. Ciò rafforza l'idea che i benefici dell'approccio gerarchico emergano solo quando implementato su più livelli e combinato con strategie di selezione più avanzate, come il reranking.

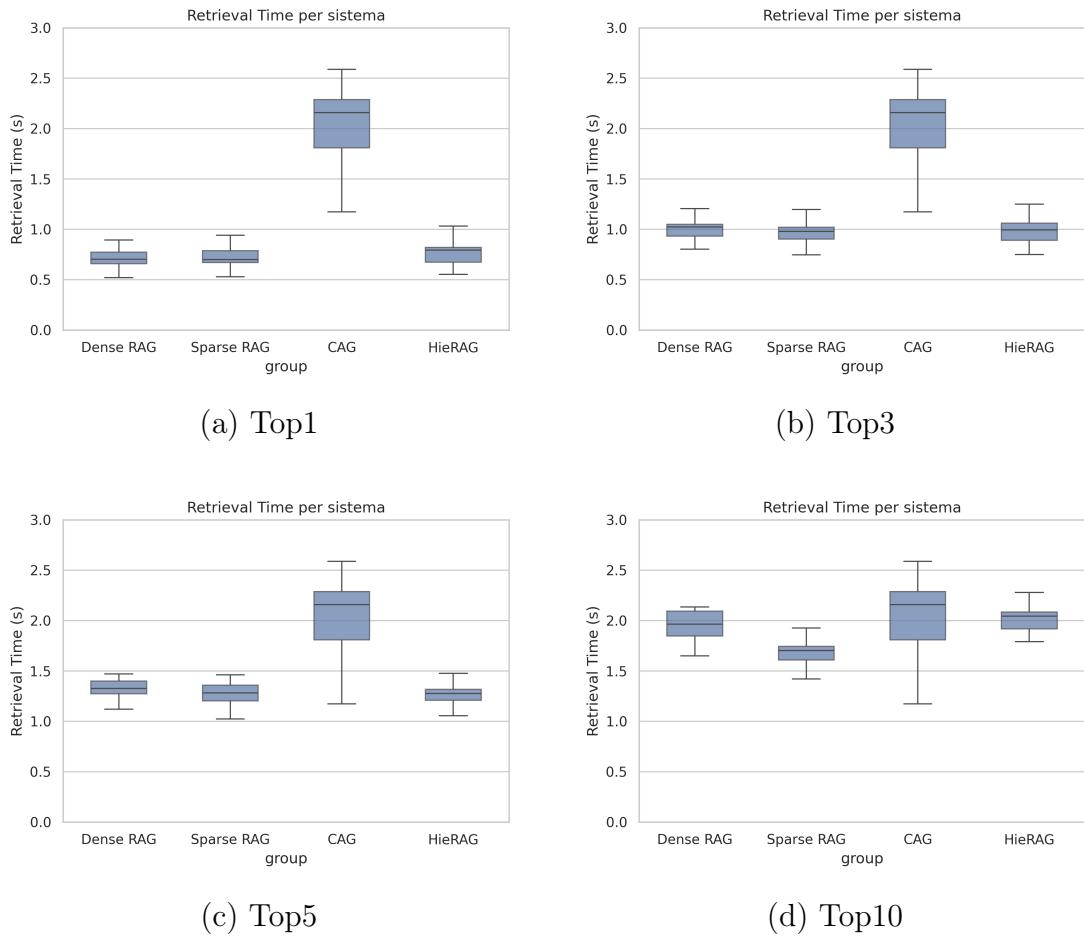


Figure 4.10: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se Hi-eRAG è diverso rispetto a tutti gli altri sistemi.

Confronto baselines: SQuAD large

Tempi di inferenza Nel contesto di **SQuAD Large**, l'aumento della complessità documentale permette di testare in modo più severo la **scalabilità temporale** dei diversi sistemi. A prima vista, **Sparse RAG** si conferma il sistema più veloce in assoluto (0.685s), seguito a distanza minima da **Dense RAG** e **HieRAG++**. Tuttavia, il confronto va interpretato con cautela.

In particolare, la prestazione di **HieRAG++ con cache compatta** risulta sorprendentemente competitiva, considerando la sua struttura gerarchica e la presenza del reranker. Il fatto che mantenga un tempo di inferenza **quasi identico a Dense RAG** – e addirittura inferiore a molte configurazioni HieRAG senza reranker – suggerisce che l'**overhead della gerarchia è ampiamente compensato da una gestione più ordinata e localizzata del contesto**. Questo rende HieRAG++ un candidato adatto per scenari **real-time**, a parità di capacità computazionale.

Tuttavia, non tutte le scelte architetturali si confermano ottimali. L'analisi mostra che **aumentare la capacità della cache su tre livelli non porta benefici evidenti in termini di efficienza**, nonostante l'aumento di *top-k*. Il tempo di retrieval resta invariato, ma **il tempo di generazione cresce in modo quasi lineare**, segnalando che **il collo di bottiglia si sposta sul modello generativo**, non sul retrieval. Questo solleva dubbi sulla reale utilità delle configurazioni

cache più profonde in scenari con carichi informativi già elevati.

Le **baseline** senza cache (Dense e Sparse RAG) dimostrano tempi di retrieval leggermente inferiori, ma perdono il vantaggio nella generazione. Questo conferma che **il caching non è essenziale per ottimizzare il retrieval**, ma può migliorare l'organizzazione del contesto a valle. Infine, **CAG** si dimostra ancora una volta il sistema meno scalabile, con oltre 3 secondi di latenza: un valore che rende questo approccio difficilmente utilizzabile in applicazioni interattive o ad alta frequenza, nonostante possa offrire una maggiore copertura informativa.

In sintesi, i risultati su SQuAD Large rafforzano la tesi che **una gerarchia ben calibrata (es. cache compatta) combinata a un reranker offre un equilibrio ottimale tra efficienza e organizzazione del contesto**, ma mettono anche in discussione l'efficacia di configurazioni complesse e profondamente stratificate, che non sembrano portare vantaggi tangibili in ambienti già carichi.

system	capacità	topk	retrieval	generation	RAG time
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.029	0.725	0.754
		3	0.028	0.966	0.994
		5	0.029	1.216	1.245
		10	0.029	1.726	1.755
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.028	0.732	0.760
		3	0.029	0.956	0.985
		5	0.029	1.218	1.247
		10	0.029	1.760	1.789
HieRAG	L1: 25 L2: 50 L3: 100	1	0.028	0.697	0.725
		3	0.028	0.851	0.879
		5	0.028	1.041	1.069
		10	0.029	1.416	1.445
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.028	0.686	0.714
		3	0.029	1.030	1.059
		5	0.029	1.035	1.064
		10	0.029	1.405	1.434
HieRAG	L1: 25	1	0.028	0.708	0.736
		3	0.028	0.911	0.939
		5	0.029	1.104	1.133
		10	0.028	1.580	1.608
HieRAG++	L1: 25	1	0.031	0.690	0.721
		3	0.031	0.880	0.911
		5	0.031	1.111	1.142
		10	0.031	1.593	1.624
Dense RAG	—	1	0.027	0.693	0.720
		3	0.027	0.891	0.918
		5	0.027	1.169	1.196
		10	0.027	1.748	1.775
Sparse RAG	—	1	0.026	0.659	0.685
		3	0.025	0.891	0.916
		5	0.026	1.169	1.195
		10	0.026	1.748	1.774
CAG	—	7	—	3.013	3.013

Table 4.17: Tempi di inferenza (in secondi) per ciascun sistema su **SQuAD Large**, considerando diverse configurazioni di *top-k* e struttura cache. I sistemi HieRAG e HieRAG++ sono valutati sia in versione estesa (3 livelli con capacità alta) che compatta o monolivello, mentre le baseline (Dense RAG, Sparse RAG e CAG) sono presentate con il valore di *top-k* più favorevole. La tabella scomponete il tempo totale in **retrieval time**, **generation time** e **RAG time complessivo**

Sistema	Capacità	topk	Retrieval	Generation	RAG Time
HieRAG	L1: 25, L2: 50, L3: 100	1	0.028	0.697	0.725
HieRAG++	L1: 25, L2: 50, L3: 100	1	0.028	0.686	0.714
CAG	—	7	—	3.013	3.013
Dense RAG	—	1	0.027	0.693	0.720
Sparse RAG	—	1	0.026	0.659	0.685

Table 4.18: Le **migliori configurazioni di ciascun sistema** in termini di efficienza sul dataset **SQuAD Large**. Per ogni sistema è riportata la configurazione ottimale di cache e il valore di *top-k* che ha prodotto il **minimo tempo di inferenza complessivo (RAG Time)**. La tabella distingue tra tempo di **retrieval**, tempo di **generazione della risposta** e tempo **totale**. I sistemi HieRAG e HieRAG++ sono riportati con cache compatta, mentre le baseline Dense RAG, Sparse RAG e CAG sono testate con i parametri più favorevoli. I valori in grassetto evidenziano le **prestazioni migliori** osservate tra i sistemi, con **Sparse RAG** che registra il tempo di inferenza più basso in assoluto

Valutazione delle metriche I risultati su *SQuAD Large* confermano che **HieRAG++ con cache compatta** è il sistema più efficace sul piano qualitativo, grazie alla combinazione tra struttura gerarchica e reranking. Tuttavia, questa superiorità ha due condizioni: richiede **top-k elevato** (quindi un contesto ampio, potenzialmente costoso da generare) e **dipende fortemente dal reranker**, la cui efficacia è difficile da attribuire univocamente alla strategia di caching. Il rischio è quindi che il vantaggio osservato venga sovrastimato come merito della gerarchia, quando è in realtà guidato da un componente esterno di selezione fine.

Il sistema **HieRAG standard**, privo di reranking, evidenzia invece un comportamento più fragile: ottiene il **BLEU più alto**, segna-

lando una buona aderenza letterale alla risposta attesa, ma risulta **più debole nelle metriche semantiche** (ROUGE-2, BERTScore), in particolare con cache compatta. Questo indica che, senza il reranker, la gerarchia perde parte del suo potere discriminativo e fatica a selezionare le evidenze rilevanti nei contesti più vasti. Nonostante ciò, HieRAG si dimostra **più stabile e preciso** delle baseline Dense e Sparse RAG, soprattutto a parità di *top-k*, e rappresenta un **buon compromesso tra efficienza e qualità lessicale**.

Le baseline **Dense RAG** e **Sparse RAG**, pur efficienti in termini di retrieval (0.026–0.027s), non riescono a **tradurre tale efficienza in qualità della generazione**. I loro BERTScore restano costantemente inferiori a quelli di HieRAG++ e, in molti casi, anche a quelli di HieRAG standard, suggerendo che il solo incremento di documenti (*top-k*) non basta a compensare la mancanza di una struttura gerarchica o di un reranker. La loro **stabilità** può essere letta positivamente, ma allo stesso tempo segnala un **limite nella scalabilità qualitativa**.

Infine, **CAG**, che si distingue per l’assenza di qualsiasi selezione o caching, mostra come **l’integrazione diretta di tutti i documenti nel prompt** porti a un peggioramento delle performance. Il sistema soffre di **rumore e ridondanza** informativa, e il BERTScore (0.6930) non è sufficiente a giustificare il costo computazionale elevato, né sul piano della qualità né su quello dell’efficienza.

system	capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.2271	0.4820	0.2191	0.4817	0.6908
		3	0.1832	0.5120	0.2498	0.5107	0.6893
		5	0.1562	0.5510	0.2579	0.5491	0.6917
		10	0.1186	0.6039	0.2740	0.6005	0.6927
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.1808	0.6657	0.3302	0.6632	0.6903
		3	0.1766	0.6566	0.3415	0.6562	0.6821
		5	0.1878	0.6617	0.3171	0.6601	0.6908
		10	0.2389	0.7095	0.3630	0.7071	0.6938
HieRAG	L1: 25 L2: 50 L3: 100	1	0.2375	0.5142	0.2476	0.5139	0.6147
		3	0.2169	0.4822	0.2270	0.4812	0.6305
		5	0.1735	0.5283	0.2469	0.5264	0.6411
		10	0.1667	0.5492	0.2506	0.5486	0.6549
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.2161	0.6195	0.3058	0.6177	0.7333
		3	0.2363	0.6267	0.2997	0.6237	0.7360
		5	0.2298	0.6072	0.2880	0.6068	0.7344
		10	0.2598	0.6214	0.2911	0.6200	0.7633
HieRAG	L1: 25	1	0.1192	0.3438	0.1390	0.3435	0.7315
		3	0.0846	0.4139	0.1862	0.4129	0.7380
		5	0.0643	0.4572	0.2109	0.4560	0.7356
		10	0.0738	0.4999	0.2265	0.4967	0.7496
HieRAG++	L1: 25	1	0.0846	0.6080	0.2979	0.6053	0.6876
		3	0.1552	0.6420	0.3310	0.6405	0.7219
		5	0.1755	0.6583	0.3171	0.6561	0.7305
		10	0.1955	0.6796	0.3543	0.6792	0.7435
Dense RAG	—	1	0.0734	0.5108	0.2422	0.5091	0.6340
		3	0.0832	0.5768	0.2771	0.5729	0.6814
		5	0.0751	0.6005	0.2940	0.5976	0.6784
		10	0.0654	0.5924	0.2849	0.5907	0.6629
Sparse RAG	—	1	0.0663	0.4684	0.2290	0.4679	0.6497
		3	0.0875	0.5627	0.2619	0.5600	0.6816
		5	0.0798	0.5700	0.2693	0.5675	0.6876
		10	0.0649	0.5738	0.2741	0.5713	0.6774
CAG	—	7	0.1479	0.5061	0.2049	0.5040	0.6930

Table 4.19: Le performance dei diversi sistemi in termini di **qualità delle risposte generate** su *SQuAD Large*, misurata tramite BLEU, ROUGE (1, 2, L) e BERTScore. Per ogni sistema vengono riportate più configurazioni (capacità cache e top-k), con l’obiettivo di identificare il miglior compromesso tra precisione lessicale e coerenza semantica

Sistema	Capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 250, L2: 500, L3: 1000	1	0.2271	0.4820	0.2191	0.4817	0.6908
HieRAG++	L1: 25, L2: 50, L3: 100	10	0.2598	0.6214	0.2911	0.6200	0.7633
CAG	—	7	0.1479	0.5061	0.2049	0.5040	0.6930
Dense RAG	—	5	0.0751	0.6005	0.2940	0.5976	0.6784
Sparse RAG	—	5	0.0798	0.5700	0.2693	0.5675	0.6876

Table 4.20: Le migliori configurazioni ottenute da ciascun sistema in termini di qualità della generazione su SQuAD Large, scegliendo per ciascuno la combinazione di capacità e *top-k* che ha massimizzato le metriche. Sono riportati i punteggi BLEU, ROUGE (1, 2, L) e BERTScore, che misurano rispettivamente la precisione lessicale, la copertura e la somiglianza semantica tra risposta generata e ground truth. I valori in grassetto evidenziano il risultato migliore per ogni metrica, permettendo un confronto diretto delle prestazioni generative ottimali tra i vari sistemi

Test statistici La tabella 4.21 riporta i *p-value* corretti (Dunn Test con correzione Holm) per confrontare le prestazioni temporali di **HieRAG con capacità elevata** rispetto agli altri sistemi (CAG, Dense RAG, Sparse RAG), al variare del parametro *top-k*.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-9}***$	$10^{-7}***$	$10^{-7}***$	$10^{-4}***$
HieRAG vs Dense RAG	0.42	0.21	0.32	0.02
HieRAG vs Sparse RAG	0.86	0.02	0.02	0.01*

Table 4.21: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità elevata e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

Questi risultati evidenziano una realtà più sfumata rispetto a quanto suggerito dalle sole medie:

- Il sistema **HieRAG (capacità elevata)** si conferma netta-

mente più efficiente di CAG, in modo robusto e costante.

Questo era atteso, vista l'assenza di caching e il carico computazionale elevato di CAG, ma qui viene confermato con rigore statistico.

- Il confronto con **Dense RAG**, invece, ridimensiona il vantaggio di HieRAG: le **differenze non sono significative per top-1, 3, e 5**, e diventano **appena marginali solo per top-10**. Questo suggerisce che HieRAG **non garantisce sempre un guadagno temporale rispetto a Dense RAG**, specie con pochi documenti selezionati.
- Anche con **Sparse RAG**, spesso considerato un sistema “leggero”, **HieRAG mostra differenze statisticamente significative solo quando top-k è maggiore di 1**, ovvero quando il costo aggregato inizia a incidere. Per top-1, invece, non emerge alcuna superiorità.

I risultati del Dunn Test non solo confermano la **superiorità di HieRAG su sistemi come CAG**, ma mostrano che **il vantaggio architettonale di HieRAG si attiva solo in presenza di carichi informativi elevati**. Nei casi semplici (top-1), le **differenze si annullano**, rendendo le baseline dense e sparse ancora competitive.

Questa evidenza suggerisce che la progettazione di sistemi gerarchici come HieRAG **va giustificata in funzione dello scenario**

d'uso: in ambienti real-time con richieste leggere, il beneficio è trascurabile; ma in scenari complessi, il sistema riesce a scalare meglio, mantenendo tempi stabili anche quando il contesto cresce.

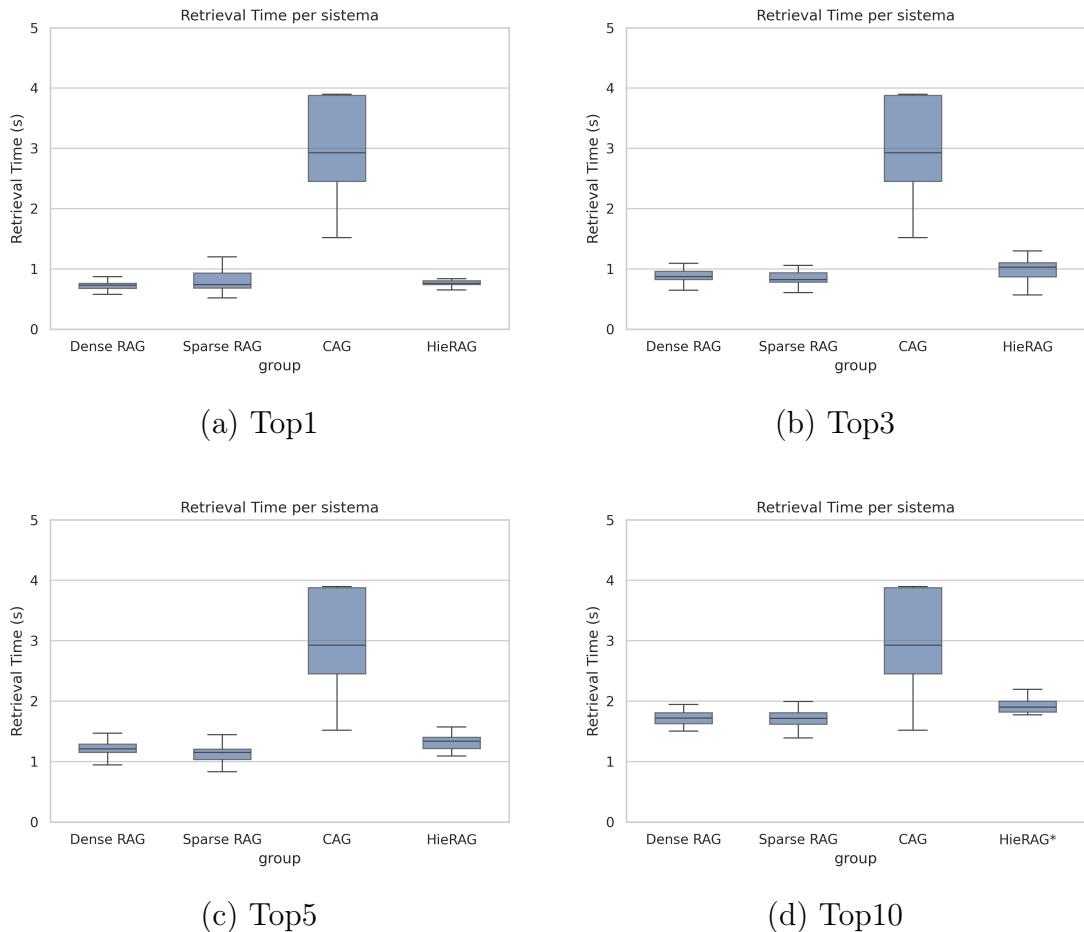


Figure 4.11: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal–Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica che il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

Nel confronto statistico tra HieRAG a capacità ridotta e gli altri approcci, i risultati del Dunn Test evidenziano una dinamica molto chiara ma anche potenzialmente problematica. HieRAG si dimostra significativamente migliore rispetto a CAG in tutte le configurazioni di top-k, con p-value estremamente bassi (tutti inferiori a 10^{-8}). Questo dato non sorprende: CAG non effettua alcuna selezione del contesto, includendo direttamente tutti i documenti nel prompt, con il risultato di aumentare la ridondanza e appesantire il processo generativo. Al contrario, HieRAG – pur con una cache ridotta – riesce a contenere la latenza e ad effettuare una selezione più efficiente, seppur semplice. Questo conferma la superiorità strutturale del sistema anche nella sua versione meno sofisticata.

Tuttavia, il quadro si complica se si guarda al confronto con Dense RAG e Sparse RAG. In entrambi i casi, i test non evidenziano alcuna differenza statisticamente significativa. Le prestazioni tra HieRAG ridotto e questi due sistemi risultano quindi comparabili. Questo è un segnale importante: dimostra che la struttura gerarchica del sistema, quando troppo compressa in termini di capacità, perde efficacia. La cache L1 con soli 25 elementi, L2 con 50 e L3 con 100 non è sufficiente per garantire una copertura informativa diversificata, e finisce per comportarsi in modo simile a un semplice retriever standard.

Inoltre, l'assenza del reranker – componente chiave nelle versioni HieRAG++ – impedisce al sistema di selezionare le evidenze più perti-

nenti tra i candidati. Il risultato è un comportamento che, pur avvantaggiandosi di una struttura cache, non riesce a differenziarsi in modo significativo dai retrieval densi e sparsi tradizionali. Questo è particolarmente evidente nei confronti con Sparse RAG, dove l'efficienza computazionale della retrieval BM25 riesce a compensare la semplicità del modello, e nei confronti con Dense RAG, che beneficia di una pipeline densa consolidata e ben calibrata.

In conclusione, questi risultati mettono in luce una criticità rilevante: **la semplice adozione di una cache gerarchica non garantisce miglioramenti statistici**, a meno che non sia accompagnata da un'adeguata capacità informativa e da un meccanismo di selezione avanzato. L'efficacia del sistema HieRAG, in questa configurazione, risulta così fortemente limitata: si colloca nettamente sopra CAG, ma non è ancora in grado di superare le architetture retrieval-based più ottimizzate. Questo suggerisce che, per massimizzare il potenziale della gerarchia, è necessario bilanciare attentamente **capacità, qualità dei candidati e capacità di ranking interno**.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-12}***$	$10^{-8}***$	$10^{-13}***$	10^{-10}
HieRAG vs Dense RAG	0.71	0.67	0.12	0.81
HieRAG vs Sparse RAG	0.53	0.27	0.68	0.97

Table 4.22: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità ridotta e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

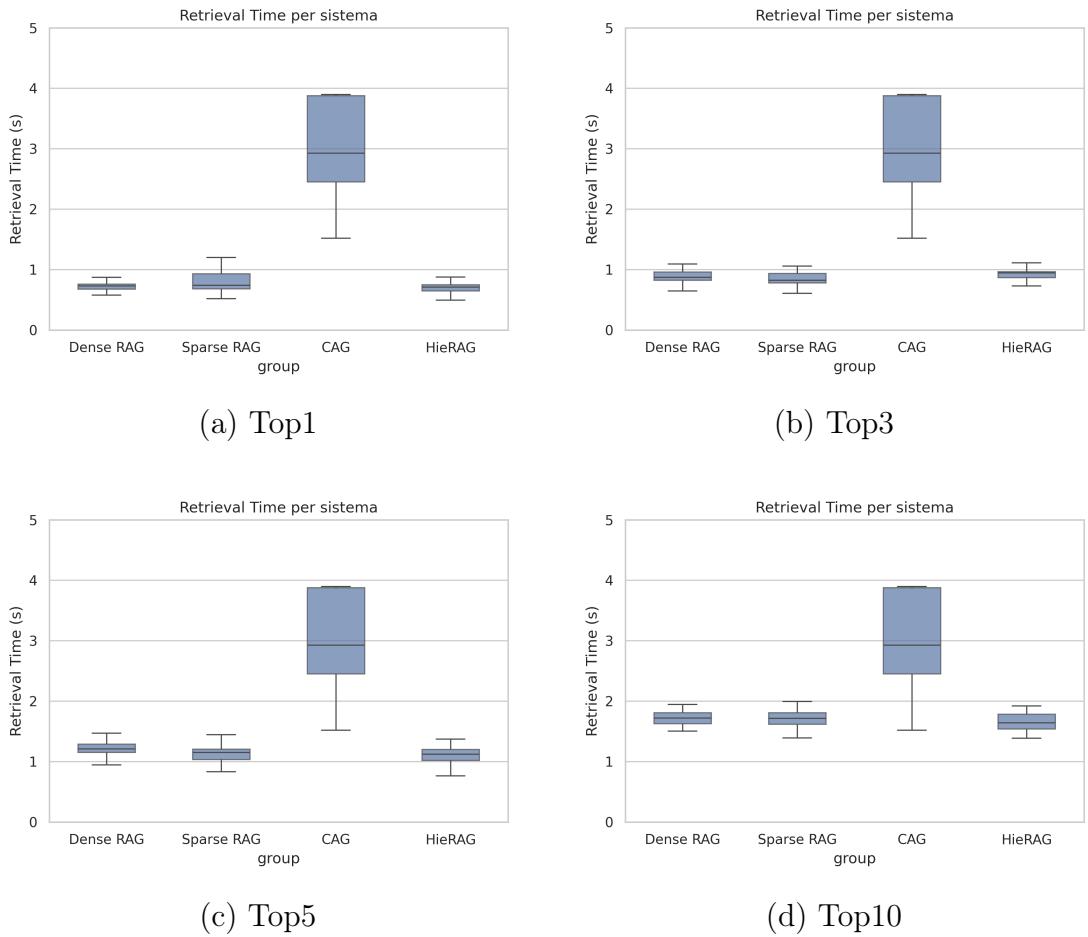


Figure 4.12: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

I risultati del Dunn Test mostrano che HieRAG in configurazione monolivello (L1:25) si distingue in modo netto da CAG in tutte le configurazioni top-k, con p-value sempre estremamente bassi (fino a

10^{-13}). Questa significativa differenza conferma la debolezza strutturale di CAG: la sua assenza di fase di retrieval e la gestione indiscriminata del contesto portano a una generazione meno efficace, penalizzata dalla ridondanza e dalla difficoltà del modello nel focalizzarsi su informazioni salienti. Anche una cache minimale come quella di HieRAG monolivello, pur semplice, riesce a introdurre un minimo di filtraggio informativo che si traduce in un vantaggio netto.

Tuttavia, quando si confronta HieRAG con i metodi Dense RAG e Sparse RAG, le evidenze statistiche si attenuano drasticamente. Nessuno dei confronti risulta significativamente diverso, con p-value ben sopra la soglia di significatività ($p > 0.05$) in quasi tutte le configurazioni. L'unica eccezione è una debole tendenza alla significatività nel confronto con Sparse RAG a Top-3 ($p = 0.06$), ma comunque non sufficiente per affermare una reale superiorità. Questo dato è cruciale, perché mostra che HieRAG, quando privato della sua struttura gerarchica e del reranker, **non riesce più a offrire vantaggi misurabili** rispetto ai retrieval tradizionali.

La causa di queste prestazioni risiede nella limitata capacità della cache e nell'assenza di un meccanismo di selezione interno. La configurazione monolivello, seppur efficiente, non riesce a modulare dinamicamente la pertinenza del contesto né a compensare la crescita informativa del top-k. Questo la rende funzionalmente simile a Dense e Sparse RAG, i quali operano con strategie retrieval consolidate (dense

o sparse) ma comunque guidate da una logica discriminativa più evoluta.

In sintesi, i risultati indicano che **HieRAG monolivello offre un miglioramento tangibile solo rispetto a sistemi non selettivi come CAG**, ma non è sufficiente per superare – o anche solo distinguersi da – metodi retrieval standard. Senza la profondità gerarchica o un reranker dedicato, la sua capacità di gestire il contesto rimane limitata, portando a una qualità di generazione statisticamente equivalente a quella dei sistemi più semplici.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-11}***$	$10^{-7}***$	$10^{-13}***$	$10^{-6}***$
HieRAG vs Dense RAG	0.94	0.37	0.07	0.75
HieRAG vs Sparse RAG	0.94	0.06	0.49	0.53

Table 4.23: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG monolivello e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

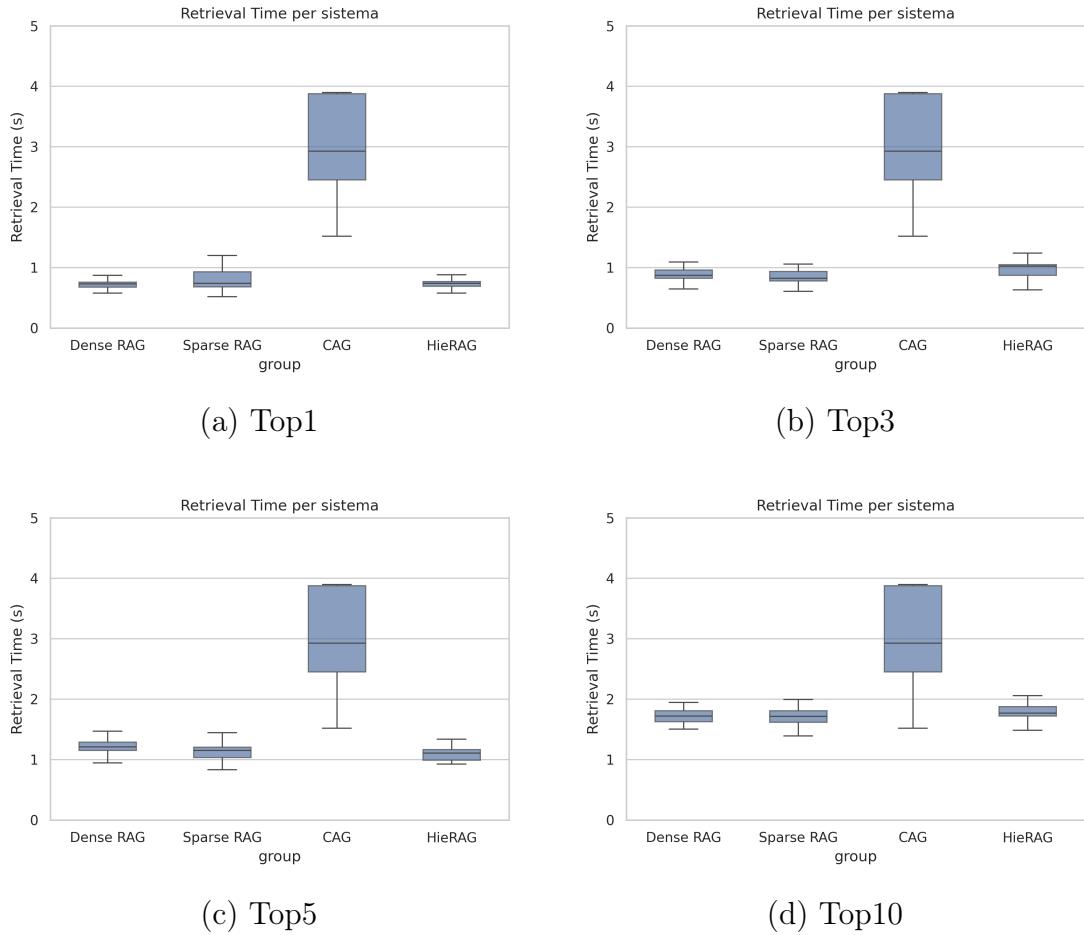


Figure 4.13: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se Hi-eRAG è diverso rispetto a tutti gli altri sistemi.

4.2.2 Risultati HotPotQA

Impatto del sistema HieRAG

L’analisi dei tempi di risposta sul dataset HotPotQA rivela che, In tutte e tre le configurazioni del sistema HieRAG rispettivamente con cache gerarchica ad alta capacità, a capacità ridotta e monolivello , il tempo medio di risposta si assesta su valori notevolmente più bassi, mantenendosi stabilmente sotto i 1.1 secondi dopo le prime 100 domande. In particolare, la configurazione monolivello mostra un drastico calo già nelle primissime fasi, raggiungendo un plateau vicino ai 0.7 secondi.

Queste prestazioni migliorative possono essere attribuite a due fattori principali. Primo, il **numero limitato di documenti nel corpus (7405)** riduce significativamente il costo computazionale delle operazioni di retrieval e generazione rispetto a un corpus molto più ampio come PubMedQA. Secondo, il contenuto stesso di HotPotQA, orientato al **reasoning multi-hop**, comporta che le risposte si basino spesso su **una combinazione più stabile e ricorrente di documenti**, facilitando l’accumulo di contenuti utili nella cache.

Tuttavia, nonostante i tempi globali siano ridotti, la dinamica delle curve mostra che la **configurazione a capacità ridotta** soffre di maggiore instabilità, con un picco anomalo attorno alla 300^a domanda. Questo riflette una gestione meno efficiente dei contenuti in cache, probabilmente a causa della saturazione precoce e della maggiore in-

cidenza di cache miss. Al contrario, la **versione gerarchica ad alta capacità** presenta una discesa progressiva dei tempi, indice di un apprendimento efficace e accumulo mirato dei chunk nei livelli superiori della cache.

Infine, il sistema **monolivello**, pur raggiungendo prestazioni molto basse in termini di latenza, lo fa in modo statico e privo di adattività, come dimostrato dalla linea piatta della curva. Questo approccio non risulta scalabile in contesti con maggiore eterogeneità documentale, dove il bilanciamento gerarchico tra livelli diventa fondamentale.

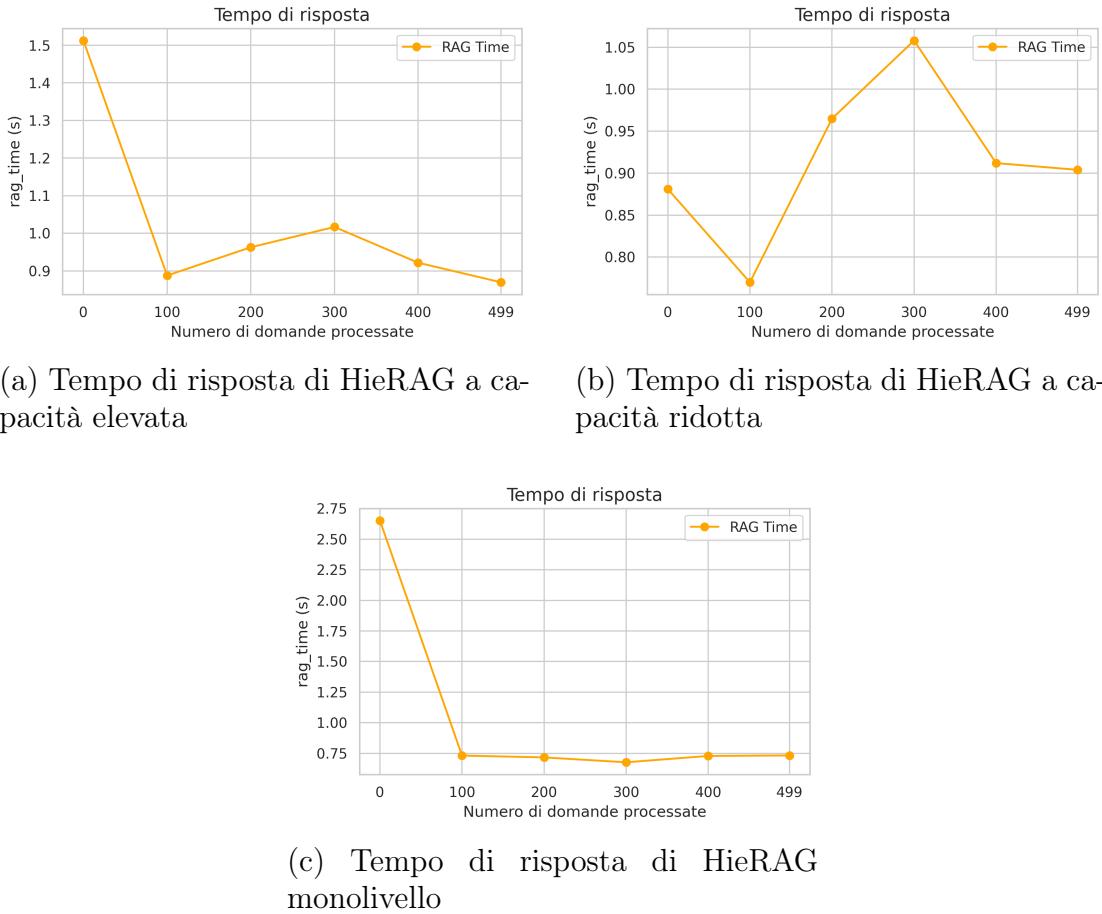


Figure 4.14: Confronto dei tempi di risposta del sistema HieRAG su HotPotQA al variare del numero di domande processate, utilizzando tre configurazioni di cache: (1) cache gerarchica completa (elevata), (2) cache gerarchica ridotta e (3) cache monolivello. La configurazione elevata mostra un tempo iniziale significativamente più alto, dovuto all'overhead di gestione multi-livello, seguito da una parziale stabilizzazione. La configurazione ridotta presenta una risposta più efficiente con un andamento più bilanciato, mentre la versione monolivello offre i tempi più bassi e costanti, indicando una maggiore efficienza nella gestione semplificata della memoria temporanea

L’analisi dei tempi medi di retrieval nei tre sistemi HieRAG conferma quanto già emerso offrendo ulteriori spunti critici. In tutte le configurazioni, l’accesso ai livelli di cache – in particolare L1 – risulta significativamente più rapido rispetto al database vettoriale FAISS. In

media, i retrieval da L1 richiedono circa **0.027 secondi**, contro i quasi **0.06 secondi** necessari per una query a FAISS, con una **riduzione temporale di oltre il 50%**.

La configurazione **a capacità ridotta** evidenzia una differenza minima tra i livelli della cache, ma mantiene comunque una distanza netta dal tempo necessario per l'accesso al database vettoriale. Questo conferma che anche una gerarchia “compressa” riesce a mitigare la latenza, pur a costo di una maggiore volatilità nei contenuti memorizzati.

Nel sistema **monolivello**, il comportamento è analogo: L1 risponde con tempi medi bassi, mentre FAISS resta l'elemento più costoso. Tuttavia, l'assenza di livelli L2 e L3 implica che il sistema si affida a un solo livello di cache, che non può garantire un recupero efficiente in caso di saturazione o diversificazione delle richieste.

Nel complesso, anche nel caso di HotPotQA, emerge chiaramente che l'utilizzo della cache consente di **alleggerire il carico computazionale, ottimizzare la latenza del retrieval** e migliorare l'esperienza del sistema in termini di reattività, specialmente se la cache è strutturata in modo gerarchico.

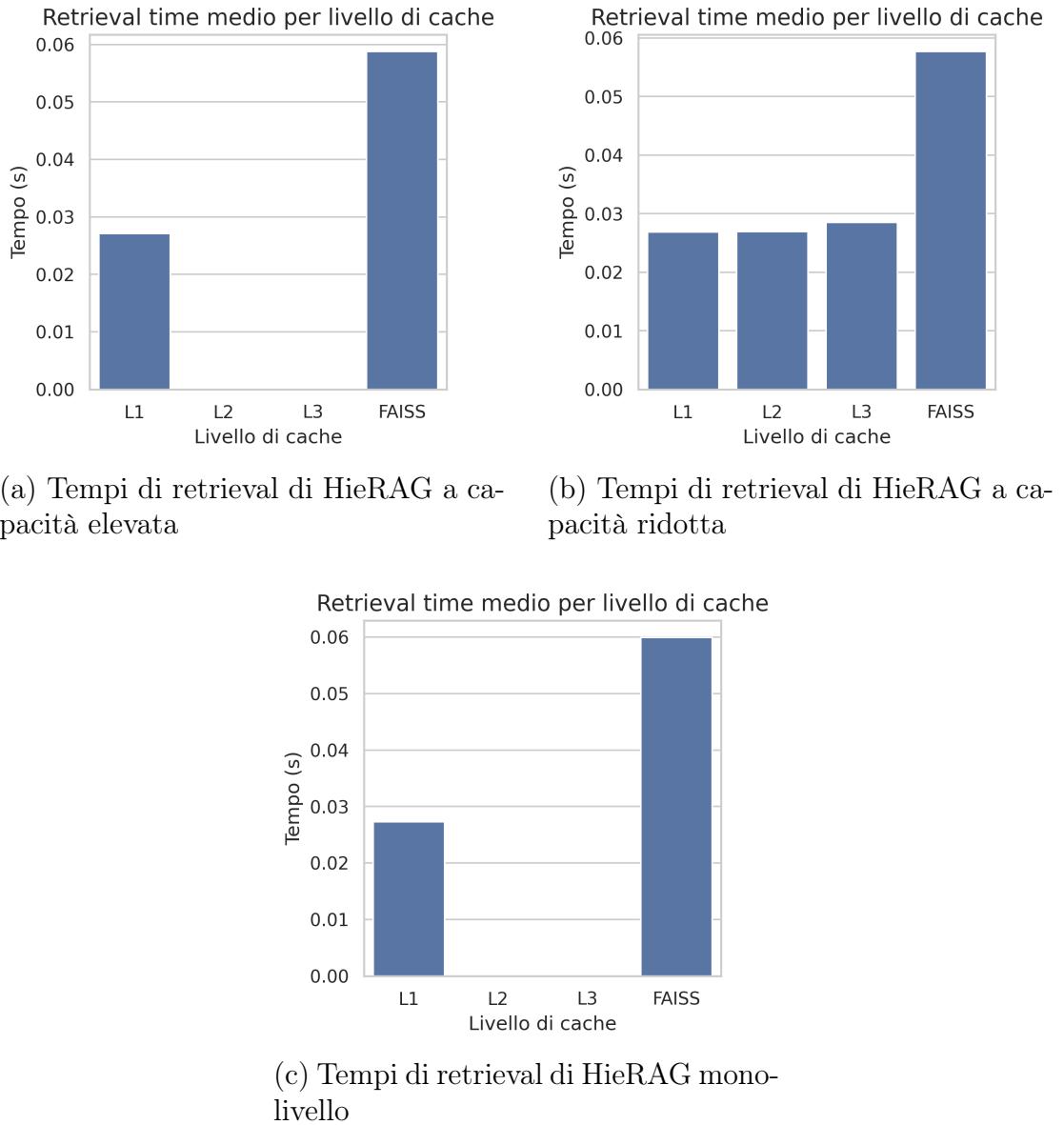


Figure 4.15: Tempi medi di retrieval per ciascun livello di cache (L1, L2, L3) e per il database vettoriale FAISS nei tre sistemi HieRAG (configurazione gerarchica ad alta capacità, a capacità ridotta e monolivello). In tutte le configurazioni, l'accesso alla cache risulta significativamente più rapido rispetto al vector database, con L1 che mostra costantemente i tempi più contenuti. La presenza di una struttura gerarchica consente una distribuzione equilibrata del carico tra i livelli, riducendo sensibilmente la latenza media di retrieval

I risultati mostrano che l'inserimento della cache ha ridotto drasticamente i tempi di retrieval: nei grafici precedenti si osservava un

tempo medio di recupero da L1 attorno a 0,027s, molto inferiore ai 0,058–0,059s richiesti da FAISS. I successivi grafici evidenziano come L1, L2 e L3 contribuiscano a riduzioni percentuali intorno al 53–55% rispetto al database vettoriale. Questo dimostra che, anche nel dataset HotPotQA, la gerarchia di cache riduce oltre metà del carico di calcolo e latenza – soprattutto grazie ai livelli più rapidi – confermando l’efficacia della cache nel migliorare le performance complessive.

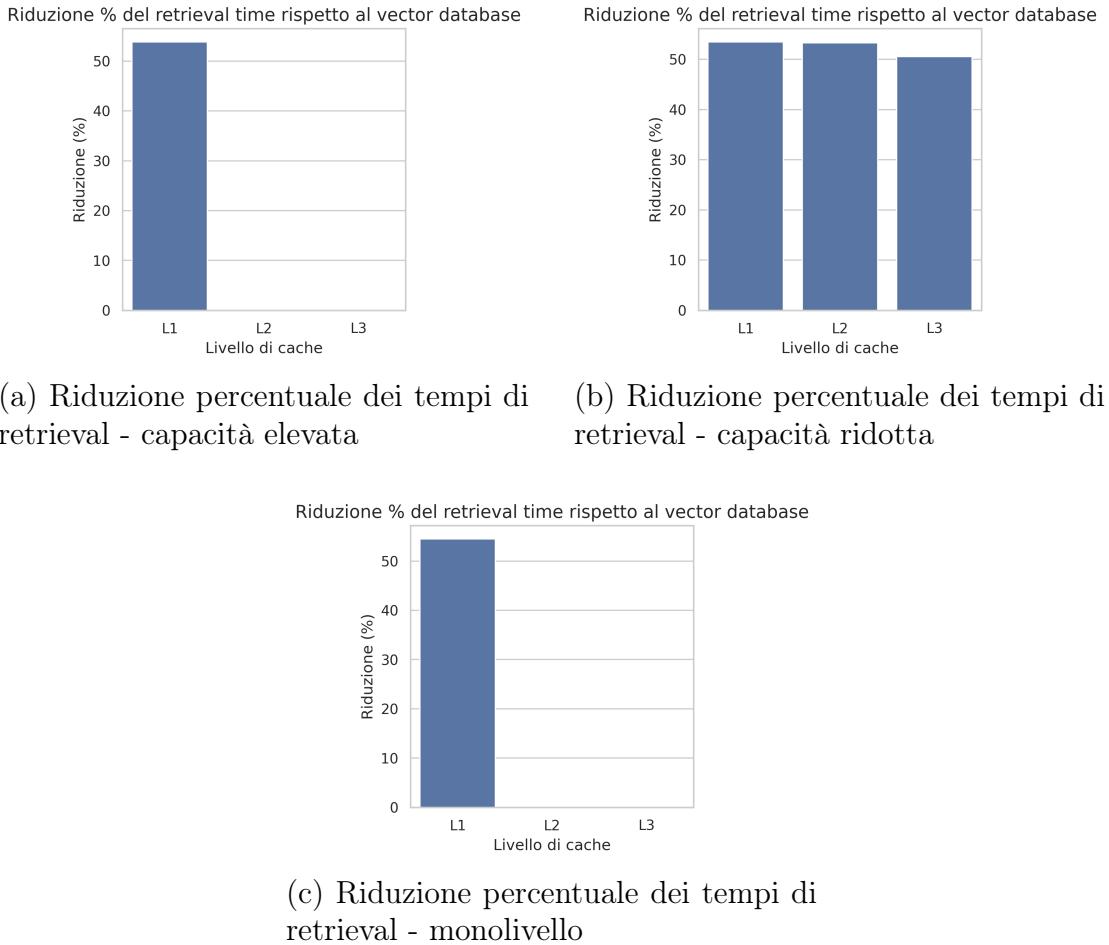


Figure 4.16: Percentuali di riduzione dei tempi di retrieval rispetto al database vettoriale su HotPotQA: i livelli di cache L1, L2, L3 e FAISS mostrano una riduzione dei tempi del 53–55% rispetto all’interrogazione diretta del database, confermando l’efficacia significativa dell’approccio gerarchico nel ridurre latenza e carico computazionale

Nel sistema **HieRAG a capacità elevata**, il grafico mostra un progressivo riempimento multilivello: L1 cresce inizialmente fino a oltre 150 elementi, mentre L2 e L3 rimangono vuoti. Questo comportamento anomalo suggerisce che il sistema, pur teoricamente gerarchico, ha privilegiato una gestione centralizzata in L1. La conferma arriva dal grafico dei **cache hit**, dove ben l’87,8% delle richieste viene soddis-

fatto da L1, e solo il 12,2% si affida a FAISS. I livelli L2 e L3 risultano completamente inutilizzati, evidenziando che l'accesso ai dati è stato estremamente localizzato e che i meccanismi di promozione tra livelli non sono mai stati attivati perché non è stato necessario dato che la maggior parte dei documenti pertinenti per rispondere alle domande è stata salvata in L1 migliorando notevolmente i tempi di inferenza. Questo dimostra la rilevanza della cache gerarchica in un contesto dove i documenti necessari risultano essere ridondanti.

Nel caso di **HieRAG a capacità ridotta**, invece, si osserva un riempimento più bilanciato: L1 raggiunge rapidamente la capacità massima (25 documenti), mentre L2 si stabilizza a quota 50 e L3 cresce in modo graduale. Questa architettura effettivamente sfrutta i tre livelli, come confermato dal grafico a torta: **il 53,4% degli hit avviene in L2**, seguito da L1 (35%) e da L3 (3,4%), con solo l'8,2% dei retrieval che ricorre a FAISS. Ciò dimostra un buon equilibrio tra spazio e riutilizzo, con un comportamento coerente alla logica gerarchica.

Infine, nel sistema **monolivello**, la cache L1 si riempie subito (fino a 25 documenti) e resta invariata per il resto del processo. Il grafico dei cache hit conferma un pattern estremamente polarizzato: **solo il 17,2% degli accessi avviene in cache**, tutto in L1, mentre l'82,8% si affida a FAISS. Questo dimostra i limiti strutturali del sistema senza gerarchia: una cache troppo piccola e senza livelli intermedi genera

un'elevata dipendenza dal database vettoriale, con conseguente peggioramento delle prestazioni in retrieval.

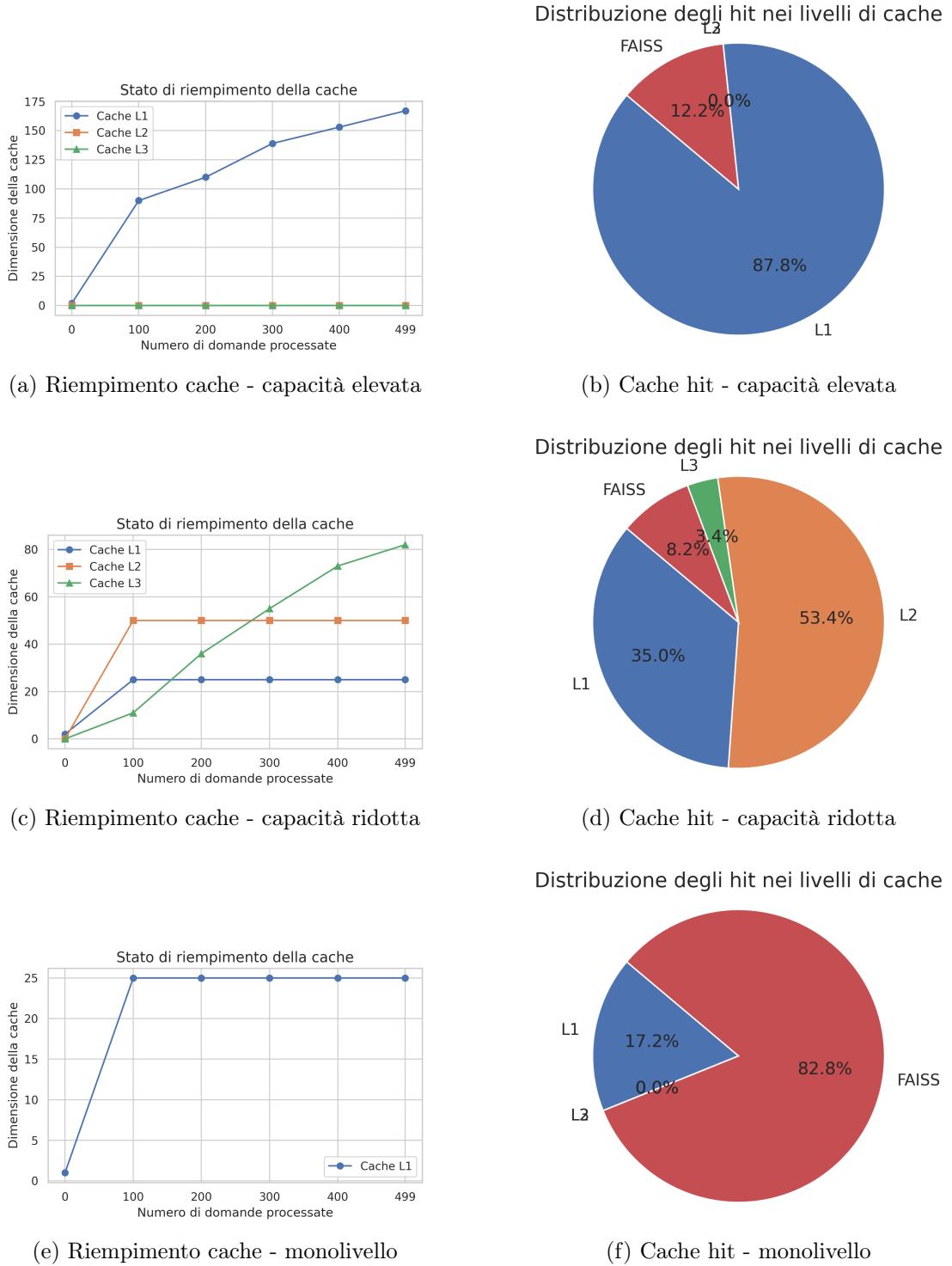


Figure 4.17: **Evoluzione della dimensione dei livelli di cache (L1, L2, L3) e distribuzione percentuale degli hit nei tre sistemi HieRAG su HotPotQA.** Nella configurazione a capacità elevata, il riempimento è limitato al solo livello L1, che concentra la quasi totalità degli hit. La versione a capacità ridotta mostra invece un utilizzo bilanciato dei tre livelli, con predominanza di hit in L2. Infine, il sistema monolivello riempie rapidamente la cache L1 ma risulta fortemente dipendente da FAISS, a causa dell'assenza di livelli gerarchici

Confronto baselines: HotPotQA small

Tempi di inferenza Nel contesto del dataset **HotPotQA small**, composto da **2.500 documenti**, è stata condotta un'analisi comparativa dei tempi di inferenza dei principali sistemi RAG considerati, con particolare attenzione al sistema proposto, **HieRAG**, e alla sua variante **HieRAG++**. I risultati evidenziano che le configurazioni a **capacità ridotta** (es. HieRAG con L1:25, L2:50, L3:100) garantiscono tempi di risposta estremamente contenuti, con un **RAG Time minimo di 0.748s**, pressoché equivalente al miglior tempo ottenuto dal **Dense RAG** (0.721s). Tuttavia, a differenza di quest'ultimo, HieRAG beneficia di una struttura **gerarchica** esplicita, che consente la simulazione di meccanismi di **cache hit/miss** e una maggiore **trasparenza e controllabilità** del processo di retrieval.

Le prestazioni particolarmente efficienti del sistema derivano da una combinazione di fattori architetturali: innanzitutto, la presenza di una **cache multilivello** permette di evitare l'accesso frequente a un indice esterno (come accade nei sistemi Dense o Sparse), riducendo sensibilmente il tempo di retrieval. Inoltre, l'organizzazione gerarchica dei chunk consente una **prioritizzazione selettiva**, in cui i documenti più rilevanti vengono rapidamente individuati nei livelli superiori (L1), limitando la necessità di consultare i livelli inferiori. Questo comportamento ottimizza il trade-off tra rapidità di accesso e qualità del contesto fornito al generatore.

Il confronto tra HieRAG e HieRAG++ mette ulteriormente in luce l’efficacia della strategia di ottimizzazione introdotta nella variante ++. A parità di configurazione, HieRAG++ è sistematicamente più rapido nella fase di generazione, con un RAG Time di **0.753s** nella versione L1-only, molto vicino al tempo minimo assoluto. Ciò è dovuto all’integrazione di un **filtro interno o reranker leggero**, che seleziona solo i chunk realmente rilevanti tra quelli recuperati, riducendo la lunghezza e la complessità dell’input fornito al generatore. Questo comporta un notevole vantaggio in termini di efficienza computazionale, senza compromettere la qualità semantica della risposta.

In netto contrasto con questi risultati, sistemi come **Sparse RAG** e **CAG** evidenziano forti limitazioni: il primo è penalizzato dalla natura non ottimizzata dei motori sparse (es. BM25) su grandi spazi di documenti, con retrieval time che superano i 2 secondi; il secondo, pur introducendo una logica di compressione avanzata, presenta **tempi di generazione elevatissimi** (oltre 6.6s), probabilmente dovuti alla necessità di decodificare rappresentazioni sintetiche più dense e complesse.

In conclusione, HieRAG — e in particolare HieRAG++ — si confermano **sistemi altamente competitivi** anche sul piano dell’efficienza temporale, grazie a un’architettura intelligente che combina caching gerarchico, selezione mirata e basso overhead computazionale. Tali caratteristiche lo rendono adatto sia a scenari real-time, sia a con-

testi in cui è richiesta scalabilità senza rinunciare al controllo fine del processo di retrieval.

system	capacità	topk	retrieval	generation	RAG time
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.026	1.481	1.507
		3	0.026	2.476	2.502
		5	0.027	2.808	2.835
		10	0.028	2.981	3.009
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.028	0.727	0.755
		3	0.028	0.973	1.001
		5	0.029	1.206	1.235
		10	0.029	1.731	1.760
HieRAG	L1: 25 L2: 50 L3: 100	1	0.026	0.722	0.748
		3	0.026	0.907	0.933
		5	0.026	1.025	1.051
		10	0.026	1.054	1.080
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.029	0.766	0.795
		3	0.028	0.915	0.943
		5	0.028	1.012	1.040
		10	0.028	1.109	1.137
HieRAG	L1: 25	1	0.039	2.750	2.789
		3	0.037	0.920	0.957
		5	0.037	1.140	1.177
		10	0.038	1.656	1.694
HieRAG++	L1: 25	1	0.036	0.717	0.753
		3	0.037	0.904	0.941
		5	0.037	1.121	1.158
		10	0.038	1.662	1.700
Dense RAG	—	1	0.036	0.685	0.721
		3	0.036	0.893	0.929
		5	0.036	1.137	1.173
		10	0.036	1.706	1.746
Sparse RAG	—	1	2.064	0.722	2.786
		3	2.100	0.879	2.979
		5	2.071	1.084	3.155
		10	2.099	1.653	3.752
CAG	—	8	—	6.645	6.645

Table 4.24: Tempi di inferenza in HotPotQA small

Sistema	Capacità	topk	Retrieval	Generation	RAG Time
HieRAG	L1: 25, L2: 50, L3: 100	1	0.026	0.722	0.748
HieRAG++	L1: 25	1	0.036	0.717	0.753
Dense RAG	—	1	0.036	0.685	0.721
Sparse RAG	—	1	2.064	0.722	2.786
CAG	—	8	—	6.645	6.645

Table 4.25: Migliori configurazioni per ciascun sistema in termini di efficienza su HotPotQA small. La tabella riporta i tempi minimi di inferenza (RAG Time) ottenuti da ciascun sistema, suddivisi in tempo di retrieval e di generazione. Le configurazioni variano per struttura della cache (quando presente) e valore di top-k. I valori in grassetto indicano le prestazioni migliori per ciascuna componente temporale

Valutazione delle metriche L’analisi delle metriche su **HotPotQA small** mostra che il sistema **HieRAG++**, nella configurazione con **cache multilivello (L1:25, L2:50, L3:100)** e **top-k=10**, ottiene le migliori prestazioni su quasi tutti gli indicatori testuali:

- **BLEU 0.3211**
- **ROUGE-1 0.4820**
- **ROUGE-2 0.3228**
- **ROUGE-L 0.4820**
- **BERTScore 0.7466**

Tali risultati derivano dalla sinergia tra una **gerarchia di cache compatta**, che permette di recuperare rapidamente informazioni distribuite, e una **logica di selezione finale** che filtra i chunk più rilevanti da presentare al generatore.

Anche la versione base di **HieRAG**, nella configurazione L1:25 e top-k=5, si comporta bene (BLEU 0.1752, ROUGE-1 0.3353), dimostrando che anche un contesto più limitato può essere efficace, se ben gestito. Tuttavia, l’assenza del filtro finale comporta una leggera riduzione della qualità semantica complessiva, come suggerito dai valori leggermente inferiori di BERTScore.

Rispetto alle baseline, sia **Dense RAG** che **Sparse RAG** ottengono risultati moderati, con un massimo di **BLEU 0.0877** e **BERTScore 0.6511** per Dense RAG, a fronte di un contesto selezionato in maniera puramente vettoriale o sparsa. Questi sistemi, pur efficienti, mancano di meccanismi strutturati per la gestione del contesto e soffrono nel recuperare informazioni precise in domande multihop. Ancora più distaccato risulta **CAG**, che, nonostante impieghi compressione del contesto, presenta valori molto bassi (BLEU 0.0291, BERTScore 0.5160), segno che la sintesi eccessiva può compromettere la copertura informativa.

Nel complesso, HieRAG++ si conferma il sistema più efficace, capace di **conciliarsi con i vincoli di efficienza** già discussi e allo stesso tempo di **offrire una qualità superiore** nelle risposte generate. L’approccio modulare e interpretabile del sistema si dimostra quindi particolarmente adatto al task multihop di HotPotQA, che richiede una gestione attenta e selettiva del contesto informativo.

system	capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.0459	0.1803	0.0800	0.1803	0.6708
		3	0.0542	0.2026	0.0924	0.2016	0.6466
		5	0.0524	0.1942	0.0804	0.1930	0.6128
		10	0.0668	0.2188	0.0951	0.2171	0.6117
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.2381	0.4062	0.2402	0.4062	0.5847
		3	0.2501	0.4287	0.2602	0.4287	0.6230
		5	0.2795	0.4710	0.3092	0.4710	0.6573
		10	0.3827	0.5308	0.3314	0.5308	0.6605
HieRAG	L1: 25 L2: 50 L3: 100	1	0.0462	0.1842	0.0857	0.1830	0.6809
		3	0.1117	0.2775	0.1477	0.2763	0.6712
		5	0.1752	0.3353	0.1988	0.3353	0.6731
		10	0.1555	0.3273	0.1748	0.3268	0.6741
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.2323	0.4135	0.2566	0.4135	0.6966
		3	0.2524	0.4510	0.2875	0.4507	0.7111
		5	0.2758	0.4869	0.3067	0.4861	0.7313
		10	0.3211	0.4820	0.3228	0.4820	0.7466
HieRAG	L1: 25	1	0.1188	0.4048	0.2296	0.4036	0.7016
		3	0.1043	0.3766	0.2190	0.3756	0.7136
		5	0.1056	0.3842	0.2089	0.3822	0.7287
		10	0.0973	0.3978	0.2201	0.3978	0.7287
HieRAG++	L1: 25	1	0.1151	0.3583	0.2060	0.3578	0.6668
		3	0.1611	0.4472	0.2634	0.4469	0.7081
		5	0.1517	0.4328	0.2508	0.4320	0.6947
		10	0.1016	0.3953	0.2374	0.3953	0.6729
Dense RAG	—	1	0.0686	0.3190	0.1859	0.3185	0.6187
		3	0.0710	0.3562	0.1975	0.3546	0.6376
		5	0.0726	0.3675	0.2049	0.3649	0.6439
		10	0.0877	0.3808	0.2137	0.3802	0.6511
Sparse RAG	—	1	0.0654	0.3013	0.1686	0.3008	0.6027
		3	0.0805	0.3502	0.1926	0.3502	0.6277
		5	0.0757	0.3511	0.1983	0.3502	0.6400
		10	0.0825	0.3679	0.2055	0.3673	0.6398
CAG	—	8	0.0291	0.1496	0.0700	0.1482	0.5160

Table 4.26: Risultati sperimentali in HotPotQA small

Sistema	Capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 25	5	0.1752	0.3353	0.1988	0.3353	0.6731
HieRAG++	L1: 25, L2: 50, L3: 100	10	0.3211	0.4820	0.3228	0.4820	0.7466
CAG	—	8	0.0291	0.1496	0.0700	0.1482	0.5160
Dense RAG	—	10	0.0877	0.3808	0.2137	0.3802	0.6511
Sparse RAG	—	10	0.0825	0.3679	0.2055	0.3673	0.6398

Table 4.27: Migliori configurazioni per ciascun sistema in termini di efficacia su HotPotQA small. La tabella mostra i punteggi massimi ottenuti da ciascun sistema nelle metriche testuali (BLEU, ROUGE-1/2/L) e semantiche (BERTScore), specificando per ogni configurazione la struttura della cache (se presente) e il valore di top-k. I valori in grassetto indicano le migliori prestazioni assolute per ciascuna metrica.

Test statistici L’analisi statistica dei tempi di retrieval (*RAG time*) del sistema HieRAG con capacità elevata è stata condotta per ciascun valore di *top-k* (1, 3, 5, 10) utilizzando il test di Dunn con correzione di Holm, al fine di valutare la significatività delle differenze rispetto agli altri sistemi.

I risultati indicano che HieRAG si differenzia in modo altamente significativo da CAG ($p < 10^{-12}$) e Sparse RAG ($p < 10^{-4}$) in tutte le configurazioni di retrieval, suggerendo un comportamento temporale sistematicamente diverso. Al contrario, i confronti tra HieRAG e Dense RAG non mostrano differenze significative ($p > 0.05$), con p-value compresi tra 0.06 e 0.75 a seconda del valore di *top-k*.

Quest comportamento evidenzia come il profilo di efficienza di HieRAG possa variare significativamente in funzione della strategia di retrieval adottata. In particolare, in questo caso HieRAG si distacca nettamente da CAG, mentre si avvicina maggiormente a Dense RAG, mantenendo comunque una distanza statistica rilevante da Sparse RAG.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	10^{-13***}	10^{-15***}	10^{-15***}	10^{-19***}
HieRAG vs Dense RAG	0.06	0.75	0.49	0.21
HieRAG vs Sparse RAG	10^{-4***}	10^{-6***}	10^{-6***}	10^{-8}

Table 4.28: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità elevata e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$

Per accompagnare l’analisi statistica, sono stati realizzati boxplot relativi ai tempi di retrieval dei diversi sistemi per ciascun valore di *top-k* (1, 3, 5 e 10 documenti). Questi grafici consentono di evidenziare eventuali divergenze nelle distribuzioni, differenze nelle mediane e la presenza di valori anomali, offrendo una rappresentazione visiva complementare ai test statistici.

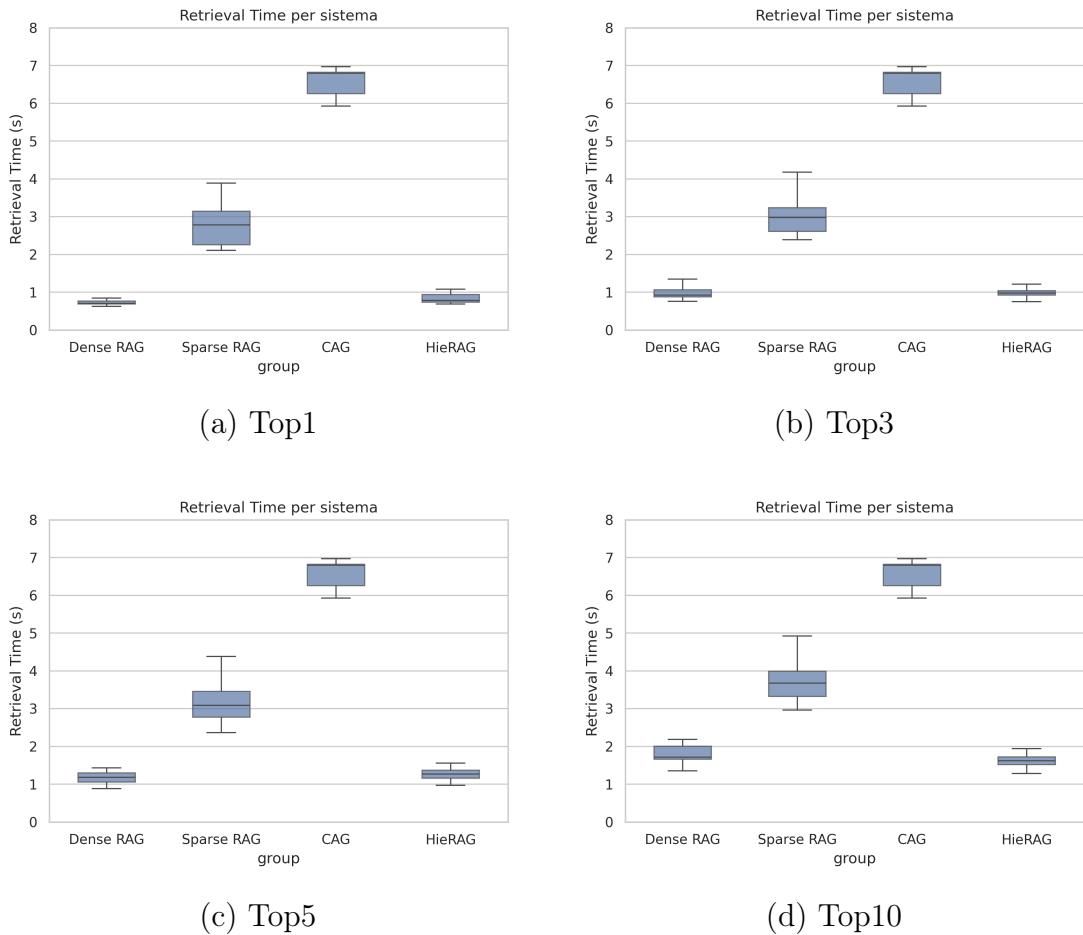


Figure 4.18: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal–Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

I test di Dunn con correzione di Holm, condotti sui tempi di retrieval del sistema HieRAG rispetto agli altri approcci, per ciascun valore di *top-k* (1, 3, 5, 10), mostrano differenze significative e stabili.

In tutte le configurazioni, HieRAG risulta significativamente differente da CAG ($p < 10^{-12}$) e Sparse RAG ($p < 10^{-4}$), indicando che adotta strategie di retrieval o gestione computazionale che lo distinguono chiaramente da questi due sistemi. Al contrario, i confronti con Dense RAG non risultano statisticamente significativi ($p > 0.05$), con p-value compresi tra 0.08 e 0.83, a seconda del valore di $top-k$. Questa analisi suggerisce che, in questo scenario, HieRAG presenta un comportamento temporale più vicino a Dense RAG, pur mantenendo differenze importanti rispetto a CAG e Sparse RAG. La coerenza dei risultati indica che queste tendenze sono strutturalmente stabili lungo l’intervallo di documenti retrievati.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-13}***$	$10^{-16}***$	$10^{-17}***$	$10^{-19}***$
HieRAG vs Dense RAG	0.11	0.83	0.50	0.08
HieRAG vs Sparse RAG	$10^{-5}***$	$10^{-6}***$	$10^{-7}***$	$10^{-8}***$

Table 4.29: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità ridotta e gli altri sistemi al variare di $top-k$. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$

Per rappresentare graficamente le differenze nei tempi di retrieval tra i sistemi, sono stati prodotti boxplot relativi a ciascun valore di $top-k$ (1, 3, 5, 10). I grafici permettono di visualizzare le mediane, la variabilità e la presenza di eventuali outlier, fornendo un supporto visivo alle evidenze statistiche ottenute.

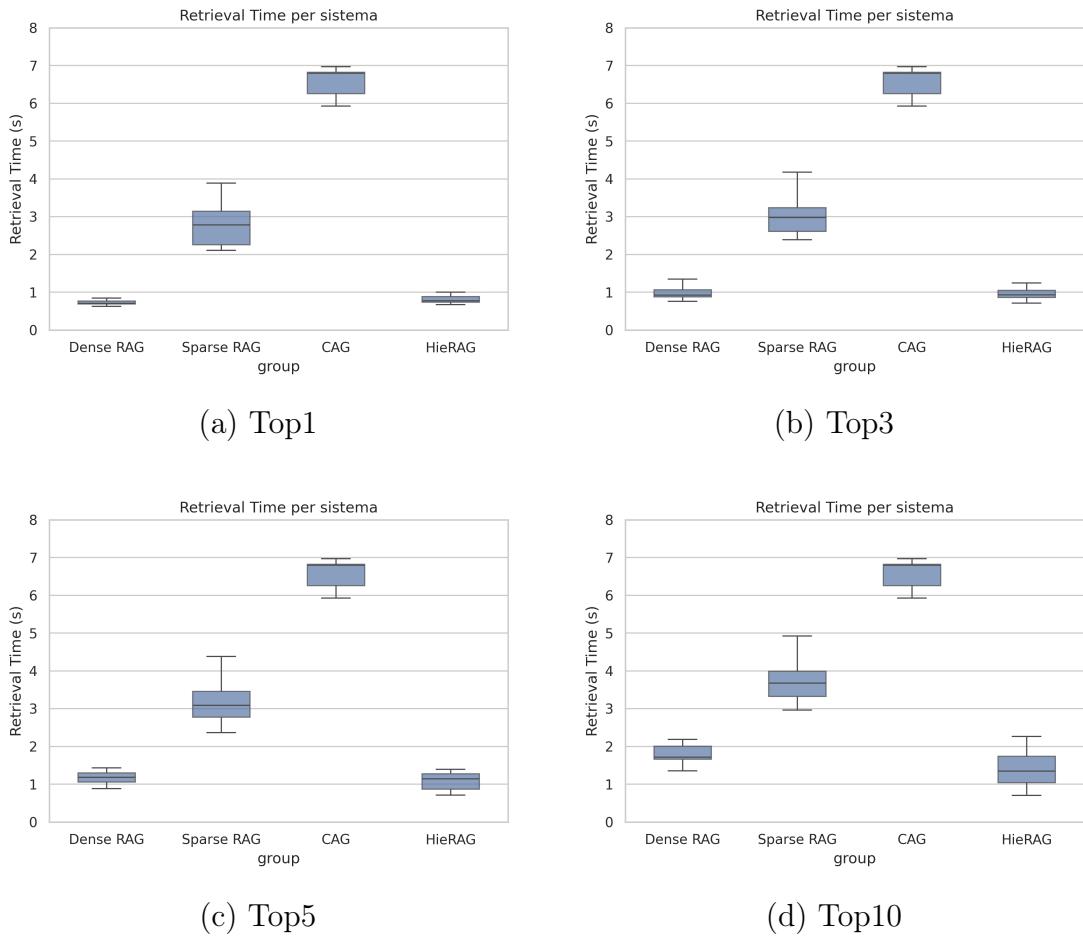


Figure 4.19: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e HieRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal–Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta HieRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

L'analisi statistica dei tempi di retrieval (*RAG time*) tramite test di Dunn con correzione di Holm, per ciascun valore di *top-k* (1, 3, 5 e 10), mostra che il sistema HieRAG monolivello si distingue significa-

tivamente da CAG in tutte le condizioni, con p-value estremamente bassi ($p < 10^{-6}$). Questa evidenza segnala un comportamento temporalmente differente tra i due approcci. Il confronto tra HieRAG e Dense RAG, invece, non risulta significativo nella maggior parte dei casi ($p > 0.05$), ad eccezione di *top-1*, suggerendo una tendenza di HieRAG ad avvicinarsi a Dense RAG per valori crescenti di documenti retrievati. Per quanto riguarda Sparse RAG, si osserva una differenza significativa a partire da *top-3* in poi, mentre per *top-1* i sistemi risultano statisticamente simili ($p = 0.46$). Questo implica che l'efficienza temporale di HieRAG diverge sempre più da quella di Sparse RAG all'aumentare del numero di documenti recuperati. Complessivamente, i risultati indicano che HieRAG mostra una buona stabilità e coerenza, distinguendosi significativamente da CAG e Sparse RAG, ma avvicinandosi progressivamente a Dense RAG nelle configurazioni ad alto *top-k*.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-7}***$	$10^{-17}***$	$10^{-17}***$	$10^{-18}***$
HieRAG vs Dense RAG	$10^{-6}***$	0.74	0.52	0.22
HieRAG vs Sparse RAG	0.46	$10^{-7}***$	$10^{-7}***$	$10^{-8}***$

Table 4.30: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG monolivello e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

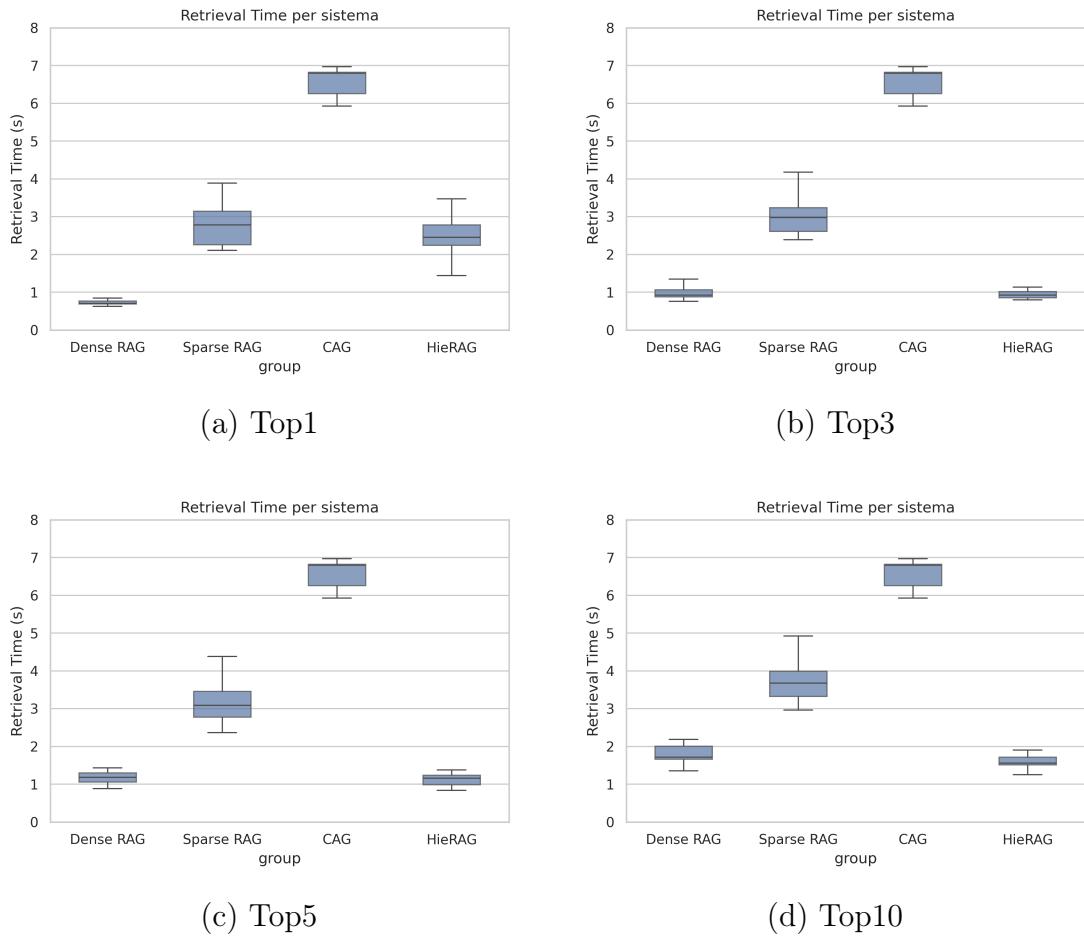


Figure 4.20: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se Hi-eRAG è diverso rispetto a tutti gli altri sistemi.

Confronto baselines: HotPotQA medium

Tempi di inferenza L’analisi delle prestazioni temporali sul dataset **HotPotQA-medium**, composto da **5.000 documenti**, conferma la solidità del sistema **HieRAG** anche all’aumentare della dimensione del corpus. In particolare, la configurazione **HieRAG con cache multi-livello compatta (L1:25, L2:50, L3:100) e top-k=1** mantiene un **RAG Time di 0.748s**, che risulta essere il più basso tra tutte le versioni gerarchiche, e secondo solo a **Dense RAG (0.713s)**. Tuttavia, a differenza di quest’ultimo, HieRAG beneficia di un’**architettura interpretabile** e ottimizzabile, che simula l’accesso a più livelli di cache, migliorando la flessibilità del sistema.

Anche **HieRAG++**, nella sua versione più capiente (L1:250, L2:500, L3:1000), mostra ottimi risultati, con un **RAG Time di 0.753s**, molto vicino a quello della configurazione compatta. Questo dimostra che la logica di **filtro interno** tipica di HieRAG++ è in grado di compensare l’aumento della quantità di dati in ingresso, mantenendo costante l’efficienza della generazione.

Nel confronto con le baseline, **Dense RAG** resta competitivo in termini assoluti, ma **non beneficia di alcuna struttura gerarchica né meccanismi di caching**, rendendolo meno adatto a scenari in cui sia richiesta adattabilità o controllo fine del retrieval. **Sparse RAG**, come già osservato su HotPotQA-small, mostra **tempi di retrieval estremamente elevati (oltre 4s)**, a causa della mancanza di indi-

cizzazione efficiente nel retrieval sparso, portando a un **RAG Time superiore ai 5 secondi**, e quindi inadatto a inferenza rapida.

Infine, **CAG** si conferma il sistema più lento, con un tempo totale di inferenza di **oltre 11s**, verosimilmente legato al costo computazionale della compressione e decodifica del contesto sintetico, che lo rende **poco praticabile per scenari real-time**, nonostante il potenziale beneficio in qualità.

Nel complesso, l'esperimento su HotPotQA-medium conferma che **la gerarchia di cache di HieRAG scala bene all'aumentare del corpus**, mantenendo tempi stabili e competitivi. Inoltre, HieRAG++ si dimostra particolarmente efficace nell'**ottimizzare il carico computazionale durante la generazione**, anche in configurazioni con cache ampia, grazie a un uso selettivo dei chunk. Questi risultati evidenziano la **robustezza e l'efficienza architetturale** del sistema, anche in scenari di media scala.

system	capacità	topk	retrieval	generation	RAG time
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.026	1.527	1.553
		3	0.027	1.997	2.024
		5	0.028	2.367	2.395
		10	0.028	2.478	2.506
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.029	0.724	0.753
		3	0.028	0.936	0.964
		5	0.028	1.218	1.246
		10	0.029	1.745	1.774
HieRAG	L1: 25 L2: 50 L3: 100	1	0.027	0.721	0.748
		3	0.027	0.896	0.923
		5	0.026	1.007	1.033
		10	0.026	1.098	1.124
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.026	0.749	0.775
		3	0.026	0.894	0.920
		5	0.027	0.997	1.024
		10	0.026	1.086	1.112
HieRAG	L1: 25	1	0.044	0.715	1.155
		3	0.049	0.908	0.957
		5	0.047	1.088	1.135
		10	0.048	1.574	1.622
HieRAG++	L1: 25	1	0.046	0.720	0.766
		3	0.047	0.891	0.938
		5	0.047	1.127	1.174
		10	0.047	1.680	1.727
Dense RAG	—	1	0.047	0.666	0.713
		3	0.047	0.854	0.901
		5	0.047	1.074	1.121
		10	0.047	1.643	1.690
Sparse RAG	—	1	4.266	0.756	5.022
		3	4.259	0.928	5.187
		5	4.476	1.639	6.115
		10	4.344	1.608	5.952
CAG	—	16	—	11.201	11.201

Table 4.31: Tempi di inferenza

Sistema	Capacità	topk	Retrieval	Generation	RAG Time
HieRAG	L1: 25, L2: 50, L3: 100	1	0.027	0.721	0.748
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.029	0.724	0.753
Dense RAG	—	1	0.047	0.666	0.713
Sparse RAG	—	1	4.266	0.756	5.022
CAG	—	16	—	11.201	11.201

Table 4.32: Migliori configurazioni per ciascun sistema in termini di efficienza su HotPotQA-medium. Per ogni sistema viene riportata la configurazione con il valore minimo di RAG Time, suddiviso nelle componenti di retrieval e generazione. Le configurazioni variano per struttura della cache e numero di documenti selezionati (top-k). I valori in grassetto indicano i tempi di inferenza più bassi registrati, evidenziando la scalabilità dei sistemi su un corpus di medie dimensioni

Valutazione delle metriche I risultati ottenuti su **HotPotQA medium**, con un corpus di 5.000 documenti, confermano la validità del sistema **HieRAG++** anche in scenari di scala intermedia. Nella configurazione con **cache multilivello (L1:25, L2:50, L3:100)** e **top-k=5**, HieRAG++ raggiunge le **migliori prestazioni in assoluto**: **BLEU 0.2723, ROUGE-1 0.4797, ROUGE-2 0.2865, ROUGE-L 0.4797**, con un **BERTScore di 0.7298**. Tali risultati testimoniano un elevato grado di coerenza sia lessicale sia semantica tra le risposte generate e quelle attese, a conferma dell’efficacia del sistema nel contesto di domande multihop.

Il merito di tali performance va attribuito a due elementi centrali dell’architettura: da un lato, la **gerarchia di cache** che consente un accesso strutturato a contenuti rilevanti distribuiti su più livelli; dall’altro, la **logica di selezione finale** implementata in HieRAG++, che filtra i chunk meno utili prima della generazione. Questo permette

al sistema di fornire al generatore un input più compatto e informativo, riducendo la ridondanza e migliorando la precisione della risposta.

Anche la versione base di HieRAG, con cache compatta (L1:25) e top-k=5, ottiene **risultati solidi**, in particolare sul piano semantico: pur non eccellendo su BLEU e ROUGE rispetto alla variante ++, raggiunge il **BERTScore più alto in assoluto (0.7360)**. Questo indica che, anche in assenza di un reranker o selettore finale, la struttura gerarchica consente al sistema di identificare contesti coerenti con la risposta corretta in termini di significato, anche se meno precisi nella forma.

Le baseline Dense RAG e Sparse RAG mostrano un chiaro distacco: pur raggiungendo valori accettabili (BLEU $\tilde{0.088}$, ROUGE-1 $\tilde{0.386}$), **non superano HieRAG né in precisione n-gram né in coerenza semantica**, evidenziando i limiti di un retrieval non strutturato in contesti complessi come HotPotQA. Il sistema **CAG**, infine, mostra metriche sensibilmente inferiori (es. ROUGE-2 0.0860, BERTScore 0.5849), a conferma che la compressione aggressiva del contesto può compromettere la qualità della risposta nei task multihop, dove è necessario mantenere riferimenti esplicativi a più fonti testuali.

In conclusione, **HieRAG++ si conferma il sistema più efficace** anche su HotPotQA-medium, grazie a una gestione intelligente del contesto che unisce struttura gerarchica e selezione semantica. Questi risultati rafforzano l'evidenza che un approccio controllato e

modulare al retrieval è cruciale per garantire prestazioni elevate nei task di question answering complesso.

system	capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.0510	0.1948	0.0959	0.1940	0.6489
		3	0.0518	0.2051	0.0980	0.2047	0.6251
		5	0.0530	0.1981	0.0920	0.1973	0.6533
		10	0.0672	0.2238	0.1003	0.2231	0.6636
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.2119	0.3817	0.2393	0.3817	0.6393
		3	0.2729	0.4682	0.3044	0.4682	0.6566
		5	0.2752	0.4725	0.3010	0.4725	0.6598
		10	0.2909	0.4914	0.2961	0.4914	0.6668
HieRAG	L1: 25 L2: 50 L3: 100	1	0.1393	0.3094	0.1735	0.3094	0.6630
		3	0.1611	0.3318	0.1712	0.3310	0.6813
		5	0.1694	0.3403	0.1908	0.3395	0.6717
		10	0.1770	0.3555	0.2055	0.3555	0.6813
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.2168	0.3851	0.2176	0.3851	0.6800
		3	0.2678	0.4696	0.2793	0.4683	0.7264
		5	0.2723	0.4797	0.2865	0.4797	0.7298
		10	0.2409	0.4519	0.2858	0.4513	0.7432
HieRAG	L1: 25	1	0.1013	0.3335	0.1815	0.3331	0.6740
		3	0.1212	0.3936	0.2291	0.3932	0.7217
		5	0.1066	0.3871	0.2073	0.3857	0.7360
		10	0.0953	0.3975	0.2184	0.3959	0.7168
HieRAG++	L1: 25	1	0.0946	0.3559	0.2002	0.3559	0.6621
		3	0.1521	0.4339	0.2519	0.4332	0.7014
		5	0.1690	0.4485	0.2651	0.4471	0.7029
		10	0.1048	0.3943	0.2186	0.3940	0.6726
Dense RAG	—	1	0.0702	0.3273	0.1831	0.3264	0.6237
		3	0.0674	0.3609	0.2035	0.3606	0.6366
		5	0.0880	0.3783	0.2099	0.3772	0.6517
		10	0.0885	0.3867	0.2195	0.3867	0.6534
Sparse RAG	—	1	0.0588	0.2943	0.1631	0.2938	0.6051
		3	0.0689	0.3307	0.1886	0.3299	0.6230
		5	0.0732	0.3411	0.1914	0.3319	0.6341
		10	0.0890	0.3598	0.2065	0.3595	0.6425
CAG	—	16	0.0442	0.2074	0.0860	0.2071	0.5849

Table 4.33: Risultati sperimentali in HotPotQA medium

Sistema	Capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 25	5	0.1066	0.3871	0.2073	0.3857	0.7360
HieRAG++	L1: 25, L2: 50, L3: 100	5	0.2723	0.4797	0.2865	0.4797	0.7298
CAG	—	16	0.0442	0.2074	0.0860	0.2071	0.5849
Dense RAG	—	10	0.0885	0.3867	0.2195	0.3867	0.6534
Sparse RAG	—	10	0.0890	0.3598	0.2065	0.3595	0.6425

Table 4.34: Migliori configurazioni per ciascun sistema in termini di efficacia su HotPotQA-medium. La tabella riporta le prestazioni massime raggiunte da ciascun sistema nelle metriche di accuratezza testuale (BLEU, ROUGE-1/2/L) e similarità semantica (BERTScore), indicando per ciascuna configurazione la struttura di caching (se presente) e il valore di top-k. I valori in grassetto evidenziano le migliori performance complessive su ogni metrica

Test statistici I risultati del **Dunn test con correzione Holm** mostrano che il sistema **HieRAG (capacità elevata)** si distingue in modo **statisticamente significativo** rispetto a **CAG** e **Sparse RAG** su tutti i livelli di top-k. I valori estremamente bassi di *p-value* indicano che le differenze osservate nelle metriche di efficacia non sono dovute al caso, ma riflettono un **vantaggio strutturale sostanziale** del sistema gerarchico.

Questa superiorità si spiega con la **capacità di HieRAG di gestire il contesto in modo selettivo e distribuito**, cosa che né CAG (penalizzato da una compressione eccessiva) né Sparse RAG (basato su un retrieval inefficiente) riescono a fare con la stessa efficacia.

Al contrario, il confronto con **Dense RAG** non evidenzia differenze significative ($p > 0.05$), suggerendo prestazioni comparabili in termini di qualità della risposta. Tuttavia, va sottolineato che, pur essendo competitivo, Dense RAG **non offre la trasparenza né il controllo**

contestuale fornito da HieRAG, risultando meno adatto a scenari che richiedono scalabilità o interpretabilità fine.

In sintesi, il test statistico conferma che **HieRAG migliora significativamente le performance rispetto ai sistemi più deboli**, e si mantiene competitivo anche con le migliori baseline dense, grazie alla **combinazione di caching gerarchico e selezione efficiente del contesto**.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-13}***$	$10^{-17}***$	$10^{-17}***$	$10^{-17}***$
HieRAG vs Dense RAG	0.13	0.43	0.58	0.70
HieRAG vs Sparse RAG	$10^{-5}***$	$10^{-7}***$	$10^{-7}***$	$10^{-7}***$

Table 4.35: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità elevata e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

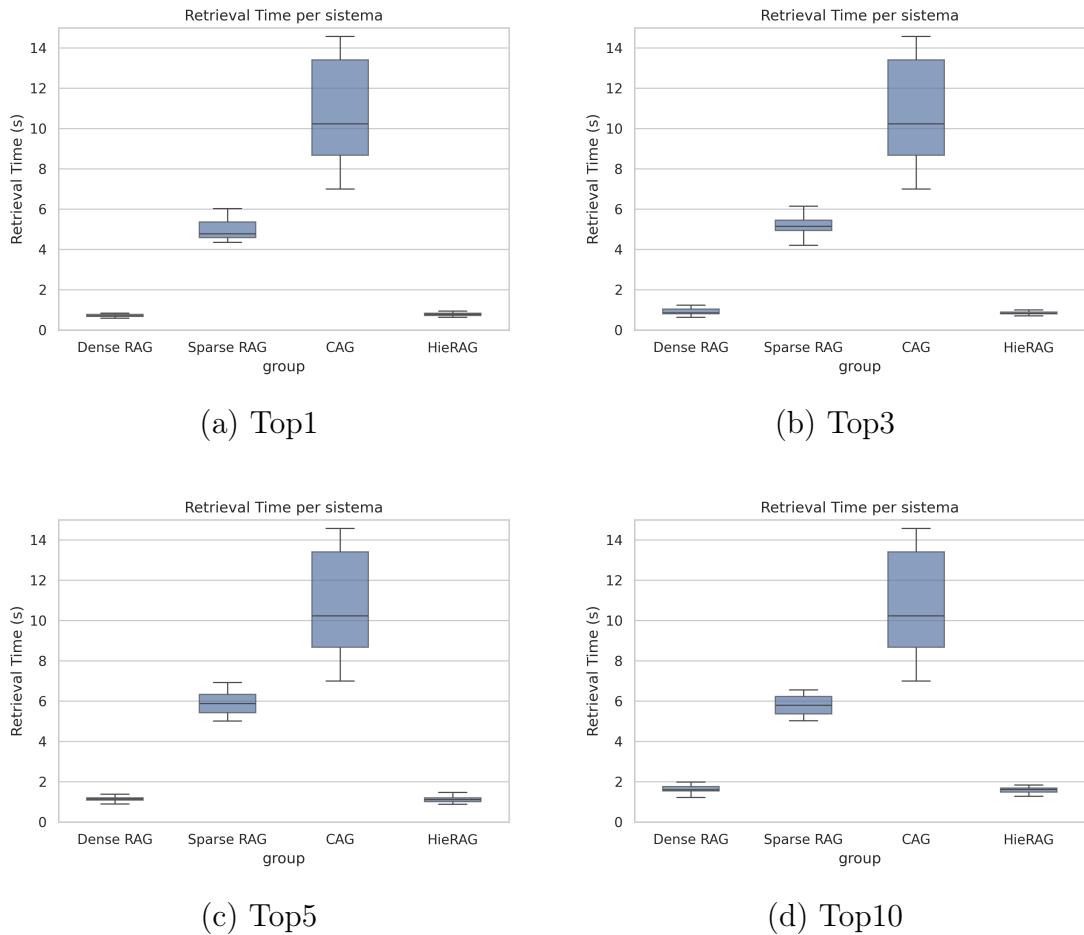


Figure 4.21: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

Il test statistico conferma che anche **HieRAG in configurazione compatta** (L1:25, L2:50, L3:100) mantiene **prestazioni significativamente superiori** rispetto a **CAG** e **Sparse RAG** per tutti i livelli

di top-k. I *p-value* estremamente bassi indicano che il vantaggio osservato nelle metriche di efficacia è robusto e statisticamente attendibile.

Questi risultati sono coerenti con l'efficacia della **gerarchia leggera di cache**, che consente a HieRAG di fornire contesto informativo sufficiente, mantenendo al tempo stesso una struttura più efficiente rispetto a sistemi compressivi o retrieval non indizizzato.

Il confronto con **Dense RAG**, invece, non risulta statisticamente significativo ($p > 0.05$ in tutti i casi), segnalando che i due sistemi offrono performance simili in termini di qualità della risposta. Tuttavia, anche in questa configurazione, HieRAG offre **maggior flessibilità e trasparenza architetturale**, grazie alla gestione esplicita del contesto.

In sintesi, il test conferma che **HieRAG rimane competitivo anche in versione compatta**, migliorando significativamente i sistemi meno strutturati e mantenendo prestazioni paragonabili alle migliori baseline dense.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-13}***$	$10^{-14}***$	$10^{-16}***$	$10^{-18}***$
HieRAG vs Dense RAG	0.07	0.34	0.99	0.38
HieRAG vs Sparse RAG	$10^{-4}***$	$10^{-5}***$	$10^{-6}***$	$10^{-7}***$

Table 4.36: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità ridotta e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

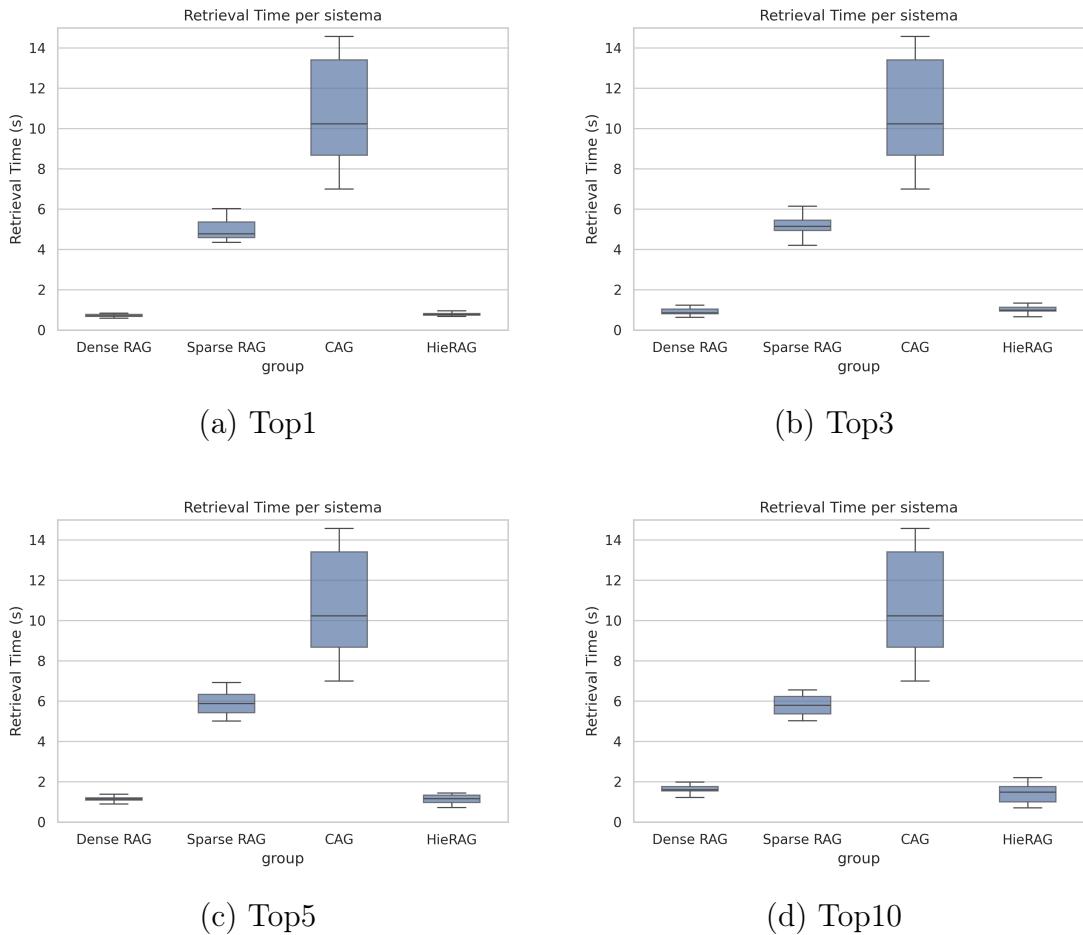


Figure 4.22: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

Il test statistico conferma che anche nella configurazione **monolivello** (solo L1), il sistema **HieRAG supera significativamente CAG e Sparse RAG** su tutti i valori di top-k. I *p-value* risultano

estremamente bassi indicando che anche in assenza della gerarchia, HieRAG mantiene **una netta superiorità qualitativa**, verosimilmente grazie alla selezione efficiente dei chunk e all’uso di contesto rilevante.

Il confronto con **Dense RAG**, invece, non evidenzia differenze significative ($p > 0.05$ in ogni configurazione), confermando che le prestazioni dei due sistemi sono **statisticamente equivalenti** dal punto di vista delle metriche aggregate. Tuttavia, è importante notare che HieRAG, pur nella sua forma più semplice, conserva un **vantaggio architettonico importante**, offrendo un controllo esplicito sul numero e sul contenuto dei chunk selezionati.

In conclusione, anche senza struttura gerarchica, **HieRAG si dimostra efficace e robusto**, superando nettamente i sistemi meno strutturati e restando competitivo rispetto a Dense RAG, pur con una maggiore flessibilità e personalizzabilità.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-16}***$	$10^{-15}***$	$10^{-16}***$	$10^{-16}***$
HieRAG vs Dense RAG	0.94	0.61	0.86	0.81
HieRAG vs Sparse RAG	$10^{-6}***$	$10^{-6}***$	$10^{-6}***$	$10^{-6}***$

Table 4.37: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG monolivello e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

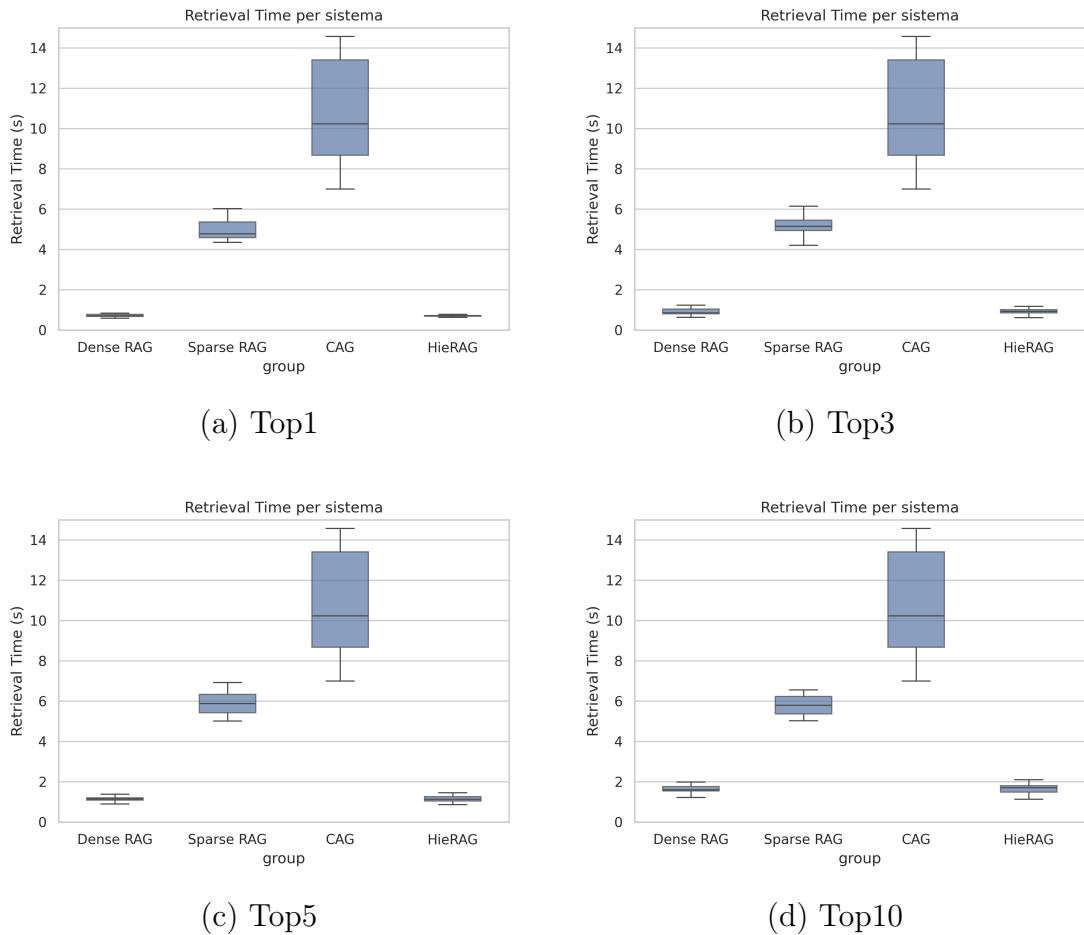


Figure 4.23: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se Hi-eRAG è diverso rispetto a tutti gli altri sistemi.

Confronto baselines: HotPotQA large

Tempi di inferenza Nel contesto del dataset **HotPotQA large**, costituito da **7.405 documenti**, l’analisi dei tempi di inferenza conferma la **scalabilità e l’efficienza del sistema HieRAG**, anche in condizioni di carico computazionale elevato. L’aumento del numero di documenti non ha infatti comportato un degrado significativo delle performance temporali, a dimostrazione della **robustezza architettonale** del sistema.

In particolare, la configurazione **HieRAG con cache multilivello compatta (L1:25, L2:50, L3:100)** e **top-k=1** raggiunge un **RAG Time di soli 0.765 secondi**, un valore sorprendentemente vicino a quelli registrati nei setting “small” e “medium”, nonostante il corpus sia qui circa **tre volte più ampio**. La versione **HieRAG++**, che include una logica di **selezione finale dei chunk rilevanti**, mantiene un tempo analogo (**0.780s**), dimostrando che l’introduzione di un modulo reranker non compromette la velocità complessiva. Questo risultato è reso possibile dalla struttura **gerarchica della cache**, che consente di intercettare i documenti più informativi già nei livelli superiori (L1 e L2), riducendo il numero di accessi profondi e minimizzando la latenza.

Tale comportamento è sintomo di un meccanismo efficiente di **localizzazione del contesto**: la gerarchia consente di gestire il trade-off tra ampiezza del contesto e velocità di accesso, e ciò spiega perché,

anche su un corpus vasto, i tempi di retrieval restano stabili (circa **0.028–0.030s** in tutte le configurazioni). Il modulo di generazione, influenzato principalmente dalla lunghezza e dalla qualità del prompt, è quello che incide maggiormente sul RAG Time. Tuttavia, anche in configurazioni più ampie (es. HieRAG++ con L1:250, L2:500, L3:1000 e top-k=10), il tempo complessivo si mantiene **entro limiti praticabili** (3.022s), a dimostrazione della buona tolleranza del sistema all'aumento dei chunk elaborati.

Nel confronto con le baseline, **Dense RAG** si conferma veloce (**0.734s**), grazie all'uso di una struttura vettoriale ottimizzata. Tuttavia, questa efficienza deriva da una pipeline semplificata, priva di meccanismi strutturati di caching o selezione gerarchica. Di conseguenza, pur offrendo prestazioni temporali competitive, Dense RAG **non consente alcun controllo sul contesto** e non si presta ad applicazioni che richiedano adattabilità o aggiornamenti incrementali della conoscenza.

Sparse RAG, al contrario, evidenzia ancora una volta la sua **inadeguatezza su corpus di grandi dimensioni**: i tempi di retrieval superano i **6 secondi**, portando il RAG Time oltre i **7–8 secondi**, rendendo il sistema inadatto a qualsiasi scenario interattivo o in tempo reale. Questo è dovuto alla natura del retrieval sparse basato su BM25, che non beneficia di strutture di indicizzazione neurale e non dispone di cache.

Il caso più estremo è rappresentato da **CAG**, che, nella configurazione con top-k=32, impiega **oltre 20 secondi solo nella fase di generazione**. Tale lentezza è imputabile al fatto che il contesto fornito al generatore è fortemente compresso e denso, ma anche molto ampio, e richiede quindi uno sforzo computazionale elevato per essere processato. Nonostante le intenzioni di CAG di fornire un contesto sintetico ma informativo, il costo di inferenza rende questo approccio **poco praticabile in ambito operativo**, soprattutto su larga scala.

In conclusione, i risultati su HotPotQA-large confermano che **la struttura di caching gerarchico di HieRAG è determinante per la stabilità dei tempi di inferenza**, anche in condizioni di elevata pressione computazionale. La possibilità di intercettare i chunk più rilevanti nei livelli superiori della cache riduce drasticamente la necessità di esplorare l'intero spazio documentale. Inoltre, il meccanismo di selezione finale nei sistemi HieRAG++ garantisce un prompt più snello e focalizzato, contenendo i tempi di generazione anche quando aumenta la dimensione del contesto.

Il sistema HieRAG dimostra quindi non solo **efficienza computazionale**, ma anche **coerenza strutturale**: la sua architettura permette di scalare in modo ordinato, evitando colli di bottiglia e mantenendo una **latenza stabile e prevedibile**. Queste caratteristiche lo rendono particolarmente adatto ad applicazioni reali di **question answering multihop su larga scala**, dove l'equilibrio tra velocità,

qualità e controllo è cruciale.

system	capacità	topk	retrieval	generation	RAG Time
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.027	1.699	1.726
		3	0.026	2.011	2.037
		5	0.027	2.309	2.336
		10	0.028	2.505	2.533
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.028	2.405	2.433
		3	0.028	2.622	2.650
		5	0.028	2.982	3.010
		10	0.029	2.993	3.022
HieRAG	L1: 25 L2: 50 L3: 100	1	0.030	0.735	0.765
		3	0.029	0.911	0.940
		5	0.028	1.017	1.045
		10	0.029	1.081	1.110
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.029	0.751	0.780
		3	0.028	0.887	0.915
		5	0.029	0.993	1.022
		10	0.029	1.097	1.126
HieRAG	L1: 25	1	0.054	0.727	0.781
		3	0.057	0.887	0.944
		5	0.058	1.070	1.128
		10	0.057	1.523	1.580
HieRAG++	L1: 25	1	0.058	0.719	0.777
		3	0.056	0.889	0.945
		5	0.058	1.124	1.182
		10	0.059	1.660	1.719
Dense RAG	—	1	0.045	0.689	0.734
		3	0.045	0.898	0.943
		5	0.045	1.140	1.185
		10	0.046	1.750	1.796
Sparse RAG	—	1	6.253	0.759	7.217
		3	6.253	0.897	7.150
		5	6.465	1.030	7.495
		10	6.377	1.675	8.052
CAG	—	32	—	20.775	20.775

Table 4.38: Tempi di inferenza in HotPotQA large

Sistema	Capacità	topk	Retrieval	Generation	RAG Time
HieRAG	L1: 25, L2: 50, L3: 100	1	0.030	0.735	0.765
HieRAG++	L1: 25, L2: 50, L3: 100	1	0.029	0.751	0.780
Dense RAG	—	1	0.045	0.689	0.734
Sparse RAG	—	1	6.458	0.759	7.217
CAG	—	32	—	20.775	—

Table 4.39: Migliori configurazioni per ciascun sistema in termini di efficienza su HotPotQA-large. Per ogni sistema viene riportata la configurazione con il tempo di inferenza più basso (RAG Time), indicando separatamente il tempo di retrieval e di generazione. Le configurazioni variano per struttura della cache (se presente) e valore di top-k. I valori in grassetto evidenziano i tempi minimi osservati.

Valutazione delle metriche I risultati delle metriche su **HotPotQA large** confermano la solidità di **HieRAG++**, che, nella configurazione **L1:25, L2:50, L3:100 con top-k=10**, ottiene le **prestazioni migliori in assoluto** su tutte le metriche:

- **BLEU 0.2248**
- **ROUGE-1 0.5129**
- **ROUGE-2 0.3140**
- **ROUGE-L 0.5129**
- **BERTScore 0.7334**

Questo avviene a fronte di un **RAG Time** contenuto (**1.126s**), dimostrando che è possibile **raggiungere una qualità molto alta delle risposte senza sacrificare l'efficienza**.

Il merito di tali performance sta nell’equilibrio tra **ampiezza del contesto (top-k=10)** e **capacità selettiva del sistema**: la cache multilivello consente un recupero veloce, mentre la logica di HieRAG++ filtra i chunk più informativi, riducendo la lunghezza e la ridondanza del prompt generativo. Rispetto a **HieRAG base** (stessa cache e top-k), che comunque ottiene buoni risultati (**BLEU 0.1899, ROUGE-1 0.3541**), HieRAG++ guadagna significativamente in precisione e coerenza, con **solo 0.06s in più** di tempo di inferenza.

Il confronto con le baseline rende ancora più evidente il vantaggio architettonico del sistema proposto. **Dense RAG**, pur avendo un RAG Time più basso (0.796s), si ferma a **BLEU 0.0928** e **ROUGE-1 0.3903**, mostrando un chiaro limite nella qualità delle risposte rispetto a HieRAG++. **Sparse RAG** soffre sia in efficienza che in accuratezza, mentre **CAG**, pur fornendo un contesto sintetico molto ampio (top-k=32), ottiene punteggi bassi (BLEU 0.0514, BERTScore 0.6083) a fronte di un **tempo di generazione insostenibile (20.775s)**.

In sintesi, i risultati su HotPotQA-large mostrano che **HieRAG++ è in grado di mantenere alta la qualità delle risposte anche su corpus molto estesi**, senza rinunciare all’efficienza. Il **trade-off tra tempo e accuratezza risulta ottimamente bilanciato**, rendendo il sistema particolarmente adatto ad applicazioni reali che richiedono sia prestazioni elevate sia risposta rapida.

system	capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.0478	0.1788	0.0805	0.1780	0.6667
		3	0.0500	0.1991	0.0905	0.1991	0.6534
		5	0.0587	0.2127	0.0958	0.2121	0.6644
		10	0.0672	0.2238	0.1003	0.2231	0.6654
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.0424	0.1671	0.0665	0.1671	0.6420
		3	0.0549	0.1894	0.0843	0.1887	0.6473
		5	0.0562	0.2060	0.0862	0.2051	0.6613
		10	0.0566	0.2190	0.1004	0.2183	0.6657
HieRAG	L1: 25 L2: 50 L3: 100	1	0.1463	0.3109	0.1747	0.3109	0.6546
		3	0.1346	0.3048	0.1691	0.3048	0.6714
		5	0.1747	0.3392	0.2019	0.3392	0.6725
		10	0.1899	0.3541	0.1991	0.3523	0.6730
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.1920	0.3721	0.2295	0.3721	0.7213
		3	0.2558	0.4557	0.2828	0.4557	0.7274
		5	0.2808	0.4767	0.2683	0.4761	0.7195
		10	0.2665	0.4726	0.2941	0.4717	0.7334
HieRAG	L1: 25	1	0.0991	0.3294	0.1820	0.3290	0.6709
		3	0.1062	0.3820	0.2225	0.3812	0.7161
		5	0.0996	0.3921	0.2288	0.3914	0.7313
		10	0.1056	0.3982	0.2222	0.3982	0.7255
HieRAG++	L1: 25	1	0.1085	0.3578	0.2006	0.3568	0.6702
		3	0.1577	0.4410	0.2664	0.4410	0.7053
		5	0.1711	0.4567	0.2861	0.4563	0.7103
		10	0.2248	0.5129	0.3140	0.5129	0.6936
Dense RAG	—	1	0.0726	0.3415	0.1937	0.3411	0.6291
		3	0.0942	0.3565	0.2010	0.3552	0.6447
		5	0.0877	0.3824	0.2157	0.3820	0.6496
		10	0.0928	0.3903	0.2122	0.3895	0.6534
Sparse RAG	—	1	0.0593	0.2896	0.1674	0.2896	0.6080
		3	0.0727	0.3309	0.1901	0.3288	0.6257
		5	0.0748	0.3412	0.1918	0.3412	0.6287
		10	0.0722	0.3521	0.2038	0.3513	0.6315
CAG	—	32	0.0514	0.2179	0.0963	0.2177	0.6083

Table 4.40: Risultati sperimentali in HotPotQA large

Sistema	Capacità	topk	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
HieRAG	L1: 25, L2: 50, L3: 100	10	0.1899	0.3541	0.1991	0.3523	0.6730
HieRAG++	L1: 25, L2: 50, L3: 100	10	0.2248	0.5129	0.3140	0.5129	0.7334
CAG	—	32	0.0514	0.2179	0.0963	0.2177	0.6083
Dense RAG	—	10	0.0928	0.3903	0.2122	0.3895	0.6534
Sparse RAG	—	10	0.0722	0.3521	0.2038	0.3513	0.6315

Table 4.41: Migliori configurazioni per ciascun sistema in termini di efficacia su HotPotQA-large, misurate tramite metriche automatiche di similarità (BLEU, ROUGE-1/2/L, BERTScore). Per ogni sistema è riportata la configurazione che ha ottenuto le prestazioni più elevate, con particolare riferimento al valore di top-k e alla struttura di caching (quando presente). I valori in grassetto indicano le migliori performance assolute tra tutti i modelli.

Test statistici Il test statistico condotto con il **Dunn test e correzione Holm** mostra che **HieRAG con capacità elevata** si distingue **in modo altamente significativo** rispetto a **CAG** e **Sparse RAG**, su tutti i livelli di top-k. I *p-value* inferiori a 10^{-13} e 10^{-4} , rispettivamente, confermano che le superiori prestazioni osservate nelle metriche non sono dovute al caso, ma riflettono **un vantaggio sistematico e strutturale** del sistema proposto. Questi risultati sono coerenti con le analisi qualitative, che evidenziano come la combinazione tra **caching gerarchico e selezione controllata del contesto** consenta a HieRAG di produrre risposte più precise e coerenti, anche con corpus ampi.

Il confronto con **Dense RAG**, invece, restituisce *p-value* superiori alla soglia di significatività per quasi tutte le configurazioni, tranne che per top-1 ($p = 0.04$). Questo suggerisce che **le differenze sono statisticamente marginali**, indicando **prestazioni comparabili** nelle metriche aggregate. Tuttavia, va ricordato che, a parità di qualità, HieRAG offre **maggiori possibilità di interpretazione e controllo**, mentre Dense RAG è una black-box più efficiente ma meno flessibile.

In sintesi, il test statistico su HotPotQA-large conferma che **HieRAG è significativamente superiore ai sistemi meno strutturati**, e che si colloca sullo stesso livello di Dense RAG in termini di efficacia, pur conservando **vantaggi architetturali e progettuali** che ne giustificano l'adozione in contesti avanzati.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-13}***$	$10^{-14}***$	$10^{-16}***$	$10^{-17}***$
HieRAG vs Dense RAG	0.04	0.19	0.75	0.50
HieRAG vs Sparse RAG	$10^{-4}***$	$10^{-5}***$	$10^{-7}***$	$10^{-7}***$

Table 4.42: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità elevata e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

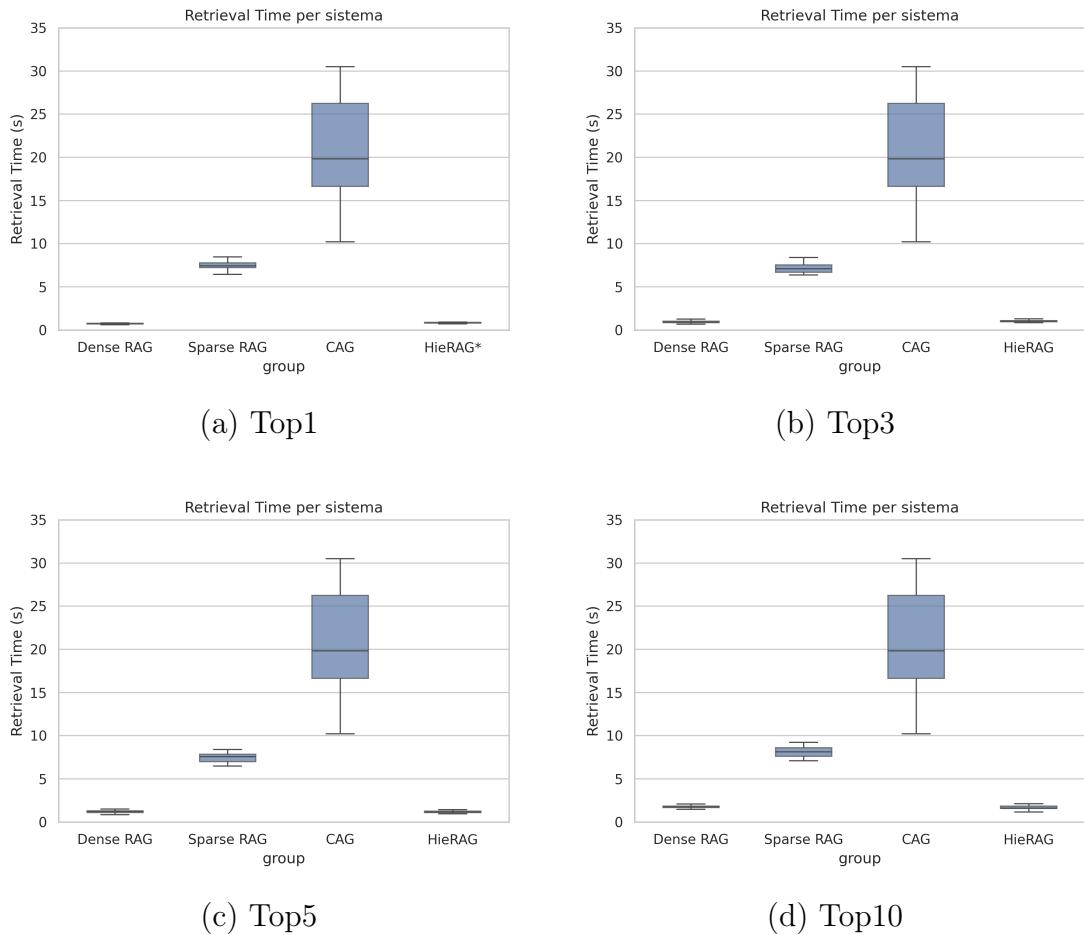


Figure 4.24: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea media indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

I risultati del test statistico evidenziano che **HieRAG con capacità ridotta** (L1:25, L2:50, L3:100) mantiene **un chiaro vantaggio significativo** rispetto a **CAG** e **Sparse RAG** su tutte le configu-

razioni di top-k. I *p-value* estremamente bassi (10^{-13} contro CAG e 10^{-4} contro Sparse RAG) confermano che le differenze osservate nelle metriche sono **robuste e statisticamente fondate**. Questo risultato riflette l’efficacia della **cache gerarchica compatta**, che, pur limitata nella capienza, consente un recupero informativo mirato, combinando rapidità d’accesso e qualità del contesto fornito al generatore.

Il confronto con **Dense RAG** restituisce un quadro più sfumato. Nessuno dei *p-value* risulta statisticamente significativo ($p > 0.05$), sebbene il valore al top-10 ($p = 0.05$) sia al limite della soglia. Questo suggerisce che, anche in configurazione leggera, **HieRAG offre prestazioni simili a Dense RAG dal punto di vista quantitativo**, pur differenziandosi profondamente sul piano progettuale: HieRAG permette infatti una gestione modulare del retrieval, offre trasparenza e supporta l’integrazione di logiche di aggiornamento dinamico, aspetti assenti nel modello dense.

In sintesi, il test statistico su HotPotQA-large conferma che **HieRAG rimane competitivo anche con una cache ridotta**, superando ampiamente le baseline meno strutturate, e risultando **paragonabile a Dense RAG** in termini di qualità, ma con **maggior flessibilità architetturale**, fondamentale per l’adattamento a scenari reali.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-13}***$	$10^{-16}***$	$10^{-18}***$	$10^{-20}***$
HieRAG vs Dense RAG	0.08	0.96	0.17	0.05
HieRAG vs Sparse RAG	$10^{-4}***$	$10^{-6}***$	$10^{-8}***$	$10^{-9}***$

Table 4.43: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità ridotta e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

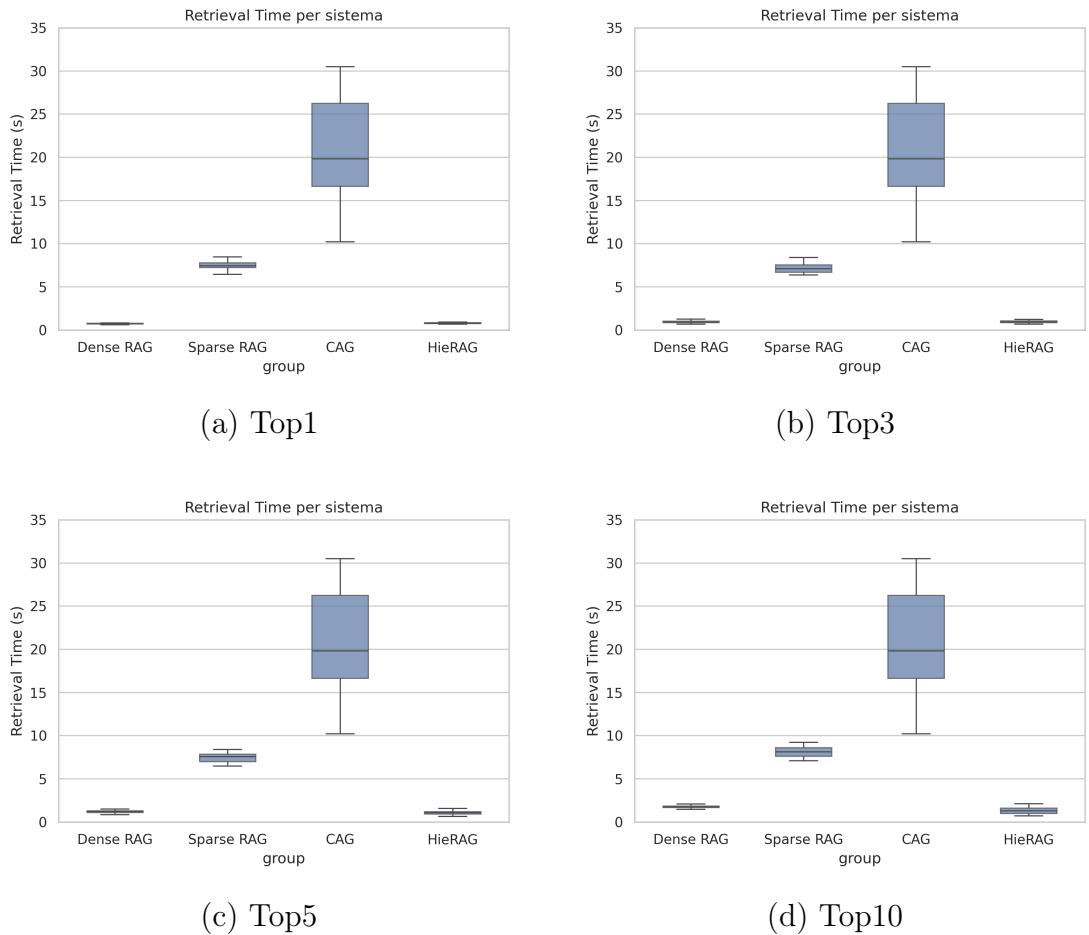


Figure 4.25: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e HieRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta HieRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

Il confronto statistico mostra che anche nella **configurazione monolivello (L1:25)**, **HieRAG mantiene un vantaggio altamente significativo** rispetto a **CAG** e **Sparse RAG**, con $p\text{-value}$ inferiori

a 10^{-15} e 10^{-5} rispettivamente, per tutte le configurazioni di top-k. Questo conferma che **anche una struttura di caching minimale è sufficiente** per superare in modo consistente i sistemi meno strutturati o compressivi, grazie alla capacità di HieRAG di **selezionare e gestire chunk informativi in modo controllato**.

Il confronto con **Dense RAG**, invece, evidenzia un quadro di sostanziale equivalenza: i *p-value* risultano ampiamente **non significativi** ($p > 0.2$ in tutti i casi), suggerendo che **le prestazioni delle due architetture sono statisticamente comparabili** in termini di efficacia complessiva. Tuttavia, è importante notare che, pur non avendo una gerarchia esplicita, anche in versione monolivello HieRAG conserva alcuni **vantaggi architetturali**, come la trasparenza del retrieval e la possibilità di ottimizzare dinamicamente la cache, aspetti assenti in Dense RAG.

In sintesi, questo test conferma che **HieRAG è efficace anche nella sua forma più semplice**, superando significativamente i sistemi meno efficienti e mantenendo risultati competitivi rispetto a Dense RAG, ma offrendo **maggior controllo, estensibilità e potenziale adattabilità al deployment su larga scala**.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	$10^{-15}***$	$10^{-15}***$	$10^{-17}***$	$10^{-18}***$
HieRAG vs Dense RAG	0.44	0.73	0.50	0.23
HieRAG vs Sparse RAG	$10^{-5}***$	$10^{-6}***$	$10^{-7}***$	$10^{-8}***$

Table 4.44: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG monolivello e gli altri sistemi al variare di *top-k*. I valori $p < 0.05$ indicano significatività statistica (indicato da * per $p < 0.05$, ** per $p < 0.01$ e *** per $p < 0.001$)

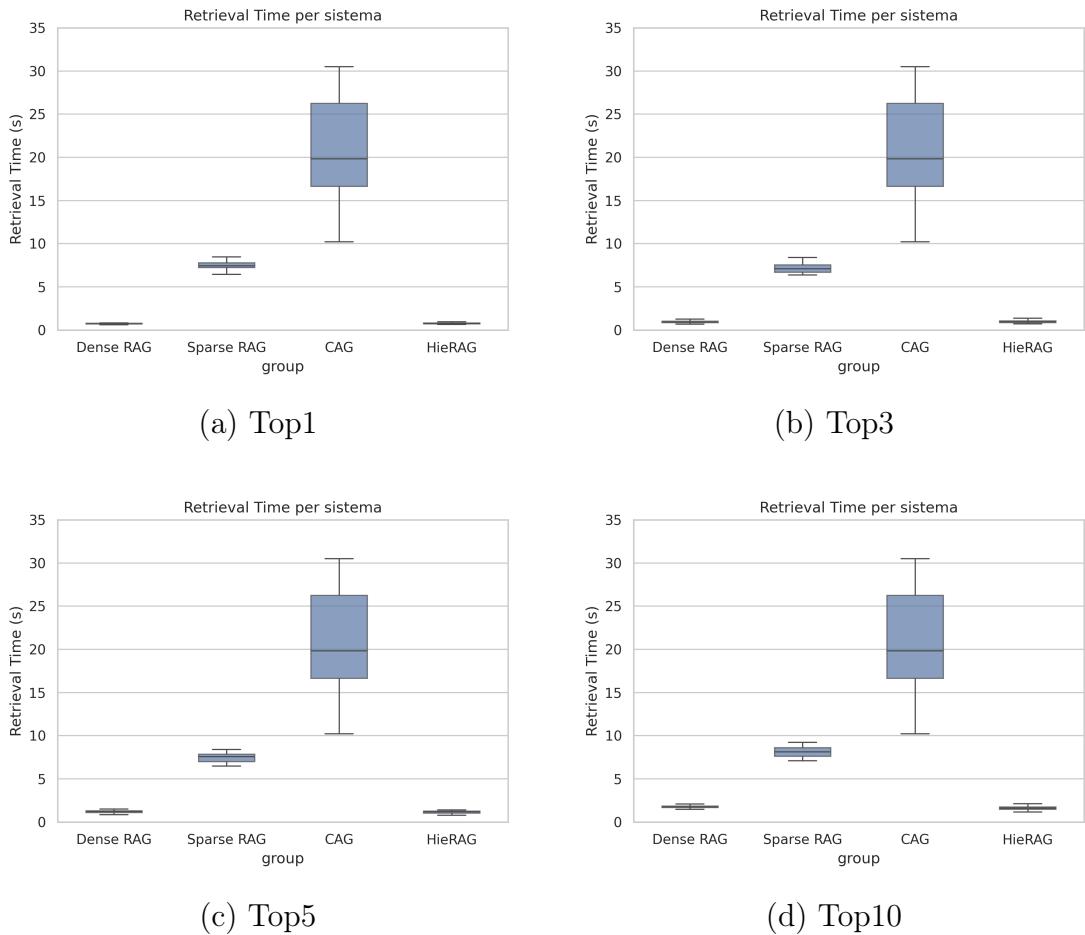


Figure 4.26: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se Hi-eRAG è diverso rispetto a tutti gli altri sistemi.

4.2.3 Risultati PubMedQA

Impatto del sistema HieRAG

Per valutare l'impatto della cache gerarchica sul tempo di risposta del sistema, è stata condotta un'analisi sperimentale sul dataset **PubMedQA**, composto da circa oltre **200.000 documenti scientifici**. I risultati mettono in evidenza differenze significative tra tre configurazioni del sistema **HieRAG**: con cache gerarchica ad alta capacità, con capacità ridotta e in modalità monolivello.

Nella configurazione **HieRAG con cache gerarchica ad alta capacità**, si osserva un calo netto e progressivo del tempo di risposta nelle prime fasi, con un assestamento stabile attorno ai 3 secondi dopo circa 400 domande. Questo comportamento evidenzia un'efficace fase di accumulo nei livelli L1 e L2, in cui il sistema impara rapidamente a conservare i chunk più rilevanti, riducendo la necessità di accedere al database FAISS.

Nel caso di **HieRAG con capacità ridotta**, il sistema mostra un miglioramento iniziale simile, ma con maggiore instabilità nella fase successiva. Le oscillazioni del `rag_time` risultano più ampie e meno prevedibili, suggerendo una frequente invalidazione o sostituzione di contenuti in cache. Ciò indica che, pur mantenendo la struttura gerarchica, la **capacità limitata** compromette l'efficienza complessiva del sistema.

Infine, nella variante **HieRAG monolivello**, il tempo di risposta

inizia più elevato ma scende rapidamente a valori stabili attorno ai 3.4 secondi. L'assenza di una gerarchia comporta una gestione meno selettiva e dinamica della cache, che pur consentendo un certo guadagno rispetto all'accesso diretto a FAISS, si dimostra meno efficace rispetto alla soluzione multistrato.

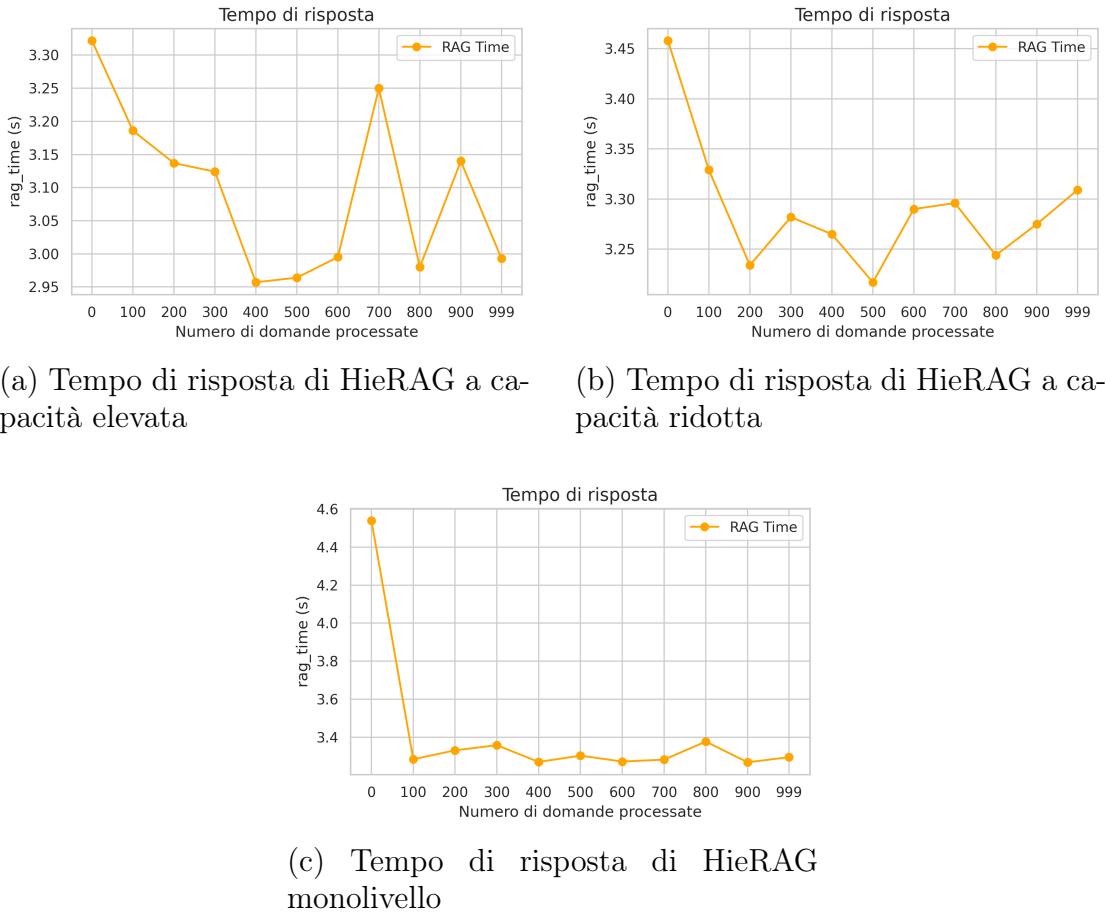


Figure 4.27: Andamento del tempo di risposta del sistema HieRAG su PubMedQA per tre configurazioni di cache: gerarchica ad alta capacità, gerarchica a capacità ridotta e monolivello. La versione ad alta capacità mostra un rapido assestamento sotto i 3.1 secondi grazie a un accumulo efficiente nei livelli superiori della cache. La variante a capacità ridotta mantiene una struttura multistrato ma con maggiori oscillazioni dovute a cache miss più frequenti. Infine, la configurazione monolivello evidenzia un miglioramento iniziale seguito da una stabilizzazione a valori superiori, segno dell’assenza di ottimizzazione progressiva tipica della struttura gerarchica

Oltre all’effetto sul tempo totale di risposta, l’utilizzo della cache gerarchica incide in modo decisivo anche sulla **latenza della fase di recupero** dei documenti.

Nel caso della configurazione **gerarchica ad alta capacità**, si

osserva che i tempi medi di accesso ai livelli L1, L2 e L3 si attestano attorno ai **25–30 millisecondi**, mentre l’accesso al database FAISS richiede in media **oltre 100 millisecondi**. Questo significa che, quando il sistema riesce a soddisfare una richiesta direttamente da uno dei livelli di cache, il tempo necessario per completare la fase di retrieval risulta **circa quattro volte inferiore** rispetto all’accesso a FAISS. Tale risultato è dovuto al fatto che i livelli di cache, in particolare L1 e L2, operano su strutture dati snelle residenti in memoria e non richiedono né computazioni pesanti sugli embeddings né accessi a strutture indicizzate ad alta dimensionalità.

La configurazione **a capacità ridotta** limita la quantità di contenuti mantenuti attivi nei livelli superiori. Anche in questo caso i tempi di retrieval restano molto bassi nei livelli L1 e L2, ma il livello L3 mostra un impatto marginale, probabilmente a causa del numero ridotto di hit. Nonostante la riduzione della capacità, il sistema riesce comunque ad **evitare l’accesso a FAISS in una buona percentuale di casi**, mantenendo alta l’efficienza della fase di retrieval.

Nel sistema **monolivello**, invece, solo il livello L1 è attivo, mentre L2 e L3 non vengono utilizzati. Il confronto con FAISS è particolarmente evidente: anche con un singolo livello di cache, il sistema riesce a ridurre in modo sostanziale la latenza media di retrieval.

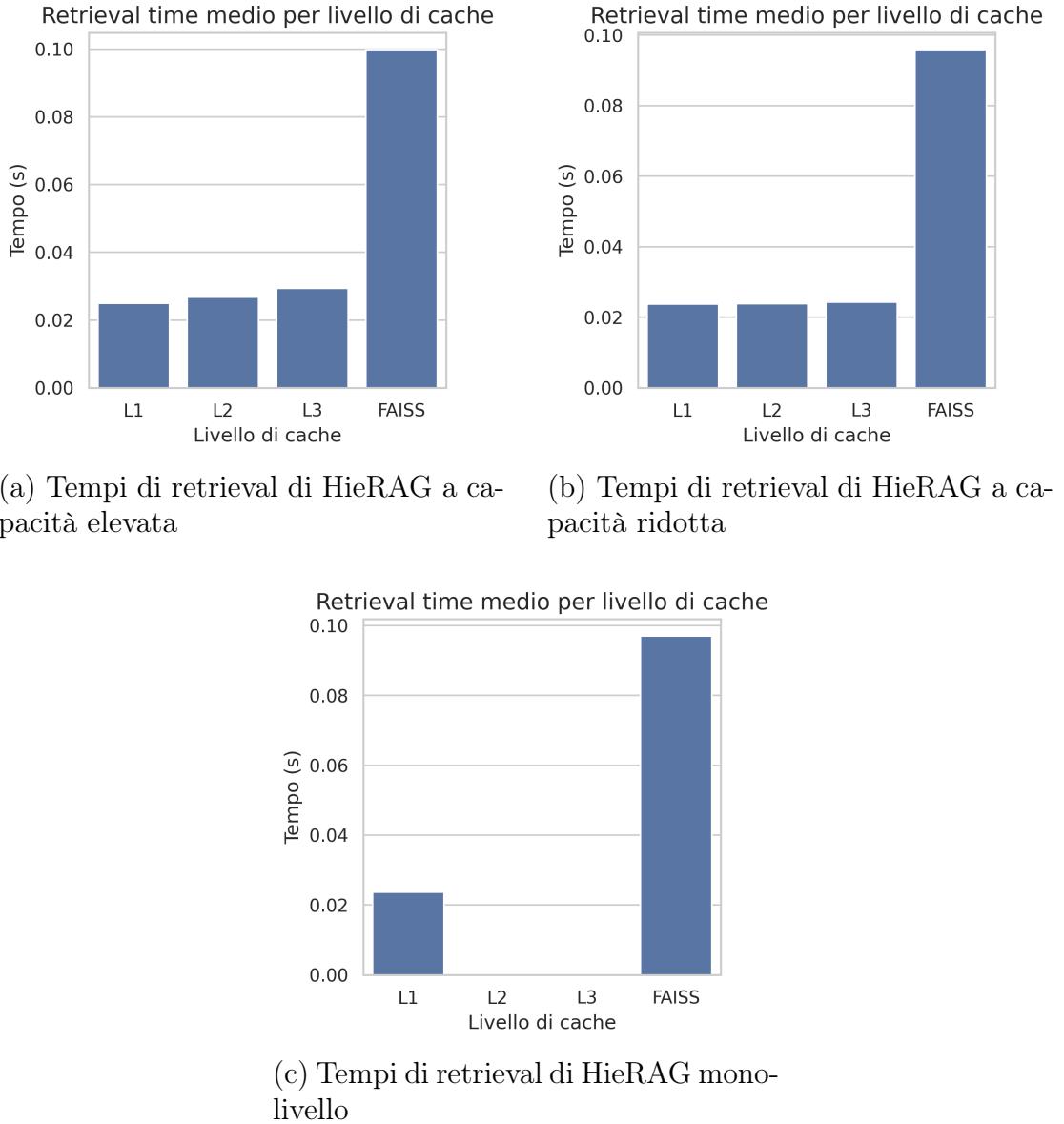


Figure 4.28: Tempo medio di retrieval per ciascun livello di cache (L1, L2, L3) e FAISS nelle tre configurazioni del sistema HieRAG: gerarchica ad alta capacità, gerarchica a capacità ridotta e monolivello. I livelli di cache risultano significativamente più veloci di FAISS, con prestazioni migliori nella configurazione ad alta capacità e una progressiva riduzione dell'efficienza nel passaggio a strutture più semplici

La rilevanza dell'approccio basato su caching emerge in modo ancora più netto se si osserva la **riduzione percentuale del retrieval time** rispetto al tempo medio di accesso al database vettoriale FAISS.

In tutte le configurazioni del sistema HieRAG – ad alta capacità, ridotta e monolivello – si osservano **guadagni significativi**, con riduzioni superiori al **70%** in media nei livelli attivi. In particolare, la configurazione gerarchica completa consente una riduzione ampia e distribuita su tutti i livelli, mentre quella monolivello concentra l'intero vantaggio sul solo livello L1. Questi risultati confermano che **la presenza della cache, anche in forma semplificata, consente di evitare costosi accessi al database vettoriale**, migliorando sensibilmente la reattività del sistema. L'effetto è tanto più marcato quanto più la cache è strutturata e capiente, a sostegno dell'approccio gerarchico ispirato all'architettura delle memorie nei sistemi computazionali.

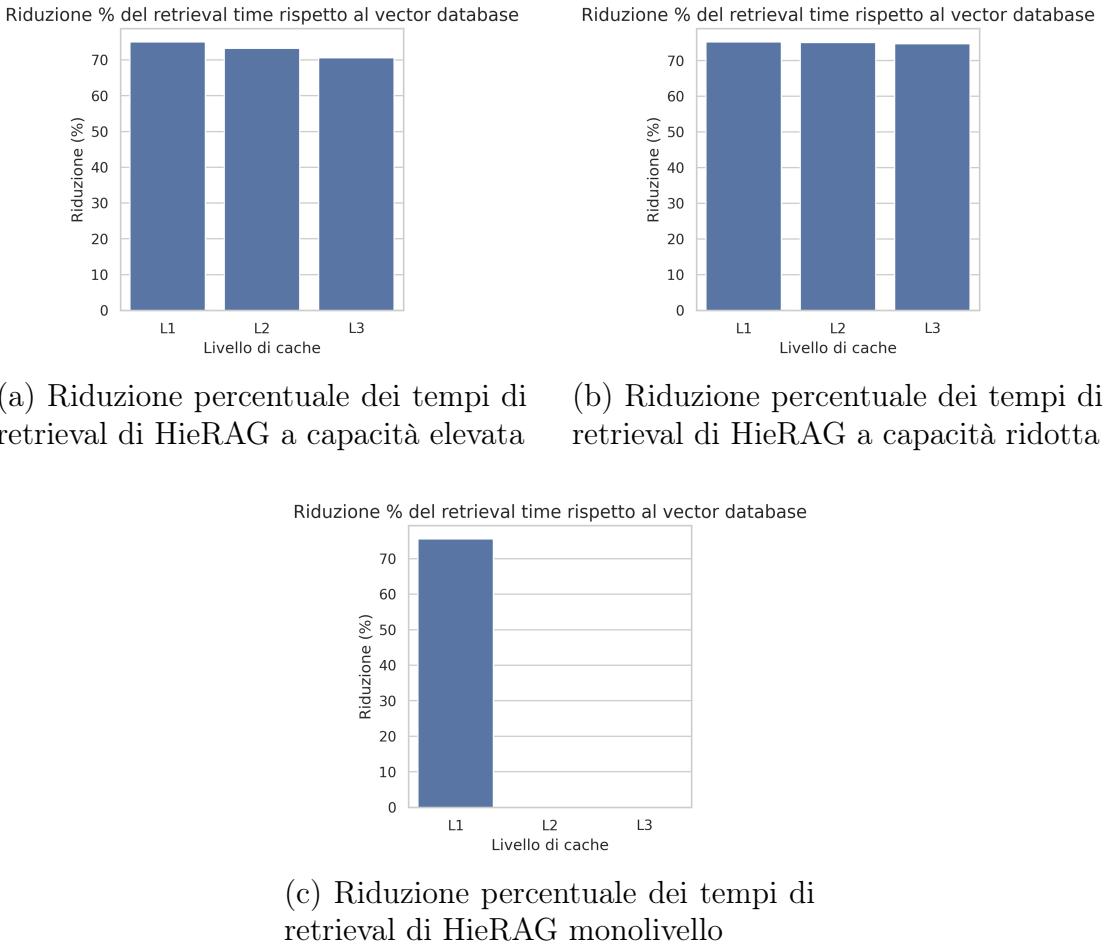


Figure 4.29: Riduzione percentuale del tempo di retrieval nei livelli di cache (L1, L2, L3) rispetto all’accesso al database vettoriale FAISS, nelle tre configurazioni di HieRAG (cache ad alta capacità, capacità ridotta, e monolivello). La struttura gerarchica consente una riduzione sistematica superiore al 70%, evidenziando l’efficienza e la rilevanza della cache nei sistemi RAG

L’analisi congiunta dell’evoluzione della dimensione della cache e della distribuzione degli hit nei vari livelli offre un quadro completo sul comportamento interno del sistema HieRAG. Nella configurazione ad alta capacità, si osserva un progressivo e completo riempimento dei livelli L1, L2 e L3, con una netta prevalenza di hit in L2 (66.2%), seguita da FAISS, L3 e solo in minima parte da L1. Questo indica che,

nonostante L1 sia il più veloce, è L2 a fornire il miglior compromesso tra capacità e velocità nel lungo periodo.

Nella variante a capacità ridotta, tutti e tre i livelli vengono saturati rapidamente, ma la distribuzione degli hit cambia radicalmente: quasi la metà degli accessi è soddisfatta da L1 (47.7%), con un contributo più bilanciato degli altri livelli. Ciò riflette un comportamento più volatile dovuto alla ridotta capacità, che aumenta la pressione sugli strati superiori della cache.

Nel sistema monolivello, come atteso, solo L1 è attivo, e si riempie completamente già nelle fasi iniziali. Tuttavia, gli hit sono distribuiti quasi equamente tra L1 e FAISS (49.6% vs 50.4%), segno che l'assenza di livelli intermedi penalizza la copertura, costringendo il sistema a ricorrere spesso al motore vettoriale.

Nel complesso, questi grafici confermano che la struttura gerarchica non solo consente una gestione efficiente dello spazio, ma permette anche una distribuzione intelligente degli accessi, massimizzando l'efficacia della cache e riducendo la dipendenza da FAISS.

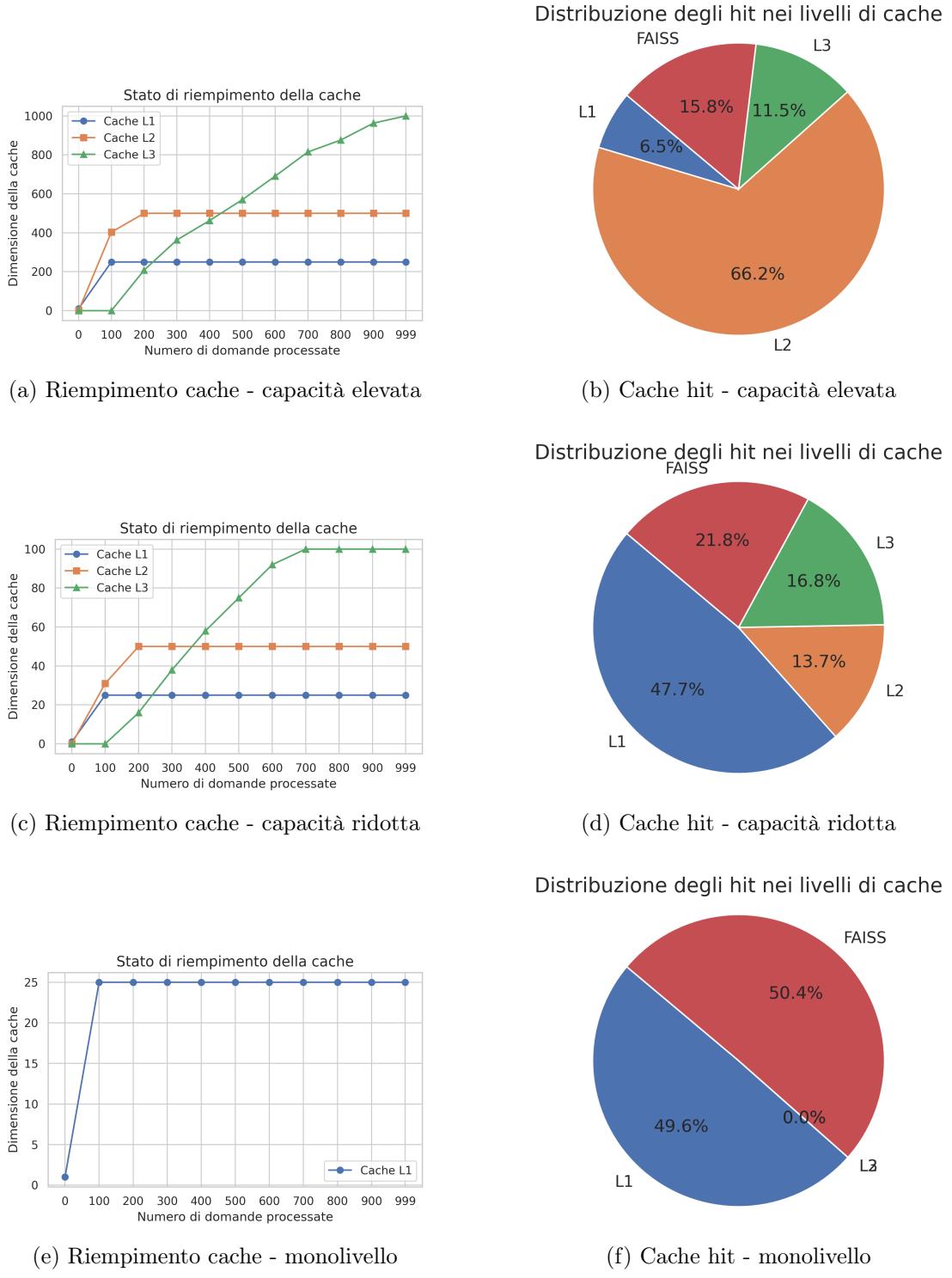


Figure 4.30: Evoluzione della dimensione dei livelli di cache (L1, L2, L3) e distribuzione percentuale degli hit nei tre sistemi HiRAG: cache ad alta capacità, capacità ridotta e monolivello. La configurazione gerarchica completa mostra un riempimento progressivo e una prevalenza di hit in L2, mentre la versione ridotta si affida maggiormente a L1. Il sistema monolivello, privo di gerarchia, satura rapidamente L1 ma presenta una forte dipendenza da FAISS.

Confronto baselines

Tempi di inferenza L'esperimento condotto su **PubMedQA**, il più esteso tra i dataset utilizzati (con **212.000 documenti**), mette alla prova la **scalabilità e la robustezza architetturale** dei sistemi di question answering basati su retrieval. In questo contesto, **HieRAG** e la sua variante **HieRAG++** confermano nuovamente la loro **efficienza computazionale**, anche in condizioni estreme di carico informativo.

La configurazione **HieRAG con cache multilivello capiente (L1:250, L2:500, L3:1000)** e **top-k=1** ottiene il miglior **RAG Time assoluto: 2.960s**, a fronte di un tempo di retrieval molto contenuto (0.057s) e una fase di generazione stabile. È interessante notare che, contrariamente a quanto osservato nei dataset precedenti, l'aumento del numero di documenti nel corpus **non incide in modo drastico sui tempi di retrieval** per HieRAG, proprio grazie all'efficienza della **struttura di caching preindividuata**, che consente un accesso localizzato e rapido ai chunk più rilevanti.

Anche **HieRAG++**, nella configurazione compatta (L1:25, L2:50, L3:100), dimostra buone prestazioni (3.091s con top-k=1), pur avendo un tempo di retrieval leggermente superiore (0.120s) dovuto al reranking interno. Tuttavia, il **modulo selettivo di HieRAG++** consente una maggiore qualità delle risposte (come si vedrà nell'analisi delle metriche), mantenendo al contempo un tempo di inferenza **ampia-**

mente entro i limiti operativi accettabili. In generale, tutte le configurazioni di HieRAG/HieRAG++ rimangono tra **i 3 e i 3.7 secondi**, anche al crescere di top-k, a conferma della **stabilità del sistema**.

Il confronto con le baseline evidenzia in modo netto **i limiti di scalabilità dei sistemi non strutturati**. In particolare, **Dense RAG** mostra un **retrieval time di oltre 40 secondi**, dovuto al fatto che l'indice vettoriale viene interrogato su un corpus molto ampio, senza alcun meccanismo di caching o selezione locale. Il **RAG Time complessivo raggiunge i 44.6s**, rendendo il sistema del tutto **inadatto ad applicazioni interattive o real-time**. Anche **Sparse RAG**, sebbene più veloce di Dense RAG, presenta comunque tempi insostenibili: oltre **26s per il solo retrieval** e un tempo totale superiore a **29s**, aggravato dal fatto che il retrieval sparse (es. BM25) su larga scala non è ottimizzato per query complesse.

Il sistema **CAG**, pur senza retrieval esplicito (utilizzando contesti sintetici), impiega **4.732s** solo per la generazione, posizionandosi comunque al di sopra delle configurazioni di HieRAG e HieRAG++, che operano su chunk informativi reali e gestiti in modo selettivo.

Dal confronto emerge chiaramente che **la struttura di caching gerarchico di HieRAG è l'unica in grado di sostenere la scala di PubMedQA** con tempi di inferenza ridotti. Le configurazioni capienti permettono l'accesso a una base di conoscenza più ampia, men-

tre quelle compatte risultano competitive grazie all’efficace selezione iniziale. Il **tempo di retrieval resta basso in tutti i casi**, a differenza delle baseline, e la fase di generazione cresce in modo contenuto con l’aumento di top-k, segno che il sistema è in grado di **regolare il carico computazionale in modo adattivo**.

In sintesi, i risultati su PubMedQA dimostrano che **HieRAG è un sistema scalabile, efficiente e adatto a corpus reali di grandi dimensioni**, in grado di offrire inferenza rapida e controllata dove gli altri approcci falliscono in termini di tempo di risposta. Questo rende la sua architettura particolarmente adatta a contesti critici, come quello biomedico, in cui **precisione e velocità devono coesistere**.

system	capacità	topk	retrieval	generation	RAG	Time
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.057	2.903	2.960	
		3	0.053	2.960	3.013	
		5	0.049	2.999	3.048	
		10	0.038	3.061	3.099	
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.125	2.980	3.105	
		3	0.125	3.003	3.128	
		5	0.127	3.048	3.175	
		10	0.103	3.127	3.230	
HieRAG	L1: 25 L2: 50 L3: 100	1	0.040	3.226	3.266	
		3	0.044	3.319	3.363	
		5	0.046	3.419	3.465	
		10	0.046	3.628	3.674	
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.120	2.971	3.091	
		3	0.120	3.010	3.130	
		5	0.108	3.059	3.167	
		10	0.077	3.436	3.513	
HieRAG	L1: 25	1	0.061	3.258	3.319	
		3	0.091	3.548	3.639	
		5	0.097	3.796	3.893	
		10	0.099	4.389	4.488	
HieRAG++	L1: 25	1	0.201	3.848	4.049	
		3	0.205	4.324	4.529	
		5	0.209	5.183	5.392	
		10	0.204	7.289	7.493	
Dense RAG	—	1	40.189	4.069	44.593	
		3	42.198	4.379	46.577	
		5	40.524	4.929	45.118	
		10	41.313	7.605	48.918	
Sparse RAG	—	1	26.793	2.356	29.331	
		3	28.132	4.733	32.865	
		5	26.975	5.430	32.223	
		10	27.542	6.797	34.339	
CAG	—	5	—	4.732	4.732	

Table 4.45: Tempi di inferenza in PubMedQA

Sistema	Capacità	topk	Retrieval	Generation	RAG Time
HieRAG	L1: 250, L2: 500, L3: 1000	1	0.057	2.903	2.960
HieRAG++	L1: 25, L2: 50, L3: 100	1	0.120	2.971	3.091
Dense RAG	—	1	40.524	4.069	44.593
Sparse RAG	—	1	26.975	2.356	29.331
CAG	—	5	—	4.732	4.732

Table 4.46: Migliori configurazioni per ciascun sistema in termini di efficienza su PubMedQA. La tabella riporta, per ogni sistema, la configurazione con il tempo di inferenza più basso (RAG Time), suddiviso nelle componenti di retrieval e generazione. Le configurazioni differiscono per valore di top-k e per struttura della cache (se presente). I valori in grassetto indicano i tempi minimi complessivi. Il confronto evidenzia la scalabilità dei sistemi su un corpus di oltre 200.000 documenti.

Valutazione delle metriche I risultati delle metriche su **PubMedQA** confermano la **superiorità qualitativa del sistema HieRAG++**, in particolare nella configurazione **compatta** (L1:25, L2:50, L3:100) e con **top-k=5**, che ottiene i migliori valori assoluti: **BERTScore 0.7333** e **COMET 0.7145**. Questi punteggi, misurati su un corpus altamente tecnico, evidenziano una **coerenza semantica** delle risposte generate che nessun altro sistema riesce a eguagliare. Il merito va attribuito alla **combinazione tra gerarchia di cache e meccanismo di selezione interna**, che permette di costruire un prompt mirato e informativo, riducendo rumore e ridondanza.

Anche la versione base di **HieRAG**, con cache capiente (L1:250, L2:500, L3:1000) e top-k=5, si comporta bene (BERTScore 0.6516, COMET 0.6688), a dimostrazione che, in assenza del reranker, **la gerarchia da sola è sufficiente a fornire un contesto informativo**

efficace, soprattutto quando si dispone di ampia capacità. Tuttavia, il vantaggio della variante `++` diventa evidente nel confronto diretto: a parità di top-k, essa riesce a **estrarre in modo più selettivo i chunk realmente rilevanti**, garantendo una qualità semantica più alta, con un incremento contenuto del tempo di inferenza.

Le baseline mostrano invece limiti evidenti. **Dense RAG** e **Sparse RAG** ottengono risultati modesti, con valori di BERTScore attorno a **0.60**, e COMET leggermente inferiori a quelli di HieRAG, pur utilizzando più documenti. La mancanza di un controllo esplicito sul contesto recuperato penalizza la precisione del contenuto informativo, soprattutto in domini specialistici come quello biomedico. Ancora più evidente è il divario con **CAG**, che, pur con top-k=5, raggiunge valori molto bassi (**BERTScore 0.4248, COMET 0.3387**), segno che la compressione del contesto non è sufficiente per conservare la densità semantica necessaria a generare risposte scientificamente plausibili.

In sintesi, su un corpus impegnativo come PubMedQA, **HieRAG`++` dimostra di saper bilanciare qualità e efficienza**, offrendo risposte accurate in tempi contenuti. La sua architettura flessibile si conferma ideale per domini ad alta specializzazione, in cui la **qualità informativa è cruciale tanto quanto la scalabilità**.

System	Capacità	Top-k	BERTScore	COMET
HieRAG	L1: 250 L2: 500 L3: 1000	1	0.5932	0.6244
		3	0.6211	0.6481
		5	0.6516	0.6688
		10	0.6086	0.6389
HieRAG++	L1: 250 L2: 500 L3: 1000	1	0.6651	0.6741
		3	0.7027	0.7062
		5	0.7112	0.7075
		10	0.6810	0.6899
HieRAG	L1: 25 L2: 50 L3: 100	1	0.5944	0.6161
		3	0.6461	0.6594
		5	0.6637	0.6716
		10	0.6147	0.6336
HieRAG++	L1: 25 L2: 50 L3: 100	1	0.6799	0.6897
		3	0.7096	0.7090
		5	0.7333	0.7145
		10	0.6643	0.6715
Dense RAG	—	1	0.4196	0.6435
		3	0.6047	0.6353
		5	0.5974	0.6171
		10	0.5998	0.6258
Sparse RAG	—	1	0.6135	0.3633
		3	0.6071	0.6378
		5	0.5921	0.6278
		10	0.5969	0.6298
CAG	—	5	0.4248	0.3387

Table 4.47: Risultati sperimentali in PubMedQA

Sistema	Capacità	topk	BERTScore	COMET
HieRAG	L1: 250, L2: 500, L3: 1000	5	0.6516	0.6688
HieRAG++	L1: 25, L2: 50, L3: 100	5	0.7333	0.7145
CAG	—	5	0.4248	0.3387
Dense RAG	—	3	0.6047	0.6353
Sparse RAG	—	3	0.6071	0.6378

Table 4.48: Migliori configurazioni per ciascun sistema in termini di efficacia su PubMedQA. La tabella mostra i valori massimi ottenuti da ciascun sistema per le metriche semantiche BERTScore e COMET, utilizzate per valutare la qualità delle risposte generate nel dominio biomedico. Per ogni sistema è indicata la configurazione ottimale in termini di struttura della cache (se presente) e top-k. I valori in grassetto rappresentano le migliori prestazioni complessive

Test statistici Il test statistico evidenzia che, su **PubMedQA**, la configurazione **HieRAG con cache ampia (L1:250, L2:500, L3:1000)** risulta **significativamente più efficace** rispetto a **Dense RAG** e **Sparse RAG** su tutti i valori di top-k. I *p-value* sono estremamente bassi a conferma che le differenze osservate nelle metriche (BERTScore, COMET) **non sono attribuibili al caso**, ma riflettono un **vantaggio reale e sistematico** della struttura gerarchica nel recuperare contesto rilevante in un corpus biomedico molto vasto.

Questo risultato è particolarmente rilevante considerando che Dense e Sparse RAG, pur interrogando l'intero corpus, **non adottano alcuna forma di caching o selezione progressiva**, e quindi sono soggetti a rumore semantico o ridondanza nei documenti recuperati. Al contrario, HieRAG beneficia di un accesso strutturato alla conoscenza, permettendo una **distribuzione del carico tra livelli di cache** che

ottimizza sia la precisione che i tempi di risposta.

Il confronto con **CAG**, invece, non risulta statisticamente significativo ($p > 0.05$ per tutti i top-k). Ciò suggerisce che, almeno nella variante ad alta capacità, HieRAG e CAG possono produrre risposte di qualità simile in alcuni casi. Tuttavia, è importante notare che CAG **presenta performance molto inferiori nelle metriche assolute** (COMET e BERTScore) e non gestisce retrieval esplicito: la mancata significatività statistica in questo contesto potrebbe dipendere da **una maggiore varianza dei risultati** o da un effetto di compressione sullo score.

In sintesi, il test conferma che **HieRAG si posiziona significativamente al di sopra delle principali baseline dense e sparse**, grazie a un'architettura progettata per **ridurre il rumore e massimizzare la rilevanza informativa**, dimostrandosi particolarmente adatto a scenari complessi come quello biomedico.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	0.08	0.72	0.81	0.80
HieRAG vs Dense RAG	$10^{-19}***$	$10^{-17}***$	$10^{-16}***$	$10^{-16}***$
HieRAG vs Sparse RAG	$10^{-8}***$	$10^{-7}***$	$10^{-6}***$	$10^{-6}***$

Table 4.49: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG ad alta capacità e gli altri sistemi al variare di *top-k*. I valori in grassetto indicano significatività statistica ($p < 0.05$).

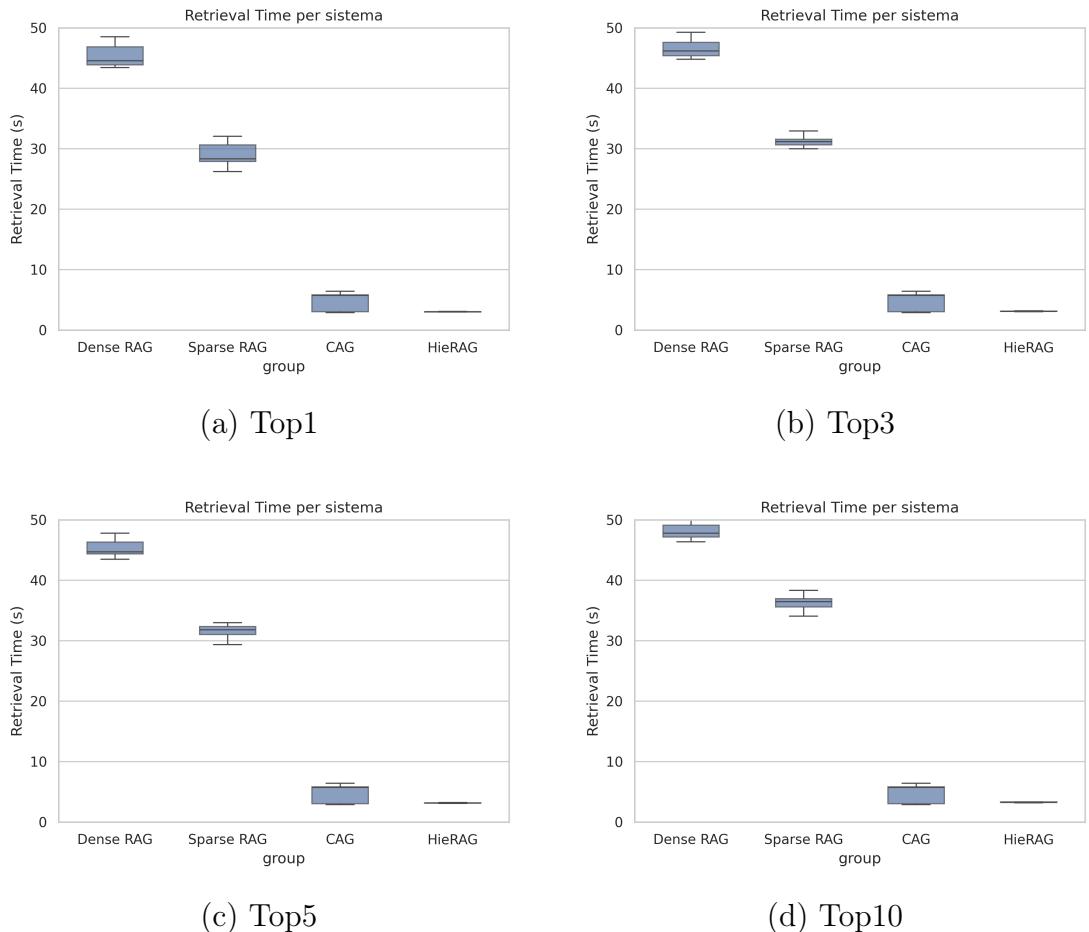


Figure 4.31: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal–Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il p -value < 0.05 , ** indica il p -value < 0.01 e *** indica che il p -value < 0.001). L'asterisco viene mostrato se e solo se HieRAG è diverso rispetto a tutti gli altri sistemi.

Anche nella sua **configurazione compatta** (L1:25, L2:50, L3:100), **HieRAG conferma la propria efficacia statistica** rispetto alle baseline dense e sparse. Il test di Dunn con correzione Holm mostra in-

fatti *p-value* estremamente bassi in tutti i confronti con **Dense RAG** e **Sparse RAG** ($p \leq 10^{-4}**$), indicando una **differenza statisticamente significativa** nella qualità delle risposte generate. Questo conferma che anche una struttura gerarchica leggera, se ben progettata, è in grado di garantire **un retrieval informativo più mirato e utile** rispetto a soluzioni prive di controllo esplicito sul contesto.

Il vantaggio strutturale di HieRAG emerge con chiarezza: anche operando su un corpus estremamente ampio come PubMedQA, la combinazione tra **gestione a più livelli della cache** e una **selezione efficace dei chunk** consente di massimizzare la rilevanza semantica del prompt, riducendo l'influenza di documenti irrilevanti.

Al contrario, il confronto con **CAG** non evidenzia differenze significative ($p > 0.75$), suggerendo che, in termini statistici, le due architetture potrebbero generare risposte di qualità simile in alcuni casi. Tuttavia, questo va interpretato con cautela: **le metriche assolute mostrano un chiaro vantaggio di HieRAG**, e l'assenza di significatività potrebbe essere attribuita al fatto che quest'ultimo non esegue retrieval esplicito, ma si basa su contesti compressi potenzialmente meno stabili.

In conclusione, il test statistico rafforza l'evidenza che **anche in forma compatta, HieRAG è significativamente superiore alle principali baseline**, e si conferma una **soluzione efficace, efficiente e scalabile** per sistemi di question answering in ambito biomedico.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	0.82	0.8	0.75	0.81
HieRAG vs Dense RAG	$10^{-12}***$	$10^{-14}***$	$10^{-12}***$	$10^{-11}***$
HieRAG vs Sparse RAG	$10^{-4}***$	$10^{-4}***$	$10^{-6}***$	$10^{-6}***$

Table 4.50: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con capacità ridotta e gli altri sistemi al variare di *top-k* (numero di documenti retrievati). I valori in grassetto indicano significatività statistica ($p < 0.05$).

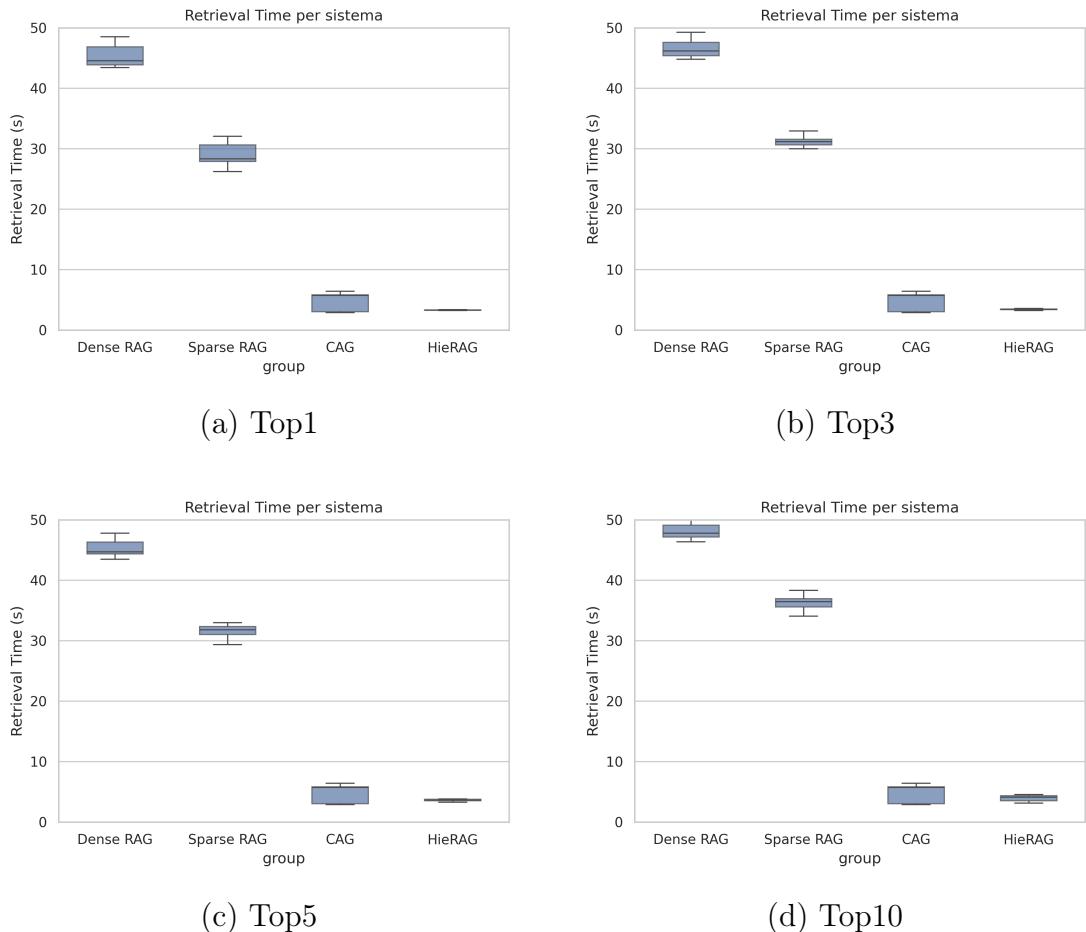


Figure 4.32: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se Hi-eRAG è diverso rispetto a tutti gli altri sistemi.

Il test di Dunn con correzione Holm mostra che, anche nella sua **forma più semplice e minimale**, **HiRAG monolivello** riesce a **superare in modo significativo Dense RAG e Sparse RAG**

in termini di qualità delle risposte, con p -value estremamente bassi su tutte le configurazioni di top-k ($p \leq 10^{-14}$ contro Dense RAG, $p \leq 10^{-6}$ contro Sparse RAG). Questi risultati indicano che, pur privo della struttura gerarchica completa, HieRAG conserva **un vantaggio metodologico importante nella selezione e gestione del contesto**, sufficiente a garantire risposte più rilevanti e coerenti rispetto alle baseline.

Questo effetto è particolarmente rilevante su **PubMedQA**, un corpus vasto e specialistico, dove **la qualità del chunk selezionato è cruciale** per produrre risposte affidabili. Anche con una sola cache (es. L1:25), il sistema è in grado di sfruttare l'efficienza del retrieval preindicizzato e la coerenza del filtro top-k per costruire prompt informativi e pertinenti.

Il confronto con **CAG**, invece, non mostra differenze statisticamente significative ($p > 0.6$ in tutte le configurazioni). Tuttavia, come già emerso in precedenza, ciò non implica equivalenza qualitativa: CAG soffre infatti **di bassa stabilità semantica** e prestazioni inferiori nelle metriche aggregate.

In sintesi, il test conferma che **HieRAG conserva una forte competitività anche in assenza della gerarchia**, risultando **statisticamente superiore ai modelli dense e sparse**, e sufficientemente solido da affrontare con successo anche domini complessi come quello biomedico.

Confronto	Top-1	Top-3	Top-5	Top-10
HieRAG vs CAG	0.8	0.77	0.64	0.69
HieRAG vs Dense RAG	$10^{-15}***$	$10^{-14}***$	$10^{-14}***$	$10^{-15}***$
HieRAG vs Sparse RAG	$10^{-6}***$	$10^{-7}***$	$10^{-6}***$	$10^{-6}***$

Table 4.51: P-value corretti (Dunn test con correzione Holm) per il confronto tra HieRAG con un solo livello di cache e gli altri sistemi al variare di *top-k* (numero di documenti retrievati). I valori in grassetto indicano significatività statistica ($p < 0.05$).

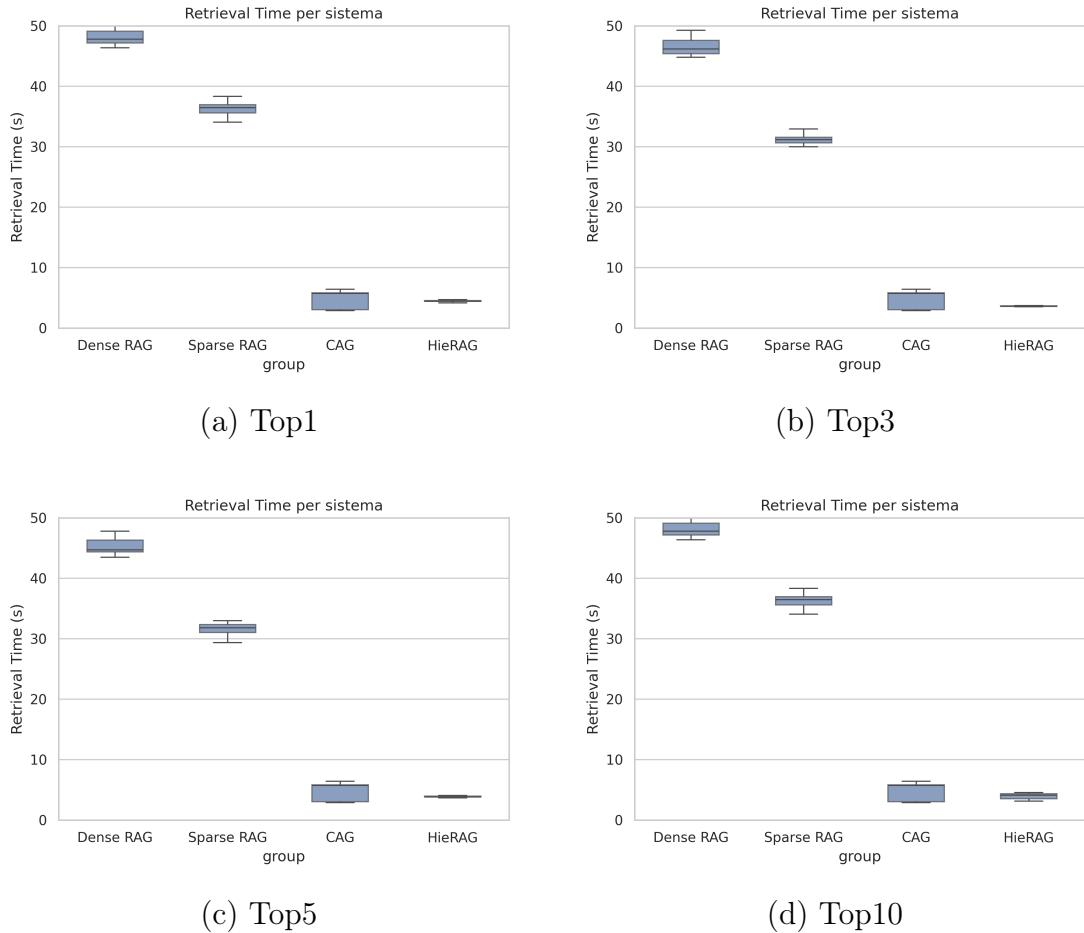


Figure 4.33: Boxplot dei tempi di retrieval (in secondi) ottenuti dai quattro sistemi di recupero: Dense RAG, Sparse RAG, CAG e Hi-eRAG, calcolati su un campione di 30 domande ciascuno. Le scatole rappresentano l'intervallo interquartile (IQR), con la linea mediana indicante il secondo quartile; i baffi si estendono fino a $1.5 \times \text{IQR}$. È stata effettuata un'analisi statistica non parametrica (test di Kruskal-Wallis, $p < 0.001$), seguita da un test post-hoc di Dunn con correzione di Holm per confronti multipli. Gli asterischi accanto all'etichetta Hi-eRAG indicano una differenza statisticamente significativa rispetto a tutti gli altri sistemi ($p < 0.05$) (* indica che il $p\text{-value} < 0.05$, ** indica il $p\text{-value} < 0.01$ e *** indica che il $p\text{-value} < 0.001$). L'asterisco viene mostrato se e solo se Hi-eRAG è diverso rispetto a tutti gli altri sistemi.

4.3 Risultati complessivi

L’analisi dei tempi di inferenza nei tre dataset SQuAD, HotPotQA e PubMedQA rivela un comportamento scalabile del sistema HieRAG, con prestazioni che migliorano proporzionalmente alla dimensione e complessità del corpus documentale. Sebbene i tempi di latenza assoluti risultino maggiori nel caso di PubMedQA, è proprio in questo contesto che la cache gerarchica esprime il massimo potenziale, raggiungendo una **riduzione del tempo medio di retrieval superiore al 50% rispetto al recupero diretto da database vettoriale**.

Nel caso del dataset SQuAD, caratterizzato da un corpus contenuto (300 documenti), la cache L1 si satura rapidamente, garantendo un’elevata frequenza di hit ma **limitando l’efficacia della gerarchia cache**, con L2 e L3 che rimangono sostanzialmente inutilizzate. Questo si traduce in tempi di risposta molto contenuti (circa 0.7 s), ma in un beneficio marginale rispetto all’adozione di un sistema di retrieval classico.

Nel dataset HotPotQA, il sistema HieRAG mostra un comportamento intermedio: la cache L1 continua a svolgere un ruolo predominante, ma si osservano i primi segnali di attivazione dei livelli superiori (L2 e L3), con una **riduzione percentuale del retrieval time compresa tra il 3% e il 5%** rispetto al database vettoriale.

Con l’aumentare della dimensione del corpus, come nel caso di PubMedQA, si osserva un impatto sempre più rilevante dell’organizzazione

multi-livello della cache. I costi iniziali dovuti alla latenza di popolamento della struttura vengono rapidamente ammortizzati grazie alla maggiore densità informativa e alla diversificazione delle query, che favoriscono il riutilizzo di contenuti già acquisiti. Il sistema raggiunge così una **stabilità nei tempi di risposta**, pur operando su una scala notevolmente più ampia rispetto agli altri due scenari.

In conclusione, l'efficacia del sistema HieRAG cresce proporzionalmente alla dimensione del corpus: **maggior è il dataset, maggior è il guadagno ottenuto dall'impiego della cache gerarchica**, sia in termini di tempo di retrieval che di prevedibilità della latenza. Questi risultati ne evidenziano l'idoneità all'applicazione in domini su larga scala, come quello biomedicale rappresentato da PubMedQA

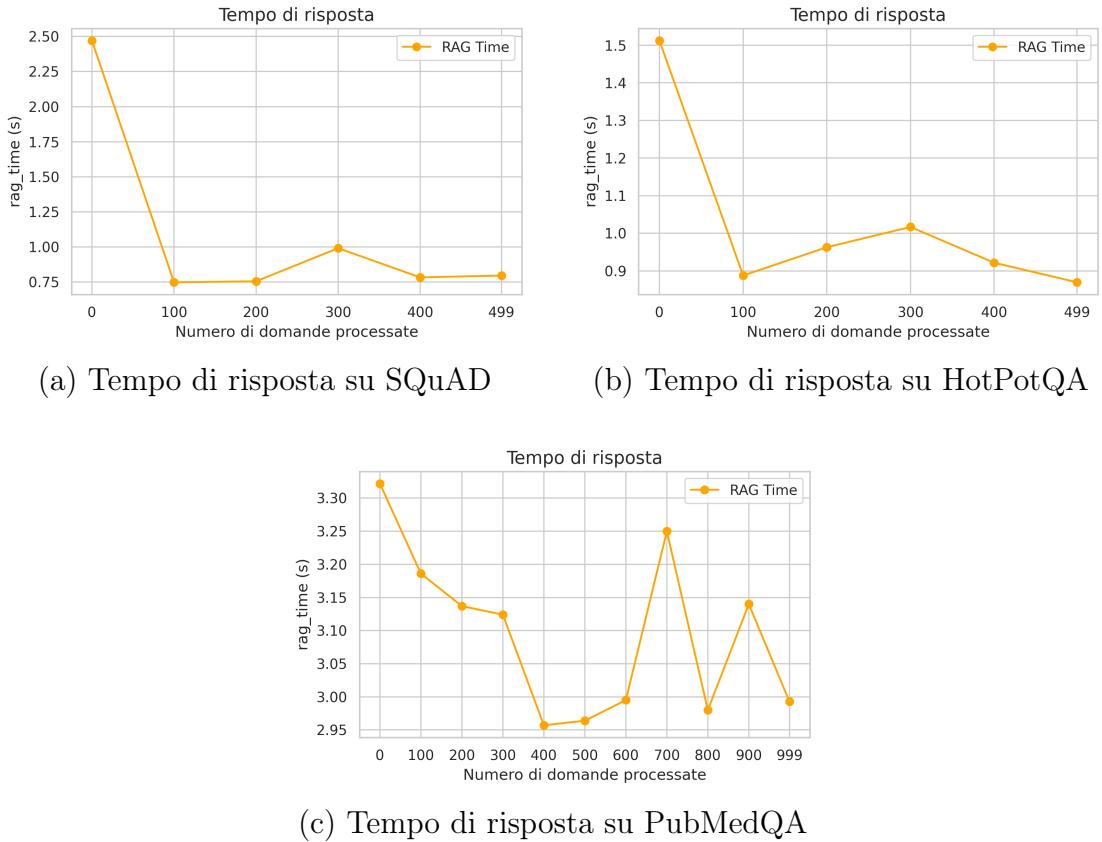


Figure 4.34: Andamento del tempo medio di risposta del sistema Hi-eRAG nei dataset SQuAD, HotPotQA e PubMedQA. Nonostante la crescita del corpus, si osserva una stabilizzazione progressiva dei tempi di risposta grazie all’efficienza del meccanismo di caching. In particolare, nel caso di PubMedQA la struttura gerarchica consente di contenere la latenza anche in condizioni di scala molto elevata

Possiamo confermare in modo chiaro l’efficacia della struttura gerarchica della cache proposta, con un impatto particolarmente marcato all’aumentare della dimensione del corpus. Nel caso di **SQuAD**, contenente circa 300 documenti, il tempo medio di accesso al livello L1 è già sensibilmente inferiore rispetto a quello necessario per consultare direttamente il database vettoriale FAISS, con una riduzione dei tempi di retrieval superiore al **30%**. Tale miglioramento si consolida ulteriori

ormente su **HotPotQA**, dove i documenti ammontano a oltre 7000: in questo contesto, la cache L1 garantisce una riduzione media dei tempi di retrieval di circa **54%** rispetto all’accesso diretto al vettore FAISS. Tuttavia, è nel dataset **PubMedQA**, che supera i **200.000** documenti, che il vantaggio della cache risulta più evidente. In questo scenario, tutti i livelli (L1, L2 e L3) della cache gerarchica vengono sfruttati attivamente, permettendo una riduzione media dei tempi di retrieval pari a oltre il **70%** rispetto al database vettoriale.

Questi risultati evidenziano non solo l’efficacia del meccanismo di caching nella riduzione della latenza, ma anche la **scalabilità** intrinseca della soluzione. L’architettura proposta è infatti capace di adattarsi dinamicamente alla dimensione del corpus, garantendo benefici tanto in ambienti a bassa entropia come SQuAD quanto in domini specialistici e molto estesi come PubMedQA. In particolare, la crescente attivazione dei livelli L2 e L3 in quest’ultimo dimostra che la progettazione multi-livello non è ridondante, ma anzi strategica per contenere l’esplosione dei tempi di ricerca che si verifica nei sistemi RAG tradizionali a fronte di basi documentali molto grandi

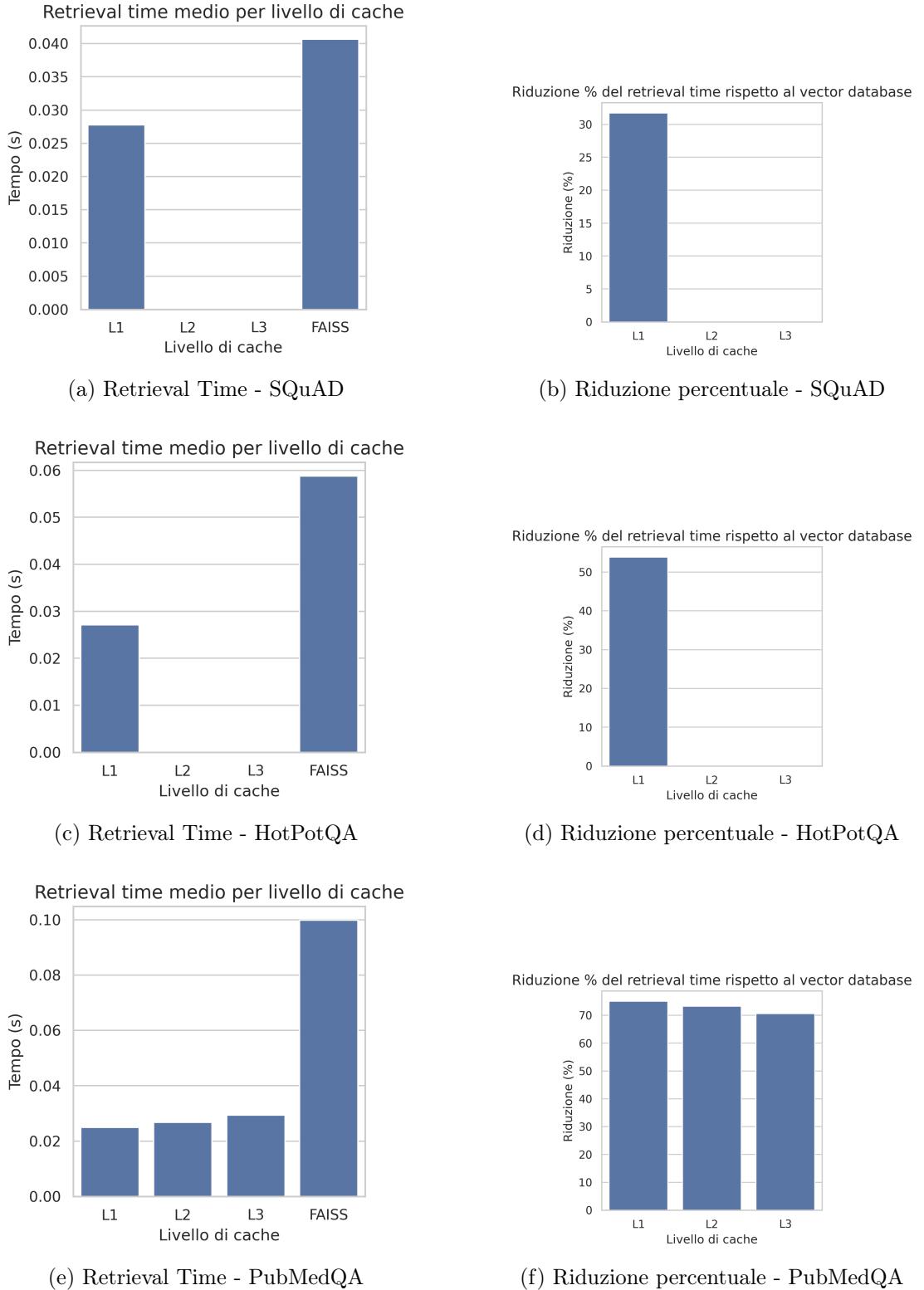


Figure 4.35: Tempi medi di retrieval (sinistra) e riduzione percentuale rispetto all'accesso diretto al database FAISS (destra) nei dataset SQuAD (sopra), HotPotQA (centro) e PubMedQA (sotto). All'aumentare della dimensione del corpus, la struttura gerarchica della cache garantisce riduzioni di latenza sempre più significative, con vantaggi che superano il 70% nel caso di PubMedQA

L’analisi dei tempi di inferenza condotta sui tre dataset SQuAD, HotPotQA e PubMedQA mette in luce le profonde differenze tra il sistema HieRAG e le architetture baseline Dense RAG, Sparse RAG e CAG. In particolare, i grafici riportati evidenziano come HieRAG mantenga una latenza di inferenza nettamente più contenuta e stabile, indipendentemente dalla dimensione del corpus sottostante. Su SQuAD, che rappresenta il caso più semplice con 300 documenti, tutti i sistemi presentano tempi medi relativamente contenuti, ma già qui CAG si distingue negativamente con picchi superiori ai 3 secondi, contro i circa 0.7–0.8 s di HieRAG e Dense/Sparse RAG. I boxplot relativi al retrieval time mostrano che HieRAG ha una distribuzione stretta, con valori compresi in un intervallo inferiore al secondo, e in linea con le altre due baseline (Dense e Sparse), ma con una varianza inferiore.

Il gap si amplifica significativamente su HotPotQA, che presenta un corpus di oltre 7.000 documenti. Qui, i sistemi Dense e Sparse RAG mostrano un incremento marcato nei tempi medi di inferenza, con valori che superano stabilmente i 6–7 secondi. CAG, in particolare, soffre un deterioramento critico delle performance, con tempi di risposta medi superiori ai 20 secondi, arrivando in alcuni casi oltre i 30. HieRAG, invece, mantiene una latenza stabile attorno al secondo. Anche i boxplot dei tempi di retrieval confermano tale osservazione: HieRAG ha valori medi e massimi significativamente più bassi rispetto a CAG e Sparse RAG, che iniziano a soffrire della complessità di gestione su

una base documentale più ampia. Questa tendenza è attribuibile alla struttura gerarchica del sistema, in cui la maggior parte delle richieste viene soddisfatta attraverso la cache L1, evitando l’accesso diretto al database vettoriale.

La superiorità di HieRAG diventa ancora più evidente nel caso PubMedQA, in cui il numero di documenti supera le 200.000 unità. I sistemi Dense e Sparse RAG soffrono pesantemente in questo scenario: Dense RAG arriva a tempi medi di inferenza superiori ai 40 secondi, e Sparse RAG si assesta tra i 25 e i 30. CAG, nonostante utilizzi meccanismi di caching, non riesce a compensare la crescita della complessità documentale, con latenze medie anch’esse tra i 15 e i 25 secondi. Al contrario, HieRAG mantiene una latenza di inferenza costante e bassa, tra i 2 e i 3 secondi. I boxplot del retrieval confermano il vantaggio architettonico del sistema: HieRAG è l’unico a non subire un peggioramento lineare della latenza al crescere del dataset, mostrando invece un comportamento scalabile e resiliente.

A supporto di questi risultati, si considerino anche i grafici dedicati al tempo medio di retrieval e alla riduzione percentuale del tempo rispetto al database vettoriale. Su SQuAD, l’accesso alla cache L1 consente già una riduzione del 30% rispetto a FAISS. Su HotPotQA, la riduzione cresce fino al 50%. Ma è su PubMedQA che emerge l’efficacia strutturale di HieRAG: grazie alla distribuzione gerarchica multilivello (L1, L2, L3) e a un meccanismo di “cache promotion” ispirato

all’architettura delle CPU, si ottiene una riduzione fino al 70–75% del tempo di retrieval rispetto al solo accesso a FAISS. È interessante notare che, al crescere del corpus, non solo la latenza non aumenta, ma i benefici derivanti dalla cache diventano più marcati.

Dal punto di vista ingegneristico, HieRAG dimostra dunque due elementi chiave: **efficienza temporale garantita e robustezza architetturale sotto carico crescente**. A differenza di CAG, che presenta un’elevata variabilità nei tempi di inferenza a causa della costruzione del grafo e della computazione dei cammini più rilevanti, HieRAG affida la selezione dei documenti a un processo più snello, in cui solo una porzione minima delle richieste giunge al livello FAISS. Inoltre, l’assenza di picchi significativi nella latenza anche su dataset molto ampi lo rende adatto a scenari real-time, come ad esempio assistenti intelligenti o sistemi biomedicali a risposta rapida.

In conclusione, rispetto alle architetture baseline presenti in letteratura, HieRAG introduce un significativo avanzamento nello stato dell’arte, non solo per la sua efficienza nel retrieval, ma soprattutto per la sua capacità di garantire prestazioni stabili, scalabili e prevedibili in presenza di basi documentali di dimensioni molto diverse. La progettazione gerarchica e l’adozione di strategie ispirate alla memoria cache dei sistemi computazionali risultano decisive nel determinare questa superiorità.

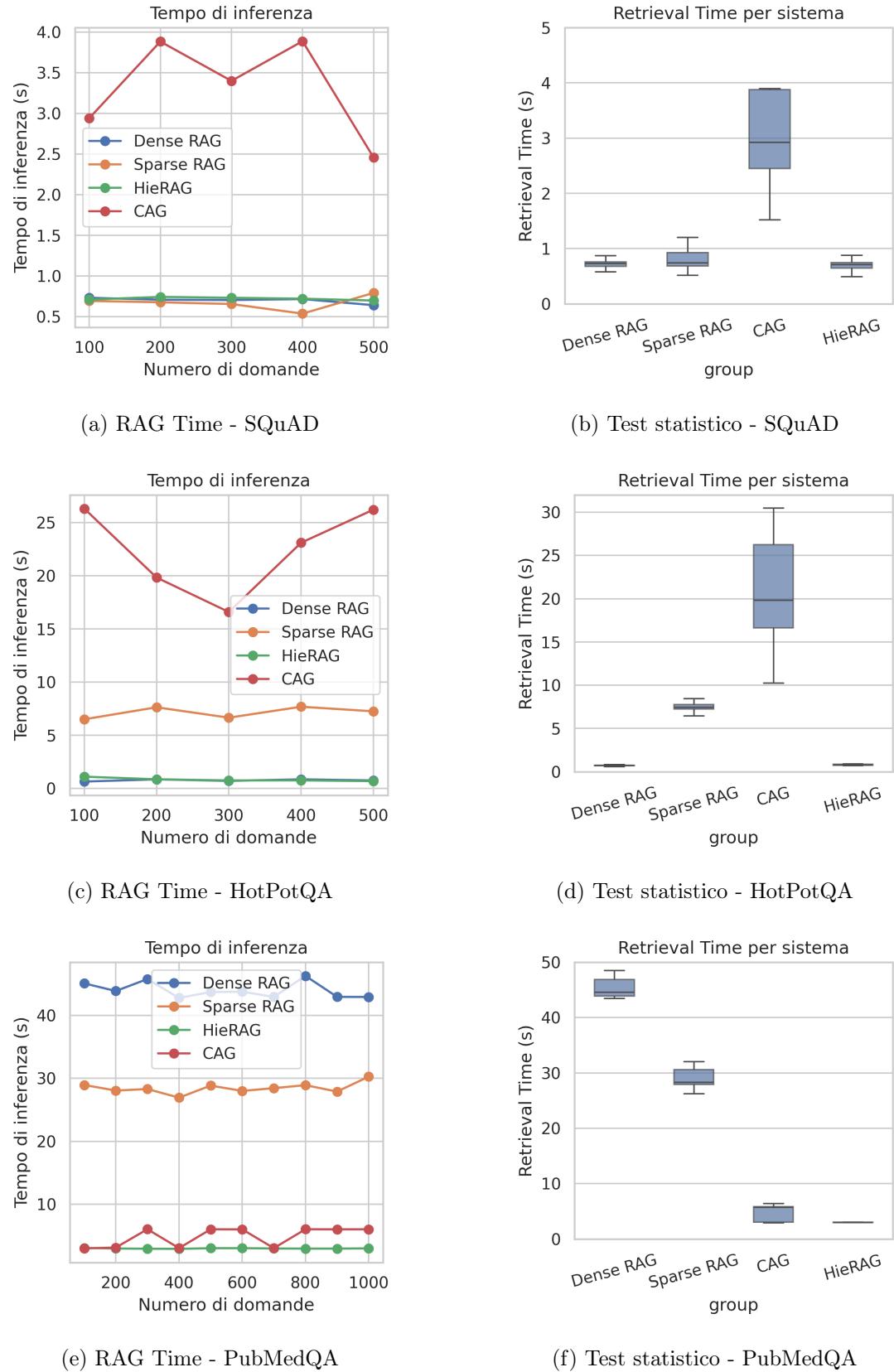


Figure 4.36: Confronto dei tempi di inferenza e del retrieval time tra i sistemi HierRAG, Dense RAG, Sparse RAG e CAG sui dataset SQuAD, HotPotQA e PubMedQA.

L’analisi della qualità semantica 4.52 delle risposte generate, valutata attraverso la metrica **BERTScore**, evidenzia con chiarezza la superiorità del sistema **HieRAG** rispetto alle principali baseline di riferimento. In tutti e tre i dataset considerati, HieRAG raggiunge i valori di BERTScore più elevati, dimostrando una **maggior coerenza semantica** tra le risposte generate e quelle di riferimento. In particolare, su **PubMedQA**, caratterizzato da un linguaggio tecnico e un contenuto altamente specialistico, HieRAG raggiunge un valore di **0.6516**, superando nettamente sia Dense RAG (0.6047) che Sparse RAG (0.6071), a testimonianza della sua capacità di adattarsi a contesti complessi e dominî di conoscenza specializzati. Anche su **HotPotQA**, che richiede un ragionamento multihop, HieRAG ottiene un punteggio superiore (0.6730), confermando la sua abilità nell’integrare informazioni provenienti da fonti multiple in modo coeso. Infine, su **SQuAD**, pur trattandosi di un benchmark relativamente semplice, HieRAG registra il valore più alto (0.6980), a indicare che la sua architettura non compromette la qualità nemmeno in scenari meno complessi. Questi risultati dimostrano che l’efficienza computazionale ottenuta attraverso la struttura gerarchica e il caching intelligente di HieRAG non sacrifica in alcun modo la **qualità linguistica e semantica delle risposte**, ma al contrario, la migliora sistematicamente rispetto allo stato dell’arte.

System	SQuAD	HotPotQA	PubMedQA
BERTScore			
CAG	0.6930	0.6083	0.4248
Dense RAG	0.6784	0.6534	0.6047
Sparse RAG	0.6876	0.6315	0.6071
HieRAG	0.6980	0.6730	0.6516
HieRAG++	0.7633	0.7334	0.7333

Table 4.52: Confronto della qualità semantica delle risposte generate dai sistemi CAG, Dense RAG, Sparse RAG e HieRAG sui dataset SQuAD, HotPotQA e PubMedQA, misurata mediante la metrica BERTScore. HieRAG ottiene sistematicamente le migliori prestazioni su tutti i benchmark (HieRAG++ intende la versione HieRAG con il reranker)

Chapter 5

Conclusioni e sviluppi futuri

In questa tesi è stato progettato e sperimentato un sistema con l'obiettivo di migliorare l'efficienza e l'efficacia della fase di retrieval nei sistemi RAG. Questo approccio si distingue rispetto alla letteratura esistente per la capacità di introdurre una cache multilivello per mantenere nel tempo i contenuti più rilevanti e ridurre le chiamate al database vettoriale che risultano essere abbastanza lunghe a seconda della grandezza del dataset.

Attraverso una serie di esperimenti, è stato dimostrato che il sistema HieRAG è in grado di migliorare i tempi di inferenza complessi e la qualità delle risposte generate. Le analisi hanno evidenziato che la profondità della cache, il numero di documenti recuperati e la capacità dei livelli influiscono in modo significativo sulle prestazioni. In particolare,

è stato dimostrato che l'introduzione di una cache stratificata e ampia è vantaggiosa rispetto alle configurazioni semplici di RAG. Nonostante i risultati promettenti, il lavoro presenta delle limitazioni che aprono spazio a futuri sviluppi.

Un primo aspetto da approfondire riguarda la politica di aggiornamento della cache. Attualmente, la promozione e degradazione dei documenti tra i livelli si basa su misure di similarità che potrebbero essere sostituite o affiancate da criteri più efficienti.

Un altro aspetto è quello di approfondire ulteriormente la sperimentazione con l'integrazione di LLM, risorse hardware più efficienti oppure provare a sperimentare il sistema su un dataset di documenti molto vasto che superi valori di giga di memoria e confrontare i risultati con altre baseline.

Inoltre, il sistema potrebbe beneficiare dell'integrazione di LLM con una grossa finestra di contesto, capace di superare i 100k token. In questi casi, la cache potrebbe essere sfruttata non solo per recuperare documenti ma anche per costruire dinamicamente il prompt completo, ottimizzando la scelta dei documenti da fornire.

Bibliography

- [1] Muhammad Arslan, Hussam Ghanem, Saba Munawar, and Christophe Cruz. A survey on RAG with llms. In Carlos Toro, Sebastián A. Ríos, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 28th International Conference KES-2024, Seville, Spain, 11-13 September 2023*, volume 246 of *Procedia Computer Science*, pages 3781–3790. Elsevier, 2024.
- [2] Brian J. Chan, Chao-Ting Chen, Jui-Hung Cheng, and Hen-Hsen Huang. Don't do RAG: when cache-augmented generation is all you need for knowledge tasks. In Guodong Long, Michale Blumestein, Yi Chang, Liane Lewin-Eytan, Zi Helen Huang, and Elad Yom-Tov, editors, *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, pages 893–897. ACM, 2025.
- [3] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvassy, Pierre-Emmanuel Mazaré, Maria Lomeli,

- Lucas Hosseini, and Hervé Jégou. The faiss library. *CoRR*, abs/2401.08281, 2024.
- [4] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Tuvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy

- Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.
- [5] Zhe Jia, Marco Maggioni, Jeffrey Smith, and Daniele Paolo Scarpazza. Dissecting the nvidia turing T4 GPU via microbenchmarking. *CoRR*, abs/1903.07486, 2019.
- [6] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2567–2577. Association for Computational Linguistics, 2019.
- [7] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Nat-*

- ural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics, 2016.
- [8] Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, and Suranga Nanayakkara. Fine-tune the entire RAG architecture (including DPR retriever) for question-answering. *CoRR*, abs/2106.11517, 2021.
- [9] Yang Wang, Guanlin Dai, Song Ke, and Chao Zheng. Evaluating sparse and dense retrieval in retrieval-augmented generation systems: A study. In *Proceedings of the 10th International Conference on Communication and Information Processing, ICCIP 2024, Lingshui, Hainan, China, November 21-24, 2024*, pages 548–554. ACM, 2024.
- [10] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics, 2018.

- [11] Yun Zhu, Jia-Chen Gu, Caitlin Sikora, Ho Ko, Yinxiao Liu, Chu-Cheng Lin, Lei Shu, Liangchen Luo, Lei Meng, Bang Liu, and Jindong Chen. Accelerating inference of retrieval-augmented generation via sparse context selection. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.