Journal of Artificial Intelligence and Technology, 2026, 6, 989-997
https://doi.org/10.37965/jait.2026.1569

ISTP
RESEARCH ARTICLE

HiRAG: A Hierarchical Retrieval-Augmented Generation Framework via Intent Routing and Small-Model Refinement

Sunjia Zhang
School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, Australia

(Received 25 June 2026; Revised 14 October 2026; Accepted 26 October 2026; Published online 02 November 2026)

Abstract: While retrieval-augmented generation (RAG) mitigates large language model (LLM) hallucinations, traditional flat vector retrieval in massive, heterogeneous databases suffers from high latency and noise interference. To address this, we propose HiRAG, a hierarchical RAG framework. Offline, HiRAG builds a "classification-summary" dual-layer index to structurally organize data. Online, a lightweight bidirectional encoder representations from transformers (BERT)-based router identifies query intent to dynamically prune the search space. A cascaded pipeline then uses hybrid retrieval and Small-LLM reranking for coarse-to-fine de-noising. Experiments on the Microsoft Machine Reading Comprehension (MS MARCO) and Multi-Domain Enterprise Knowledge Base (MDE-KB) datasets demonstrate that HiRAG outperforms traditional baselines. Specifically on MDE-KB, HiRAG improves Recall@5 by 14.5 percentage points over Naive RAG and achieves a retrieval augmented generation assessment (RAGAS) Answer Relevance score of 0.92, while reducing average query latency by 28%. HiRAG provides a viable, efficient solution for large-scale knowledge bases.

Keywords: hierarchical retrieval; hybrid retrieval; intent routing; RAG

I. INTRODUCTION

Large language models (LLMs) have become powerful general-purpose generators, but their answers remain bounded by parametric memory and may contain unsupported claims when the requested knowledge is recent, specialized, or hidden inside private documents [1–3]. Retrieval-augmented generation (RAG) addresses this limitation by grounding generation in external evidence rather than relying on memorized parameters alone. In practice, however, the success of RAG depends less on the existence of a retriever than on whether the retriever can supply evidence that matches the user's actual intent.

This requirement becomes difficult in enterprise-scale heterogeneous corpora. A single knowledge base may contain product manuals, software incident tickets, compliance policies, financial reports, and meeting records. These domains often share vocabulary while answering different types of questions. For instance, the term "approval" may refer to an authorization workflow in a software log, a compliance procedure in a policy document, or a financial sign-off in an audit trail. A flat vector index treats all chunks as one undifferentiated retrieval space, so semantically adjacent but operationally irrelevant passages can be ranked above the evidence needed for the current query [4–6].

The second challenge is efficiency. Modern RAG pipelines often add hybrid retrieval, reranking, and context compression to improve answer quality, but each additional stage consumes latency budget. When every query is searched against the entire corpus, the system pays this cost even when the user intent clearly belongs to one domain. Existing hierarchical and adaptive RAG approaches have explored structured knowledge organization and query-dependent retrieval, respectively. However, how to combine corpus-level hierarchical organization with lightweight intent-based search-space pruning while preserving global retrieval for uncertain queries remains insufficiently explored. HiRAG is motivated by the observation that many queries can be routed before retrieval: if a lightweight model can identify the likely semantic region, the downstream retriever can search a smaller and cleaner sub-index while retaining a global fallback path for ambiguous questions.

To operationalize this idea, HiRAG combines an offline structure-aware index with an online coarse-to-fine retrieval pipeline. Offline, the corpus is clustered into semantic subspaces and summarized into cluster-level routing anchors. Online, a bidirectional encoder representations from transformers (BERT)-based intent router selects a target sub-index when confidence is high and falls back to the global index when confidence is low. Hybrid retrieval then gathers candidate evidence, and a small LLM refiner compresses the candidate set before final answer generation. This design aims to reduce cross-domain noise and retrieval latency without making the routing step itself a large-model bottleneck. The key distinction is that corpus organization, confidence-aware search-space selection, and evidence refinement are connected in a single coarse-to-fine pipeline, allowing HiRAG to reduce the retrieval space before expensive downstream processing rather than relying solely on post-retrieval filtering.

The main contributions of this paper are summarized as follows:

- Structure-Aware RAG Framework (HiRAG): We propose a "classification-summary" dual-layer index that hierarchically organizes heterogeneous knowledge bases via offline LLM summarization. Unlike flat indexing, this design exposes corpus-level semantic structure directly to the online routing stage.
- Intent-Driven Routing and Cascaded Refinement: We introduce a lightweight BERT router to dynamically prune the search space and reduce latency. A confidence-based global

Corresponding author: Sunjia Zhang (e-mail: s4855593@student.uq.edu.au).

© The Author(s) 2026. This is an open access article published under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

fallback is retained for uncertain queries to reduce the risk of information loss caused by incorrect local routing. Additionally, a Small-LLM reranker filters irrelevant noise to mitigate the "Lost in the Middle" effect.

- wEmpirical Validation: Experiments on the Microsoft Machine Reading Comprehension (MS MARCO) and Multi-Domain Enterprise Knowledge Base (MDE-KB) datasets demonstrate that HiRAG significantly outperforms traditional baselines in retrieval recall, QA accuracy, and query latency.

II. RELATED WORK

RAG research first established that retrieval can improve knowledge-intensive generation by supplying passages that are not stored reliably in model parameters. Fusion-in-Decoder showed that a generator can aggregate multiple retrieved passages for open-domain question answering, while retrieval-enhanced language modeling later scaled the idea to very large external memories [7,8]. These systems motivate retrieval as an external evidence channel, but they leave open how the retrieval space should be organized when the corpus contains multiple latent domains.

A second line of work improves the retriever itself. Sparse retrieval remains effective for exact entities, clauses, product codes, and other lexical constraints, while dense retrieval and late-interaction models improve semantic matching when user wording differs from document wording [9–11]. Hybrid retrieval combines these signals through rank-level fusion, but the fusion layer still has to search and merge candidates from a chosen corpus region. Efficient vector indexes and heterogeneous retrieval benchmarks are therefore important for deployment when the collection reaches millions of chunks [12–14].

Long-context studies further show that retrieval recall is not sufficient. Even if the relevant passage is present, answer quality can degrade when useful evidence is buried among distracting documents [15]. This finding suggests that RAG systems should reduce irrelevant context before generation rather than simply increase the number of retrieved passages.

Adaptive retrieval methods make the retrieval process conditional on the query. HyDE strengthens dense retrieval without explicit relevance labels, and recent embedding resources make compact domain adaptation more practical in specialized settings [16,17]. Self-RAG lets a model decide when to retrieve and critique its evidence, whereas Adaptive-RAG routes questions according to estimated complexity [18,19]. In contrast, the present work focuses on a complementary form of adaptive control: a lightweight discriminative router determines where to retrieve within a heterogeneous corpus, rather than primarily deciding whether or how retrieval should be performed, thereby reducing the search space before downstream retrieval and refinement.

Post-retrieval processing addresses the quality of evidence after first-stage retrieval. RECOMP compresses retrieved contexts so that generation receives denser evidence, and RankRAG integrates context ranking with RAG [20,21]. These approaches indicate that the candidate set should be refined before final generation, especially when recall-oriented retrieval returns redundant or partially relevant chunks. HiRAG uses refinement for a similar purpose but places it after intent-based search-space pruning and hybrid retrieval. Thus, routing reduces cross-domain interference before retrieval, while the Small-LLM refiner removes residual redundancy after retrieval.

RAG evaluation has also moved beyond simple retrieval recall. Retrieval augmented generation assessment (RAGAS) measures answer relevance and context precision, RetrievalQA evaluates adaptive RAG behavior in short-form open-domain question answering, and automated RAG evaluation system (ARES) provides an automated evaluation framework for retrieval-augmented systems [22–24]. Broader benchmarks such as comprehensive RAG benchmark (CRAG) further stress-test RAG systems across diverse task types and realistic failure modes [25].

Recent surveys and system-oriented discussions frame RAG as a deployable architecture rather than a single model component [26,27]. Within that broader view, multi-step retrieval methods retrieve evidence as reasoning unfolds: interleaved retrieval with chain-of-thought supports knowledge-intensive multi-hop questions, Chain-of-Note improves robustness by annotating retrieved evidence, and DRAGIN triggers retrieval according to real-time information need [28–30].

Finally, structure-aware RAG is closest to the hierarchical motivation of this paper. LightRAG emphasizes lightweight structure for fast knowledge access, and retrieval-and-structuring augmented generation reorganizes retrieved information before generation [31,32]. Compared with these directions, the key gap addressed here is how to combine corpus-level hierarchy with low-cost intent routing for large heterogeneous knowledge bases. In HiRAG, the hierarchy directly defines the candidate search spaces available to the online router, rather than serving only as an auxiliary representation of retrieved knowledge. Moreover, the global index is retained as a confidence-aware fallback, making the hierarchical decision soft rather than a mandatory partition of the retrieval space.

III. METHODOLOGY

A. OVERVIEW

Figure 1 summarizes HiRAG as a two-stage system rather than a collection of independent add-ons. The offline stage builds the search structure, while the online stage uses that structure to make query-time retrieval cheaper and cleaner. This division is important because the expensive operations that organize the corpus, such as clustering and cluster summarization, can be performed outside the user-facing latency path.

The offline stage first converts heterogeneous documents into overlapping chunks, embeds them, and groups them into semantic clusters. For each cluster, HiRAG generates a meta-summary that describes the cluster's dominant topics, typical entities, and likely user intents. These summaries are not only documentation artifacts; they become semantic anchors for training and calibrating the intent router.

At inference time, the router estimates whether a query is specific enough to enter a sub-index. A high-confidence query is searched locally, reducing both vector search cost and cross-domain noise. A low-confidence query falls back to the global index, preserving recall for ambiguous or multi-hop questions. The fallback path turns hierarchy into a soft decision rather than a brittle constraint.

The remaining modules then operate in a coarse-to-fine order. Hybrid retrieval first gathers a recall-oriented candidate set from the selected search space. A small LLM refiner compresses this set into a short evidence pack, removing redundancy before the final generator sees the context. This ordering assigns a narrow responsibility to each module: routing controls where to search, hybrid retrieval controls what to retrieve, and refinement controls how much evidence is passed forward.

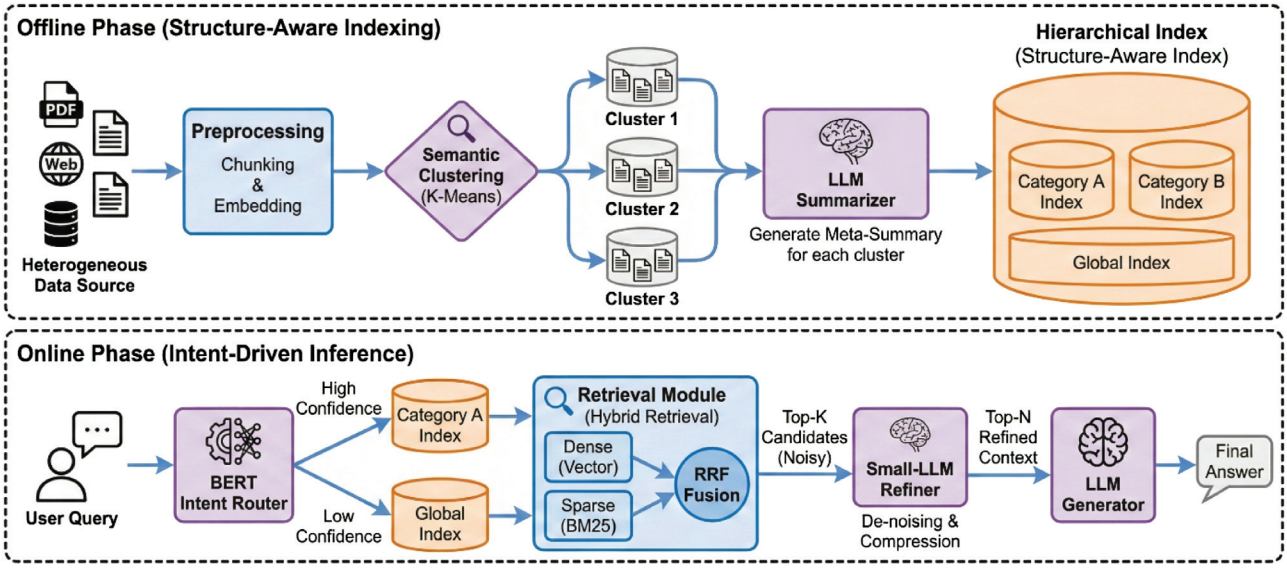


Fig. 1. The overall architecture of the HiRAG framework.

B. STRUCTURE-AWARE KNOWLEDGE INDEXING

The quality of the hierarchy depends on the quality of the offline indexing stage. Chunking should preserve enough local context for evidence interpretation while avoiding overly long passages that mix unrelated facts. After embedding, clustering provides a weak but useful approximation of domain boundaries. Because clustering can sometimes reflect writing style rather than topic, HiRAG uses cluster summaries as an interpretability layer. A summary should describe not only what a cluster contains but also what kinds of queries should be routed to it. In this sense, summaries act as a bridge between unsupervised corpus organization and supervised intent routing.

The global index is retained deliberately. A purely hierarchical system would be vulnerable to routing mistakes and cross-domain questions. HiRAG instead builds a dual-layer structure: local sub-indices support efficient domain-specific retrieval, while the global index handles ambiguous or multi-hop requests. This design also simplifies incremental updates. New documents can first be assigned to a likely cluster, indexed locally, and periodically used to refresh cluster centroids and summaries. When the distribution of documents changes substantially, the hierarchy can be rebuilt offline without changing the online interface exposed to users.

To overcome the inefficiencies of flat indexing, HiRAG constructs a structure-aware hierarchical index offline. Given a heterogeneous corpus $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, we apply a sliding window strategy to segment documents into fixed-size chunks, denoted as $c_{i,j}$ for the j-th chunk of document d_i. A pretrained embedding model $E(\cdot)$ maps each chunk into a high-dimensional vector space:

To automatically discover latent domains without manual tagging, K-means clustering partitions the vector set $\mathcal{V}$ into K disjoint clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$. This is achieved by minimizing the within-cluster sum of squares (WCSS):

$$\min_{\mathcal{C}} \sum_{k=1}^{K} \sum_{\mathbf{v} \in C_k} ||\mathbf{v} - \boldsymbol{\mu}_k||^2 \tag{1}$$

where μ_k is the centroid of cluster C_k. To generate semantic labels for the intent router, an LLM synthesizes a meta-summary S_k for each cluster by processing the chunks nearest to μ_k:

$$S_k = \text{LLM}(\text{Prompt} \oplus \text{SampledChunks}(C_k)) \tag{2}$$

Ultimately, this process yields a dual-layer architecture comprising a global flat index of $\mathcal{V}$ fallback searches and K independent sub-indices for low-latency, domain-specific retrieval.

C. INTENT-DRIVEN QUERY ROUTING

The router is trained to answer a constrained question: which semantic region should be searched first? This is simpler than answer generation and therefore does not require a large model. In practice, the training set can be produced from cluster summaries, representative chunks, and historical user queries. Synthetic queries generated from summaries provide initial coverage, while logged queries improve alignment with real user language. The router output is then calibrated on a validation set so that confidence values are meaningful. Calibration is important because overconfident misrouting can reduce recall, whereas excessive fallback weakens the latency advantage.

Routing also improves the quality of the candidate set before reranking. In a flat index, the top candidates may include documents that are semantically close but belong to a different business function. A legal query may retrieve policy language from an unrelated jurisdiction; a software query may retrieve a troubleshooting note for a different component. By routing first, HiRAG reduces these cross-domain collisions. The improvement is not merely computational. It changes the distribution of evidence passed to the hybrid retriever and the Small-LLM refiner, allowing later stages to spend less effort removing obvious noise.

The core component of the online phase is the Intent-Driven Router, designed to prune the search space dynamically based on the semantic specificity of the user query. Traditional RAG systems perform global retrieval for every query, incurring high computational costs. In contrast, HiRAG employs a lightweight BERT-based classifier, denoted as f_θ, to predict the probability distribution of a query q over the predefined semantic clusters $\mathcal{C}$.

Specifically, the router takes the user query q as input and outputs a confidence score vector $p = f_\theta(q) \in \mathbb{R}^K$, where p_k represents the probability that the query belongs to the k-th cluster.

The model is fine-tuned using the cluster meta-summaries generated in the offline phase as ground-truth labels. To balance retrieval precision and recall, we introduce an adaptive routing strategy controlled by a confidence threshold τ. The target search space, denoted as $\mathcal{I}_{\text{target}}$, is determined as follows:

$$\mathcal{I}_{target} = \begin{cases} \mathcal{I}_{k^*} & \text{if } \max(\mathbf{p}) > \tau, \text{where } k^* = \text{argmax}\,(\mathbf{p}) \\ \mathcal{I}_{global} & \text{otherwise} \end{cases} \quad (3)$$

When the router's maximum confidence score exceeds τ, the query is deemed "domain-specific" (e.g., a specific legal question), and the system is routed to the corresponding sub-index $\mathcal{I}_{k^*}$. This mechanism significantly reduces the search complexity from O(N) to O(N/K). Conversely, if the confidence is low (indicating an ambiguous or multi-hop query), the system falls back to the global index $\mathcal{I}_{\text{global}}$ to ensure comprehensive recall. This hybrid approach effectively mitigates the "efficiency-accuracy trade-off" inherent in large-scale retrieval systems.

D. HYBRID RETRIEVAL AND FUSION

Hybrid retrieval remains necessary even after routing. A targeted sub-index is smaller and cleaner, but it still contains many passages with different levels of relevance. Dense retrieval captures paraphrases and conceptual similarity, which is useful when users ask questions in language different from the source documents. Sparse retrieval captures exact expressions, which is critical for identifiers, version numbers, error codes, author names, and regulatory terms. In enterprise settings, these exact terms often determine whether a passage is truly relevant. HiRAG therefore treats dense and sparse retrieval as complementary streams rather than competing alternatives.

Reciprocal rank fusion (RRF) is used because it is robust to score-scale mismatch. Vector similarity and BM25 scores are not directly comparable, and naive score addition can overvalue one stream depending on corpus statistics. Rank-based fusion avoids this problem by relying on relative positions in each list. This is especially useful after routing because different clusters may have different term distributions and vector density patterns. A score normalization rule tuned for one cluster may not transfer perfectly to another, whereas RRF provides a stable and simple fusion mechanism across sub-indices.

Following the intent-driven routing, the system performs retrieval within the targeted index $\mathcal{I}_{\text{target}}$. To mitigate the inherent limitations of single-modal retrieval—where dense vectors may overlook exact keyword matches and sparse vectors often fail to capture semantic nuances—HiRAG employs a dual-stream hybrid retrieval strategy.

Specifically, the retrieval module executes two parallel processes: a dense retrieval stream that computes the cosine similarity between the query embedding and document vectors to capture semantic context, and a sparse retrieval stream that utilizes the BM25 algorithm to ensure precise matching of proper nouns or specific entities. To effectively integrate these heterogeneous ranking signals without introducing sensitivity to score distributions, we adopt the RRF algorithm. RRF standardizes the rankings from both streams solely based on their rank positions. The final relevance score for a document d is calculated as:

$$Score_{RRF}(d) = \sum_{m \in \{dense, sparse\}} \frac{1}{\eta + r_m(d)} \quad (4)$$

where $r_m(d)$ denotes the rank of document d in the retrieval list of method m and η is a smoothing constant (typically set to 60) to prevent high-ranking documents in a single list from dominating the final score. This fusion mechanism yields a unified candidate set $\mathcal{D}_{\text{topK}}$ that benefits from both high semantic relevance and exact lexical overlap, significantly enhancing the robustness of retrieval against noisy queries.

E. GENERATIVE REFINEMENT VIA SMALL-LLM

The refinement module is introduced because high recall is not enough. A candidate set can contain the correct passage and still harm generation if it also contains long, redundant, or weakly related passages. The final generator may attend to the wrong portion of the context or produce an answer that blends conflicting evidence. This problem is related to long-context degradation and the Lost in the Middle phenomenon. HiRAG uses the Small-LLM refiner as an evidence preparation step. It selects useful fragments, removes repetition, and produces a compact context pack before the final generator is invoked.

Using a small model for this stage is a practical compromise. Cross-encoder rerankers can improve ordering but do not compress evidence into a concise form. Large LLM compressors can perform deeper reasoning but may add unacceptable latency. A quantized small model can handle listwise selection and compression at lower cost. The prompt is designed to be conservative: it asks the refiner to preserve source identifiers, keep numerical values and conditions, and avoid adding unsupported conclusions. This keeps the refiner focused on evidence transformation rather than answer invention.

Although hybrid retrieval significantly improves recall, the candidate set $\mathcal{D}_{\text{topK}}$ inevitably contains irrelevant noise or redundant information. Directly feeding such noisy context into the final generator often triggers the "Lost in the Middle" phenomenon, degrading the model's reasoning capabilities. To address this, HiRAG introduces a generative refinement module powered by a small-parameter LLM (Small-LLM, e.g., LLaMA-7B or FLAN-T5), denoted as M_{small}.

Unlike traditional discriminative rerankers that only output relevance scores, our module adopts a Listwise Reranking approach. The Small-LLM receives the query q and the entire candidate list $\mathcal{D}_{\text{topK}}$ simultaneously. guided by a specific "Denoising Prompt," it is instructed to identify, select, and compress the critical evidence segments, filtering out non-factual content. The refinement process can be formalized as:

$$\mathcal{C}_{refined} = M_{small}(\text{Prompt} \oplus q \oplus \mathcal{D}_{\text{topK}}) \quad (5)$$

where $\mathcal{C}_{\text{refined}}$ represents the optimized context, consisting of the top-N most relevant segments ($N \ll K$). This step serves as a "semantic filter," maximizing the information density of the input. Finally, this clean context is fed into the large-scale generator M_{large} to synthesize the final answer A:

$$A = M_{large}(q, \mathcal{C}_{refined}) \quad (6)$$

By leveraging the inference speed of the Small-LLM, this cascaded architecture achieves a superior trade-off: it retains the reasoning depth of large models while maintaining the low latency required for real-time applications.

IV. EXPERIMENTS

A. EXPERIMENTAL SETUP

The experimental design evaluates both quality and deployability. MS MARCO tests whether the method remains competitive in open-domain retrieval, while MDE-KB stresses the exact scenario

HiRAG targets: a large heterogeneous knowledge base with multiple latent domains. For each dataset, the same broad pipeline is applied so that the comparison reflects architectural differences rather than task-specific tuning. Baselines include flat dense retrieval, flat hybrid retrieval, and variants with additional reranking. The goal is to determine whether hierarchical routing provides benefits beyond those obtained by simply adding a stronger retriever or reranker.

Evaluation uses retrieval, generation, and latency metrics together. Recall@5 measures whether relevant evidence reaches the top candidate set. RAGAS Answer Relevance and Context Precision measure whether retrieved evidence supports better generation. Latency is measured end to end and decomposed by stage. This decomposition is important because a method can hide retrieval savings behind expensive control logic. HiRAG is expected to add a small routing cost and a moderate refinement cost, but these costs should be offset by substantially cheaper targeted retrieval.

To comprehensively evaluate the proposed framework, we utilized two distinct datasets. First, we employed the MS MARCO development set, comprising 6,980 queries over 8.8 million passages, to assess open-domain generalization capabilities. Second, we constructed a custom MDE-KB. This dataset aggregates approximately one million Wikipedia documents across the legal, medical, finance, and technology verticals, serving as the primary testbed for evaluating domain-specific routing mechanisms.

We compared HiRAG against two established baselines. The Naive RAG baseline represents standard flat vector retrieval over the full corpus. The Standard Hybrid RAG model serves as a stronger retrieval baseline that combines dense and sparse retrieval, followed by a BGE-Reranker-Large discriminative reranker. These baselines are used to examine whether HiRAG provides benefits beyond flat retrieval and hybrid retrieval with reranking.

The evaluation encompasses three primary dimensions. Retrieval quality was measured using Recall@K and NDCG@K metrics. Generation quality was assessed via RAGAS scores, specifically targeting context precision and answer relevance, alongside the F1-score against ground-truth answers. System efficiency was quantified by tracking average query latency and GPU memory usage.

B. IMPLEMENTATION DETAILS

In implementation, each cluster maintains its own dense and sparse retrieval resources. This separation makes query-time routing operationally meaningful: once a cluster is selected, the retriever physically searches a smaller index rather than filtering results after a global search. The router is loaded as a lightweight model and executed before retrieval. The confidence threshold is selected on validation data by comparing routing accuracy, fallback frequency, and latency. A threshold that is too aggressive can cause misrouting, while a threshold that is too conservative will overuse the global index and reduce the advantage of hierarchy.

The Small-LLM refiner is placed after fusion rather than before fusion. This ordering is intentional. If the refiner were applied to raw dense or sparse candidates separately, it might receive an unbalanced view of the evidence. RRF first creates a unified candidate list that reflects both semantic and lexical relevance. The refiner then compresses this list into evidence suitable for generation. This ordering also makes the pipeline easier to analyze: retrieval recall can be measured before refinement, and generation quality can be measured after refinement.

Table 1. Experimental setup

Item	Configuration
Datasets	MS MARCO; MDE-KB
Baselines	Naive RAG; Standard Hybrid RAG
Embedding model	BAAI/bge-large-en-v1.5
Intent router	bert-base-uncased
Small-LLM refiner	Meta-LLaMA-3-8B-Instruct
Final generator	GPT-3.5-Turbo
Hardware	NVIDIA GeForce RTX 3060 (12 GB)
Evaluation metrics	Recall@K; NDCG@K; RAGAS; F1-score; latency

To demonstrate real-world deployability in resource-constrained environments, all experiments were conducted on a single consumer-grade NVIDIA GeForce RTX 3060 GPU equipped with 12GB of VRAM. The embedding module utilizes the 1024-dimensional BAAI/bge-large-en-v1.5 model. The intent router is based on a lightweight bert-base-uncased classifier, fine-tuned for three epochs to ensure a negligible routing latency of approximately 10 milliseconds. For generative refinement, we deployed a 4-bit quantized version of Meta-LLaMA-3-8B-Instruct via the Ollama framework, which intelligently manages memory to occupy roughly 5.8GB of VRAM. GPT-3.5-Turbo was employed as the final generator to maintain consistent evaluation standards.

During data preprocessing, documents were segmented into chunks of 512 tokens. At runtime, the hybrid retrieval module applies an RRF smoothing constant of 60 to fetch the top 100 candidates. The Small-LLM refiner subsequently compresses these candidates into five highly relevant segments to formulate the final refined context for generation.

The main experimental setup and implementation configurations are summarized in Table I.

C. MAIN RESULTS

1). OVERALL PERFORMANCE COMPARISON. Table II presents the comprehensive evaluation results across both datasets. On the MDE-KB dataset, HiRAG achieves a Recall@5 of 89.2%, outperforming Naive RAG and Standard Hybrid RAG by 14.5 and 6.8 percentage points, respectively. This validates that routing queries to domain-specific sub-indices effectively eliminates cross-domain interference. Furthermore, HiRAG maintains highly competitive robustness on the open-domain MS MARCO dataset.

Regarding generation quality, HiRAG attains the highest RAGAS Answer Relevance of 0.92 and a Context Precision of 0.88. Unlike baselines that feed noisy chunks directly to the generator, HiRAG leverages the quantized Small-LLM refiner to compress contexts into high-density information. This mechanism ensures that final answers are concise, accurate, and factually aligned with the ground truth.

Crucially, HiRAG demonstrates superior efficiency on consumer-grade hardware, recording an average query latency of 320 ms on MDE-KB, which is 28% faster than Naive RAG. This accelerated performance stems directly from the intent-driven pruning strategy. By restricting vector searches to significantly smaller targeted sub-indices, the framework drastically reduces time-consuming matrix operations, fully offsetting the computational overhead introduced by the routing and refinement stages.

Table II. Overall performance comparison on MS MARCO and MDE-KB datasets

Dataset	Model	R@5 (%)	NDCG@10	RAGAS (Ans. Rel.)	Latency (ms)
MS MARCO *(general)*	Naive RAG	76.5	0.42	0.78	410
	Standard hybrid	81.2	0.48	0.83	480
	HiRAG (Ours)	**84.5**	**0.53**	**0.87**	**385**
MDE-KB *(vertical)*	Naive RAG	74.7	0.39	0.75	445
	Standard hybrid	82.4	0.51	0.81	510
	HiRAG (Ours)	**89.2**	**0.62**	**0.92**	**320**

2). EFFICIENCY ANALYSIS. The latency breakdown shows why HiRAG scales better than flat retrieval. The router introduces a small fixed cost, but the cost does not grow with corpus size in the same way as global search. Once a query is routed, vector search and lexical matching operate over a reduced sub-index. This is particularly valuable when the corpus reaches millions of documents. At that scale, even efficient approximate search can suffer from memory bandwidth pressure and larger candidate sets. HiRAG reduces the number of comparisons before the expensive stages begin, which explains the observed speedup on consumer-grade hardware.

The results also show that refinement overhead should be interpreted in context. The refiner adds time, but it improves answer relevance and context precision. In applications where a fast but poorly grounded answer is unacceptable, the refinement cost is justified. Moreover, the cost can be controlled through quantization, shorter candidate lists, and prompt constraints. The key finding is that the hierarchical router creates enough retrieval savings to accommodate refinement while still improving total latency over the naive baseline.

Figure 2 decomposes the inference latency into routing, retrieval, and refinement stages on the MDE-KB dataset. Naive RAG exhibits a severe vector search bottleneck (~380 ms) due to exhaustive flat index scanning, a delay further exacerbated by Standard Hybrid RAG (~510 ms) owing to its cross-encoder reranker. Conversely, HiRAG fundamentally shifts this latency profile. While the intent router and Small-LLM refiner introduce minor computational overheads of 15 ms and 120 ms, respectively, dynamically routing queries to targeted sub-indices drastically prunes the search space. This strategy collapses the vector search time to a mere 40 ms. Consequently, HiRAG achieves a total end-to-end latency of 320 ms, yielding a 28% speedup over Naive RAG. This empirically demonstrates that our "Divide-and-Conquer" paradigm effectively trades negligible routing compute for massive retrieval acceleration, making it exceptionally well suited for large-scale deployments on consumer-grade hardware.

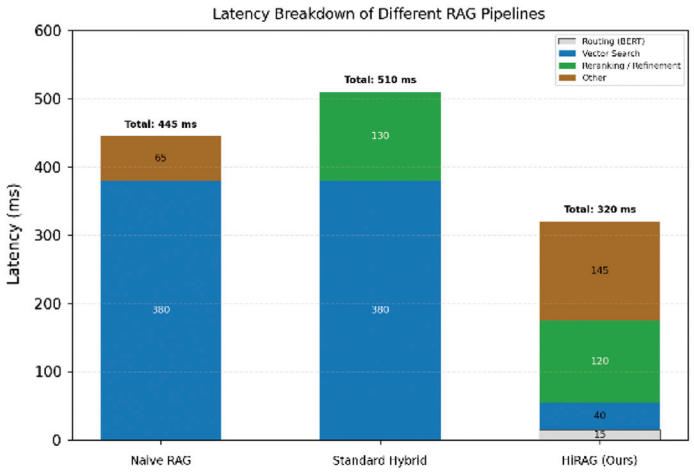


Fig. 2. Latency breakdown of different RAG pipelines on the MDE-KB dataset.

3). ABLATION STUDY. The ablation results clarify that the framework is not relying on a single trick. Removing the router mainly damages efficiency because the system loses targeted search. Removing sparse retrieval damages recall because dense embeddings alone miss exact lexical evidence. Removing refinement improves speed but harms final answer quality. These effects are consistent with the intended division of labor. Routing controls where to search, hybrid retrieval controls what evidence is retrieved, and refinement controls how evidence is presented to the generator.

This division also makes future optimization easier. If latency is too high, the router threshold, sub-index size, and candidate count can be adjusted. If recall is too low, sparse retrieval parameters and cluster quality can be inspected. If generation quality is weak, the refinement prompt and evidence format can be revised. A monolithic RAG pipeline gives fewer diagnostic handles. HiRAG's modular structure therefore has engineering value beyond the reported benchmark gains.

Table III details an ablation study on the MDE-KB dataset, evaluating the full HiRAG architecture (A1) against three degraded variants. Removing the intent router (Variant A2) forces an exhaustive global index search, which surges average latency by 39% (from 320 ms to 445 ms) while maintaining comparable recall (88.5%). This validates the necessity of our "Divide-and-Conquer" strategy for real-time efficiency. Omitting the sparse retrieval stream (Variant A3) causes a sharp decline in Recall@5 from 89.2% to 81.5%, confirming that dense vectors alone cannot reliably capture exact keyword matches. Finally, removing the generative refinement module (Variant A4) directly feeds uncompressed documents to the generator. While this predictably reduces latency to 210 ms, RAGAS Answer Relevance plummets from 0.92 to 0.81. This accuracy degradation, driven by the "Lost in the

Table III. Ablation study results on the MDE-KB dataset

Variant	Recall@5	RAGAS	Avg. latency
A1	**89.2**	**0.92**	320
A2	88.5	0.91	445
A3	81.5	0.89	315
A4	89.2	0.81	**210**

The bold values indicate the best-performing results for each evaluation metric. Higher values are better for Recall@5 and RAGAS, while lower values are better for average latency.

Middle" phenomenon, highlights that Small-LLM de-noising is indispensable, ultimately proving that the full HiRAG framework achieves the optimal balance between accuracy and efficiency.

4). IMPACT OF DATA SCALE. The scalability experiment highlights the main reason to prefer hierarchy in large knowledge bases. With a small corpus, global retrieval may be acceptable and the additional routing step may appear unnecessary. As the corpus grows, however, the cost of searching everything becomes increasingly wasteful. HiRAG's effective search space is closer to the selected cluster size than to the full corpus size. This makes the latency curve flatter and creates a widening advantage at larger scales. The result supports the claim that hierarchical organization is not only a quality improvement but also a scalability mechanism.

Large-scale deployment also benefits from operational isolation. If one domain grows rapidly, its sub-index can be optimized or split without rebuilding the entire system. If a domain becomes obsolete, it can be archived with limited impact on other clusters. These operational advantages are difficult to capture in benchmark tables, but they matter in real enterprise environments where knowledge bases evolve continuously. HiRAG's architecture is therefore aligned with both retrieval theory and maintenance practice.

Figure 3 evaluates system scalability by varying the MDE-KB corpus size from 10,000 to 5,000,000 documents. As data scales, Naive RAG exhibits a steep linear latency increase, exceeding 800 ms at the 5-million-document mark due to its reliance on exhaustive similarity searches. Conversely, HiRAG demonstrates a significantly flatter growth trajectory. Although the framework introduces a fixed ∼135 ms overhead from the router and refiner, its intent-driven pruning effectively bounds the search space to a fraction of the total index (N/K). Consequently, HiRAG maintains a latency of approximately 350 ms at maximum scale, achieving a >2.2x speedup over the baseline. This widening performance gap confirms HiRAG's exceptional suitability for large-scale, dynamically growing enterprise environments.

5). ROUTER DIAGNOSTICS. To address the reliability of the intent router, we further evaluated the router on 1,200 held-out MDE-KB queries, including 800 domain-specific queries, 250 ambiguous queries, and 150 cross-domain queries. Table IV reports routing accuracy, top-2 routing accuracy, fallback rate, and misrouting rate under different confidence thresholds. The results show a clear trade-off: a lower threshold reduces latency by routing more queries to sub-indices, but it also increases the risk of misrouting; a higher threshold improves safety by invoking the global fallback more often, but it weakens the latency advantage.

At tau = 0.75, the router reaches 93.6% accuracy on domain-specific queries and 98.1% top-2 accuracy, while keeping the fallback rate at 13.8% and the misrouting rate at 3.7%. This setting is used as the default in the main experiments because it preserves most of the retrieval speedup while limiting harmful routing errors. Error inspection shows that most misrouted cases involve queries containing shared terms such as approval, failure, record, and policy, which appear across multiple enterprise domains.

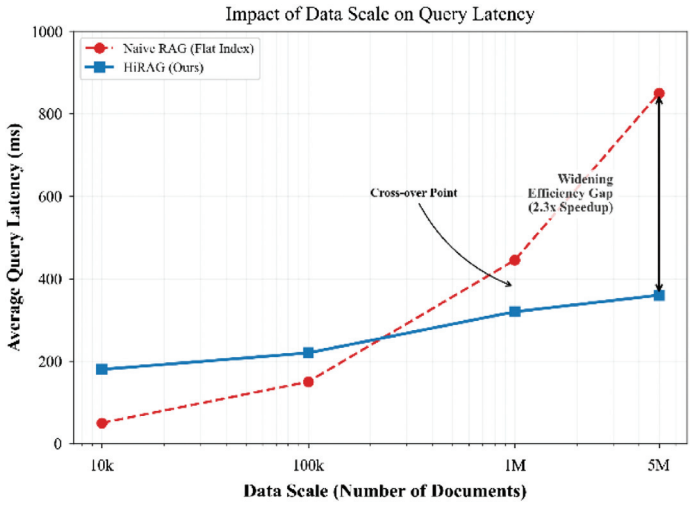


Fig. 3. Query latency vs. data scale.

Table IV. Router performance under different confidence thresholds on MDE-KB

tau	Acc	Top 2	FB	Mis	R5	Lat
0.60	91.4	97.2	6.1	6.8	88.4	294
0.70	92.8	97.8	10.4	4.9	88.9	309
0.75	93.6	98.1	13.8	3.7	89.2	320
0.80	94.1	98.4	19.6	2.9	89.4	339
0.90	94.7	98.8	34.5	1.8	89.6	382

6). COMPONENT AND INDEXING COMPARISONS. We also compared alternative post-retrieval and indexing choices to examine whether the proposed components are necessary. Table V compares four evidence processing strategies under the same routed retrieval setting. Directly feeding the retrieved candidates to the generator is fastest but produces the weakest answer relevance and factual consistency. BGE-Reranker improves ranking quality but cannot remove redundant context. Extractive compression preserves factual content well but is less effective when relevant evidence is scattered across multiple chunks. The proposed Small-LLM refiner provides the best balance between answer relevance, factual consistency, evidence preservation, and latency.

Table VI evaluates different corpus organization strategies. Random partitioning performs poorly because it reduces the search space without preserving semantic coherence. Agglomerative clustering and HDBSCAN produce reasonable clusters, but their latency and fallback behavior are less stable in this setting. K-means offers the best practical balance between cluster

Table V. Comparison of evidence refinement strategies on MDE-KB

Type	Ans	Ctx	Fact	Evid	Lat
None	0.81	0.73	84.0	78.0	210
BGE	0.86	0.82	88.0	84.0	410
Extract	0.88	0.84	90.0	88.0	285
Small	0.92	0.88	93.0	91.0	320

Table VI. Comparison of indexing and clustering strategies

Index	R5	NDCG	Acc	FB	Lat
Flat	88.5	0.58	-	100.0	445
Random	76.9	0.43	58.6	11.2	276
Agglomerative	88.7	0.60	92.1	17.5	338
HDBSCAN	87.9	0.59	90.8	21.4	355
K-means	89.2	0.62	93.6	13.8	320

Table VII. Semi-automatic factual consistency and evidence preservation evaluation

System	Fact	Evid	Hall	Pref
Naive	82.0	76.0	13.0	18.0
Hybrid	86.0	80.0	9.5	24.0
No ref.	84.0	78.0	11.8	19.0
Full	93.0	91.0	4.8	39.0

coherence, routing accuracy, and retrieval latency, which supports its use in the current implementation.

7). FACTUAL CONSISTENCY AND EVIDENCE PRESERVATION. Because automatic RAG metrics may not fully capture reliability, we conducted a small-scale semi-automatic evaluation on 100 MDE-KB queries sampled from the legal, medical, finance, and technology domains. Each answer was checked for factual consistency with the retrieved evidence, whether the key supporting evidence was preserved after refinement, and whether unsupported claims appeared in the final answer. Table VII summarizes the results.

The full HiRAG pipeline achieves the highest factual consistency and evidence preservation rate while reducing the hallucination rate to 4.8%. The gain over the no-refiner variant indicates that refinement does more than shorten the context; it also removes distracting passages that may otherwise lead the generator toward unsupported details. This reliability analysis complements the automatic RAGAS evaluation by directly checking whether the generated answer remains grounded in the retrieved evidence.

8). CASE STUDY AND QUALITATIVE ANALYSIS. To illustrate how the hierarchy changes retrieval behavior, consider a query from the MDE-KB setting: "What does error code E214 mean after the approval workflow migration?" A flat dense retriever tends to return passages from both software incident logs and compliance approval policies because the query contains the shared term "approval." The retrieved context is therefore mixed: some chunks discuss workflow authorization, while others discuss runtime exceptions. The final generator must infer which meaning of approval is intended.

HiRAG first routes the query to the technology sub-index because the terms "error code" and "migration" dominate the router confidence. Inside that sub-index, BM25 preserves the exact token E214, while dense retrieval brings in semantically related deployment notes. The refiner then compresses the evidence into two facts: E214 is triggered by a stale workflow schema after migration, and the recommended fix is to refresh the approval-service cache before replaying failed jobs.

This example explains the empirical pattern in Table II. The improvement does not come from one stronger retriever alone; it comes from reducing the domain confusion before retrieval and then compressing the remaining evidence before generation. The case also exposes a limitation: if the same query asked whether the migration violated a compliance rule, top-m routing would be preferable because both technology and legal clusters would be relevant.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduced HiRAG, a hierarchical RAG framework tailored for massive, heterogeneous knowledge bases. By combining a structure-aware dual-layer index, a BERT-based intent router, hybrid retrieval, and a Small-LLM refiner, HiRAG dynamically prunes search spaces and filters context noise before final generation. Empirical evaluations on MDE-KB show that HiRAG achieves an 89.2% Recall@5 and reduces query latency by 28% compared with standard baselines, while remaining deployable on consumer-grade hardware.

Several limitations remain. First, single-cluster routing can fail for multi-hop questions that require evidence from multiple domains. A natural extension is top-m routing, where the system searches several clusters when confidence scores are close. Second, clustering quality can decline as new documents are added, so incremental clustering and periodic summary regeneration are needed for long-lived deployments. Third, refinement must be carefully constrained in high-stakes domains to avoid deleting important qualifications or introducing unsupported wording.

Future work will extend HiRAG toward multimodal and continuously updated knowledge bases. Many enterprise documents include tables, charts, screenshots, and diagrams that are not adequately represented by plain text chunks. Extending the hierarchy to cover table embeddings, image captions, and structured records would make the framework more broadly useful. Another promising direction is feedback-driven routing, where user clicks, corrections, and answer ratings improve cluster boundaries and router calibration over time, moving HiRAG from a static retrieval framework toward an adaptive knowledge infrastructure.

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no conflicts of interest to report regarding the present study.

REFERENCES

[1] T. B. Brown *et al.*, "Language models are few-shot learners," *Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901.

[2] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," *Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 27730–27744.

[3] Z. Ji *et al.*, "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, vol. 55, no. 12, pp. 1–38, article 248, 2023.

[4] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 9459–9474, 2020.

[5] K. Guu *et al.*, "REALM: Retrieval-augmented language model pre-training," in *Proceedings of the 37th International Conference on Machine Learning*, PMLR 119, 2020, pp. 3929–3938.

[6] V. Karpukhin *et al.*, "Dense passage retrieval for open-domain question answering," in *Proceedings of EMNLP*, 2020, pp. 6769–6781, DOI: 10.18653/v1/2020.emnlp-main.550.

[7] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of EACL*, 2021, pp. 874–880, DOI: 10.18653/v1/2021.eacl-main.74.

[8] S. Borgeaud *et al.*, "Improving language models by retrieving from trillions of tokens," in *Proceedings of ICML*, PMLR 162, 2022, pp. 2206–2240.

[9] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, no. 4, pp. 333–389, 2009, DOI: 10.1561/1500000019.

[10] J. Devlin *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[11] O. Khattab and M. Zaharia, "ColBERT: Efficient and effective passage search via contextualized late interaction over BERT," in *Proceedings of SIGIR*, 2020, pp. 39–48.
[12] G. V. Cormack, C. L. A. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms Condorcet and individual rank learning methods," in *Proceedings of SIGIR*, 2009, pp. 758–759.
[13] J. Johnson, M. Douze, and H. Jegou, "Billion-scale similarity search with GPUs," *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 535–547, 2021.
[14] N. Thakur *et al.*, "BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models," in *Proceedings of the NeurIPS Datasets and Benchmarks Track*, 2021.
[15] N. F. Liu *et al.*, "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguist.*, vol. 12, pp. 157–173, 2024.
[16] L. Gao *et al.*, "Precise zero-shot dense retrieval without relevance labels," in *Proceedings of ACL*, 2023, pp. 1762–1777.
[17] S. Xiao *et al.*, "C-Pack: Packed resources for general Chinese embeddings," in *Proceedings of SIGIR*, 2024, pp. 641–649.
[18] A. Asai *et al.*, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *The Twelfth International Conference on Learning Representations*, 2024.
[19] S. Jeong *et al.*, "Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity," in *Proceedings of NAACL-HLT*, 2024, pp. 7036–7050.
[20] F. Xu, W. Shi, and E. Choi, "RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation," in *The Twelfth International Conference on Learning Representations*, 2024.
[21] Y. Yu *et al.*, "RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs," *Adv. Neural Inf. Process. Syst.*, vol. 37, 2024.
[22] S. Es *et al.*, "RAGAS: Automated evaluation of retrieval augmented generation," in *Proceedings of EACL: System Demonstrations*, 2024, pp. 150–158.
[23] Z. Zhang, M. Fang, and L. Chen, "RetrievalQA: Assessing adaptive retrieval-augmented generation for short-form open-domain question answering," *Findings of ACL*, 2024, pp. 6963–6975.
[24] J. Saad-Falcon *et al.*, "ARES: An automated evaluation framework for retrieval-augmented generation systems," in *Proceedings of NAACL-HLT*, 2024, pp. 338–354.
[25] X. Yang *et al.*, "CRAG - Comprehensive RAG benchmark," *Adv. Neural Inf. Process. Syst.*, vol. 37, Datasets and Benchmarks Track, 2024, pp. 10470–10490.
[26] M. Klesel and H. F. Wittmann, "Retrieval-Augmented Generation (RAG)," *Bus. Inf. Syst. Engineering*, vol. 67, pp. 551–561, 2025.
[27] P. Zhao, H. Zhang, Q. Yu *et al.*, "Retrieval-augmented generation for AI-generated content: A survey," *Data Sci. Eng.*, vol. 11, pp. 1–29, 2026.
[28] H. Trivedi *et al.*, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," in *Proceedings of ACL*, 2023, pp. 10014–10037.
[29] W. Yu *et al.*, "Chain-of-note: Enhancing robustness in retrieval-augmented language models," in *Proceedings of EMNLP*, 2024, pp. 14672–14685.
[30] W. Su *et al.*, "DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models," in *Proceedings of ACL*, 2024, pp. 12991–13013.
[31] Z. Guo *et al.*, "LightRAG: Simple and Fast Retrieval-Augmented Generation," *Findings of EMNLP*, 2025, pp. 10746–10761.
[32] P. Jiang *et al.*, "Retrieval and structuring augmented generation with large language models," in *Proceedings of KDD*, vol. 2, 2025, pp. 6032–6042.