A Unified Framework for Context-Aware and Relation-Aware Graph Retrieval-Augmented Generation

Anonymous Author(s)*

Abstract

Retrieval-Augmented Generation (RAG) has emerged as a paradigm for enhancing large language models (LLMs) with external knowledge, yet existing graph-based methods face a fundamental limitation: entity-centric and chunk-centric approaches operate on representations anchored to original text without true knowledge fusion. While entity-centric methods connect logically related content and chunk-centric methods preserve context, both retrieve information separately through similarity search, missing emergent understanding from their synthesis. In this paper, we propose **Hy-GRAG**, a hierarchical graph-based RAG framework that transcends source documents by addressing three core challenges: constructing summaries that genuinely integrate contextual and relational information, leveraging these synthesized representations to access emergent knowledge during retrieval, and efficiently updating hierarchical structures for dynamic corpora. Specifically, we design hierarchical index structures over hybrid graphs with both chunk and entity nodes, then iteratively cluster them and generate LLM-based summaries. Then, we design context and relation-aware retrieval that searches across all abstraction levels while expanding through community membership. Moreover, we enable dynamic knowledge update through attachment-based algorithms with only local re-summarization. Experimental results show that **HyGRAG** improves the average accuracy of multi-hop reasoning tasks by 9.7%, with less than a 2% decrease in performance on corpus expansion tasks, while maintaining reasonable efficiency. [1]

CCS Concepts

• **Networks** → **Online social networks**; • **Computing methodologies** → *Knowledge representation and reasoning*.

Keywords

Graph RAG, context-aware, relation-aware.

[1]Our codes are available at https://anonymous.4open.science/r/HyGRAG/.

Figure 1: Illustration of method performance using Qwen3-8B across relation-aware (MuSiQue) and context-aware (MultiHop-RAG) datasets, including representative cases.

1 Introduction

Retrieval-Augmented Generation (RAG) systems have been developed to enhance large language models (LLMs) by integrating external knowledge sources [4][15]. This integration allows LLMs to access up-to-date information and domain-specific knowledge, addressing the inherent limitations of parametric memory [3]. Recent advances in RAG have moved beyond simple document retrieval to incorporate graph structures that explicitly model relationships between concepts, entities, and documents [9][25]. These graph-based RAG (GraphRAG) systems leverage the rich structural information in knowledge graphs to improve both retrieval accuracy and reasoning capabilities, making them particularly effective for complex queries that require understanding interconnected information [16][7].

Current GraphRAG methods have explored two distinct directions to harness graph structures for retrieval augmentation. Entity-centric approaches such as GraphRAG [2], HippoRAG [7], and Hi-RAG [10] build knowledge graphs where nodes represent entities extracted from text and edges encode their semantic relationships. These methods excel at multi-hop reasoning by enabling traversal along relational paths—for instance, answering "Which company acquired the developer of GPT-3?" by following edges from GPT-3 to OpenAI to Microsoft. In contrast, chunk-centric methods like RAPTOR [20] and EraRAG [28] organize text chunks into hierarchical structures based on semantic similarity. They create tree-like indexes where leaf nodes contain original text chunks and parent

nodes store increasingly abstract summaries. This design preserves full contextual information while enabling retrieval at different levels of granularity.

However, both approaches exhibit fundamental limitations that restrict their effectiveness. Entity-centric methods suffer from information loss during the entity extraction process. Named entity recognition and relation extraction models introduce errors that compound throughout the system, and the abstraction from text to entities discards valuable contextual information needed for factual question answering. Our analysis shows that these methods often underperform simple dense retrieval on factual QA benchmarks. Chunk-centric methods face the opposite problem: while they maintain high accuracy on factual queries, they cannot capture explicit relationships between entities scattered across different chunks. This makes them ineffective for queries requiring logical reasoning, such as finding all products affected by a specific supply chain disruption. The fundamental issue is that existing methods optimize for either contextual completeness or relational structure, but real-world queries often require both.

Our key insight is that simply combining chunk and entity representations in a hybrid graph does not solve the fundamental problem: both representations remain anchored to their original textual sources without true knowledge fusion. While knowledge graphs can connect logically related content that is textually distant, they cannot integrate the logical information from relations with the background context from chunks. During retrieval, these two aspects are still accessed separately through similarity search, missing the opportunity for synergistic understanding. Consider a query about "the impact of renewable energy adoption on manufacturing industries"—entity-based retrieval might find relationships between solar panels and factories, while chunk-based retrieval might return documents about energy costs, but neither captures the emergent understanding that comes from synthesizing these perspectives. This motivates our approach: we cluster entities and chunks together and generate summaries that simultaneously encode both relational logic and contextual background, creating new knowledge representations that transcend the original text.

Implementing this knowledge fusion approach presents three core technical challenges. First, how to construct summaries that genuinely integrate context and relations rather than simply concatenating them. Second, how to leverage these hybrid summaries during retrieval to access knowledge beyond the original corpus. The summaries represent emergent understanding that doesn't exist in any single source document—the retrieval mechanism must be able to match queries to these synthesized insights while still providing access to supporting details. Third, how to efficiently update such complex hierarchical structures when the corpus changes. Real-world knowledge bases receive continuous updates, but rebuilding the entire hierarchy for each new document would be computationally prohibitive.

To address these challenges, we propose **HyGRAG**, a GraphRAG framework with three key innovations. For summary construction, we design a two-stage approach: first clustering nodes using graph structure-aware embeddings that capture both textual similarity and relational connectivity, then prompting LLMs with structured templates that explicitly request synthesis of contextual and relational information within each community. This produces summaries that represent genuine knowledge fusion rather than simple aggregation. For retrieval, we implement a bi-level mechanism that operates across both context and relation dimensions. Context-aware retrieval searches across all hierarchy levels—from specific chunks to abstract community summaries—enabling access to emergent knowledge. Relation-aware retrieval then expands results by collecting entities from retrieved communities and filtering their associated triplets, ensuring logical completeness. For dynamic updates, we develop an attachment-based algorithm where new content is matched to the most similar community at the appropriate abstraction level, with local re-summarization propagating only along affected paths. This preserves the stability of unrelated portions while incorporating new knowledge. Experimental results show that **HyGRAG** improves the average accuracy of multi-hop reasoning tasks by 9.7%, with less than a 2% decrease in performance on corpus expansion tasks, while maintaining reasonable efficiency.

In summary, we make three contributions:

- We propose hierarchical indexing for GraphRAG that unifies context and relational information through multi-level abstraction, enabling retrieval across different granularities.
- We develop a retrieval mechanism that combines context-aware and relation-aware strategies, leveraging community structures to find information that flat approaches miss.
- We achieve significant performance gains across a range of tasks, including a 6.2% improvement in Factual Accuracy and a 9.7% increase in Multi-Hop Reasoning, with improvements reaching up to 12.2% on the HotpotQA dataset.

2 Related Work

2.1 Graph-based RAG

Retrieval-Augmented Generation (RAG)[4] mitigates hallucinations[11] in large language models by integrating external knowledge during generation. RAG involves two coupled steps: retrieval and generation. Given a query, the system retrieves the top-k relevant text chunks from a preprocessed external corpus. The LLM then generates an answer conditioned on both the retrieved chunks and the query, producing more factual and contextually appropriate responses.

Vanilla RAG[14] improves factual grounding but treats the corpus as unstructured text, limiting inter-document reasoning and multi-hop inference. To overcome this, recent work incorporates graph structures into retrieval[19], giving rise to graph-based RAG methods. These can be categorized as Context-aware, focusing on text chunks, and Relation-aware, explicitly leveraging graph structures.

Context-aware Methods enhance generation by retrieving segmented text chunks as context. Vanilla RAG retrieves top-k chunks based on embedding similarity. Advanced approaches cluster and summarize chunks to capture high-order information. RAPTOR[20] recursively clusters chunks into hierarchical summaries, while EraRAG[28] organizes the corpus into multi-layered summaries

using LSH, efficiently handling high-order information in dynamic corpora.

Relation-aware Methods construct textual knowledge graphs to encode structured relationships. LightRAG[6] integrates graph structures into the text indexing ,and retrieve by high/low-level keywords. GraphRAG[2] cluster communities and summarize them to provide holistic context. HiRAG[10] and ArchRAG[24] further leverage hierarchical graphs to bridge local entity details and global insights by applying attributed entities. HippoRAG[7] and HippoRAG 2[8] enhance multi-hop reasoning by navigating graph connectivity via Personalized PageRank, improving performance on complex QA tasks.

2.2 Dynamic Retrieval

Most graph-based RAG systems assume a static corpus, making updates costly. LightRAG[6] uses a modular retriever to add new documents dynamically. HippoRAG[7] also supports incremental updates, as it leverages the KG only for retrieval. EraRAG[28] introduces LSH-based clustering, enabling localized updates by re-segmenting and re-summarizing only affected parts. However, entity-based community clustering methods still require full reconstruction.

3 Method

Overview. Our proposed framework, **HyGRAG**, addresses the fundamental trade-off between context-aware and relation-aware retrieval through a unified architecture. The system consists of four key components: (1) **Hierarchical Index Structure Construction** constructs a hierarchical indexing structure with hybrid graph that captures both chunk-based contextual information and entity-based relational information; (2) **Context and Relation-Aware Retrieval** performs retrieval across micro and macro granularities; (3) **Retrieval-Augmented Efficient Generation** ensemble heirarchical context; (4) **Dynamic Knowledge Update** supports incremental knowledge integration without full reconstruction.

3.1 Hierarchical Index Structure Construction

This module operates entirely in the offline phase, ensuring no impact on online retrieval efficiency. We construct a hybrid knowledge graph from raw text that simultaneously captures chunk-based contextual information and entity-based relational structures.

Text Chunking and Chunk-Level Graph Construction. Given a corpus $\mathcal{D}$, we first segment it into overlapping text chunks $C = \{c_1, c_2, \ldots, c_n\}$. Each chunk c_i preserves the contextual information of the original text, enabling inference LLMs to better understand its semantic content. For each chunk, we compute its embedding representation:

$$\mathbf{e}_i^c = \text{LM}_{\text{Embedding}}(c_i), \tag{1}$$

where we employ BGE-M3 as the embedding model to encode text chunks into dense vector representations. Based on embedding similarity, we construct a chunk-based graph $\mathcal{G}^c = (\mathcal{V}^c, \mathcal{E}^c)$, where $\mathcal{V}^c = C$. We create edges between chunks based on shared entities rather than shallow embedding similarity. Specifically, if two chunks c_i and c_j share more than l common entities, we create an edge $(c_i, c_j) \in \mathcal{E}^c$:

$$(c_i, c_j) \in \mathcal{E}^c \Leftrightarrow |\mathcal{E}_i \cap \mathcal{E}_j| > l, \tag{2}$$

where $\mathcal{E}_i$ and $\mathcal{E}_j$ are the entity sets extracted from chunks c_i and c_j respectively, and l is set to 3 in our implementation. This approach establishes meaningful connections between chunks through shared entities, providing stronger semantic relationships than surface-level embedding similarity.

Entity-Level Graph Construction. Then we employ LLMs to extract knowledge graph triplets from the raw text. Specifically, for each text chunk c_i, the LLM identifies entities and their relationships, generating a set of triplets:

$$\mathcal{T}_i = \{(h, r, t) \mid h, t \in \mathcal{E}_i, r \in \mathcal{R}_i\}, \tag{3}$$

where $\mathcal{E}_i$ and $\mathcal{R}_i$ are the entity and relation sets extracted from c_i, respectively. By merging triplets from all chunks, we obtain the global entity set $\mathcal{V}^e = \bigcup_{i=1}^n \mathcal{E}_i$ and relation set $\mathcal{E}^e = \bigcup_{i=1}^n \mathcal{T}_i$, forming the entity-based knowledge graph $\mathcal{G}^e = (\mathcal{V}^e, \mathcal{E}^e)$.

Hybrid Graph Construction. To bridge the gap between chunk-based context and entity-based relations, we connect the two graphs through membership relationships. For each entity $e \in \mathcal{V}^e$, we identify the set of text chunks containing that entity:

$$C(e) = \{c_i \in C \mid e \text{ is mentioned in } c_i\}. \tag{4}$$

We create cross-layer edges $\mathcal{E}^{ce}$ connecting entities to their containing chunks:

$$\mathcal{E}^{ce} = \{(e, c) \mid e \in \mathcal{V}^e, c \in C(e)\}. \tag{5}$$

The final hybrid knowledge graph is defined as:

$$\mathcal{G}_{\text{hybrid}} = (\mathcal{V}^c \cup \mathcal{V}^e, \mathcal{E}^c \cup \mathcal{E}^e \cup \mathcal{E}^{ce}). \tag{6}$$

This graph contains two types of nodes (chunk nodes and entity nodes) and three types of edges (chunk-to-chunk edges, entity-to-entity edges, and cross-layer edges).

Hierarchical Indexing. While the hybrid graph enriches information representation, it also increases the complexity of online retrieval. To efficiently retrieve both relevant contextual and relational information simultaneously, we propose to construct a hierarchical tree-structured index for the hybrid graph.

First, we use Cleora to generate structure-aware embeddings for all nodes $v \in \mathcal{V}^c \cup \mathcal{V}^e$ in the hybrid graph:

$$\mathbf{z}_v = \text{Cleora}(\mathcal{G}_{\text{hybrid}}, v). \tag{7}$$

Then, we employ hyperplane-based Locality-Sensitive Hashing (LSH) for efficient clustering. For each node embedding $\mathbf{z}_v \in \mathbb{R}^d$, we randomly sample k hyperplanes $\{\mathbf{h}_1, \ldots, \mathbf{h}_k\}$ and compute its hash code:

$$\text{hash}(\mathbf{z}_v) = [\text{sign}(\mathbf{z}_v \cdot \mathbf{h}_1), \ldots, \text{sign}(\mathbf{z}_v \cdot \mathbf{h}_k)] \tag{8}$$

Nodes with similar hash codes (small Hamming distance) are assigned to the same bucket. For each bucket B, if its size satisfies $S_{\min} \leq |B| \leq S_{\max}$, it is retained; otherwise, split or merge operations are performed:

$$B_{\text{adjusted}} = \begin{cases} \text{Split}(B) & \text{if } |B| > S_{\max} \\ \text{Merge}(B, B_{\text{neighbor}}) & \text{if } |B| < S_{\min} \\ B & \text{otherwise} \end{cases} \tag{9}$$
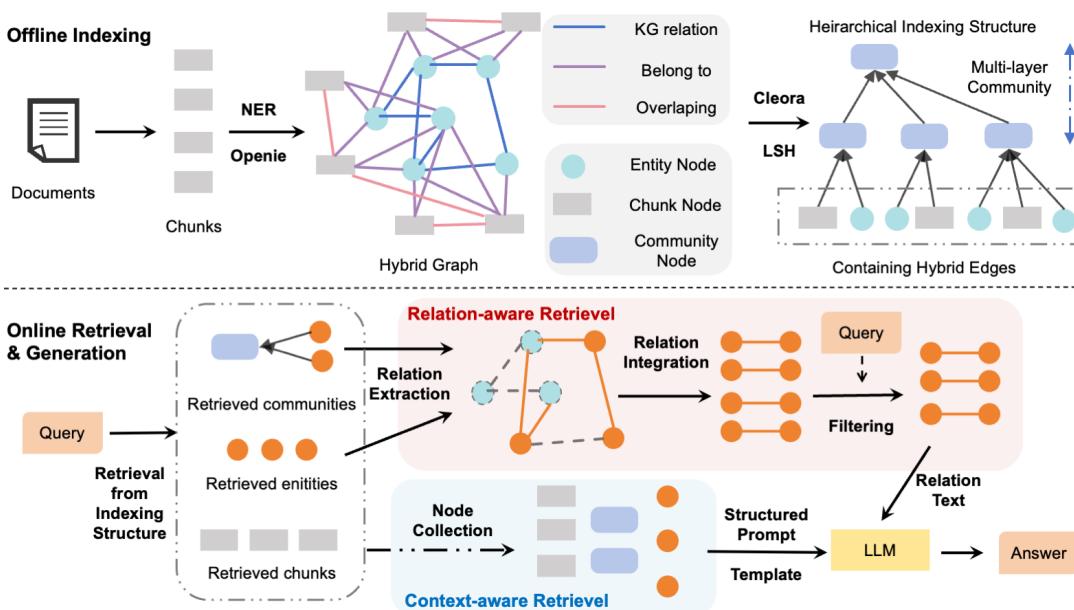
Figure 2: Overall architecture of HyGRAG.

For each adjusted bucket (forming a community) C, we generate summary representations through a two-step process:

$$t_C = \text{LLM}_{\text{summarize}}(\{v \mid v \in C\}), \tag{10}$$

$$s_C = \text{LM}_{\text{Embedding}}(t_C), \tag{11}$$

where we utilize Llama3.1-8B-Instruct to generate concise textual summaries t_C that capture the semantic essence of nodes within each community, and then employ BGE-M3 to encode these summaries into dense vector representations s_C.

We treat summary nodes as nodes in the next layer and recursively apply the above hashing, bucketing, and summarization process up to a predefined number of layers. Let $\mathcal{L}_\ell$ denote the node set at layer ℓ, where $\mathcal{L}0 = \mathcal{V}^c \cup \mathcal{V}^e$ represents the leaf nodes. The hierarchical construction continues until the layer index ℓ reaches the preset maximum depth L, resulting in an L-layer hierarchical index tree:

$$\mathcal{T}\text{index} = \mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_L. \tag{12}$$

This hierarchical tree serves as an efficient retrieval database during the dual-stage retrieval phase, where leaf nodes correspond to original nodes from the hybrid graph and upper-layer nodes contain semantic summaries at coarser granularities, supporting multi-scale context and relation-aware retrieval.

3.2 Context and Relation-Aware Retrieval

During the online retrieval phase, we employ a bi-level approach that captures information at both context and relation levels to provide comprehensive and contextually rich results.

Query Encoding. Given a user query q, we first encode it using the same embedding model employed during indexing:

$$\mathbf{q} = \text{LM}_{\text{Embedding}}(q), \tag{13}$$

where we utilize BGE-M3 to generate the query embedding $\mathbf{q} \in \mathbb{R}^d$ that serves as the basis for similarity computation across all retrieval levels.

Context-Aware Retrieval. From the hierarchical indexing structure $\mathcal{T}_{\text{index}}$ containing community, chunk, and entity nodes, we first perform context-aware retrieval by computing similarity scores across all node types. We retrieve the top-k most similar nodes for each type:

$$\mathcal{P}_{\text{retrieved}} = \text{TopK}(\{C \in \bigcup_{\ell=1}^{L} \mathcal{L}_\ell \mid \text{sim}(\mathbf{q}, s_C)\}, k), \tag{14}$$

$$C_{\text{retrieved}} = \text{TopK}(\{c \in \mathcal{V}^c \mid \text{sim}(\mathbf{q}, \mathbf{e}_c^c)\}, k), \tag{15}$$

$$\mathcal{E}_{\text{retrieved}}^{\text{context}} = \text{TopK}(\{e \in \mathcal{V}^e \mid \text{sim}(\mathbf{q}, \mathbf{z}_e)\}, k). \tag{16}$$

These retrievals capture content most relevant to the query from a contextual perspective across different granularities.

Relation-Aware Retrieval. To capture logical relationships, we construct a comprehensive entity set by combining entities from retrieved communities and context-based entity retrieval:

$$\mathcal{E}_{\text{all}} = \mathcal{E}_{\text{retrieved}}^{\text{context}} \cup \bigcup_{C \in \mathcal{P}_{\text{retrieved}}} \{e \mid e \in \mathcal{V}^e, (e, C) \in \mathcal{E}^{ce}\}. \tag{17}$$

We then extract all relations connected to these entities from leaf nodes:

$$\mathcal{R}_{\text{all}} = \{(h, r, t) \in \mathcal{E}^e \mid h \in \mathcal{E}_{\text{all}} \vee t \in \mathcal{E}_{\text{all}}\}. \tag{18}$$

Since $|\mathcal{R}_{\text{all}}|$ can be large, potentially introducing noise and LLM overhead, we filter by computing triplet embeddings and selecting the top-k most relevant:

$$\mathcal{R}_{\text{retrieved}} = \text{TopK}(\{(h, r, t) \in \mathcal{R}_{\text{all}} \mid \mathbf{z}_{(h,r,t)} = \text{LM}_{\text{Embedding}}(h \oplus r \oplus t)\}, k), \tag{19}$$

where $\oplus$ denotes concatenation.

Integrated Retrieval Strategy. The final retrieval combines context-aware and relation-aware perspectives, yielding four sets of k elements each:

$$\mathcal{R}_{\text{final}} = \{\mathcal{P}_{\text{retrieved}}, C_{\text{retrieved}}, \mathcal{E}_{\text{retrieved}}^{\text{context}}, \mathcal{R}_{\text{retrieved}}\}, \qquad (20)$$

providing community summaries for high-level understanding, chunks for detailed context, entities for key concepts, and relations for logical connections.

Computational Complexity Analysis. Similar to other retrieval-augmented approaches, **HyGRAG** employs a FAISS vector store with HNSW indexing to efficiently retrieve the top-k most similar items. Let N denote the total number of indexed nodes and d the embedding dimension. The HNSW-based search achieves logarithmic complexity for each index: $O(\log N_i \cdot d)$, and retrieving from entity-, chunk-, and community-level indexes results in:

$$O((\log N_e + \log N_c + \log N_p) \cdot d) = O(\log N \cdot d). \qquad (21)$$

For each retrieved entity, relation extraction and relevance scoring introduce an additional cost of $O(k_e \cdot \bar{d} \cdot d)$, where $\bar{d}$ is the average node degree. The overall retrieval complexity is thus:

$$O(\log N \cdot d + k_e \cdot \bar{d} \cdot d), \qquad (22)$$

which maintains sub-linear scaling with respect to graph size while enabling multi-granularity and relation-aware retrieval.

Compared with traditional chunk-aware RAG systems that only perform a single chunk-level retrieval $O(\log N_c \cdot d)$, our method introduces additional but lightweight costs from entity and community retrievals $O((\log N_e + \log N_p) \cdot d + k_e \cdot \bar{d} \cdot d)$.

3.3 Retrieval-Augmented Efficient Generation

Given the retrieved nodes from bi-level retrieval, we extract and organize information for LLM generation. For community nodes $C \in \mathcal{P}_{\text{retrieved}}$, we obtain their summaries t_C. For entity nodes $e \in \mathcal{E}_{\text{retrieved}}^{\text{context}}$, we extract their textual representations. For relation triplets $(h, r, t) \in \mathcal{R}_{\text{retrieved}}$, we preserve their structured format. For chunk nodes $c \in C_{\text{retrieved}}$, we directly use their textual content.

We combine these four information types using a structured prompt template:

$$\text{Context} = \mathcal{P}(\{t_C\}_{C \in \mathcal{P}_{\text{retrieved}}}, \mathcal{E}_{\text{retrieved}}^{\text{context}}, \mathcal{R}_{\text{retrieved}}, C_{\text{retrieved}}), \quad (23)$$

where $\mathcal{P}$ organizes community summaries, entities, relation triplets, and chunk contexts hierarchically.

The final response is generated by:

$$y = \text{LLM}_{\text{generate}}(q, \text{Context}), \qquad (24)$$

leveraging community summaries for high-level understanding, entities for key concepts, relation triplets for logical reasoning, and chunk contexts for detailed background information.

3.4 Dynamic Knowledge Update

In real-world scenarios, knowledge corpora evolve continuously, requiring efficient update mechanisms. Our clustering-based hierarchical structure enables fast integration of new content without full graph reconstruction.

Update Processing. Given new text content d_{new}, we first segment it into chunks $C_{\text{new}} = \{c_{\text{new}}^1, \ldots, c_{\text{new}}^m\}$ and extract knowledge graph triplets:

$$\mathcal{T}_{\text{new}} = \bigcup_{i=1}^{m} \{(h, r, t) \mid h, t \in \mathcal{E}_i^{\text{new}}, r \in \mathcal{R}_i^{\text{new}}\}. \qquad (25)$$

We generate a summary representation for the new content:

$$t_{\text{new}} = \text{LLM}_{\text{summarize}}(C_{\text{new}} \cup \mathcal{T}_{\text{new}}), \qquad (26)$$

$$\mathbf{s}_{\text{new}} = \text{LM}_{\text{Embedding}}(t_{\text{new}}). \qquad (27)$$

Hierarchical Attachment. We traverse the hierarchical index from bottom to top. Starting at layer $\mathcal{L}_1$, we find the most similar community:

$$C^* = \arg\max_{C \in \mathcal{L}_1} \text{sim}(\mathbf{s}_{\text{new}}, s_C). \qquad (28)$$

If $\text{sim}(\mathbf{s}_{\text{new}}, s_{C^*}) > \tau_{\text{attach}}$, we attach the new content to C^*. Otherwise, we proceed to the next layer:

$$\ell^* = \min\{\ell \mid \exists C \in \mathcal{L}_\ell : \text{sim}(\mathbf{s}_{\text{new}}, s_C) > \tau_{\text{attach}}\}. \qquad (29)$$

Upon attachment at layer ℓ^*, we update all ancestor community summaries along the path to the root:

$$\forall j \in \{\ell^*, \ldots, L\}, \quad t_{C_j} \leftarrow \text{LLM}_{\text{summarize}}(\text{Children}(C_j)), \qquad (30)$$

$$s_{C_j} \leftarrow \text{LM}_{\text{Embedding}}(t_{C_j}). \qquad (31)$$

Additionally, we establish connections between new nodes and existing leaf nodes following the original hybrid graph construction rules, creating edges when entities are shared or semantic similarity exceeds thresholds. Since all updates occur offline, online retrieval efficiency remains unaffected.

4 Experiments

In this section, we answer the following questions to validate the effectiveness of our method:

- **RQ1.** How does **HyGRAG** perform compared with existing baselines on different type of static QA tasks?
- **RQ2.** How efficient is the **HyGRAG** approach?
- **RQ3.** How robust is **HyGRAG** in corpus expansion?
- **RQ4.** What is the quality of our communities and relations, and how does it impact the performance of RAG?

4.1 Experimental Setup

4.1.1 Datasets and Metrics. For static QA, we adopt five datasets: **PopQA** [17] (factual accuracy), **MultiHop-RAG** [21], **MuSiQue** [22], and **HotpotQA** [27] (multi-hop reasoning), and **QuALITY** [18] (reading comprehension).

For evaluation, static datasets report **Accuracy** and **Recall** (QuALITY uses Accuracy only).

4.1.2 Baseline. We compare our approach with three categories of RAG baselines: (1) LLM-only: **ZeroShot**, **Chain-of-Thought (CoT)**[12]; (2) Context-aware: **VanillaRAG**[14], **RAPTOR**[20] (GMM/K-means clustering algorithm), and **EraRAG**[28]; (3) Relation-aware: **L-GraphRAG**[2](local mode of Microsoft GraphRAG), **HippoRAG**[7], **HippoRAG2**[8], **HiRAG**[10], **ArchRAG**[24] and **LightRAG**[6]. Variants of LightRAG, namely Local, Global, and Hybrid, are denoted as **L/G/H-LightRAG** for brevity.

4.1.3 Implementation Details. All static-query experiments use Llama-3.1-8B-Instruct[5] as backbone under the vLLM[13] engine with greedy decoding and top-k=5 retrieval. For a fair comparison, we maintain retrieval sizes across all node types, i.e.,entity nodes = community + chunk nodes = relations = top-k. Embeddings are generated by BGE-M3[1], a SOTA embedding model that supports both multilingual and multi-granularity retrieval, splitting text into 1,200-token chunks with 100-token overlap. Baselines follow the framework[30] or official code with default hyperparameters. Runs exceeding two days are marked as *OOT*.

Table 1: Dataset Statistics

Dataset	Tokens	Questions	of Chunks	Avg. Tokens
PopQA	3,896,179	1,399	33,595(32,158)	121.16
Quality	1,521,377	4,609	265(1,518)	1084.25
MultihopQA	1,424,248	2,556	609(1,658)	920.62
MusiqueQA	3,134,496	3,000	29,898	104.84
HotpotQA	8,139,124	3,702	66,581(66,555)	122.29

4.2 Static QA Performace

To answer the question RQ1, we present the experimental results of question answering (QA) on various types of static queries, including factual accuracy, multi-hop reasoning, and reading comprehension tasks. Our method is systematically compared with a range of baseline models to assess its effectiveness across different reasoning scenarios. Furthermore, the statistics of the HybridGraph across different datasets are summarized in the Appendix B.

4.2.1 QA results. As shown in Table 2, different categories of baseline methods exhibit substantial variations in static QA performance. LLM inference-only approaches (Zero-shot and CoT) perform poorly across all datasets, with accuracies significantly lower than retrieval-based methods. This indicates that relying solely on the parametric knowledge of LLMs is insufficient for factual and reasoning-intensive tasks.

Among context-level methods, VanillaRAG, RAPTOR, and EraRAG achieve comparable performance on factual accuracy and multi-hop reasoning datasets. This suggests that the advantage of high-order community structures is not always benificial, and the effectiveness largely depends on whether multi-hop content is aggregated into the same community during the offline indexing stage. On the Quality dataset, however, RAPTOR and EraRAG exhibit clear advantages, with EraRAG achieving the best overall performance. We attribute this to EraRAG's retrieval strategy, which enforces the selection of one non-leaf node and four leaf nodes, thereby reducing irrelevant information and facilitating semantic understanding by the LLM. Overall, RAPTOR slightly outperforms RAPTOR-K and EraRAG, mainly due to its use of a superior but expensive clustering method, though the improvements are marginal.

For relation-level methods, the LightRAG variants demonstrate divergent strengths. L-LightRAG achieves the highest recall (39.97%) on PopQA, showing that relation-aware retrieval can effectively support LLMs in factual accuracy tasks. In contrast, G-LightRAG and H-LightRAG perform relatively poorly, likely because global higher-level concepts are less suited for highly specific queries. ArchRAG

and HiRAG, as improved versions of L-GraphRAG, refines reasoning path generation by leveraging semantically similar summary entities. HiRAG achieves better performance than L-GraphRAG in multi-hop reasoning, though still leaving room for improvement. HippoRAG and its improved version, HippoRAG2, improving retrieval performance through relations, exhibit competitive results on tasks such as MultihopQA and PopQA but consistently fall short of our method in terms of overall accuracy.

Overall, **HYGRAG** achieves the best or near-best performance across all tasks. On PopQA, it reaches 72.34% accuracy and 43.51% recall, outperforming other methods by 6.2% and 8.9%, respectively. On the Quality dataset, it achieves the second-best result. In multi-hop reasoning tasks, **HYGRAG** attains 65.41% accuracy on MultihopQA, a 6.0% improvement over the strongest baseline(the metric in parentheses denotes results without explicitly prompting the model to state "Insufficient Information" when possible), and shows stable advantages on MusiqueQA and HotpotQA, with an average gain of approximately 11.1%. With the addition of inference augmentation (+Inference), which add a rule in prompt to encourage the model to leverage our domain knowledge for reasoning, MusiqueQA and HotpotQA further improve by 2.0% and 0.4%, respectively. These results demonstrate that **HYGRAG** consistently delivers robustness and superior effectiveness across factual QA, reading comprehension, and multi-hop reasoning tasks through a synergistic integration of relation and context. In addition, we record in the Appendix C.2 the results of further analyses using different dense retrievers and replacing the base LLM with Qwen3-8B[26]. The detailed results are shown in Table 9 and Table 10. We observe that across various embedding models, **HYGRAG** consistently maintains stable and competitive performance, indicating that its advantages are not limited to a specific dense retriever.

When replacing the base LLM with the newer model Qwen3-8B[26] (using a consistent *no-thinking* mode), the overall trend remains consistent with previous analyses. Notably, we observe an unexpected performance drop of context-aware methods on the MusiqueQA and HotpotQA datasets after replacing the base LLM. In contrast, **HYGRAG** maintains competitive results, achieving up to 10% higher accuracy than other relation-aware approaches. This demonstrates its robustness and adaptability across different model architectures and reasoning paradigms.

*4.2.2 Efficiency of **HYGRAG**.* To answer the question RQ2, we compare the time cost and token usage of **HYGRAG** with those of other baseline methods. As shown in Figure 3, **HYGRAG** demonstrates significant time and cost efficiency especially among relation-aware methods for online queries. However, compared with context-aware methods, the cost of **HYGRAG** is higher.
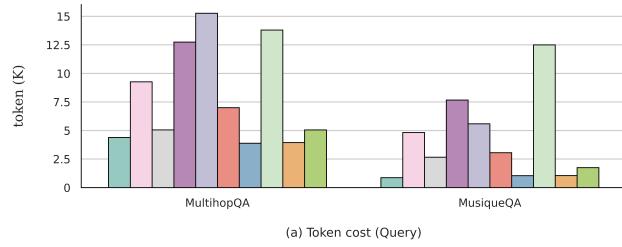
4.2.3 Robustness to Corpus Expansion. As RAG systems are increasingly deployed in real-world applications, they must become more adaptable to scenarios of continuous learning where the retrieval corpus keeps expanding. To answer the question RQ3, in order to evaluate the ability of **HYGRAG** to handle incremental corpus insertions, we designed an experiment that divides the initial corpus into different proportions and then incrementally inserts the remaining corpus as updates. We then tested the QA performance under varying proportions of incremental corpus, and the results are shown in Figure 5.

Table 2: Overall QA results(Accuracy and Recall) on static RAG query datasets using Llama-3.1-8B-Instruct.
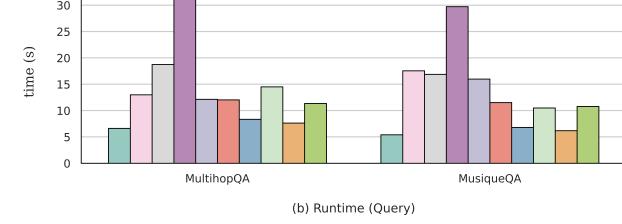
Method	Factual Accuracy		Reading Comprehension			Multi-Hop Reasoning			
	PopQA		Quality	MultihopQA		MusiqueQA		HotpotQA	
	Accuracy	Recall	Accuracy	Accuracy	Recall	Accuracy	Recall	Accuracy	Recall
Zero-shot	15.37	5.62	33.54	27.93	20.84	8.76	19.19	27.36	32.97
CoT	16.03	5.58	33.83	28.13	21.01	8.53	18.95	27.77	38.88
VanillaRAG	66.98	31.64	56.38	58.49	37.52	32.3	43.92	60.97	64.24
RAPTOR	66.33	30.98	58.08	58.72	37.88	31.17	43.47	OOT	OOT
RAPTOR-K	66.26	30.86	58.34	57.82	37.45	31.67	43.91	61.24	65.19
EraRAG	61.40	29.94	60.79	58.67	37.89	26.70	38.93	61.03	65.22
L-LightRAG	67.05	39.97	48.05	53.05	37.38	30.63	43.66	OOT	OOT
G-LightRAG	23.52	11.26	30.53	52.15	35.42	16.01	27.73	OOT	OOT
H-LightRAG	47.46	25.91	38.60	52.86	36.84	18.73	29.41	OOT	OOT
HippoRAG	55.18	23.35	46.10	58.88	31.36	20.00	32.54	51.89	56.31
HippoRAG2	68.12	38.77	45.98	61.66	33.72	30.23	39.23	OOT	OOT
L-GraphRAG	54.82	30.89	50.01	53.01	36.78	14.57	28.72	OOT	OOT
HiRAG	53.18	30.47	55.83	54.00	38.09	26.20	40.70	OOT	OOT
ArchRAG	60.64	33.44	37.77	30.81	14.04	14.27	29.22	OOT	OOT
HyGRAG	**72.34**	**43.51**	59.49	**73.79 (65.41)**	**44.05 (39.44)**	**35.87**	**48.06**	68.72	70.79
+Inference	72.38	43.62	59.79	74.33	44.09	36.6	48.36	69.10	71.08
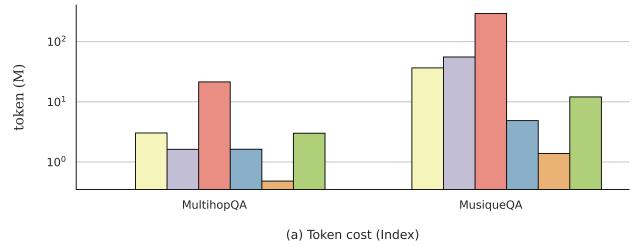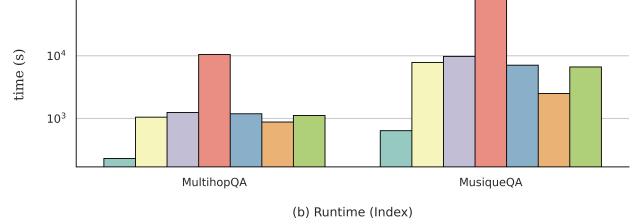
Figure 3: Comparison of query efficiency.

Figure 4: Corpus expansion indexing cost.

From the results, we observe that the larger the initial corpus proportion, the better the final QA performance. This is mainly because, compared to static construction, incremental corpus insertion leads to a slight decrease in community quality. However, this decrease is minor—around 1–2%—which demonstrates the capability of **HyGRAG** to handle continual learning scenarios effectively. In addition, we measured the reconstruction time and token consumption required by various baseline methods when 20% of the corpus is inserted, as shown in Figure 4. The results show that our proposed **HyGRAG** achieves favorable efficiency and competitive performance, indicating its strong practicality in dynamic corpus environments.

4.3 Detailed Analysis

To answer **RQ4**, we conduct detailed analyses to investigate how each component of **HyGRAG** cooperatively affects the overall performance of the system.

4.3.1 Ablation Study. Table 3 presents the results of our ablation studies conducted on multiple datasets. To assess the individual contributions of each retrieval strategy, we progressively removed specific components and examined the resulting performance changes.

As shown in Table 3, removing the chunk-level structure leads to the most pronounced degradation across all datasets, highlighting its crucial role in providing fundamental contextual grounding and factual support. Similarly, eliminating entity and relation information causes a decline in accuracy, confirming the importance of
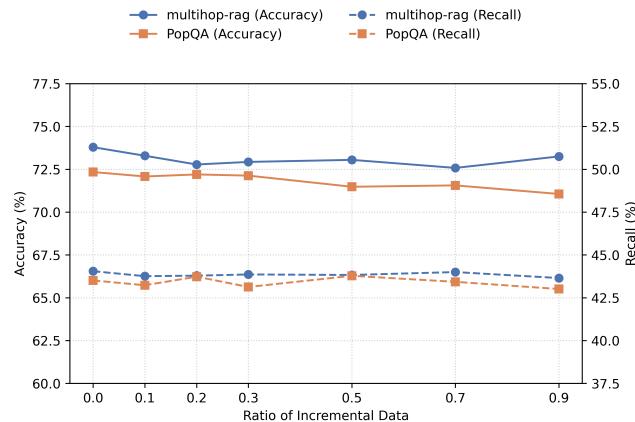
Figure 5: Corpus expansion performance.

Table 3: Results of ablation study.

	multihop-rag		quality	Popqa	
	Acc	Rec	Acc	Acc	Rec
HʏGRAG (Full)	**65.41**	**39.44**	<u>59.49</u>	**72.34**	**43.51**
w/o entity&relation	64.35	<u>39.08</u>	**60.20**	70.80	41.30
w/o community	<u>65.02</u>	39.01	58.75	<u>71.92</u>	<u>43.18</u>
w/o chunk	52.19	35.89	37.82	51.25	26.81
vanillaRAG	58.49	37.52	56.38	66.98	31.64

relational reasoning for both factual and multi-hop question answering tasks. Interestingly, on the quality dataset, this removal yields a slight performance improvement. We attribute this to the fact that, in reading comprehension tasks—particularly for smaller LLMs—explicit relational cues may sometimes mislead the model, whereas context-aware semantic signals are more essential when the answer requires inference rather than direct textual matching. In contrast, removing the community-level representation results in a relatively mild yet stable decrease, indicating that community structures primarily contribute to aggregating query-relevant higher-order knowledge, providing semantic summarization, and optimizing relation retrieval. Overall, these findings confirm that **HʏGRAG** benefits substantially from the joint design of context-aware and relation-aware mechanisms.

4.3.2 Comparative Analysis. To further examine the complementary effects of the *context-aware* and *relation-aware* components, we conducted comparative experiments. Specifically, we combined the retrieved chunks and communities from RAPTOR (the best-performing context-aware method) with the entities and relations from HiRAG (the modified method of MS GraphRAG in multi-hop reasoning path generation). We then compared the results with and without applying the same system prompt used in **HʏGRAG**.

As shown in Table 4, combining HiRAG and RAPTOR yields moderate improvements after applying the **HʏGRAG**-style prompt but still underperforms our model in both accuracy and recall. Notably, **HʏGRAG** achieves superior or comparable performance with substantially fewer tokens, demonstrating its efficiency in balancing reasoning depth and retrieval precision. These results demonstrate

Table 4: Comparative results combining HiRAG and RAPTOR.

Method	Multihop-RAG			MuSiQue		
	Acc	Rec	Tokens	Acc	Rec	Tokens
HiRAG + RAPTOR	62.36	37.39	7170.7	31.30	43.80	3277.2
+ prompt	70.27	41.37	7280.2	32.47	44.58	3768.5
HʏGRAG	**73.79**	**44.05**	**5030.3**	**35.87**	**48.06**	**1720.0**

that the integrated design of **HʏGRAG** is not a mere concatenation of relation and context representations, but rather an organic and unified framework that jointly models both aspects, enabling large language models to more effectively leverage contextual and relational knowledge during reasoning.

4.3.3 Case Study. To further illustrate the qualitative advantages of our framework, we present representative examples from the PopQA and MuSiQue datasets. For each dataset, we compare **HʏGRAG** with the best-performing baseline method. Detailed case studies of model responses are provided in the Appendix A. The results demonstrate that **HʏGRAG** effectively integrates relevant entities, relations, contextual information, and community-level knowledge to assist the model in constructing coherent reasoning chains and producing factually consistent answers. For instance, in the MuSiQue multi-hop reasoning task, baseline models such as LLightRAG and VanillaRAG fail to retrieve the correct passages through semantic search or to establish the connection between Jan Klapáč's birthplace and the target entity Prague Castle. In contrast, **HʏGRAG** successfully leverages relational and contextual information from both the chunk and community levels to capture the latent associations and infer the correct answer. Similarly, in a factual query from PopQA, under a setting involving multiple person entities, the model accurately identifies the target entity Paul Walker and retrieves the supporting evidence from key relation that he was born in Kilwinning. These cases collectively illustrate that **HʏGRAG** not only achieves unified modeling of relational and contextual knowledge but also enhances entity disambiguation and relational consistency during reasoning, thereby improving both factual accuracy and interpretability.

5 Conclusion

This work introduces a unified framework with context and relation-aware retrieval that transcends existing graph-based methods. **HʏGRAG** addresses the problem that entity-centric and chunk-centric methods operate on representations anchored to original text. By clustering hybrid nodes and generating communities that integrate both contextual and relational information, we create summary representations that go beyond source documents. Our bi-level retrieval mechanism leverages these representations across multiple levels, enabling access to insights that neither approach could achieve independently. Furthermore, **HʏGRAG**'s attachment-based update capability ensures the system efficiently incorporates new information. Overall, **HʏGRAG** demonstrates significant improvements in multi-hop reasoning (9.7% average gain) while maintaining efficiency and adaptability to dynamic corpora.

References

[1] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. arXiv:2402.03216 [cs.CL] https://arxiv.org/abs/2402.03216

[2] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130 [cs.CL] https://arxiv.org/abs/2404.16130

[3] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. arXiv:2405.06211 [cs.CL] https://arxiv.org/abs/2405.06211

[4] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL] https://arxiv.org/abs/2312.10997

[5] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[6] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. LightRAG: Simple and Fast Retrieval-Augmented Generation. arXiv:2410.05779 [cs.IR] https://arxiv.org/abs/2410.05779

[7] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2025. HippoRAG: neurobiologically inspired long-term memory for large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '24)*. Curran Associates Inc., Red Hook, NY, USA, Article 1902, 38 pages.

[8] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From RAG to Memory: Non-Parametric Continual Learning for Large Language Models. arXiv:2502.14802 [cs.CL] https://arxiv.org/abs/2502.14802

[9] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2025. GRAG: Graph Retrieval-Augmented Generation. arXiv:2405.16506 [cs.LG] https://arxiv.org/abs/2405.16506

[10] Haoyu Huang, Yongfeng Huang, Junjie Yang, Zhenyu Pan, Yongqiang Chen, Kaili Ma, Hongzhi Chen, and James Cheng. 2025. Retrieval-Augmented Generation with Hierarchical Knowledge. arXiv:2503.10150 [cs.CL] https://arxiv.org/abs/2503.10150

[11] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems* 43, 2 (Jan. 2025), 1–55. https://doi.org/10.1145/3703155

[12] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 1613, 15 pages.

[13] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.

[15] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401 [cs.CL] https://arxiv.org/abs/2005.11401

[16] Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2025. Think-on-Graph 2.0: Deep and Faithful Large Language Model Reasoning with Knowledge-guided Retrieval Augmented Generation. arXiv:2407.10805 [cs.CL] https://arxiv.org/abs/2407.10805

[17] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. arXiv:2212.10511 [cs.CL] https://arxiv.org/abs/2212.10511

[18] Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. 2022. QuALITY: Question Answering with Long Input Texts, Yes! arXiv:2112.08608 [cs.CL] https://arxiv.org/abs/2112.08608

[19] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph Retrieval-Augmented Generation: A Survey. arXiv:2408.08921 [cs.AI] https://arxiv.org/abs/2408.08921

[20] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. In *International Conference on Learning Representations (ICLR)*.

[21] Yixuan Tang and Yi Yang. 2024. MultiHop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-Hop Queries. arXiv:2401.15391 [cs.CL] https://arxiv.org/abs/2401.15391

[22] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. arXiv:2108.00573 [cs.CL] https://arxiv.org/abs/2108.00573

[23] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual E5 Text Embeddings: A Technical Report. arXiv:2402.05672 [cs.CL] https://arxiv.org/abs/2402.05672

[24] Shu Wang, Yixiang Fang, Yingli Zhou, Xilin Liu, and Yuchi Ma. 2025. ArchRAG: Attributed Community-based Hierarchical Retrieval-Augmented Generation. arXiv:2502.09891 [cs.CL] https://arxiv.org/abs/2502.09891

[25] Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. 2025. When to use Graphs in RAG: A Comprehensive Analysis for Graph Retrieval-Augmented Generation. arXiv:2506.05690 [cs.CL] https://arxiv.org/abs/2506.05690

[26] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] https://arxiv.org/abs/2505.09388

[27] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. arXiv:1809.09600 [cs.CL] https://arxiv.org/abs/1809.09600

[28] Fangyuan Zhang, Zhengjun Huang, Yingli Zhou, Qintian Guo, Zhixun Li, Wensheng Luo, Di Jiang, Yixiang Fang, and Xiaofang Zhou. 2025. EraRAG: Efficient and Incremental Retrieval Augmented Generation for Growing Corpora. arXiv:2506.20963 [cs.IR] https://arxiv.org/abs/2506.20963

[29] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. arXiv:2506.05176 [cs.CL] https://arxiv.org/abs/2506.05176

[30] Yingli Zhou, Yaodong Su, Youran Sun, Shu Wang, Taotao Wang, Runyuan He, Yongwei Zhang, Sicong Liang, Xilin Liu, Yuchi Ma, and Yixiang Fang. 2025. In-depth Analysis of Graph-based RAG in a Unified Framework. arXiv:2503.04338 [cs.IR] https://arxiv.org/abs/2503.04338

A Case Study

As shown in Table 5, the baselines fail to establish the correct reasoning chain required to answer the question. LLightRAG retrieves partial contextual evidence (i.e., Jan Klapáč's birthplace) but lacks the relational linkage to the target entity, while VanillaRAG produces an entirely irrelevant response due to the absence of structured retrieval. In contrast, **HyGRAG** successfully integrates entity-level, relational, and community-level information. It first identifies the entity Jan Klapáč and his birthplace (Prague), then connects this with the relation located in and the community summary mentioning Prague Castle. This multi-level reasoning enables the model to recover the implicit connection between the birthplace and the target entity, ultimately leading to the correct answer (Prague Castle). This case illustrates that **HyGRAG** effectively captures and organizes cross-entity relations and contextual dependencies, allowing the LLM to perform more accurate and interpretable multi-hop reasoning.

Table 6 presents a case from the PopQA dataset illustrating **HyGRAG**'s superiority in handling entity ambiguity. The query "In what city was Paul Walker born?" involves multiple homonymous entities. Baseline models like LLightRAG and RAPTOR confuse different individuals, producing inconsistent or irrelevant answers

Table 5: Case Study on musique.

Category	Multi-hop reasoning
Query	**What is the name of the castle in the city Jan Klapáč was born in? Answer: Prague Castle**
LLightRAG	Jan Klapáč's Birthplace Jan Klapáč was born in Prague, Protectorate of Bohemia and Moravia. Castle in Prague However, the data does not provide information about a specific castle in Prague where Jan Klapáč was born.
VanillaRAG	Unfortunately, the given context does not mention Jan Klapáč. However, based on the information provided, we can see that Casimir Pulaski, a Polish general and hero of the American Revolutionary War, was born in Warsaw.
HyGRAG	Analysis: To answer the user's question, we need to find the connection between Jan Klapáč's birthplace and the castle mentioned in the context. From the **Most Relevant Entities**, we see that Prague Castle is the most relevant entity, but it's not directly related to Jan Klapáč's birthplace. However, from the **Key Relationships**, we find that Jan Klapáč was born in Prague (relationship 1: jan klap –[born in]–> prague). From the **Most Relevant Documents**, we see that Document 1 mentions Jan Klapáč's birthplace as Prague, Protectorate of Bohemia and Moravia. Now, let's look for a connection between Prague and a castle. From the **Key Relationships**, we find that Prague Castle is located in the Czech Republic (relationship 3: prague castle –[located in]–> czech republic). ... From the **Community Summary**, we see that Prague Castle is a central entity associated with the architectural styles of Baroque and Mannerism, and Matthias of Arras is credited with constructing the castle. ... According to the **Community Summary**, Prague Castle is a central entity associated with the architectural styles of Baroque and Mannerism, and Matthias of Arras is credited with constructing the castle. So, the name of the castle in Prague is Prague Castle.

(e.g., Balwyn, Colac, or Glendale). In contrast, **HyGRAG** combines entity–relation reasoning with chunk-level to filter out unrelated entities and consolidate consistent evidence. It correctly identifies Kilwinning and North Ayrshire—matching the gold answer—demonstrating that its relational and hierarchical design substantially improves factual accuracy and interpretability.

B Detailed Statistics

We show the hybrid graph statistics using Llama-3.1-8B-Instruct for indexing in Table 8. Hyperparameter settings used in Table 7.

C More experiments

C.1 Robustness in Dense Retrievers

We experimented with a range of state-of-the-art embedding models. As shown in Table 9, **HyGRAG** consistently outperforms the corresponding vanilla dense retrieval baseline across all embedding models. Moreover, the performance of **HyGRAG** improves progressively as the quality of the underlying embedding model

increases, demonstrating its strong scalability and sensitivity to embedding fidelity. In our experiments, we specifically evaluated **HyGRAG** using Qwen3-Embedding-0.6B[29], text-embedding-v3[29], and multilingual-e5-large-instruct[23] as the underlying dense retrievers.

C.2 Robustness in Base LLM

We present the QA performance of various methods using the base LLM Qwen3-8B, as summarized in Table 10. When employing Qwen3-8B for both indexing and question answering, **HyGRAG** consistently achieves competitive accuracy and recall across most datasets, maintaining leading performance on MuSiQue and HotpotQA, with accuracy gains up to 21.36% over other methods.

For context-level methods, VanillaRAG, RAPTOR, and EraRAG show slight improvements compared to their performance with the previous base LLM. However, they exhibit noticeable drops on the MuSiQue and HotpotQA datasets, indicating that their performance is sensitive to changes in the base LLM. Similarly, the relation-level

Table 6: Case Study on Popqa.

Category	Factual accuracy
Query	In what city was Paul Walker born? Answer: Kilwinning | The Crossroads of Ayrshire
LLightRAG	Paul Walker was born in Balwyn or Colac, Victoria, Australia. According to the data provided, both Balwyn and Colac are listed as places where Walker was born, with Balwyn being noted as a suburb in Victoria, Australia, and Colac as a town in the same region. The information suggests that Walker's birthplace is associated with these two locations, though it does not definitively state which one is the exact place of birth. This ambiguity is reflected in the sources, which mention that Walker was born in Balwyn or Colac in 1927.
RAPTOR	Based on the provided information, there are multiple individuals named Paul Walker, and each has a different birthplace. However, the most well-known Paul Walker, the actor, was born in Glendale, California, on September 12, 1973. If you are referring to Paul Baxendale-Walker, the Arctic explorer, he was born in Shrewsbury, England, on March 9, 1966. If you are referring to the other Paul Walker, the one born in Altoona, Pennsylvania, or Memphis, the information provided is not as detailed as the other two, but it can be concluded that he was born in either Altoona, Pennsylvania, or Memphis, on March 15, 1968, or in Altoona, Pennsylvania, on an unspecified date.
HyGRAG	Analysis of Evidence and Reasoning: ... Key Relationships: 1. **Paul Walker –[was born in]–> England** 2. **Paul Walker –[born in]–> Scotland** 3. **Paul Walker –[was born in]–> Shrewsbury** 4. **Paul Walker –[born in]–> Kilwinning** 5. **Paul Walker –[born in]–> North Ayrshire** These relationships suggest multiple possible birthplaces for Paul Walker, including ... Documents: - **Document 1** states: "He was born in Shrewsbury, England on 09 March 1966." ... - **Document 5** states: "Walker was born in Franklin, Georgia, on October 22, 1935." These documents provide conflicting birthplaces for **Paul Walker**, with different cities and even different years of birth. ... Conclusion: Based on the **Key Relationships**, the most reliable answers are: - **Shrewsbury, England** - **Kilwinning** - **North Ayrshire** These are all in **Scotland** or **England**, and they are explicitly stated in the **Key Relationships**. However, the **Documents** provide conflicting information: - **Document 1** says **Shrewsbury, England** ... This suggests that there may be multiple individuals named **Paul Walker**, or that the documents refer to different people with the same name....

Table 7: Hyperparameter settings used in our experiments.

Parameter	Value
cleora_iterations	2
lsh_num_hyperplanes k	16
lsh_min_cluster_size $S_{\min}$	5
lsh_max_cluster_size $S_{\max}$	50
max_hierarchy_levels L	4
community_summary_length	300
shared_entity_threshold l	3
update_th tau_{attach}	0.65

method HippoRAG2 also shows performance declines on these datasets, and its results on Quality remain suboptimal.

In contrast, **HyGRAG** maintains stable and superior performance across all tasks, demonstrating robustness and adaptability to different LLM architectures. This suggests that **HyGRAG** 's synergistic integration of relation- and context-level information provides resilience against base model changes, while other approaches are more sensitive to variations in the underlying LLM.

D   Prompts used for HyGRAG

The Answer Generation Prompt is designed to synthesize a comprehensive and evidence-grounded answer to a user's query from a structured, multi-source context. The process begins by presenting

Table 8: Statistic of HʏGRAG using Llama-3.1-8B-Instruct.

Metric	PopQA	Quality	MultihopQA	MusiqueQA	HotpotQA
# of chunk nodes	32,157	1,518	1,658	29,898	66,559
# of entity nodes	271,776	33,440	36,884	224,846	510,457
# of community nodes	6,403	758	843	5,422	12,109
# of relations	379,564	32,938	32,968	314,145	758,380
# of chunk-chunk edge	162,273	3,690	16,466	48,264	199,675
# of chunk-entity edge	360,891	15,453	37,782	293,276	647,454

Table 9: Performance comparison of different embedding models under Vanilla and HʏGRAG settings.

Dense Retriever	Vanilla		HʏGRAG	
	Accuracy	Recall	Accuracy	Recall
bge-m3	58.49	37.52	65.41	39.44
Qwen3-Embedding-0.6B	57.98	37.45	65.49	39.39
text-embedding-v3	59.31	38.14	66.00	39.74
multilingual-e5-large-instruct	58.76	37.83	63.65	39.06

Table 10: Overall QA results(Accuracy and Recall) on static RAG query datasets using Qwen3-8B. "OOT" denotes results that could not be obtained within two days.

Method	Factual Accuracy		Reading Comprehension			Multi-Hop Reasoning			
	PopQA		Quality	MultihopQA		MusiqueQA		HotpotQA	
	Accuracy	Recall	Accuracy	Accuracy	Recall	Accuracy	Recall	Accuracy	Recall
Zero-shot	25.23	9.83	34.78	47.10	27.33	8.13	22.38	30.01	36.62
CoT	30.52	14.75	36.06	60.02	31.37	10.50	27.97	32.90	41.15
VanillaRAG	63.83	35.00	59.28	61.93	37.40	23.60	35.26	51.54	55.15
RAPTOR	63.18	31.05	60.87	60.13	36.66	23.67	35.37	54.70	59.07
EraRAG	OOT	OOT	**61.78**	61.78	36.85	OOT	OOT	OOT	OOT
L-LightRAG	71.05	44.37	58.41	55.52	38.25	33.10	47.52	OOT	OOT
HippoRAG	63.97	33.49	56.65	61.89	33.54	24.90	38.88	OOT	OOT
HippoRAG2	67.12	35.22	42.53	63.97	36.24	25.87	37.08	OOT	OOT
L-GraphRAG	58.32	33.24	58.75	55.20	38.04	18.53	32.79	OOT	OOT
HʏGRAG	**73.84**	**46.17**	60.67	**73.87 (67.57)**	**45.09 (38.87)**	**36.67**	**49.33**	**72.18**	**72.67**

the model with a rich contextual payload retrieved from a hierarchical knowledge system, which includes: hierarchical community summaries with similarity scores, a ranked list of the most relevant entities, key entity relationships structured as triplets, and the content of the most relevant source documents. Subsequently, the prompt provides a set of explicit operational rules to constrain the generation process. The framework mandates that the model must report any informational gaps by stating "Insufficient information" and is forbidden from fabricating information not present in the context, ensuring a high degree of fidelity to the source material.

The Community Summarization Prompt outlines a framework for distilling a collection of content, referred to as a knowledge community, into a structured and semantically dense summary. The prompt's primary objective is to generate a summary specifically tailored for the downstream task of creating high-quality semantic embeddings. It guides the model by defining a clear, four-part structure for the output, requiring the summary to comprehensively

cover: 1) key themes and topics, 2) important entities and their roles, 3) relationships and connections between these entities, and 4) the overall context and significance of the information. By specifying both a word limit and the explicit need for rich semantic content, the prompt ensures the generation of a concise yet information-rich text that captures the core essence of the knowledge community.

The Answer Generation and Community Summarization prompt templates are provided in the code repositoryhttps://anonymous.4open.science/r/HyGRAG/.