NodeRAG:
Structuring Graph-based RAG with Heterogeneous Nodes

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) empowers large language models to access external and private corpora, enabling factually consistent responses in specific domains. By exploiting the inherent structure of the corpus, graph-based RAG methods enrich this process by building a knowledge graph index and leveraging the structural nature of graphs. However, current graph-based RAG approaches seldom prioritize the design of graph structures. Inadequately designed graphs impede the seamless integration of diverse graph algorithms and result in workflow inconsistencies and degraded performance. To further unleash the potential of graph for RAG, we propose **NodeRAG**, a heterogeneous graph-centric framework that enables the seamless and holistic integration of graph-based methodologies into the RAG workflow. By aligning closely with the capabilities of LLMs, this framework ensures a fully cohesive and efficient end-to-end process. Through extensive experiments, we demonstrate that NodeRAG exhibits performance advantages over previous methods, including GraphRAG and LightRAG, not only in indexing time, query time, and storage efficiency but also in delivering superior question-answering performance on multi-hop benchmarks and open-ended head-to-head evaluations with minimal retrieval tokens. Our anonymous GitHub repository is available at this link.

1 Introduction

Retrieval-augmented generation (RAG) has emerged as a solution to the challenges posed by the rapid evolution of real-world knowledge domains (Fan et al., 2024), coupling large language models (LLMs) with an external retrieval mechanism to ensure the generation of factually consistent and contextually relevant information (Tonmoy et al., 2024; Shrestha et al., 2024; Liu et al., 2024). Despite recent progress, current RAG methods face notable shortcomings in handling multi-hop reasoning (Luo et al., 2023; Wang et al., 2024b) and summary-level queries (Han et al., 2024a; Wen et al., 2023) due to their insufficient utilization of data structures and lack of high-level understanding of the text corpus. Graph-based RAG methods (Tian et al., 2024; Park et al., 2023) have been proposed to enhance retrieval and question-answering performance, specifically addressing the two main challenges faced by traditional RAG approaches. Leveraging LLMs to decompose raw data into graph structures (Jiménez Gutiérrez et al., 2024; He et al., 2024) for utilizing structural information, as well as employing LLMs for summary-based enhancements (Edge et al., 2024; Guo et al., 2024) to derive insights beyond the original text, have gradually become mainstream approaches.

However, previous Graph-based RAG works (Trajanoska et al., 2023; Jiménez Gutiérrez et al., 2024) have rarely considered the critical role of graph structures, i.e., what forms of graph better support RAG. Among existing approaches, Knowledge graph approaches (Sanmartin, 2024; Wang et al., 2024b) extract triples for structure but still rely on text chunks for retrieval, often yielding incoherent or irrelevant context. Current methods aim to enrich graph information and extract deeper insights, but often face inefficiencies and inconsistencies due to poor structural design. For example, GraphRAG (Edge et al., 2024) uses a tightly coupled entity-event homogeneous structure, which limits the integration of original context and summary information. This leads to inconsistent retrieval strategies (e.g., separate local and global retrieval) and coarse-grained results, where retrieving an entity also pulls in unrelated events.

To address these limitations, we propose NodeRAG, which is built around a well-designed Heterogeneous Graph, comprehensively considering the entire process of graph indexing and search-

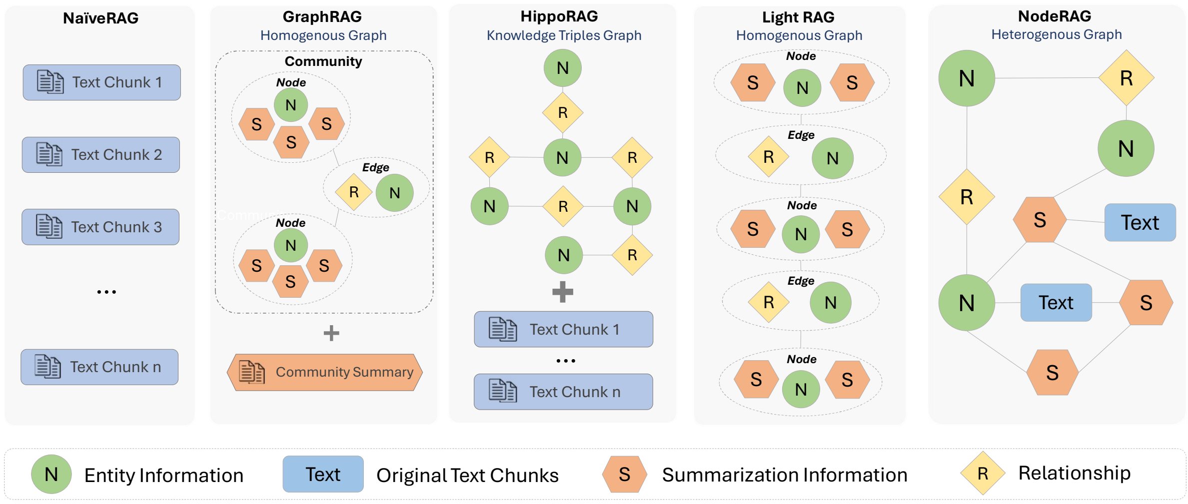

Figure 1: Comparisons between NodeRAG and other RAG systems. NaiveRAG retrieving fragmented text chunks, leads to redundant information. HippoRAG introduces knowledge graphs but lacks high-level summarization. GraphRAG retrieves community summaries but may still produce coarse-grained information. LightRAG incorporates one-hop neighbors but retrieves redundant nodes. In contrast, NodeRAG utilizes multiple node types, including high-level elements, semantic units, and relationships, enabling more precise, hierarchical retrieval while reducing irrelevant information.

ing, enabling fine-grained retrieval. The heterograph adheres to the principle of unfolding and flattening, decomposing different types of information to construct a heterogeneous fully nodalized graph where nodes serve distinct functions and roles. The heterograph encapsulates information from the original corpus and also extends beyond it, incorporating enriched insights such as key node attributes, and high-level discoveries. As illustrated in Figure 1, NodeRAG enables fine-grained, node-level retrieval tailored to user queries through graph algorithms, offering both explainable results and high-level understanding.

The key contributions of our work can be summarized in three main aspects.

(1) Better Graph Structure for RAG The graph structure serves as the foundation for graph-based RAG where significance has been overlooked. Our work emphasizes its importance and introduces a graph structure that better supports RAG.

(2) Fine-grained and Explainable Retrieval The heterograph enables fine-grained and functionally distinct nodes, allowing graph algorithms to effectively and reasonably identify key multi-hop nodes. This leads to more relevant retrieval with minimal retrieval context, enhancing both precision and interoperability.

(3) Unified-Level Information Retrieval Decomposed information from documents and extracted insights from LLMs are not treated as separate layer but are instead unified as nodes within the heterograph. This integration allows for a cohesive framework capable of handling information needs across different levels.

In addition, extensive experiments show that NodeRAG outperforms prior graph-based RAG methods in multi-hop tasks and open-ended evaluations. It achieves high retrieval precision with minimal tokens and offers system-level efficiency gains in indexing, querying, and storage, as detailed in Appendix A.

2 NodeRAG

The NodeRAG pipeline is built on a heterograph structure introduced in Section 2.1 and consists of two main stages: graph indexing and graph searching. The indexing stage includes graph decomposition, augmentation, and enrichment (Sections 2.2–2.4), integrating diverse nodes and edges using LLMs and graph algorithms. The searching stage (Section 2.5) leverages the heterograph's structure and algorithms to retrieve relevant information efficiently. Details on the graph algorithms and prompting strategies are provided in Appendices C and E, respectively.

2.1 Heterograph

The concept of the heterograph embodies the principle of comprehensive unfolding and flattening of information into a fully nodalized structure. This structure achieves its granularity through the integration of seven hetero node types: entity (N), relationship (R), semantic unit (S), attribute (A),

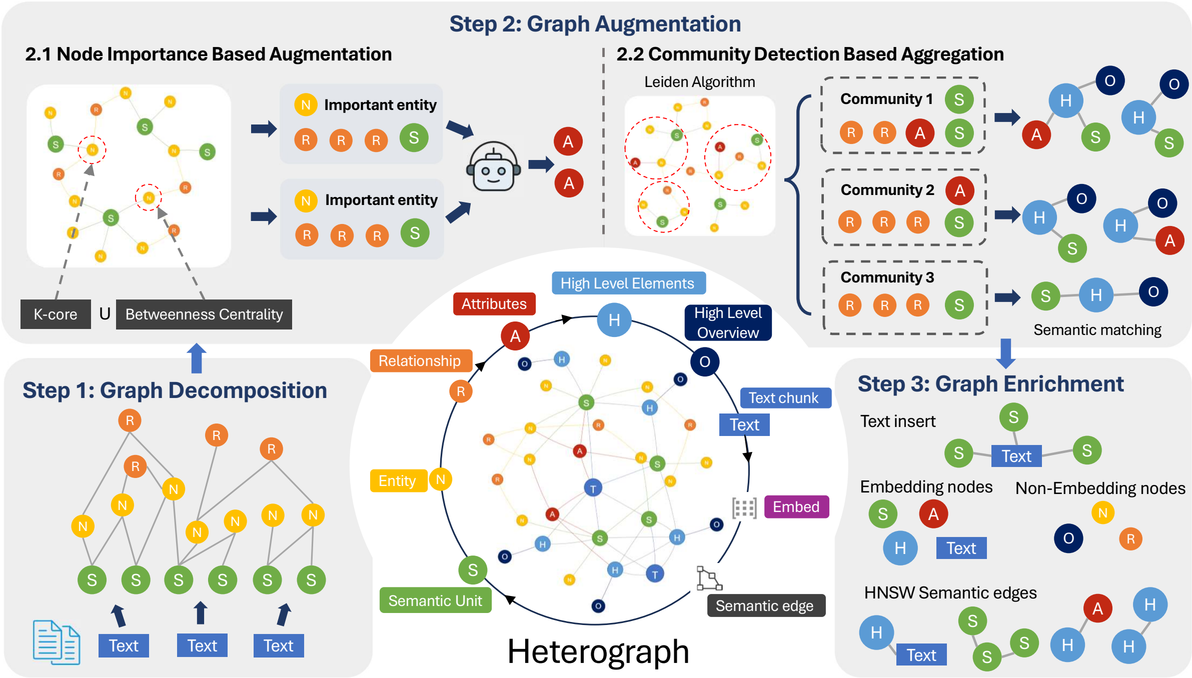

Figure 2: Main indexing workflow of NodeRAG. It illustrates the step-by-step construction of the heterograph, including the process of graph decomposition, graph augmentation, and graph enrichment

high-level elements (H), high-level overview (O), and text (T). Mathematically, the heterograph is defined as:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Psi),$$

where $\mathcal{G}$ is the heterograph, $\mathcal{V}$ represents the set of nodes, $\mathcal{E}$ is the set of edges, and $\Psi : \mathcal{V} \to \text{Types}$ is a mapping function that assigns each node $v \in \mathcal{V}$ to a specific type, where

$$\text{Types} = \{N, R, S, A, H, O, T\}.$$

For any node v, $\Psi(v)$ defines its type, with each node type performing a distinct and well-defined function, as detailed in subsequent sections and appendix C. For each $e \in \mathcal{E}$, the default weight of e is set to 1, representing a basic connection between two nodes. Furthermore, we define $\mathcal{V}_{\text{types}}$ as the subset of nodes corresponding to a subset set types $\subseteq$ Types, formally expressed as:

$$\mathcal{V}_{\text{types}} = \{v \in \mathcal{V} \mid \Psi(v) \in \text{types}\}.$$

For instance, $\mathcal{V}_{\{N,R,S\}}$ represents the subset containing only entity, relationship, and semantic unit nodes.

2.2 Graph Decomposition

First, we define a null heterograph $\mathcal{G}^0$. The initial step involves employing a LLM to decompose text chunks from the source corpus into three primary node types: semantic units (S), entities (N), and relationships (R). These nodes are then connected to construct the initial heterograph. This process can be formalized as:

$$\mathcal{G}^1 = \mathcal{G}^0 \cup \{v \in \mathcal{V}, e \in \mathcal{E} \mid \Psi(v) \in \{S, N, R\}\},$$

Where e represents the connecting edges between semantic units and entity nodes, as well as between relationship nodes and their corresponding source and target entities. An example of the different node types can be found in Appendix C.1.

Semantic unit (S) Semantic units serve as local summaries that encapsulate independent events in paraphrased form. They function as core nodes in graph augmentation and enhance search quality. In contrast, text chunks divided without semantic awareness may mix unrelated content, introducing noise and increasing entropy, which reduces the effectiveness of augmentation and retrieval.

Entity (N) **and Relationship** (R) Entities N are nodes representing entity names, while relationships R are also converted into nodes that link source and target entities. This design decouples N and R from specific events, allowing them to function independently while remaining anchored to relevant contexts.

2.3 Graph Augmentation

The heterograph $\mathcal{G}^1$ provides a foundational low-level structure. To further augment the graph, we implement node importance-based augmentation and community detection-based aggregation, which respectively capture the perspectives of individual node significance and structural cohesion within the graph.

Node Importance Based Augmentation We focus on structurally and functionally pivotal entities, processing them with LLMs to generate attribute summaries. By targeting only key entities and their semantic contexts, this approach ensures both precision and efficiency. The selection of important entities, N^*, is guided by two complementary metrics: K-core decomposition (Seidman, 1983; Kong et al., 2019) and betweenness centrality (Brandes, 2001). K-core identifies nodes in densely connected subgraphs that are critical to graph cohesion, while betweenness centrality highlights nodes that act as bridges for information flow. These metrics are denoted as $K(\mathcal{G}^1)$ and $B(\mathcal{G}^1)$, where $K(\cdot)$ and $B(\cdot)$ represent the selected entity nodes from the graph. The final set of important entities is defined as:

$$N^* = K(\mathcal{G}^1) \cup B(\mathcal{G}^1).$$

Entity attributes are constructed directly from relationships and semantic units, bypassing raw texts to avoid redundancy. Each generated attribute node is added to the graph and connected to its corresponding entity node via the edge e_a.

$$\mathcal{G}^2 = \mathcal{G}^1 \cup \{v \in \mathcal{V}, e_a \in \mathcal{E} \mid \Psi(v) \in \{A\}\}.$$

Community Detection Based Aggregation We first apply the Leiden algorithm (Traag et al., 2019) to $\mathcal{G}^2$ for community detection, assigning each node $v \in \mathcal{G}^2$ to a community $\mathcal{C}_n$. Within each $\mathcal{C}_n$, LLMs analyze the aggregated content to extract high-level elements (H) that capture the core information of the community, such as summaries and sentiment. To preserve structural coherence, each high-level node $v \in \mathcal{V}_H$ need to be semantically connected to relevant nodes $v \in \mathcal{V}_{\{S,A,H\}}$ in the same community. This is achieved via K-means clustering (MacQueen et al., 1967) on node embeddings, with the number of clusters set to $K = \sqrt{|\mathcal{V}_{\{S,A,H\}}|}$. An edge e_h is created if v and $v' \in \mathcal{V}_H$ belong to the same cluster $\mathcal{S}_k$ and community $\mathcal{C}_n$. Additionally, LLMs generate a keyword title for each high-level node, denoted as the overview node (O), used for dual search (see Section 2.5). Each $v \in \mathcal{V}_H$ and $v \in \mathcal{V}_O$ is connected via edge e_o. The resulting graph $\mathcal{G}^3$ integrates high-level elements and their connections. Formally, the resulting graph at this stage is defined as:

$$\mathcal{G}^3 = \mathcal{G}^2 \cup \{v \in \mathcal{V}, e_h, e_o \in \mathcal{E} \mid \psi(v) = \{H, O\}\}.$$

2.4 Graph Enrichment

In the previous process of generating the heterograph, $\mathcal{G}^3$ already contains a wealth of information. However, certain unique and additional details can still further enrich the heterograph, enabling it to not only preserve the entirety of the original text's information but also gain enhanced features and insights that go far beyond the source material.

Text Insertion Text chunks are not directly incorporated into $\mathcal{G}$ during graph augmentation due to their semantically incoherent structure. Nevertheless, these original chunks retain substantial value, as they contain detailed information that helps prevent both information loss and error propagation in LLM processing. Hence, it is crucial to ensure that the original content remains accessible and searchable within the graph.

$$\mathcal{G}^4 = \mathcal{G}^3 \cup \{v, e_s \mid \Psi(v) = T\},$$

where e_s denotes the edges connecting text chunks to their relevant semantic units.

Embedding Vector similarity is effective for nodes $v \in \mathcal{V}_{\{T,A,S,H\}}$, which encode rich contextual information. In contrast, nodes $v \in \mathcal{V}_{\{N,O\}}$, representing names or titles, are less suitable due to their limited semantic depth. To address this limitation, we developed a dual search mechanism. During the embedding process, we selectively embed only a subset of the graph's data, specifically $v \in \mathcal{V}_{\{T,A,S,H\}}$. This targeted embedding step is crucial for reducing storage overhead while preserving efficient search capabilities.

HNSW Semantic Edges The Hierarchical Navigable Small World (HNSW) algorithm (Malkov and Yashunin, 2018) is an approximate nearest neighbor search method that organizes data into a multi-layer graph structure to efficiently retrieve semantically similar nodes. It represents the data as a layered graph $\mathcal{H} = \{\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_m\}$, where $\mathcal{L}_0$ is the base layer containing the densest semantic similarity connections, and higher layers ($\mathcal{L}_i, i > 0$) are sparsely connected to facilitate coarse-grained navigation. $\mathcal{H}$ is built iteratively. When a new node is added, it is inserted into a random level and all layers below it, connecting to similar neighbors based on cosine similarity. Higher layers remain sparse with long-range connections, while $\mathcal{L}_0$ focuses on dense local relationships. The search starts at the sparsely connected top layer, and progressively descends to $\mathcal{L}_0$. In our work, the base layer $\mathcal{L}_0$ of the HNSW graph, which en-

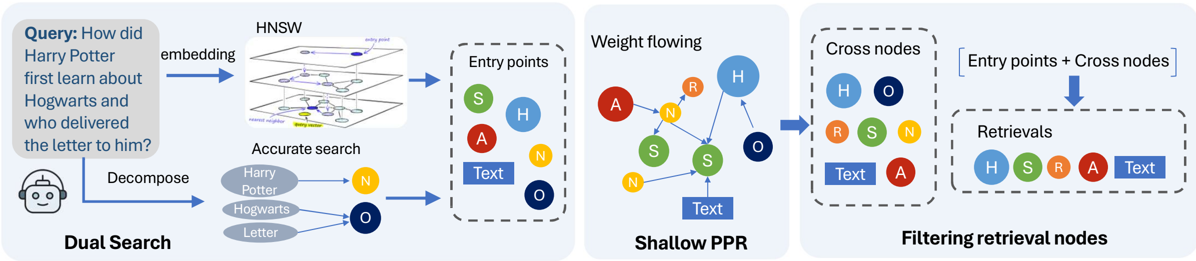

Figure 3: This figure focuses on the querying process, where entry points are extracted from the original query, followed by searching for related nodes that need to be retrieved in the heterograph.

codes semantic relations between nodes, is integrated with the heterograph $\mathcal{G}$. The updated graph, denoted as $\mathcal{G}^5$, is expressed as:

$$\mathcal{G}^5 = \mathcal{G}^4 \cup \mathcal{L}_0.$$

The inclusion of $\mathcal{L}_0$ enhances the heterograph's search capabilities by incorporating semantic dense proximity edges, augmenting its structural information in the graph.

2.5 Graph Searching

We first apply a dual search mechanism to identify entry points within the heterograph. Subsequently, a shallow Personalized PageRank (PPR) algorithm is employed to extract cross nodes. The combination of entry point nodes and cross nodes is then filtered to produce the final retrieval.

Dual Search Dual search combines exact matching on title nodes and vector similarity search on rich information nodes to identify entry points in the heterograph $\mathcal{G}$. Given a query, the LLM extracts entities N^q and embeds the query into vector ($\mathbf{q}$). The entry points are defined as:

$$\mathcal{V}_{\text{entry}} = \{v \in \mathcal{V} \mid \Phi(v, N^q, \mathbf{q})\},$$

where the condition function $\Phi(v, N^q, \mathbf{q})$ is defined as:

$$\Phi(v, N^q, \mathbf{q}) = \begin{cases} v \in \mathcal{V}_{\{N,O\}} \wedge \mathcal{M}(N^q, v), \\ v \in \mathcal{V}_{\{S,A,H\}} \wedge \mathcal{R}(\mathbf{q}, v, k). \end{cases}$$

Here, the exact matching function $\mathcal{M}(v^*, v)$ returns true if a node matches one of the extracted entities by word level string matching. Additionally, the similarity-ranking function $\mathcal{R}(\mathbf{q}, v, k)$ returns true if a node ranks among the top-k most similar to $\mathbf{q}$ based on the HNSW algorithm. Nodes $v \in \mathcal{V}_{\{N,O\}}$ are non-retrievable, as they serve solely as entry points to the graph and do not directly contribute to the retrievable content. Only nodes identified through the shallow PPR as closely related to all entry points are included in the retrieval results as cross nodes. By decoupling retrieval from direct exact matching, this approach reduces the influence of noisy or ambiguous queries, thereby improving the overall robustness of the retrieval process.

Shallow PPR Personalized PageRank (PPR) identifies relevant nodes in the heterograph $\mathcal{G}$ by simulating a biased random walk starting from a set of entry points. In our approach, we use shallow PPR, limiting the number of iterations t to ensure that relevance remains localized to the neighborhoods of the entry points. This early stop strategy prevents excessive diffusion to distant or irrelevant parts of the graph, focusing instead on multi-hop nodes near the entry points. Let P be the normalized adjacency matrix of $\mathcal{G}$, where P_{ij} represents the transition probability from node i to node j. The PPR process starts with a personalization vector $p \in \mathbb{R}^{|\mathcal{V}|}$, where $p_i = 1/|\mathcal{V}_{\text{entry}}|$ if $v_i \in \mathcal{V}_{\text{entry}}$, and $p_i = 0$ otherwise. The PPR score vector $\pi^{(t)}$ after t iterations is computed iteratively as:

$$\pi^{(t)} = \alpha p + (1-\alpha) P^\top \pi^{(t-1)}, \quad \pi^{(0)} = p,$$

where $\alpha \in (0, 1)$ is the teleport probability that balances restarting at entry points and propagating through the graph. After t iterations, the top-k nodes with the highest PPR scores for each type are selected as cross nodes, denoted as V_{cross}. In our default setting, we use $\alpha = 0.5$ and $t = 2$ to achieve a balance between exploration and convergence.

Filter Retrieval Nodes Finally, the retrieval nodes are filtered from the union of entry nodes and cross nodes to include only retrievable nodes of $v \in \mathcal{V}_{\{T,A,S,H,R\}}$. $v \in \mathcal{V}_{\{N,O\}}$, which contain only keywords without informational content, are excluded from the retrieval context. The final set of retrieval nodes is therefore defined as:

$$\mathcal{V}_{\text{retrieval}} = \{\, v \in \mathcal{V}_{\text{entry}} \cup \mathcal{V}_{\text{cross}} \mid \\ \psi(v) \in \{T, S, A, H, R\}\}$$

Part I: General comparisons

Methods	HotpotQA		MuSiQue		MultiHop		Arena-Writing		Arena-Tech		Arena-Science		Arena-Recreation		Arena-Lifestyle		Arena-FiQA	
	Acc.↑	#Token↓	Acc.↑	#Token↓	Sco.↑	#Token↓	W+T↑	#Token↓	W+T↑	#Token↓	W+T↑	#Token↓	W+T↑	#Token↓	W+T↑	#Token↓	W+T↑	#Token↓
NaiveRAG	87.50%	9.8k	39.43%	9.6k	0.56	8.9k	0.663	9.4k	0.689	9.1k	0.526	9.0k	0.720	9.3k	0.817	9.1k	0.926	9.1k
HyDE	73.00%	10.0k	33.14%	9.8k	0.53	9.4k	0.789	9.6k	0.863	9.3k	0.823	9.3k	0.777	9.5k	0.829	9.3k	0.949	9.3k
LightRAG	79.00%	7.1k	36.00%	7.4k	0.50	7.9k	0.754	6.3k	0.937	6.9k	0.840	7.1k	0.800	6.2k	0.817	6.8k	0.937	7.7k
GraphRAG	89.00%	6.6k	41.71%	6.6k	0.53	7.4k	0.749	6.4k	0.943	6.7k	0.863	6.7k	0.806	6.6k	0.863	6.8k	0.960	6.8k
NodeRAG	**89.50%**	**5.0k**	**46.29%**	**5.9k**	**0.57**	**6.1k**	**0.794**	**3.3k**	**0.949**	**3.8k**	**0.903**	**4.2k**	**0.886**	**3.4k**	**0.949**	**3.3k**	**0.977**	**3.4k**

Part II: Pairwise Comparisons

Domain	M1 vs M2	Win (M1)	Tie	Win (M2)	Domain	M1 vs M2	Win (M1)	Tie	Win (M2)	Domain	M1 vs M2	Win (M1)	Tie	Win (M2)
FiQA	NodeRAG vs GraphRAG	0.520	0.126	0.354	Recreation	NodeRAG vs GraphRAG	0.531	0.126	0.343	Writing	NodeRAG vs GraphRAG	0.691	0.120	0.189
	NodeRAG vs LightRAG	0.486	0.103	0.411		NodeRAG vs LightRAG	0.526	0.143	0.331		NodeRAG vs LightRAG	0.651	0.115	0.234
	NodeRAG vs NaiveRAG	0.749	0.034	0.217		NodeRAG vs NaiveRAG	0.800	0.017	0.183		NodeRAG vs NaiveRAG	0.851	0.018	0.131
	NodeRAG vs HyDE	0.531	0.155	0.314		NodeRAG vs HyDE	0.440	0.189	0.371		NodeRAG vs HyDE	0.349	0.228	0.423
	GraphRAG vs LightRAG	0.320	0.303	0.377		GraphRAG vs LightRAG	0.406	0.154	0.440		GraphRAG vs LightRAG	0.297	0.303	0.400
	GraphRAG vs NaiveRAG	0.754	0.092	0.154		GraphRAG vs NaiveRAG	0.714	0.080	0.206		GraphRAG vs NaiveRAG	0.691	0.092	0.217
	GraphRAG vs HyDE	0.491	0.132	0.377		GraphRAG vs HyDE	0.377	0.137	0.486		GraphRAG vs HyDE	0.177	0.126	0.697
	LightRAG vs NaiveRAG	0.711	0.106	0.183		LightRAG vs NaiveRAG	0.691	0.063	0.246		LightRAG vs NaiveRAG	0.731	0.080	0.189
	LightRAG vs HyDE	0.514	0.143	0.343		LightRAG vs HyDE	0.349	0.171	0.480		LightRAG vs HyDE	0.211	0.178	0.611
	NaiveRAG vs HyDE	0.611	0.063	0.326		NaiveRAG vs HyDE	0.674	0.069	0.257		HyDE vs NaiveRAG	0.857	0.040	0.103
Lifestyle	NodeRAG vs GraphRAG	0.640	0.114	0.246	Science	NodeRAG vs GraphRAG	0.497	0.200	0.303	Tech	NodeRAG vs GraphRAG	0.543	0.154	0.303
	NodeRAG vs LightRAG	0.623	0.131	0.246		NodeRAG vs LightRAG	0.538	0.208	0.254		NodeRAG vs LightRAG	0.497	0.137	0.366
	NodeRAG vs NaiveRAG	0.800	0.040	0.160		NodeRAG vs NaiveRAG	0.829	0.085	0.086		NodeRAG vs NaiveRAG	0.777	0.046	0.177
	NodeRAG vs HyDE	0.526	0.205	0.269		NodeRAG vs HyDE	0.423	0.280	0.297		NodeRAG vs HyDE	0.543	0.160	0.297
	GraphRAG vs LightRAG	0.429	0.120	0.451		GraphRAG vs LightRAG	0.361	0.343	0.296		GraphRAG vs LightRAG	0.400	0.234	0.366
	GraphRAG vs NaiveRAG	0.680	0.074	0.246		GraphRAG vs NaiveRAG	0.829	0.108	0.063		GraphRAG vs NaiveRAG	0.657	0.097	0.246
	GraphRAG vs HyDE	0.354	0.097	0.549		GraphRAG vs HyDE	0.354	0.172	0.474		GraphRAG vs HyDE	0.463	0.143	0.394
	LightRAG vs NaiveRAG	0.663	0.046	0.291		LightRAG vs NaiveRAG	0.828	0.119	0.053		LightRAG vs NaiveRAG	0.691	0.075	0.234
	LightRAG vs HyDE	0.349	0.120	0.531		LightRAG vs HyDE	0.308	0.189	0.503		LightRAG vs HyDE	0.463	0.097	0.440
	HyDE vs NaiveRAG	0.709	0.028	0.263		HyDE vs NaiveRAG	0.840	0.074	0.086		HyDE vs NaiveRAG	0.606	0.051	0.343

Table 1: **Part I: General Comparisons** evaluates NaiveRAG, HyDE, LightRAG, GraphRAG, and NodeRAG on HotpotQA and MuSiQue (accuracy and average tokens) and in the Arena using *Win+Tie ratios* and *average tokens*. **Part II: Pairwise Comparisons** shows the fraction of "wins" (Win(M1)), "ties" (Tie), and "losses" (Win(M2)) when comparing one RAG method against another (e.g., NodeRAG vs. GraphRAG). Bold values highlight the best performance.

3 Evaluation

We evaluate NodeRAG's performance across three different multihop benchmarks, **HotpotQA** (Yang et al., 2018), **MuSiQue** (Trivedi et al., 2022b), **MultiHop-RAG** (Tang and Yang, 2024), and an open-ended head to head evaluation **RAG-QA Arena** (Han et al., 2024b) across six domains. And we compare our method against several strong and widely used RAG methods as baseline models, including **NaiveRAG** (Lewis et al., 2020), **HyDE** (Gao et al., 2022a), **GraphRAG** (Edge et al., 2024), **LightRAG** (Guo et al., 2024). The details of these datasets and baseline models are introduced in Appendix B.

3.1 Metrics

General Comparison In the first part, we evaluated NaiveRAG, HyDE, LightRAG, GraphRAG, and NodeRAG in four benchmark data sets. For HotpotQA and MuSiQue benchmarks, we assess accuracy (Acc) to measure effectiveness and the average number of retrieved tokens (#Token) to evaluate efficiency. For the MultiHop-RAG benchmark, we adopt its original evaluation metric, Score (Sco), while still using #Token to gauge retrieval efficiency. Lastly, for the RAG-QA Arena benchmark, we continue to track #Token for efficiency and employ a win and tie ratio (W+T) against gold responses as a measure of performance across different methods.

Pairwise Comparison In this part, the evaluation focuses exclusively on the RAG-QA Arena benchmark, covering six domains: FiQA, Recreation, Writing, Lifestyle, Science, and Technology. We conduct comprehensive pairwise comparisons among all method combinations and calculate the corresponding win and tie rates for each matchup, thereby identifying the better RAG system.

3.2 Implementation details

By default, all these RAG methods are implemented with GPT 4o-mini, and the temperature is set to 0 throughout the evaluation. Meanwhile, we identify a potential unfairness in the current evaluation setup, evident in several key areas. Notably, the baselines vary in their choice of prompts used to synthesis the final response based on retrieved information. Therefore, we standardized the response prompts for every method. Our initiative to standardize these settings also benefits other methods like GraphRAG, improving their performance compared to their default setting, underscoring the broader value of establishing fair and consistent evaluation standards.

3.3 Results

General Comparison As shown in Part I of Table 1, NodeRAG consistently outperforms competing methods on HotpotQA, MuSiQue, and MultiHopRAG, demonstrating the highest accuracy while retrieving noticeably fewer tokens. For example, for MuSiQue, NodeRAG attains an accuracy of 46.29%, surpassing GraphRAG (41.71%) and LightRAG (36.00%). In HotpotQA, while NodeRAG achieves a slightly higher accuracy (89.50% vs. 89.00% for GraphRAG), it does so with only 5k retrieved tokens, which is 1.6k fewer than GraphRAG. In the RAG-QA Arena benchmark, graph-enhanced RAG systems exhibit a clear advantage over traditional approaches. Notably, NodeRAG achieves the highest win and tie ratio in each of the five domains while keeping retrieval costs minimal. For example, it attains a ratio of 94.9%, notably surpassing GraphRAG's 86.3% and LightRAG's 81.7% in the Lifestyle domain, and does so with less than half the retrieved tokens compared to the other models. It can also be noticed that graph-enhanced RAG systems generally retrieve fewer tokens than traditional RAG across all benchmarks. These results confirm NodeRAG's remarkable effectiveness and efficiency, demonstrating that our heterograph can significantly boost RAG performance across diverse tasks.

Pairwise Comparison Across all the six domains, NodeRAG consistently achieves higher win ratios against GraphRAG, LightRAG, NaiveRAG, and HyDE, demonstrating notable dominance, for instance, in the Lifestyle domain, NodeRAG achieves 0.640 win rate against GraphRAG, 0.623 against LightRAG, 0.800 against NaiveRAG and 0.526 against HyDE. GraphRAG, LightRAG, NaiveRAG, and HyDE show scattered successes, such as LightRAG edging out NaiveRAG (0.649 vs. 0.246) in Recreation, GraphRAG beats LightRAG (0.361 vs. 0.296) in Science, yet their overall win rates remain lower when compared to NodeRAG. Notably, these trends persist across other domains like Writing, Recreation, Science, and Tech, further underscoring NodeRAG's leading position, followed by LightRAG and GraphRAG, showing the superiority of our method.

In general, NodeRAG not only achieves the highest accuracy rate and the lowest retrieval token count in general benchmarks but also outperforms all other baselines in preference evaluation comparisons. This unparalleled performance in both accuracy and computational efficiency makes NodeRAG the optimal choice for a wide range of RAG tasks, from research applications to deployments in resource-constrained environments.

4 Ablation experiments

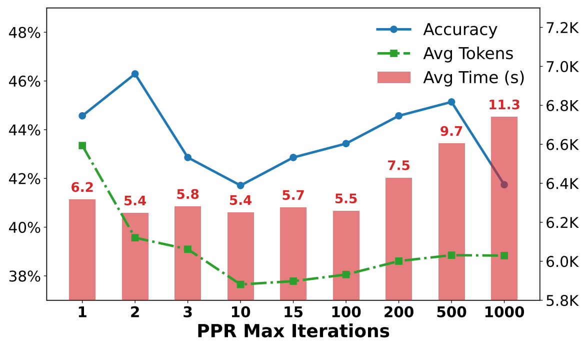

Figure 4: Ablation analysis on PPR iterations.

We conducted ablation experiments on the MuSiQue dataset, adhering to the same settings and evaluation metrics described earlier. We specifically examined the impact of four key submodules: shallow PPR, cross-node interactions, HNSW semantic edges, and dual search.

We first investigated the variation in PPR iterations and examined whether shallow PPR offers advantages. PPR, with a few iterations, performs better than deep PPR because it highlights important nodes that are closer to the entry points. Moreover, early stopping reduces unnecessary computational overhead, leading to improved retrieval efficiency.

Moreover, we evaluate the performance of applying the top-k vector similarity method to all node data in the graph. Although increasing the retrieval context, its performance remains lower than the basic version. This confirms the necessity of cross-nodes in our method, as they help identify important multi-hop nodes. Second, performing vector similarity solely on node data consistently outperforms the naive RAG approach of similarity on text chunks, demonstrating the advantages brought by graph-based data augmentation.

In addition, without integration of accurate search in dual search, accuracy drops to 44.57%, and the token count increases to 9.7k. This is because losing entity and high-level overview nodes as entry points causes nodes with long texts, such as text nodes, to have higher weights after shallow PPR. Since vector similarity entry nodes are more frequently connected to T nodes, while accurate entry nodes are more connected to S, A, and H nodes, the absence of accurate search disrupts this balance.

Finally, we investigate the effect of HNSW. HNSW introduces semantic edges to the heterograph, and removing this integration results in performance degradation. This is because HNSW enhances connectivity between semantically related nodes, enabling more efficient and meaningful retrieval.

Method	**Accuracy**	**Time (s)**	**Tokens (k)**
NodeRAG (Ours)	46.29%	4.05	5.96
w/o HNSW	41.71%	4.92	6.78
w/o Dual Search	44.57%	4.72	9.70
w/o Cross Node			
Top-$k = 10$	41.71%	4.15	4.27
Top-$k = 20$	43.43%	4.70	7.89
Top-$k = 30$	42.29%	4.80	11.62

Table 2: Ablation study of NodeRAG components.

5 Related Works

Retrieval-augmented generation Retrieval-Augmented Generation (RAG) systems (Gupta et al., 2024) boost LLM performance by retrieving domain-specific information from external sources. Traditional methods (Zhao et al., 2024; Fan et al., 2024; Lewis et al., 2020) embed queries and knowledge base entries into a shared vector space, retrieving top-K matches via similarity metrics. Though effective, these naive approaches have limitations, leading to improvements such as refined passage selection in JPR (Min et al., 2021), multi-hop reasoning in IR-CoT (Trivedi et al., 2022a), disambiguation trees in Tree of Clarifications (Kim et al., 2023), and hypothetical document generation in HyDE (Gao et al., 2022b). Other studies explore how document types affect RAG performance (Hsia et al., 2024). Challenges remain, including LLMs' context window limits (Cheng et al., 2024; Su et al., 2024) and difficulty with holistic corpus understanding (Jiang et al., 2024b). Domain-specific variants like BioRAG and MedicalRAG (Wang et al., 2024a; Wu et al., 2024; Jiang et al., 2024a) have emerged, yet RAG still struggles with tasks requiring broad synthesis, such as query-focused summarization.

RAG over Hierarchical Index To address the limitations of traditional RAG, advanced systems adopt hierarchical indexing to improve retrieval. Dense Hierarchical Retrieval (DHR) (Liu et al., 2021) combines document-level semantics with passage-level detail, while Hybrid Hierarchical Retrieval (HHR) (Arivazhagan et al., 2023) integrates sparse and dense methods for more precise retrieval. Models like RAPTOR (Sarthi et al., 2024) use tree structures for multilevel summarization. Graph-based RAGs (Trajanoska et al., 2023; Zhang et al., 2024) further enhance indexing by building knowledge graphs (Chen et al., 2020) and applying graph algorithms (Haveliwala et al., 2003). Examples include HippoRAG (Jiménez Gutiérrez et al., 2024) and KAPING (Baek et al., 2023), which improve organization and efficiency. GraphRAG (Edge et al., 2024) uses LLMs for graph construction and summary generation (Blondel et al., 2008; Traag et al., 2019), influencing works such as LightRAG (Guo et al., 2024), which balances high/low-level information with indexing efficiency. Despite these advances, existing methods underutilize the synergy between LLMs and graph structures. Our framework fills this gap through refined graph design and enhanced graph algorithms, achieving better retrieval accuracy and efficiency.

6 Conclusion

In this paper, we introduce NodeRAG, a novel framework designed to enhance RAG performance by optimizing graph structures in indexing for more effective and fine-grained retrieval. NodeRAG constructs a well-defined heterograph with functionally distinct nodes, balancing fine-grained understanding with a global perspective of the knowledge corpus. Experimental results demonstrate that NodeRAG outperforms existing methods across multi-hop reasoning benchmarks and open-ended retrieval tasks.

7 Limitations

While NodeRAG offers practical improvements in indexing and querying, its graph size and structural complexity still scale with the size of the corpus, similar to previous graph-based methods. This can limit scalability and reduce efficiency when applied to extremely large-scale corpora.

References

Manoj Ghuhan Arivazhagan, Lan Liu, Peng Qi, Xinchi Chen, William Yang Wang, and Zhiheng Huang. 2023. Hybrid hierarchical retrieval for open-domain question answering. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10680–10689.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177.

Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. 2020. Knowledge graph completion: A review. *Ieee Access*, 8:192435–192456.

Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2024. Lift yourself up: Retrieval-augmented text generation with self-memory. *Advances in Neural Information Processing Systems*, 36.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint 2404.16130*.

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. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 6491–6501.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022a. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022b. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint 2410.05779*.

Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. 2024. A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions. *arXiv preprint 2410.12837*.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, and 1 others. 2024a. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.

Rujun Han, Yuhao Zhang, Peng Qi, Yumo Xu, Jenyuan Wang, Lan Liu, William Yang Wang, Bonan Min, and Vittorio Castelli. 2024b. Rag-qa arena: Evaluating domain robustness for long-form retrieval augmented question answering. *arXiv preprint arXiv:2407.13998*.

Taher Haveliwala, Sepandar Kamvar, and Glen Jeh. 2003. An analytical comparison of approaches to personalizing pagerank. Technical report, Stanford.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Jennifer Hsia, Afreen Shaikh, Zhiruo Wang, and Graham Neubig. 2024. Ragged: Towards informed design of retrieval augmented generation systems. *arXiv preprint 2403.09040*.

Xinke Jiang, Yue Fang, Rihong Qiu, Haoyu Zhang, Yongxin Xu, Hao Chen, Wentao Zhang, Ruizhe Zhang, Yuchen Fang, Xu Chu, and 1 others. 2024a. Tc-rag: Turing-complete rag's case study on medical llm systems. *arXiv preprint arXiv:2408.09199*.

Ziyan Jiang, Xueguang Ma, and Wenhu Chen. 2024b. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *arXiv preprint arXiv:2405.14831*.

Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. 2023. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. *arXiv preprint arXiv:2310.14696*.

Yi-Xiu Kong, Gui-Yuan Shi, Rui-Jie Wu, and Yi-Cheng Zhang. 2019. k-core: Theories and applications. *Physics Reports*, 832:1–32.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

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

Ye Liu, Kazuma Hashimoto, Yingbo Zhou, Semih Yavuz, Caiming Xiong, and Philip S Yu. 2021. Dense hierarchical retrieval for open-domain question answering. *arXiv preprint arXiv:2110.15439*.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

James MacQueen and 1 others. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.

Sewon Min, Kenton Lee, Ming-Wei Chang, Kristina Toutanova, and Hannaneh Hajishirzi. 2021. Joint passage ranking for diverse multi-answer retrieval. *arXiv preprint arXiv:2104.08445*.

Jinyoung Park, Ameen Patel, Omar Zia Khan, Hyunwoo J Kim, and Joo-Kyung Kim. 2023. Graph-guided reasoning for multi-hop question answering in large language models. *arXiv preprint arXiv:2311.09762*.

Diego Sanmartin. 2024. Kg-rag: Bridging the gap between knowledge and creativity. *arXiv preprint arXiv:2405.12035*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. *arXiv preprint arXiv:2401.18059*.

Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks*, 5(3):269–287.

Robik Shrestha, Yang Zou, Qiuyu Chen, Zhiheng Li, Yusheng Xie, and Siqi Deng. 2024. Fairrag: Fair human generation via fair retrieval augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11996–12005.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models. *arXiv preprint arXiv:2403.10081*.

Yixuan Tang and Yi Yang. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2024. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19080–19088.

SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. 2024. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*.

Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. 2019. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12.

Milena Trajanoska, Riste Stojanov, and Dimitar Trajanov. 2023. Enhancing knowledge graph construction using large language models. *arXiv preprint arXiv:2305.04676*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022a. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022b. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Chengrui Wang, Qingqing Long, Meng Xiao, Xunxin Cai, Chengjun Wu, Zhen Meng, Xuezhi Wang, and Yuanchun Zhou. 2024a. Biorag: A rag-llm framework for biological question reasoning. *arXiv preprint arXiv:2408.01107*.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024b. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214.

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2023. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv preprint arXiv:2308.09729*.

Junde Wu, Jiayuan Zhu, Yunli Qi, Jingkun Chen, Min Xu, Filippo Menolascina, and Vicente Grau. 2024. Medical graph rag: Towards safe medical large language model via graph retrieval-augmented generation. *arXiv preprint arXiv:2408.04187*.

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 preprint arXiv:1809.09600*.

Yuzhe Zhang, Yipeng Zhang, Yidong Gan, Lina Yao, and Chen Wang. 2024. Causal graph discovery with retrieval-augmented generation based large language models. *arXiv preprint arXiv:2402.15301*.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

A Comparison of RAG System Performance

Datasets	Corpus Size	Index Time			Storage Usage			Query Time				Average Retrieval Tokens			
		Graph	Light	Node	Graph	Light	Node	Graph-L	Graph-G	Light	Node	Graph-L	Graph-G	Light	Node
HotpotQA	1.93M	66min	39min	21min	227MB	461MB	214MB	2.66s	26.69s	5.58s	3.98s	6680.65	810529	7176.73	5079.40
Musique	1.84M	76min	90min	25min	255MB	492MB	250MB	2.94s	22.65s	6.53s	4.05s	6616.84	1111073	7458.34	5960.25
MultiHop	1.41M	50min	58min	24min	141MB	276MB	137MB	4.15s	34.45s	7.10s	4.89s	7367.54	920780	8920.00	5259.99
Arena-Fiqa	1.65M	45min	49min	19min	112MB	240MB	117MB	8.95s	28.94s	13.35s	8.86s	6819.45	713560	7721.73	3381.72
Arena-Lifestyle	1.64M	52min	59min	18min	138MB	278MB	125MB	7.54s	33.09s	10.43s	6.79s	6860.26	895964	6822.32	3350.35
Arena-Recreation	0.93M	34min	33min	10min	89MB	172MB	80MB	5.10s	23.10s	8.01s	6.90s	6669.95	564636	6249.31	3448.38
Arena-Science	1.43M	43min	46min	17min	116MB	236MB	111MB	8.05s	35.79s	14.28s	8.85s	6759.15	778051	7111.80	4284.13
Arena-Tech	1.72M	54min	54min	14min	133MB	276MB	139MB	7.35s	28.64s	8.89s	6.74s	6755.46	741690	6922.55	3821.78
Arena-Writing	1.82M	50min	71min	13min	151MB	309MB	157MB	5.65s	40.12s	10.70s	5.40s	6477.72	877354	6364.59	3373.34

Table 3: Performance metrics for RAG methods, including Index Time, Storage Usage, Query Time, and Retrieval Tokens across various datasets. *Graph* denotes GraphRAG, with *Graph-l* representing its local mode and *Graph-G* its global mode. *Light* refers to LightRAG in hybrid mode, while *Node* represents our proposed method.

The table 3 presents the system performance of mainstream graph-based RAG methods and our proposed approach. Compared to previous work, our method demonstrates superior performance across multiple datasets and in open-ended head-to-head evaluations, while also achieving better system-level efficiency. All evaluations in the table were conducted using the default indexing settings of each RAG method, with the query settings and the prompt details provided in Appendix B.2. Notably, our method demonstrates a significant advantage in indexing time, which is crucial for practical deployment. This advantage is attributed to the construction process of our Hetero Graph, which not only creates a more fine-grained and semantically meaningful graph structure but also carefully considers the algorithmic complexity of the retrieval process.

NodeRAG also exhibits relatively better storage efficiency. Although the total number of nodes in our expanded graph is significantly larger than in previous graph structures, the combination of selective embedding and dual search effectively reduces the number of embedded nodes, leading to a more efficient storage strategy. Moreover, our unified information retrieval approach results in reduced query time. While the GraphRAG local search (Graph-l) relies purely on vector similarity—similar to our "without cross-node" setting mentioned in Section 4—and achieves faster search speeds, its global mode (Graph-G) experiences significantly higher query times, exceeding 20 seconds with a concurrency of 16. This is due to its reliance on LLM-based traversal of all community information, leading to a substantial number of retrieval tokens. Given the considerable time and computational overhead associated with Graph-G queries, we conducted a full evaluation only on the MuSiQue dataset. For other datasets, query time and retrieval token statistics were estimated based on a sample of 20 selected queries. Further details on the ablation study of GraphRAG can be found in the Appendix B.4.

In contrast, our method leverages the heterograph and graph algorithms to achieve unified information retrieval, effectively capturing meaningful information needs across multiple levels within a single framework while maintaining efficient query speed. Finally, the nodes within the heterograph are connected in a fine-grained structure, ensuring that more relevant text is retrieved with relatively fewer retrieval tokens.

B Experiment details

B.1 Datasets

We evaluate Node RAG's performance across four different benchmarks: HotpotQA, MuSiQue, MultiHop-RAG and RAG-QA Arena. However, the original question formats of HotpotQA and MuSiQue required selecting the most relevant passages from multiple documents, incorporating multi-hop reasoning details. This setup no longer aligns with mainstream RAG methods, as modern approaches perform indexing over an entire corpus and subsequently retrieve information from the indexed data. To adapt to this paradigm, we concatenate all passages into a unified corpus, transforming the task into retrieving multi-hop relevant information from the entire corpus. This modification makes the task more challenging compared to the original setting. The evaluation metrics for HotpotQA and MuSiQue are divided into two aspects: the quality of the retrieved documents and the accuracy of the final answer, measured by metrics such as F1 score. However, current RAG methods retrieve not only text chunks but also more flexible forms of information, making it difficult to assess retrieval quality using traditional top-k document evaluation. Moreover, metrics like F1 score have become less effective in evaluating answers generated by modern generative models. Therefore, we adopt the *LLM-as-a-Judge* approach, leveraging LLMs to assess the final accuracy of the generated answers.The MultiHop and RAG-QA Arena dataset settings provide a strong evaluation framework for current RAG methods. Therefore, we follow the original benchmark's proposed testing methodology and evaluation metrics. Further details regarding the benchmark settings are described below.

HotpotQA is a multi-hop question-answering dataset where each question requires combining information from multiple documents to find the correct answer. It encourages deeper reasoning by providing supporting facts—specific sentences from the texts that lead to the solution. Questions range widely across domains and often involve bridging or comparison to ensure more complex, multi-step reasoning. This makes HotpotQA a critical benchmark for evaluating advanced reading comprehension models. We sampled 200 questions from the final dataset for evaluation.

MuSiQue is also a multi-hop question-answering dataset that challenges models to combine information across multiple documents in a structured, step-by-step manner. Each question is designed to require several reasoning steps, ensuring that simple "shortcut" approaches do not suffice. As a result, MuSiQue serves as a rigorous test of advanced reading comprehension, demanding that systems accurately connect disparate pieces of evidence to arrive at correct answers. We also sample 175 questions for the evaluation

MultiHop-RAG is a multi-hop question-answering dataset that includes four distinct question types: comparison query, null query, inference query, and temporal query. From this dataset, we curated 375 questions to evaluate our approach. Each query in MultiHop requires synthesizing information from multiple sources, testing a model's ability to perform bridging inferences, handle temporal relationships, and make higher-order logical connections. This diversity in question types provides a rigorous benchmark for assessing whether RAG methods can integrate scattered pieces of evidence.

RAG-QA Arena is a new evaluation framework designed to assess the quality of retrieval-augmented generation (RAG) systems on long-form question answering. It builds on Long-form RobustQA (LFRQA), a dataset of 26K queries across seven domains including writing, tech, science, recreation and lifestyle. Each LFRQA entry features a coherent, human-written answer grounded in multiple documents. RAG-QA Arena leverages LLMs as evaluators, directly comparing a system's generated answer with the 'gold' long-form answer from LFRQA. Experimental results show that these model-based comparisons correlate highly with human judgments, making it a challenging yet reliable benchmark for testing both cross-domain robustness and the ability to produce integrated, long-form responses.

B.2 Baselines

We compare NodeRAG against several strong and widely used RAG methods. By default, all these RAG methods implement their indexing process using GPT-4o-mini. However, we identify a potential unfairness in the current evaluation setup, particularly in several key areas. To ensure the correctness and validity of the evaluation data, it is crucial to standardize both the final answer response prompt and the model temperature settings. Using different response prompts or varying temperature settings for answer generation introduces inconsistencies, as a higher temperature setting may yield responses that receive a better LLM preference score compared to those generated with a lower temperature. A critical point to consider is that, as RAG methods, the primary focus of evaluation should be the quality of the retrieved context rather than the final generated answer. Therefore, to ensure that final accuracy metrics accurately reflect the quality of the retrieved context, the final answer generation process and model settings should remain consistent across all methods. Hence, we set the temperature to 0 across the entire evaluation and standardized response prompts for every method. The unified prompt is illustrated in appendix E. Our initiative to standardize these settings also benefits other methods, such as GraphRAG, improving their performance compared to their default settings. This underscores the broader value of establishing fair and consistent evaluation standards. Additionally, traditional evaluation methods such as top-k retrieval comparison have become increasingly difficult to apply uniformly, as retrieval is no longer restricted to isolated text chunks. To address this challenge, we propose a new evaluation standard that leverages retrieval tokens as an efficiency metric. This approach ensures that retrieval methods achieve better effectiveness while utilizing fewer retrieval tokens, promoting a more efficient and fair comparison framework. Current methods can only control the number of retrieval tokens through hyperparameter tuning. Although precise control over the exact number of tokens is not possible, we consider maintaining the average number of retrieval tokens within the range of 5K to 10K to be a reasonable and fair comparison criterion. Below, we provide a detailed introduction to each method along with its specific settings for reference.

Naive RAG This method serves as a standard baseline among all existing RAG systems. It first divided input document into several text chunks and encoded them into a vector space utilizing text embeddings. Then retrieve related text chunks based on similarity of query representations. The number of retrieval tokens can be adjusted through the top-k parameter.

HyDE HyDE serves as an improved method over traditional RAG systems. It first generates "hypothetical" texts that capture the essence of a query. It then uses this generated text to retrieve relevant documents from a large corpus, employing vector similarity in an embedding space. This method modifies the input query at the frontend without altering the text chunks or their embeddings. Therefore, we can still use the top-k parameter to control the number of retrieval tokens.

GraphRAG This approach starts by segmenting the input text into chunks and extracting the entities and relationships within them, forming a graph structure. This graph is then divided into multiple communities at different levels. At query time, GraphRAG identifies the relevant entities from the question and synthesizes answers by referencing these corresponding community summaries. Compared to traditional RAG methods, GraphRAG provides a more structured and high-level understanding of the entire document. Through our experiments, we observed that under the default settings, the number of queries in GraphRAG's local mode resulted in a higher retrieval token count than the naive retrieval approach. To ensure a fair comparison, we proportionally reduced its parameters and standardized its prompt to match our unified prompt. The ablation study in Appendix B.4 demonstrates that after these adjustments, GraphRAG's accuracy improved, further validating the fairness of our evaluation methodology. Additionally, we analyzed both the local and global modes of GraphRAG. Our findings indicate that the global mode introduces significant additional overhead in terms of time and computational cost while providing only marginal improvements compared to the local mode. This result is further supported by our ablation study, which shows that the local mode achieves better efficiency and effectiveness.

LightRAG LightRAG is an improved approach based on GraphRAG, designed to minimize computational overhead while enhancing the comprehensiveness of retrieved information through dual-level retrieval. This leads to more efficient retrieval and a better balance between effectiveness and speed compared to GraphRAG. Similar to GraphRAG, the default settings of LightRAG result in a higher retrieval token count than the Naïve approach. To ensure a fair comparison, we proportionally adjusted its hyperparameters to maintain the number of retrieval tokens within the range of 5K to 10K.

B.3 NodeRAG Graph Statistics

The table 4 presents the number of each type of node in the indexed graph for each dataset, including entity (N), relationship (R), semantic unit (S), attribute (A), high-level elements (H), high-level overview (O), and text (T). These counts are detailed in the type statistics section. Additionally, the graph statistics provide information on the total number of nodes, the number of non-HNSW edges, HNSW edges, and

Datasets	Corpus Tokens	Type Statistics							Graph Statistics			
		T	S	N	R	A	O	H	Nodes	Non-HNSW Edge	HNSW Edge	Edge
HotpotQA	1.93M	1985	15905	88863	56578	684	4479	4479	172603	283543	487731	759812
MuSiQue	1.84M	1907	18714	99840	61964	795	5700	5700	193922	316029	583126	888966
MultiHop-RAG	1.41M	1532	10986	43184	29286	685	2289	2289	90144	171410	203199	367486
Arena-Fiqa	1.65M	1821	9027	32470	27422	508	1714	1714	74605	143916	154109	295165
Arena-Lifestyle	1.64M	1794	9400	39464	27895	518	2221	2221	83461	149225	174461	318073
Arena-Recreation	0.93M	1003	5542	26382	16938	413	1969	1969	54180	93228	117915	207449
Arena-Science	1.43M	1583	8010	32232	23092	551	2515	2515	70425	127719	149424	276963
Arena-Tech	1.72M	1910	10837	37724	29691	534	2633	2633	85888	167950	193159	354033
Arena-Writing	1.82M	1937	11008	42723	29338	705	4435	4435	94259	149552	298565	442397

Table 4: Comprehensive dataset statistics, detailing corpus size, type statistics (T, S, N, R, A, O, H), and graph statistics. The graph statistics include the number of document compilation nodes, HNSW semantic edges, and total edges. Each value represents a key metric relevant to graph-based document processing and retrieval.

the total number of edges. The data indicate that the number of HNSW edges is comparable to that of non-HNSW edges, highlighting the integration of semantic connections within the graph. Notably, overlapping edges are removed when merging non-HNSW and HNSW edges. For instance, in the MultiHop-RAG benchmark, there are 171,410 non-HNSW edges and 203,199 HNSW edges. However, the total number of edges after merging is 367,486, which is only 7,123 fewer than the sum of both edge types. This indicates the uniqueness of these two types of edges and highlights the effectiveness of the HNSW algorithm.

B.4 Graph RAG Ablation

Method	Accuracy	Avg. Processing Time	Avg. Tokens
GraphRAG (default)	37.14%	4.82s	10.4k
Graph-L	41.71%	2.94s	6.6k
Graph-G	33.14%	22.65s	1.11M

Table 5: Performance Comparison of GraphRAG Variants. Default is the default setting. Local and global represent the local and global modes under unified prompt and hyperparameter settings.

The default setting of GraphRAG, along with its own prompting mechanism, is not standardized for evaluation, as both the number of retrieval tokens and the choice of prompts significantly impact performance. Hence, we introduce a unified prompt and adjust the hyperparameters of GraphRAG to ensure a fair comparison within a specific range. As shown in the table B.4, GraphRAG with our unified prompt achieves higher performance, demonstrating that the original prompting strategy is not optimal for this task. This further ensures fairness in comparison, as performance is influenced solely by the quality of the retrieved context. Moreover, the global mode of GraphRAG requires significantly longer processing time and incurs higher computational costs due to the LLM analyzing all community summaries, leading to increased complexity and resource consumption. Additionally, for multi-hop question answering, this approach results in degraded performance. Therefore, we conducted an exploratory ablation study only on the MuSiQue dataset, while for other datasets, we estimated query time and retrieval token statistics based on sampled queries.

C Algorithm details

C.1 Terminology

Abbr.	Full Name	Description	Function	Example
T	Text	Full-text chunks from the original source. They contain rich detailed information, although it integrates a large amount of unrelated semantic information.	Retrievable; Entry points from vector similarity	*"Hinton was awarded the Nobel Prize in 2023 for his groundbreaking contributions to artificial intelligence, particularly in deep learning. His pioneering work on backpropagation laid the foundation for modern neural networks, influencing both academia and industry. The recognition came amid increasing discussions on the ethical implications of AI, with Hinton himself advocating for responsible AI development and regulation."*
S	Semantic Unit	Local summaries that are independent and meaningful events summarized from text chunks. They serve as a middle layer between text chunks and entities, acting as the basic units for graph augmentation and semantic analysis.	Retrievable; Entry points from vector similarity.	*"Hinton was awarded the Nobel Prize for inventing backpropagation."*
A	Attribute	Attributes of key entities, derived from relationships and semantic units around important entities.	Retrievable; Entry points from vector similarity.	*"Geoffrey Hinton, often referred to as the "Godfather of Deep Learning," is a pioneer in the field of artificial intelligence. In 2024, he was awarded the Nobel Prize for his contributions to AI and deep learning. "*
H	High-Level Element	Insights summarizing graph communities. Encapsulates core information or any high level ideas from a community.	Retrievable; Entry points from vector similarity.	*"Due to the increasing importance of AI, the Nobel Prize is awarded to scholars who have made tremendous contributions to the field of AI."*
O	High-Level Overview	Titles or keywords summarizing high-level elements.	Non-Retrievable; Entry points from accurate search.	*"AI significance"*
R	Relationship	Connections between entities represented as nodes. Acts as connector nodes and secondary retrievable node.	Retrievable; Non-Entry points	*"Hinton received the Nobel Prize."*
N	Entity	Named entities such as people, places, or concepts.	Non-Retrievable; Entry points from accurate search..	*"Hinton," "Nobel Prize"*

Table 6: Node Types in the heterograph

C.2 K-core & Betweenness centrality

In this subsection, we present the methodology for identifying important entities and generating their attribute summaries, ensuring alignment with the mathematical framework established in the main text.

The selection of important entities, denoted as N^*, is based on two fundamental structural graph metrics: K-core decomposition and betweenness centrality. These metrics collectively ensure that the selected nodes are not only structurally integral but also play a pivotal role in facilitating information flow.

The K-core decomposition, denoted as $K(\mathcal{G}^1)$, identifies nodes within densely connected subgraphs, ensuring that selected entities contribute significantly to the structural cohesion of the graph. Meanwhile, betweenness centrality, denoted as $B(\mathcal{G}^1)$, highlights nodes that serve as critical intermediaries between different regions of the graph, capturing entities essential for information dissemination.

The process of identifying important entities follows the steps outlined in Algorithm 1.

Algorithm 1 Identification of Important Entities

Input: Graph $\mathcal{G}^1 = (\mathcal{V}, \mathcal{E})$
Output: Important entity set N^*
Step 1: Compute K-core decomposition
Compute the core threshold:

$$k_{\text{default}} = \lfloor \log(|\mathcal{V}|) \times \left(\frac{\sum_{v \in \mathcal{V}} \deg(v)}{|\mathcal{V}|} \right)^{1/2} \rfloor$$

Extract the K-core subgraph:

$$K(\mathcal{G}^1) = \{ v \in \mathcal{V} \mid \deg_{\mathcal{G}^1}(v) \geq k_{\text{default}} \}$$

Step 2: Compute betweenness centrality
for each $v \in \mathcal{V}$ **do**
Approximate betweenness centrality using shortest-path sampling:

$$b(v) = \text{betweenness\_centrality}(\mathcal{G}^1, k = 10)$$

end for
Compute the average betweenness centrality:

$$\bar{b} = \frac{\sum_{v \in \mathcal{V}} b(v)}{|\mathcal{V}|}$$

Compute the scale factor:

$$\text{scale} = \lfloor \log_{10}(|\mathcal{V}|) \rfloor$$

Step 3: Select important nodes
for each $v \in \mathcal{V}$ **do**
if $b(v) > \bar{b} \times \text{scale}$ **then**
Add v to $B(\mathcal{G}^1)$
end if
end for
Compute the final set of important entities:

$$N^* = K(\mathcal{G}^1) \cup B(\mathcal{G}^1)$$

Return N^*

C.3 Semantic Matching within Community

To establish meaningful semantic relationships among high-level element nodes, we propose the *Semantic Matching within Community* algorithm. This algorithm ensures that entities with strong semantic similarities are connected within their respective communities. The motivation behind this approach is

Algorithm 2 Semantic Matching within Community

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, node embeddings $\Phi(\mathcal{V})$, community partition $\{\mathcal{C}_n\}$
Output: Semantic edges $\mathcal{E}_h$
Step 1: Select high-level element nodes
Extract nodes with labels S, A, or H:

$$V_{\{S,A,H\}} = \{v \in \mathcal{V} \mid \psi(v) \in \{S, A, H\}\}$$

Step 2: Apply K-means clustering to node embeddings
Set number of clusters:

$$K = \sqrt{|\mathcal{V}_{\{S,A,H\}}|}$$

Perform K-means clustering on $\mathcal{V}_{\{S,A,H\}}$), obtaining clusters $\{\mathcal{S}_k\}$
Step 3: Establish semantic edges within communities
for each community $\mathcal{C}_n$ **do**
  for each cluster $\mathcal{S}_k$ **do**
    Identify nodes within the community and cluster:

$$\mathcal{V}_{\mathcal{C}_n,\mathcal{S}_k} = \mathcal{V}_{\{S,A,H\}} \cap C_n \cap S_k$$

    for each pair (v, v') where $v \in \{S, A\}, v' \in H$ **do**
      Add semantic edge:

$$e_h(v, v') \in \mathcal{E}_h$$

    end for
  end for
end for
Return $\mathcal{E}_h$

to organically integrate H nodes into the graph structure by establishing connections with semantically related nodes within the same community. Formally, the process is summarized in Algorithm 2.

The algorithm begins by identifying nodes that belong to three specific categories: structure nodes (S), attribute nodes (A), and high-level nodes (H). These nodes are collectively defined as:

$$\mathcal{V}_{\{S,A,H\}} = \{v \in \mathcal{V} \mid \psi(v) \in \{S, A, H\}\}$$

Since these nodes exhibit inherent semantic relationships, we cluster them based on their embeddings, which capture their contextual meaning. To partition the nodes into semantically similar groups, we apply the K-means clustering algorithm (MacQueen et al., 1967) to the embedding representations of $\mathcal{V}_{\{S,A,H\}}$.

which balances computational efficiency and granularity. This clustering process results in a partitioning of nodes into K semantic clusters, denoted as $\mathcal{S}_k$, where each cluster contains nodes with closely related semantic representations.

After clustering, the algorithm establishes edges between semantically related nodes within the same community. Communities are predefined structural subgroups in the graph, denoted as $\mathcal{C}_n$, ensuring that local relationships are preserved. For each community-cluster pair, semantic edges are introduced between nodes in $\mathcal{V}_{\{S,A\}}$ and nodes in $\mathcal{V}_H$. Specifically, for any node pair (v, v'), where $v \in \mathcal{V}_{\{S,A\}}$ and $v' \in \mathcal{V}_H$, an edge $e_h(v, v')$ is established if both nodes belong to the same community and the same semantic cluster.

By integrating semantic matching within community constraints, this algorithm enhances the structural integrity of the graph while maintaining computational feasibility. The choice of K-means clustering efficiently groups nodes with similar semantic properties, while the enforcement of community constraints ensures that edges are only formed between nodes that naturally belong to the same substructure. Consequently, the proposed method balances semantic consistency and graph locality, making it well-suited for applications requiring structured knowledge representation and retrieval.

C.4 Dual Search

To efficiently locate relevant entry points within the Hetero Graph $\mathcal{G}$, we propose the *Dual Search* algorithm, which integrates exact matching on structured nodes and vector similarity search on rich information nodes. This hybrid approach ensures a balance between precision and recall by leveraging both symbolic and dense representations. The core idea is to utilize exact string matching for well-structured nodes while employing approximate nearest neighbor search for nodes containing rich contextual information. By doing so, the algorithm improves both retrieval accuracy and robustness to query variations.

Given a query, a LLM extracts a set of relevant entities, denoted as N^q, and embeds the query into a vector representation $\mathbf{q}$. Entry points in the graph are then determined by:

$$\mathcal{V}_{\text{entry}} = \{v \in \mathcal{V} \mid \Phi(v, N^q, \mathbf{q})\},$$

where the condition function $\Phi(v, N^q, \mathbf{q})$ determines whether a node qualifies as an entry point:

$$\Phi(v, N^q, \mathbf{q}) = \begin{cases} v \in \mathcal{V}_{\{N,O\}} \wedge \mathcal{M}(N^q, v), \\ v \in \mathcal{V}_{\{S,A,H\}} \wedge \mathcal{R}(\mathbf{q}, v, k). \end{cases}$$

Here, the exact matching function $\mathcal{M}(N^q, v)$ returns true if node v matches one of the extracted entities in N^q. This ensures that titles or named nodes such $\mathcal{V}_{N,O}$ are retrieved deterministically. Meanwhile, the similarity-ranking function $\mathcal{R}(\mathbf{q}, v, k)$ applies HNSW, selecting the top-k most similar nodes to $\mathbf{q}$ in vector space. This is particularly beneficial for context-rich nodes for $v \in \mathcal{V}_{S,A,H}$, which may not contain exact query terms but remain semantically relevant.

One of the major strengths of Dual Search is its ability to mitigate errors from noisy or ambiguous queries. Since N and O nodes serve exclusively as entry points and are not directly retrieved, any errors in exact matching do not propagate into the retrieval results. Furthermore, Personalized PageRank (PPR) is applied to refine retrieval by selecting highly weighted cross nodes—nodes that are both structurally significant and semantically aligned with multiple entry points. This graph-based expansion helps compensate for potential errors in entity extraction and ensures robustness against query variations. Another advantage of this approach is its scalability. By employing HNSW, the algorithm avoids the computational overhead of exhaustive similarity searches, making it efficient even on large-scale graphs.

C.5 Increment update

NodeRAG supports incremental updates. For any newly added data, the system applies the procedures $\mathcal{G}^1$ to $\mathcal{G}^5$ only to the new portion, and then concatenates the resulting graph with the existing graph. Nodes with identical names are merged automatically, including their associated high-level elements. During the retrieval process, graph algorithms can be applied to identify nearby or semantically related nodes, ensuring the new nodes are integrated into the retrieval results.

Although graph construction and algorithmic processing are carried out in batches, the use of heterogeneous graph structures and graph algorithm properties effectively mitigates inconsistencies introduced by batching. This design enables the integration of new data without the need to re-index the entire database, thereby avoiding significant computational overhead.

D Case Study

Question1	Why isn't there public transport on Christmas day in the UK?
NodeRAG	The absence of public transport on Christmas Day in the UK can be attributed to a combination of historical, cultural, and economic factors. **Historically**, public transport services were more prevalent on Christmas Day **before World War II**, as many people worked on that day and required transportation. However, over the years, societal norms have shifted, leading to a significant reduction in demand for public transport services on Christmas Day. **In contemporary society**, Christmas is widely recognized as a time for family gatherings and celebrations, which diminishes the need for public transport. Many people choose to spend the day at home or with relatives, resulting in a lower demand for transit services. Consequently, transport companies have adapted to this change by suspending services on Christmas Day, allowing transit workers to enjoy the holiday with their families as well. Additionally, the **operational costs** associated with running public transport services on Christmas Day can be significant. With minimal passenger demand, the financial viability of providing such services becomes questionable. Railways and other transport providers often use this time for maintenance and engineering work, taking advantage of the reduced disruption to carry out necessary improvements.
GraphRAG	Public transport in the UK is significantly limited on Christmas Day due to a combination of historical, cultural, and operational factors. Traditionally, Christmas Day is a public holiday in the UK, and many businesses, including public transport services, close to allow employees to spend time with their families and celebrate the holiday. This practice is rooted in the cultural significance of Christmas, which emphasizes family gatherings and community celebrations. The limited operation of public transport on Christmas Day is also influenced by the low demand for services. Historically, fewer people travel on this day compared to other holidays or weekends, as many individuals prefer to stay home or visit family. As a result, transport providers often find it economically unfeasible to run full services when the number of passengers is expected to be minimal. In recent years, some transport services have begun to offer limited services on Christmas Day, particularly in major cities like London, where there may be a demand for travel to events or gatherings. However, these services are typically not as extensive as those available on other public holidays or regular days. Overall, the combination of cultural practices, low demand, and operational considerations contributes to the lack of public transport on Christmas Day in the UK.
LightRAG	In most parts of the world, official or religious holidays are taken very seriously. On such days, employers typically have to pay higher rates to staff who are willing to work, but that effort often doesn't pay off due to low customer turnout. Families tend to arrange their travel plans by Christmas Eve to avoid the need for transportation on Christmas Day. Christmas avoids unnecessary travel because it's typically a day filled with home activities, and it allows workers in various roles to have a day off. The lack of public demand and the staff's resistance contributed to the lack of services on Christmas Day. Christmas Day is also an opportune time to perform engineering work on railways with minimal disruption. The railroad industry is capable of completing extensive projects during this period, thanks to thorough planning and pre-staging.
LLM Decision	ORDER: {"1": "GraphRAG", "2": "NodeRAG"} Both answers provide a comprehensive explanation, but answer 2 adds **historical context** about public transport before World War II and mentions the use of the holiday for maintenance work, offering slightly **more depth**. ORDER: {"1": "LightRAG", "2": "NodeRAG"} Both answers provide a comprehensive explanation, but answer 2 adds **historical context** about the change in public transport services since World War II, which **enriches the explanation**. Therefore, answer 2 is slightly better.

Table 7: Sample Table with Multi-line Text

Question2	What are rabbit hides used for in Minecraft?
NodeRAG	In Minecraft, rabbit hides serve a couple of useful purposes, particularly in crafting and resource management. One of the primary uses of rabbit hides is to craft leather. Players can combine four rabbit hides in a 2x2 crafting grid to create one piece of leather. This is especially beneficial in Survival Multiplayer (SMP) scenarios where finding other animals for leather can be challenging. Additionally, rabbits can drop a unique item known as the **"rabbit's foot"**, which is an important ingredient for brewing potions. The rabbit's foot is specifically used in the crafting of the **Potion of Leaping**, which grants players the ability to jump higher for a limited time. Thus, rabbit hides and their associated drops contribute significantly to crafting and potion-making in the game, enhancing the overall gameplay experience.
GraphRAG	In Minecraft, rabbit hides are primarily used to craft leather. Players can combine four rabbit hides in a 2x2 crafting grid to create one piece of leather. This crafting option is particularly useful for players who may not have access to cows, which are the traditional source of leather in the game. Additionally, rabbit hides can be beneficial in multiplayer servers that utilize entity clears, as finding animals can be challenging. By using rabbit hides to craft leather, players have an alternative method to obtain this valuable resource, which is essential for crafting various items, including armor and books.
LightRAG	In Minecraft, Rabbit Hides are primarily used as a crafting resource. They can be combined with other materials to create leather, which is an essential item in the game. Leather is used for crafting various items, including armor (like leather armor), books, item frames, and more. The process of obtaining Rabbit Hides involves hunting rabbits, which can be found in various biomes. Once collected, players can use the hides in their crafting recipes to enhance their gameplay experience through resource management and item creation. This highlights the importance of Rabbit Hides within the broader crafting mechanics of Minecraft.
LLM Decision	ORDER: {"1": "TGRAG", "2": "GraphRAG"} Both answers are truthful, but answer 1 provides more **detailed information** about the uses of rabbit hides, including the additional use of **rabbit's foot for potion-making** Therefore, answer 1 is better. ORDER: {"1": "TGRAG", "2": "LightRAG"} Both answers are truthful, but answer 1 provides more **detailed information** about the uses of rabbit hides, including the crafting of leather and **the rabbit's foot for potions**. Therefore, answer 1 is better.

Table 8: Sample Table with Multi-line Text

E Overview Prompting in NodeRAG

---Role---
You are a thorough assistant responding to questions based on retrieved information.

---Goal---
1. **Provide a clear and accurate response**. Carefully review and verify the retrieved data, and integrate any relevant necessary knowledge to comprehensively address the user's question.
2. **Do not fabricate information**. If you are unsure of the answer, just say so.
3. **Do not include details not supported by the provided evidence.**

---Target response length and format---
Multiple Paragraphs

---Retrieved Context---
{info}

---Query---
{query}

Response Prompt

--Goal--
Generate a concise summary of the given entity, capturing its essential attributes and important relevant relationships.

--Requirement--
1. The summary should read like a character sketch in a novel or a product description, providing an engaging yet precise overview.
2. Ensure the output **only includes the summary** of the entity without any additional explanations or metadata.
3. The **length must not exceed 2000** words but can be shorter if the input material is limited.
4. Focus on **distilling the most important insights** with a smooth narrative flow, highlighting the entity's core traits and meaningful connections.

--Input--
Entity: {entity}
Related Semantic Units: {semantic_units}
Related Relationships: {relationships}

Community Summary Prompt

--Goal--
Please break down the following query into a single list.

--Requirement--
1. Each item in the list should either be a **main entity** (such as a key noun or object).
2. If you have high confidence about the user's intent or domain knowledge, you may also include **closely related terms**.
3. If uncertain, please **only extract entities and semantic chunks** directly from the query. Please try to reduce the number of common nouns in the list. Ensure all elements are organized within one unified list.

--Input--
Query: {query}

Question Decompose Prompt

--Goal--
You will be given a string containing tuples representing relationships between entities. **Your task is to reconstruct each relationship in the following format**:
{'source': 'ENTITY_A', 'relation': 'RELATION_TYPE', 'target': 'ENTITY_B'}.

--Requirement--
1. The format of these relationships is incorrect and needs to be **reconstructed**.
2. The **correct format** should be: **'ENTITY_A,RELATION_TYPE,ENTITY_B',** where each tuple contains three elements: two entities and a relationship type.
3. Please ensure the output **follows this structure**, accurately mapping the entities and relationships provided.

--Input--
Incorrect relationships tuple string: {relationship}

Relationship Reconstruction Prompt

--Goal--

Given a text, segment it into multiple semantic units, each containing detailed descriptions of specific events or activities.

Perform the following tasks:

--Steps--

1. Provide a summary for each semantic unit while retaining all crucial details relevant to the original context.

2. Extract all entities directly from the original text of each semantic unit, not from the paraphrased. Format each entity name in UPPERCASE. You should extract all entities including times, locations, people, organizations and all kinds of entities.

3. From the entities extracted in Step 2, **list all relationships within the semantic unit and the corresponding original context** in the form of string separated by comma : "**ENTITY_A, RELATION_TYPE, ENTITY_B**". The **RELATION_TYPE** could be a descriptive sentence, while the entities involved in the relationship must come from the entity names extracted in Step 2. Please make sure the string contains three elements representing two entities and the relationship type.

--Requirement--

1. Temporal Entities: Represent time entities based on the available details without filling in missing parts. Use specific formats based on what parts of the date or time are mentioned in the text.

2. Each semantic unit should be represented as a dictionary containing three keys: **semantic_unit** (a paraphrased summary of each semantic unit), **entities** (a list of entities extracted directly from the original text of each semantic unit, formatted in UPPERCASE), and **relationships** (a list of extracted relationship strings that contain three elements, where the relationship type is a descriptive sentence). All these dictionaries should be stored in a list to facilitate management and access.

--Example--

Text:
In September 2024, Dr. Emily Roberts traveled to Paris to attend the International Conference on Renewable Energy. During her visit, she explored partnerships with several European companies and presented her latest research on solar panel efficiency improvements. Meanwhile, on the other side of the world, her colleague, Dr. John Miller, was conducting fieldwork in the Amazon Rainforest. He documented several new species and observed the effects of deforestation on the local wildlife. Both scholars' work is essential in their respective fields and contributes significantly to environmental conservation efforts.

Text Decomposition Prompt Part 1

Output:
```
[
 {
  semantic_unit: In September 2024, Dr. Emily Roberts attended the International Conference on Renewable Energy in Paris, where she presented her research on solar panel efficiency improvements and explored partnerships with European companies.,
  entities: ["DR. EMILY ROBERTS", "2024-09", "PARIS", "INTERNATIONAL CONFERENCE ON RENEWABLE ENERGY", "EUROPEAN COMPANIES", "SOLAR PANEL EFFICIENCY"],
  relationships:
  [
   "DR. EMILY ROBERTS, attended, INTERNATIONAL CONFERENCE ON RENEWABLE ENERGY",
   "DR. EMILY ROBERTS, explored partnerships with, EUROPEAN COMPANIES",
   "DR. EMILY ROBERTS, presented research on, SOLAR PANEL EFFICIENCY"
  ]
 },
 {
  semantic_unit: Dr. John Miller conducted fieldwork in the Amazon Rainforest, documenting several new species and observing the effects of deforestation on local wildlife.",
  entities: ["DR. JOHN MILLER", "AMAZON RAINFOREST", "NEW SPECIES", "DEFORESTATION", "LOCAL WILDLIFE"],
  relationships:
  [
   "DR. JOHN MILLER, conducted fieldwork in, AMAZON RAINFOREST",
   "DR. JOHN MILLER, documented, NEW SPECIES",
   "DR. JOHN MILLER, observed the effects of, DEFORESTATION on LOCAL WILDLIFE"
  ]
 },
 {
  semantic_unit: "The work of both Dr. Emily Roberts and Dr. John Miller is crucial in their respective fields and contributes significantly to environmental conservation efforts.",
  entities: ["DR. EMILY ROBERTS", "DR. JOHN MILLER", "ENVIRONMENTAL CONSERVATION"],
  relationships: [
   "DR. EMILY ROBERTS, contributes to, ENVIRONMENTAL CONSERVATION",
   "DR. JOHN MILLER, contributes to, ENVIRONMENTAL CONSERVATION"
  ]
 }
]
```

--**Real Input**--

Text:{text}

Text Decomposition Prompt Part 2