NoLLMRAG: LLM-Free Makes Graph-Based RAG Highly Efficient, Effective and Generalizable

Anonymous authors
Paper under double-blind review

Abstract

Graph-based Retrieval-Augmented Generation (graph-based RAG) improves retrieval relevance and multi-hop reasoning compared to traditional RAG by constructing a graph that models relationships among text chunks. However, existing methods heavily rely on LLMs during indexing, resulting in inefficiency and unstable performance across LLM scales. Moreover, during retrieval, the lack of effective mechanisms for extracting query keywords and filtering irrelevant chunks further leads to redundant retrieval, introducing noise and degrading answer quality. To address these limitations, we propose NoLLMRAG, a novel graph-based RAG framework which is LLM-free during indexing and retrieval. It builds a three-layer heterogeneous graph index without LLMs, leverages a graph-statistics-driven keyword extraction to select keywords from queries that are aligned with the corpus, and applies a clustering-based retrieval on co-occurrence subgraphs to select more relevant chunks for generation. Experiments on three datasets and three LLMs demonstrate that NoLLMRAG achieves an average improvement of 41.27% over the strongest baseline, with indexing speedup of up to 300× and QA speedup of up to 15×, and maintains robust adaptability for real-time corpus expansion, highlighting the superior performance, efficiency and generalization across LLMs and domains.

1 Introduction

With the rapid advancement of large language models (LLMs), Retrieval-Augmented Generation (RAG) has emerged as a powerful paradigm to enhance LLM performance by incorporating external knowledge. Traditional RAG frameworks follow a three-stage pipeline: (i) segmenting documents into chunks and embedding them into a vector database; (ii) retrieving top-k relevant chunks for a query via vector similarity search; (iii) concatenating retrieved chunks with the query for LLM generation (Lewis et al., 2020; Gao et al., 2024). While effective at mitigating knowledge incompleteness and temporal staleness (Tu et al., 2024; Lyu et al., 2025), these frameworks treat chunks independently, ignoring structural relationships, which limits performance in multi-document reasoning tasks and makes retrieved contents fragmented or insufficiently relevant, reducing coherence and factual accuracy.

To address these challenges, recent work has explored graph-based RAG frameworks. These approaches build a graph-based index from corpus and perform retrieval over the graph to gather supporting content for LLMs generation. For instance, GraphRAG (Edge et al., 2024) extracts entities and relations from chunks using LLMs to build the index graph. Then it employs community detection and summary capabilities of LLMs to build multi-level community summaries. During retrieval, it traverses to obtain the most relevant text information to support LLMs in generating the final answer. LightRAG (Guo et al., 2025) similarly constructs a graph-based indexing and leverages a dual-level retrieval paradigm to capture both low-level and high-level information from the graph, enhancing response diversity and comprehensiveness. These methods leverage graph structures to establish structured relationships and contextual connections between text chunks, effectively enhancing the relevance of retrieved information and improving the accuracy of answer generation in RAG systems.

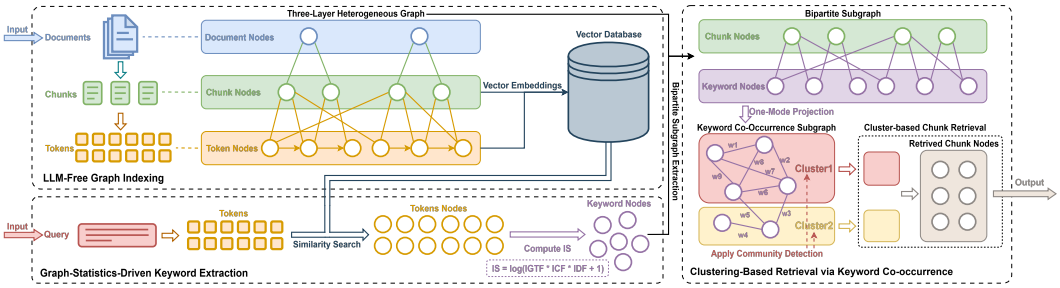


Figure 1: Overall Architecture of NoLLMRAG

Despite improvements, these methods still exhibit major limitations: (i) inefficient graph construction, hindering real-time applications; (ii) lack of robust graph-building strategies across different domains and LLM scales; (iii) retrieval redundancy introducing noise and impairing generation. These limitations primarily stem from the heavy reliance on LLMs during the indexing stage, leading to performance bottlenecks and unstable graph quality. Moreover, the keyword extraction from querie is misaligned with corpus characteristics and the retrieval process lacks effective filtering mechanisms, which causes redundant node retrieval, resulting in introducing noise and ultimately degrading answer quality.

To overcome these challenges, we propose NoLLMRAG, a novel graph-based RAG framework that eliminates LLM dependency during indexing and retrieval, and reduces redundant information in retrieval. First, to achieve rapid indexing and stable graph quality, NoLLMRAG constructs a three-layer heterogeneous indexing graph entirely without LLMs. On top of this foundation, it employs a graph-statistics-driven keyword extraction algorithm that leverages multi-level graph-based statistics to efficiently extract corpus-aware keywords from queries. Finally, a clustering-based retrieval algorithm exploits keyword co-occurrence and community detection to form semantically coherent keyword clusters, retrieving more relevant chunks to filter noise and support higher-quality generation. Together, these innovations form a cohesive pipeline that synergistically enables extreme efficiency, strong performance, and broad generalization across different LLMs and domains.

We conduct a comprehensive evaluation across three types of datasets and three LLMs of different sizes, where NoLLMRAG consistently achieves highly competitive results in all experimental settings, with an average improvement of 41.27% over the strongest baselines. In terms of efficiency, it delivers orders-of-magnitude speedups, achieving up to 300× faster indexing and up to 15× faster QA times. In scenarios requiring real-time corpus expansion, NoLLMRAG demonstrates strong adaptability, maintaining high efficiency with minimal fluctuations, and showing a tendency to improve indexing efficiency as the corpus grows. These results validate NoLLMRAG's strong performance, exceptional efficiency, and broad generalization, highlighting its practical value for real-world applications across different resource levels and domains.

2 Related Work

Graph-based Retrieval-Augmented Generation (graph-based RAG) improves upon traditional RAG frameworks by modeling corpus as a graph. Unlike traditional RAG, which treats text chunks as independent units, graph-based RAG leverages graph to capture connections across multiple chunks, reducing content fragmentation and enhancing the coherence of retrieved information, which improves the factual accuracy of generated answers, particularly in complex reasoning tasks (Zhu et al., 2025; Han et al., 2025).

GraphRAG (Edge et al., 2024) is the first method to adopt this paradigm. It constructs an entity knowledge graph from corpus using LLMs, applies community detection to generate multi-level summaries, and retrieves the most relevant context by traversing the graph, which significantly improves the comprehensiveness and diversity of generated answers. However, its reliance on repeated LLM calls during indexing and retrieval leads to high computational cost and long latency, making it impractical for large-scale or frequently updated corpora.

To improve efficiency, some recent work has focused on reducing the dependency of LLMs in graph construction. LightRAG (Guo et al., 2025) removes the multi-level summaries and introduces a dual-level retrieval strategy to reduce LLM calls while improving answer quality. E^{2}GraphRAG (Zhao et al., 2025) further enhances efficiency by building an entity graph with SpaCy (Honnibal et al., 2020) instead of LLMs and designing an adaptive retrieval strategy while maintaining competitive QA performance.

For further performance gains, HippoRAG 2 (Gutiérrez et al., 2025) enhances the Personalized PageRank used in HippoRAG (Gutierrez et al., 2024) with deeper passage integration and more effective online LLM usage, pushing it closer to human-like long-term memory and achieving notable gains. PathRAG (Chen et al., 2025) takes a different angle by retrieving key relational paths based on flow-based pruning from the graph, which reduces redundant information and generate more logical and coherent responses. MiniRAG (Fan et al., 2025), in contrast, targets resource-constrained scenarios. It introduces a semantic-aware heterogeneous graph unifying chunks and entities, along with a lightweight topology-enhanced retrieval strategy, which allows small-parameter LLMs to achieve competitive performance.

3 METHODOLOGY

In this section, we present NoLLMRAG, a novel graph-based RAG framework designed to enhance both the efficiency, effectiveness, and generalization. Our method comprises three main components: (1) **LLM-free graph indexing**, which builds a three-layer heterogeneous graph without using LLMs, (2) **Graph-statistics-driven keyword extraction**, which extracts keywords from queries that are more aligned with the corpus characteristics, and (3) **Clustering-based retrieval via keyword co-occurrence**, which aims to retrieve more relevant information. An overview of the architecture is illustrated in Figure 1.

3.1 LLM-FREE GRAPH INDEXING

Existing graph-based indexing approaches heavily depend on LLMs, which not only introduces significant computational overhead and latency, but also leads to unstable graph quality across different LLMs. To address these limitations, we propose an **LLM-free graph indexing** method that constructs a three-layer heterogeneous graph directly from raw corpus data without using LLMs.

Dual-Stage Corpus Segmentation. In the first stage, each document D_i in the raw corpus is segmented into non-overlapping text chunks $[C_{i,1}, C_{i,2}, \ldots, C_{i,m_i}]$. The length of each chunk l_{chunk} should be set much lower than that in existing graph-based RAG methods, enabling finer-grained filtering during subsequent retrieval. In the second stage, we utilize the traditional NLP toolkit spaCy (Honnibal et al., 2020) to segment each chunk $C_{i,j}$ into a sequence of word-level tokens $[T_{i,j,1}, T_{i,j,2}, \ldots, T_{i,j,l_{i,j}}]$. Specifically, we perform tokenization, followed by lemmatization to normalize token forms and stop-word filtering to remove common non-informative tokens. As a result, we obtain three levels of structured representations: the document collection, the chunk sequences for each document, and the token sequences for each chunk. The dual-stage corpus segmentation ensures that both local context and global document structure are preserved.

Three-Layer Heterogeneous Graph Construction. Based on the structured representations, we construct a three-layer heterogeneous graph that models the hierarchical structure among documents, text chunks, and tokens. The graph comprises three types of nodes: Document Nodes $\mathcal{V}^D$, Chunk Nodes $\mathcal{V}^C$ and Token Nodes $\mathcal{V}^T$, connected by edges fall into three fundamental types: Document–Chunk Edges $\mathcal{E}^{DC}$, Chunk–Token Edges $\mathcal{E}^{CT}$ and Token–Token Edges $\mathcal{E}^{TT}$. The details of node and edge construction are provided in Appendix B.1. Formally, the three-layer heterogeneous graph is represented as:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \quad \text{where } \mathcal{V} = \mathcal{V}^D \cup \mathcal{V}^C \cup \mathcal{V}^T,\ \mathcal{E} = \mathcal{E}^{DC} \cup \mathcal{E}^{CT} \cup \mathcal{E}^{TT}. \tag{1}$$

This construction captures both hierarchical semantics (document-to-chunk, chunk-to-token) and fine-grained syntactic continuity (token-to-token), providing a unified graph-based representation.

Vector Indexing for Chunk and Token Nodes. To support efficient semantic retrieval across the graph, we construct vector indices for both chunk and token nodes. Specifically, for each chunk node $v^C \in \mathcal{V}^C$ and each token node $v^T \in \mathcal{V}^T$, we compute a corresponding dense vector representation

via pre-trained embedding models based on the node's text content: the full chunk text for v^C and the token string for v^T. The resulting embeddings are then stored in a vector database to enable fast vector similarity search.

Our graph indexing method is entirely LLM-free, relying only on lightweight linguistic preprocessing, to construct a hierarchical graph-based index which captures multi-level relationships among documents, chunks and tokens and retains the full textual content of the corpus. It drastically improves the speed of graph indexing and reduces the consumption of computational resources.

3.2 Graph-Statistics-Driven Keyword Extraction

At the beginning of the retrieval stage, most existing approaches rely on LLMs to extract keywords from user queries, which heavily dependent on the semantic understanding of LLMs and typically identify words that are semantically significant within the query itself, while often neglecting their importance in the corpus. This disconnect can lead to the selection of keywords that are meaningful in isolation but poorly aligned with the corpus.

To address this issue, we propose a graph-statistics-driven keyword extraction algorithm that leverages the topological statistics of the three-layer heterogeneous graph described in Section 3.1 to evaluate the importance of each token in the user query, thereby extracting keywords that are more meaningful and important in the corpus.

More concretely, we systematically design a composite importance scoring formula to quantify the significance of a token. It is based on the following graph-based statistics (see Appendix B.2 for formal definitions):

- **Inverse Global Term Frequency ($IGTF$):** reflecting the rarity of a token across the entire corpus.
- **Inverse Chunk Frequency (ICF):** measuring how unique a token is across chunks.
- **Inverse Document Frequency (IDF):** indicating the specificity of a token across documents.

These statistics at different granularity levels are combined into a unified importance score:

$$IS(v_T^q) = \ln\left(IGTF(v_T^q) \cdot ICF(v_T^q) \cdot IDF(v_T^q) + 1\right) \tag{2}$$

By integrating statistics from multiple structural levels, IS captures a more holistic view of the importance of a token. The following describes the comprehensive pipeline for the extraction of keywords from a query based on IS.

Given a user query, we adopt the dual-stage corpus segmentation described in Section 3.1, resulting in the corresponding token set $\mathbb{T}^q$. For each token $t_i^q \in \mathbb{T}^q$, based on the vector indices established in Section 3.1, we efficiently identify the token node with the highest similarity, thus getting the token node set $\mathcal{V}_T^q$. For each token node $v_T^q \in \mathcal{V}_T^q$, we compute its IS using Equation 2, and then select the tokens whose importance scores exceed a fixed ratio $\tau \in (0, 1)$ of the maximum score among $\mathcal{V}_T^q$:

$$\mathcal{V}_T^{key} = \left\{ v_T^q \in \mathcal{V}_T^q \,\middle|\, IS(v_T^q) > \tau \cdot \max_{v_T' \in \mathcal{V}_T^q} IS(v_T') \right\} \tag{3}$$

where $\mathcal{V}_T^{key}$ is the set of token nodes identified as keywords, serving as the entry point for the subsequent retrieval.

The graph-statistics-driven keyword extraction algorithm offers several advantages. First, it's LLM-free, fully interpretable and stable. Second, by incorporating corpus-aware graph statistics from multiple structural levels, it naturally aligns extracted keywords with the corpus.

3.3 Clustering-Based Retrieval via Keyword Co-Occurrence

Motivated by the insight that queries, especially complex ones, often encompasses multiple latent sub-topics, we propose a clustering-based retrieval algorithm to mine these sub-topics for more precise chunk retrieval. Rather than retrieving based on isolated keywords, we exploit keyword co-occurrence to form semantically meaningful keyword clusters, and perform fine-grained retrieval based on these clusters to further reduce redundancy and improve retrieval quality.

Co-Occurrence Subgraph Extraction. For keyword nodes $\mathcal{V}_T^{key}$ extracted in Section 3.2, we restrict our attention to the bipartite subgraph formed between $\mathcal{V}_T^{key}$ and their connected chunk nodes in $\mathcal{G}$. A one-mode projection of this bipartite subgraph onto $\mathcal{V}_T^{key}$ yields a keyword co-occurrence subgraph $\mathcal{G}_{CO} = (\mathcal{V}_T^{key}, \mathcal{E}_{CO})$, in which the edge weights represent the number of shared chunks between token pairs.

Keyword Clustering with Community Detection. We apply Leiden (Traag et al., 2019) community detection to $\mathcal{G}_{CO}$, yielding clusters of keywords $\{L_1, L_2, \ldots, L_k\}$ that tend to appear together in chunks. These clusters are assumed to represent semantically coherent subtopics.

Cluster-Based Chunk Retrieval. For each keyword cluster L_i, we retrieve relevant chunks as follows. First, token nodes in the cluster are sorted in descending order based on their weighted degrees in $\mathcal{G}_{CO}$, yielding a token node sequence $\mathcal{V}_T^{L_i} = \left[v_1^T, v_2^T, \ldots, v_{|L_i|}^T\right]$. Then, starting from v_1^T, we iteratively intersect the chunk sets $\mathbb{S}_j^C$ connected to v_j^T to refine the candidate set $\mathbb{S}_{curr}$. If $\mathbb{S}_{curr} \cap \mathbb{S}_j^C$ becomes empty, $\mathbb{S}_{curr}$ is saved as an intermediate result set $\mathbb{S}_{inter}$ and reset. To maintain context continuity, if $\mathbb{S}_j^C$ overlaps with the union of previously visited chunk sets, we use this overlap to reinitialize $\mathbb{S}_{curr}$, otherwise we reset $\mathbb{S}_{curr}$ to $\mathbb{S}_j^C$. After resetting, the same iterative intersection process is repeated from the current token node v_j^T until all token nodes in the cluster have been processed. This process yields multiple intermediate result sets $\{\mathbb{S}_{inter}^j\}_{j=1}^{n_{inter}}$, whose union forms the final set of retrieved chunk nodes $\mathbb{S}_i$ for cluster L_i. All $\mathbb{S}_i$ are then combined via union to produce the overall retrieval result:

$$\mathbb{S}_{re} = \{v_{re1}^C, v_{re2}^C, \ldots, v_{rek_{re}}^C\} \tag{4}$$

where each $v_{rek}^C \in \mathcal{V}_C$ represents the retrieved chunk node, and k_{re} is the cardinality of S_{re}.

The pseudocode of Cluster-Based Chunk Retrieval is provided in Appendix B.3.

We analyze the computational complexity of the clustering-based retrieval via keyword co-occurrence. Let k be the number of query keywords and d_{key} be the average degree of keywords nodes in the bipartite subgrap. The overall complexity is approximately $O(k^2 \cdot d_{avg})$. Since k and d_{avg} is typically small in practice, our method achieves high efficiency. A detailed complexity derivation is provided in Appendix B.4.

This clustering-based retrieval via keyword co-occurrence method improves retrieval quality through two key advantages. First, clustering keywords based on co-occurrence captures latent subtopics, enabling semantically coherent and context-aware retrieval. Second, the iterative intersection strategy within each cluster filters out loosely related content, enhancing precision and reducing redundancy by focusing on chunks most relevant to the cluster's core semantics.

3.4 Retrieval-Augmented Answer Generation

To leverage the retrieved chunk nodes $\mathbb{S}_{re}$ for enhancing the LLM's answers, we employ the prebuilt vector indices in Section 3.1 to efficiently compute the semantic similarity between each chunk node's content and the query. The chunk nodes are then ranked by similarity score, and the top-k_{chunk} highest-scoring chunk nodes are selected for context augmentation. To mitigate the "lost in the middle" issue of LLMs (Liu et al., 2024), we sequentially incorporate the contents of the selected chunk nodes into the prompt in descending order of similarity, followed by appending the user's original query at the end. The assembled prompt is then input to the LLM to generate an answer. This simple prompting strategy effectively leverages the retrieved chunks, improving the response performance of LLMs. The exact prompt template used is provided in Appendix D.1.

4 Experimental Settings

We conduct comprehensive experiments to evaluate our proposed NoLLMRAG framework against established baselines. The experiments are designed to validate its superior performance, efficiency, and generalization across LLMs of various sizes and multiple datasets.

4.1 Evaluation Datasets

To comprehensively evaluate NoLLMRAG, we select three categories of datasets: simple QA, multi-hop QA, and real-world application QA.

- **Simple QA.** The questions in this type of QA task are related to individual entities, which can be answered simply once the relevant passages concerning those entities are retrieved. Following HippoRAG 2, we employ the same 1,000-query subset of **PopQA** (Mallen et al., 2023) with the same corpus drawn from the December 2021 Wikipedia dump.
- **Multi-hop QA.** This type of QA task requires models to perform cross-document reasoning to answer complex questions. We select **MultiHop-RAG** (Tang & Yang, 2024), a benchmark for multi-hop RAG systems, as the evaluation dataset.
- **Real-World Application QA.** This category focuses on QA tasks that are grounded in authentic, dynamic environments, reflecting the complexity of real-life user interactions. We employ the **LiHua-World** (Fan et al., 2025) dataset as a representative benchmark for real-world application scenarios.

The details of these datasets are provided in Appendix C.1.

4.2 Baselines

We compare our proposed NoLLMRAG against the following state-of-the-art and representative graph-based RAG methods:

- **LightRAG** (Guo et al., 2025). A graph-augmented RAG framework that employs a dual-level retrieval architecture, combining low-level entity-specific retrieval with high-level concept discovery to enhance both precision and semantic awareness.
- **MiniRAG** (Fan et al., 2025). A RAG system tailored for small LLMs in on-device scenarios, introducing a semantic-aware heterogeneous graph index with a lightweight topology-enhanced retrieval that avoids reliance on deep semantic understanding.
- **HippoRAG 2** (Gutiérrez et al., 2025). An advanced RAG framework inspired by human-like long-term memory, integrating knowledge graph structures and PPR-based retrieval, which is enhanced by deeper passage integration and more effective LLM use.

4.3 Metrics

Following MiniRAG, we assess system performance using two key metrics. **Accuracy (acc)** measures the semantic consistency between the RAG system's response and the gold answer, counting semantically equivalent responses as correct. **Error Rate (err)** captures cases where the system produces confidently incorrect answers that are fully inconsistent with the gold answer. To evaluate these two metrics, we employ the LLM-as-a-Judge paradigm (Zheng et al., 2023; Gu et al., 2025), which excels in determining whether a response is semantically equivalent to the gold answer rather than relying on exact string matching, offering a more reliable evaluation than traditional lexical metrics such as BLEU or ROUGE. The evaluation prompt we used is detailed in Appendix D.2

To measure the efficiency of these RAG systems, we introduce two additional metrics. **Indexing Time per Document** measures the average time required to construct the graph index for a single document. **QA Time per Query** measures the average end-to-end response time for answering a query.

4.4 Implementation Details

For hyperparameters, in our NoLLMRAG implementation, we empirically set the chunk size l_{chunk} to 100 tokens, configure k_{chunk}, the max number of chunk nodes which are selected for prompt construction, to 30, and set the threshold ratio τ for keyword selection to 0.5. For all other baselines, we adopt their officially recommended hyperparameter settings, which are provided in Appendix C.2. For model selection, we evaluate across three LLMs scales to ensure robustness and generality of our method:

Table 1: Comparison of accuracy (acc) and error rate (err) of different graph-based RAG methods across multiple LLMs on three datasets. **Bold** values represent the best performance. "–" indicates cases where the method fail to generate effective responses.

Method	LLMs	LiHua-World		MultiHop-RAG		PopQA	
		acc	err	acc	err	acc	err
LightRAG	Qwen2.5-3B -Instruct	11.15%	12.87%	35.84%	22.34%	8.50%	27.80%
MiniRAG		9.73%	14.76%	26.84%	25.31%	–	–
HippoRAG 2		–	–	–	–	–	–
NoLLMRAG		**68.45%**	19.31%	**53.01%**	23.32%	**62.80%**	4.20%
LightRAG	Qwen2.5-7B -Instruct	13.66%	11.93%	38.97%	39.40%	19.50%	57.90%
MiniRAG		11.15%	8.63%	32.20%	43.86%	18.30%	61.20%
HippoRAG 2		49.92%	30.30%	49.80%	31.81%	**72.20%**	7.80%
NoLLMRAG		**73.16%**	16.17%	**69.05%**	17.49%	70.50%	5.00%
LightRAG	GPT-4o-mini	56.99%	20.88%	64.91%	19.29%	**81.30%**	9.40%
MiniRAG		53.85%	19.47%	68.27%	19.48%	70.80%	9.30%
HippoRAG 2		62.95%	23.23%	69.80%	24.33%	77.80%	4.50%
NoLLMRAG		**75.51%**	16.80%	**76.96%**	18.19%	74.20%	5.00%

- **GPT-4o-mini** (OpenAI, 2024), a high-performance proprietary model that serves as a strong upper bound for system capability.
- **Qwen2.5-7B-Instruct** (Team, 2024), one of the most popular open-source models, widely used for on-premise deployment due to its balance between capability and hardware cost.
- **Qwen2.5-3B-Instruct** (Team, 2024), a lightweight open-source model suitable for resource-constrained settings.

For GPT-4o-mini and Qwen2.5-7B-Instruct, we use BGE-M3 (Chen et al., 2024) as the embedding model, while for Qwen2.5-3B-Instruct, we adopt all-MiniLM-L6-v2 (Reimers & Gurevych, 2019), an extremely lightweight embedding model, to simulate low-resource deployment environments. For experimental environment, we provide the details in Appendix C.3.

5 Results and Analysis

5.1 Performance Analysis

We now present our main QA experimental results. More detailed experimental results are provided in Appendix C.4. The statistics for all constructed three-layer heterogeneous graphs are shown in Appendix C.5.

Overall performance. Table 1 reports accuracy and error rates across three datasets and three LLMs. NoLLMRAG achieves the highest accuracy in 7/9 settings, with an average improvement of 20.24 percentage points (41.27% relative) over the strongest baseline. Even when excluding the Qwen2.5-3B-Instruct model due to the extremely poor performance of the baselines on it, NoLLMRAG still outperforms the best baseline by 8.90 points (13.84% relative) with an average accuracy of 73.23%, while simultaneously reducing the error by 35.5%. The observed improvements highlight the impact of NoLLMRAG's design in delivering consistently strong performance.

Robust Cross-Model Generalization. NoLLMRAG delivers a remarkable 232.06% average improvement on Qwen2.5-3B-Instruct over the strongest baseline, while still achieving substantial 23.73% and 5.90% gains on Qwen2.5-7B-Instruct and GPT-4o-mini, respectively. Remarkably, while other baselines perform poorly on Qwen2.5-3B-Instruct with some even failing to yield effective results, NoLLMRAG remains highly effective, achieving 68.45% accuracy on LiHua-World, which even surpasses the strongest baseline on GPT-4o-mini at 62.95%. These results show NoLLMRAG's robust generalization across LLM scales and its insensitivity to the underlying LLM,

Table 2: Comparison of Indexing Time per Document (IT) and QA Time per Query (QT) of different RAG methods on three datasets using Qwen2.5-7B-Instruct (in seconds). **Bold** values represent the fastest time.

Method	LiHua-World		MultiHop-RAG		PopQA	
	IT	QT	IT	QT	IT	QT
LightRAG	54.36	4.52	170.98	5.46	15.21	2.45
MiniRAG	30.41	11.19	113.92	13.93	30.09	31.08
HippoRAG 2	9.44	2.80	20.23	5.34	6.08	2.54
NoLLMRAG	**0.38**	**2.04**	**1.22**	**5.25**	**0.10**	**2.08**

making it suitable for both resource-constrained and high-performance scenarios, further highlighting the robustness and superiority of its graph construction and retrieval algorithm design.

Superior Performance across Datasets. NoLLMRAG consistently dominates on LiHua-World and MultiHop-RAG, achieving the highest accuracy across all model scales. For instance, on Qwen2.5-7B-Instruct, NoLLMRAG delivers 46.55% and 38.65% relative gains over the best-performing baseline on LiHua-World and MultiHop-RAG respectively, indicating it's effectiveness for complex multi-step reasoning tasks. For PopQA, on Qwen2.5-3B-Instruct, NoLLMRAG achieves 62.80% accuracy and 4.20% error, outperforming the strongest baseline (LightRAG) by exceptional 54.3 percentage points and reducing error by 23.6 percentage points. Although NoLLMRAG ranks second and third in accuracy on Qwen2.5-7B-Instruct and GPT-4o-mini, its accuracy remains highly competitive (70.50% and 74.20%), and its error rates are reduced by 35.9% and 46.8%, respectively, compared to the best baseline, suggesting that even when its accuracy is slightly below the top baseline, NoLLMRAG generates significantly fewer incorrect answers, making its outputs more reliable and trustworthy for simple QA scenarios.

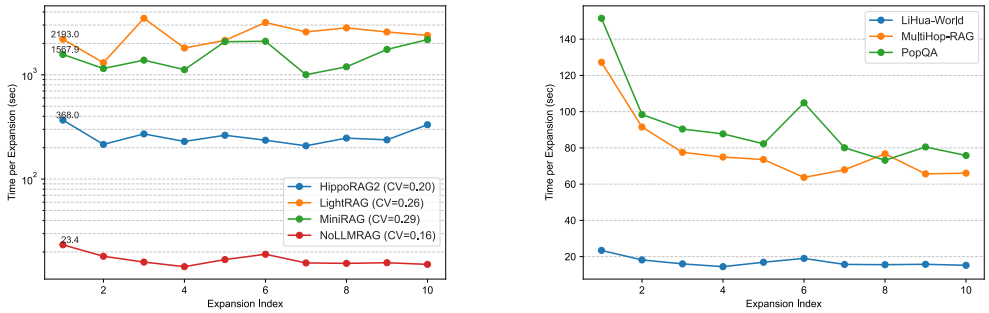
For a qualitative illustration, we provide a detailed case study in Appendix C.6.

5.2 Efficiency Analysis

To ensure fair efficiency evaluation, we conduct all experiments using the locally deployed Qwen2.5-7B-Instruct model, as some baselines are not available on Qwen2.5-3B-Instruct, and GPT-4o-mini relies on API calls that may suffer from network-induced latency fluctuations.

Latency Comparison. Table 2 summarizes the efficiency results, highlighting that NoLLMRAG consistently achieves order-of-magnitude improvements over all baselines. In terms of indexing, speedups range from over 16× to over 300×, with the largest gain observed on PopQA, where NoLLMRAG requires only 0.10 seconds per document, compared to 30.09 seconds for MiniRAG. Even relative to the strongest baseline, HippoRAG 2, NoLLMRAG delivers substantial speedups ranging from over 16× to 60× across datasets. For QA, NoLLMRAG continues to achieve substantial improvements over all baselines, with the largest speedup reaching nearly 15×. Even when compared to the best-performing baseline for each dataset, NoLLMRAG still reduces latency by 27.14%, 1.69%, and 15.10%, respectively. These dramatic efficiency gains demonstrate NoLLMRAG's particular suitability for large-scale corpora scenarios and real-time QA applications.

Real-Time Corpus Expansion. To assess efficiency in real-time corpus expansion, we partition each dataset into ten approximately equal segments and add them sequentially, measuring indexing time after each addition. As shown in Figure 2a, on LiHua-World, NoLLMRAG is significantly faster than all baselines, requiring only average 17 seconds per expansion, with the smallest coefficient of variation (CV) among all methods, indicating stable performance (CV formula in Appendix C.7). Moreover, Figure 2b shows that, when adding corpus segments of similar size, the indexing time shows a decreasing trend. This phenomenon can be attributed to the fact that, as the indexed corpus expands, an increasing proportion of tokens in newly added documents already exist in the graph nodes and vector database, reducing the number of new tokens that require indexing, which is quantitatively supported by the results reported in Appendix C.8. These findings highlight NoLLMRAG's strong scalability and efficiency for real-time corpus expansion scenarios.



(a) Corpus expansion time on LiHua-World measured with different methods (log scale)

(b) Corpus expansion time on different datasets measured with NoLLMRAG

Figure 2: Comparison of corpus expansion time.

Table 3: Comparison of accuracy (acc) and error rate (err) of NoLLMRAG and its ablations across multiple LLMs on three datasets. **Bold** values indicate best performance.

Methods	LLMs	LiHua-World		MultiHop-RAG		PopQA	
		acc	err	acc	err	acc	err
NoLLMRAG	Qwen2.5-3B -Instruct	**68.45%**	19.31%	**53.01%**	23.32%	**62.80%**	4.20%
w/o Graph stats		62.32%	16.64%	46.91%	29.15%	53.10%	6.50%
w/o Clustering		67.35%	19.15%	50.59%	32.55%	59.20%	8.40%
NoLLMRAG	Qwen2.5-7B -Instruct	73.16%	16.17%	**69.05%**	17.49%	**70.50%**	5.00%
w/o Graph stats		71.27%	13.03%	59.00%	18.11%	46.50%	3.00%
w/o Clustering		**75.98%**	11.93%	67.14%	20.97%	61.60%	5.10%
NoLLMRAG	GPT-4o-mini	75.51%	16.80%	**76.96%**	18.19%	**74.20%**	5.00%
w/o Graph stats		73.78%	13.97%	65.41%	13.93%	47.70%	3.70%
w/o Clustering		**78.02%**	12.09%	74.65%	20.34%	65.70%	5.70%

5.3 Ablation Study

We focus our ablation study on graph-statistics-driven keyword extraction and clustering-based retrieval via keyword co-occurrence. The implementation details of the ablation study are provided in Appendix C.9.1.

For graph-statistics-driven keyword extraction, we replace it with a commonly used LLM-based approach without using any graph statistics. As shown in Table 3, this substitution results in significant accuracy drops across all datasets and LLMs, with the maximum decline reaching 26.50 percentage points (on PopQA with gpt-4o-mini), demonstrating that graph-statistics-driven keyword extraction plays a crucial role in capturing corpus-aware keywords. Interestingly, on PopQA, the LLM-based method performs worse when using the stronger LLMs (Qwen2.5-7B-Instruct and GPT-4o-Mini) compared to the smallest model (Qwen2.5-3B-Instrcut), likely because stronger LLMs extract excessive keywords, in turn causing redundant retrieval. In addition, this substitution also reduces QA efficiency, as detailed in Appendix C.9.2. **For clustering-based retrieval via keyword co-occurrence**, we remove the keyword clustering step and treat the whole keywords as a single cluster. Table 3 shows that, this causes accuracy to decline in seven of nine settings, with the max drop of 8.9 percentage points, confirming that keyword clustering effectively reduces redundancy and improves the quality of candidate chunks. The occasional minor gains in two settings may be attributed to cases where treating all keywords as a single cluster happens to capture more relevant chunks.

6 CONCLUSION

We present NoLLMRAG, a novel graph-based RAG framework that achieves both exceptional efficiency and strong performance. It utilizes LLM-free graph indexing method to greatly accelerate corpus indexing while providing multi-level structural information for downstream processing. The graph-statistics-driven keyword extraction ensures better alignment between query keywords and the corpus, and clustering-based retrieval via keyword co-occurrence effectively captures latent subtopics, focusing on the most relevant chunks while reducing redundancy. Comprehensive experiments on three types of QA datasets and three LLMs with varying parameter scales demonstrate that NoLLMRAG consistently outperforms state-of-the-art graph-based RAG methods in both accuracy, efficiency, and generalization.

7 REPRODUCIBILITY STATEMENT

We make every effort to ensure the reproducibility of our results. The detailed methodology of NoLLMRAG is presented in Section 3, while the experimental settings are described in Section 4. All datasets and baselines used in our experiments are publicly available. To further facilitate reproducibility, we provide an anonymous link to our source code in the supplementary materials.

REFERENCES

Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. Pathrag: Pruning graph-based retrieval augmented generation with relational paths, 2025. URL https://arxiv.org/abs/2502.14902.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2024. URL https://arxiv.org/abs/2404.16130.

Tianyu Fan, Jingyuan Wang, Xubin Ren, and Chao Huang. Minirag: Towards extremely simple retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2501.06713.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A survey on llm-as-a-judge, 2025. URL https://arxiv.org/abs/2411.15594.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2410.05779.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=hkujvAPVsg.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=LWH8yn4HS2.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qi He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. Retrieval-augmented generation with graphs (graphrag). *CoRR*, abs/2501.00309, January 2025. URL https://doi.org/10.48550/arXiv.2501.00309.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL https://doi.org/10.5281/zenodo.1212303.

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. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Yuanjie Lyu, Zhiyu Li, Simin Niu, Feiyu Xiong, Bo Tang, Wenjin Wang, Hao Wu, Huanyong Liu, Tong Xu, and Enhong Chen. Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models. *ACM Trans. Inf. Syst.*, 43(2), January 2025. ISSN 1046-8188. doi: 10.1145/3701228. URL https://doi.org/10.1145/3701228.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

Ollama. Ollama github repository, 2025. URL https://github.com/ollama/ollama.

OpenAI. Gpt-4o mini: advancing cost-efficient intelligence, 2024. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Yixuan Tang and Yi Yang. Multihop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=t4eB3zYWBK.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), March 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-41695-z. URL http://dx.doi.org/10.1038/s41598-019-41695-z.

Shangqing Tu, Yuanchun Wang, Jifan Yu, Yuyang Xie, Yaran Shi, Xiaozhi Wang, Jing Zhang, Lei Hou, and Juanzi Li. R-eval: A unified toolkit for evaluating domain knowledge of retrieval augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 5813–5824. ACM, August 2024. doi: 10.1145/3637528.3671564. URL http://dx.doi.org/10.1145/3637528.3671564.

Yibo Zhao, Jiapeng Zhu, Ye Guo, Kangkang He, and Xiang Li. E^{2}graphrag: Streamlining graph-based rag for high efficiency and effectiveness, 2025. URL https://arxiv.org/abs/2505.24226.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. Graph-based approaches and functionalities in retrieval-augmented generation: A comprehensive survey, 2025. URL https://arxiv.org/abs/2504.10499.

APPENDIX

A LLM USAGE STATEMENT

In preparing this manuscript, we used large language models (LLMs) solely to aid in writing and polishing the text. Specifically, LLMs were employed to improve clarity, grammar, and phrasing, without contributing to the conceptual design, experimental setup, analysis, or results of this work. All scientific content, results, and conclusions presented in this paper were generated by the authors. We take full responsibility for the accuracy and integrity of the manuscript, including any text refined with the assistance of LLMs.

B MORE DETAILS OF NOLLMRAG

B.1 DETAILS OF NODE AND EDGE CONSTRUCTION

The graph comprises three types of nodes:

- **Document Nodes** $\mathcal{V}^D$: Each node $v_i^D \in \mathcal{V}^D$ corresponds to a document D_i in the corpus.
- **Chunk Nodes** $\mathcal{V}^C$: Each node $v_{i,j}^C \in \mathcal{V}^C$ represents a text chunk $C_{i,j}$. The full textual content of each chunk is preserved within its node.
- **Token Nodes** $\mathcal{V}^T$: Each node $v_k^T \in \mathcal{V}^T$ corresponds to a unique token $T_k \in \mathbb{T}$, where $\mathbb{T}$ is the union of all token sequences $\mathbb{T}_{i,j}$.

The connecting edges between these nodes fall into three fundamental types:

- **Document–Chunk Edges** $\mathcal{E}^{DC}$: Each document node $v_i^D \in \mathcal{V}^D$ is connected to its constituent chunk nodes Each node $v_{i,j}^C \in \mathcal{V}^C$, forming undirected edges $\left(v_i^D, v_{i,j}^C\right) \in \mathcal{E}^{DC}$.
- **Chunk–Token Edges** $\mathcal{E}^{CT}$: Each chunk node $v_{i,j}^C$ is linked to the token nodes v_k^T which represent tokens in the chunk's token sequence $\mathbb{T}_{i,j}$, establishing undirected edges $\left(v_{i,j}^C, v_k^T\right) \in \mathcal{E}^{CT}$.
- **Token–Token Edges** $\mathcal{E}^{TT}$: Each document D_i is associated with a token sequence $\mathbb{T}_i$, obtained by concatenating the token sequences of its constituent chunks. The directed edges $\left(v_k^T, v_{k+1}^T\right) \in \mathcal{E}^{TT}$ are formed between the token nodes corresponding to two adjacent tokens in $\mathbb{T}_i$, to capture the sequential structure within the document.

B.2 FORMAL DEFINITIONS OF GRAPH-BASED STATISTICS

In Section 3.2, we introduced three graph-based statistics for evaluating the importance of query tokens. Here we provide the complete mathematical definitions.

- **Inverse Global Term Frequency ($IGTF$):** reflecting the rarity of a token across the entire corpus:

$$IGTF(v_T^q) = \ln\left(\frac{|\mathcal{E}^{TT}|}{\max(|\{v^T \in \mathcal{V}^D \mid (v_T^q, v^T) \in \mathcal{E}^{TT}\}|, |\{v^T \in \mathcal{V}^D \mid (v^T, v_T^q) \in \mathcal{E}^{TT}\}|)}\right) \tag{5}$$

- **Inverse Chunk Frequency (ICF):** quantifying how unique a token is across the chunks:

$$ICF(v_T^q) = \ln\left(\frac{|\mathcal{V}^C|}{|\{v^C \in \mathcal{V}^C \mid (v_T^q, v^C) \in \mathcal{E}^{CT}\}|}\right) + 1 \tag{6}$$

- **Inverse Document Frequency (IDF):** reflecting the document-level specificity of a token:

$$IDF(v_T^q) = \ln\left(\frac{|\mathcal{V}^D|}{|\{v^D \in \mathcal{V}^D \mid \exists v^C \in \mathcal{V}^C, (v_T^q, v^C) \in \mathcal{E}^{CT}, (v^C, v^D) \in \mathcal{E}^{DC}\}|}\right) + 1 \tag{7}$$

where v_T^q represents each token in a query.

B.3 Pseudocode of Cluster-Based Chunk Retrieval

To provide a clear understanding of the algorithmic procedures of the cluster-based chunk retrieval discussed in Section 3.3, we present a pseudocode representation in Algorithm 1.

Algorithm 1: Cluster-Based Chunk Retrieval

Input: Keyword clusters $\{L_i\}_{i=1}^{k}$, Co-occurrence graph $\mathcal{G}_{CO} = (\mathcal{V}_{CO}, \mathcal{E}_{CO})$
Output: Retrieved chunk nodes set $\mathbb{S}_{re}$
Function RetrieveChunks($\{L_i\}_{i=1}^{k}$, $\mathcal{G}_{CO}$)

```
S_re ← ∅;
for each cluster L_i ∈ {L_i}_{i=1}^k do
    V_T^{L_i} ← DescSortByWeightedDegree(L_i, G_CO);
    S_inter ← ∅, S_curr ← ∅;
    for each token node v_j^T ∈ V_T^{L_i} do
        S_j^C ← {v^C ∈ V^C | (v_j^T, v^C) ∈ E_CO};
        if j = 1 then
            S_curr ← S_j^C;
        else
            if S_curr ∩ S_j^C ≠ ∅ then
                S_curr ← S_curr ∩ S_j^C;
            else if S_j^C ∩ (⋃_{n=1}^{j-1} S_n^C) ≠ ∅ then
                S_inter ← S_inter ∪ S_curr;
                S_curr ← S_j^C ∩ (⋃_{n=1}^{j-1} S_n^C);
            else
                S_inter ← S_inter ∪ S_curr;
                S_curr ← S_j^C;
    S_inter ← S_inter ∪ S_curr;
    S_re ← S_re ∪ S_inter;
return S_re;
```

B.4 Computational Complexity Analysis

To theoretically validate the retrieval efficiency of NoLLMRAG, we analyze the time complexity of the clustering-based retrieval via keyword co-occurrence (Section 3.3). We utilize the following notations:

- k: The number of keywords extracted from the query (i.e., $|\mathcal{V}_T^{key}|$). This is typically a small constant relative to the corpus size.
- d_{key}: The average degree of a keyword node in the bipartite subgraph.
- d_{chunk}: The average degree of a chunk node in the bipartite subgraph. Noted that $d_{chunk} \leq k$.

We analyze the complexity by dividing the process into its three sequential stages:

Co-occurrence Subgraph Extraction. The construction of the keyword co-occurrence subgraph $\mathcal{G}_{CO}$ is equivalent to a one-mode projection of the bipartite subgraph induced by the query keywords.

- **Neighbor Discovery:** For each of the k keywords, we retrieve its connected chunk nodes (d_{key} operations). For each retrieved chunk, we identify which other query keywords are present (d_{chunk} operations). The cost is $k \cdot d_{key} \cdot d_{chunk}$. Since $d_{chunk} \leq k$, this is bounded by $O(k^2 \cdot d_{key})$.

- **Edge Weight Calculation:** For every pair of co-occurring keywords (edges in $\mathcal{G}_{CO}$), we compute the intersection of their chunk sets. The number of edges is bounded by k^2, and the intersection cost is linear to the chunk set size (d_{key}). The cost is $O(k^2 \cdot d_{key})$.

Step Complexity: $O(k^2 \cdot d_{key})$.

Keyword Clustering with Community Detection. We apply the Leiden algorithm to the subgraph $\mathcal{G}_{CO}$.

- **Leiden Algorithm:** The subgraph $\mathcal{G}_{CO}$ comprises k nodes and at most $k(k-1)/2$ edges. Since the computational complexity of the Leiden algorithm is linear to the number of edges per iteration ($O(E)$), and it converges rapidly, this step is strictly bounded by $O(k^2)$.

Step Complexity: $O(k^2)$.

Cluster-Based Chunk Retrieval. As detailed in Algorithm 1, we retrieve chunks for each cluster.

- **Sorting:** Sorting keywords by degree takes $O(k \cdot \log k)$.
- **Iterative Intersection:** We iterate through the k keywords. For each keyword, we retrieve its chunk set (size $\approx d_{key}$) and perform a set intersection with the current candidate set. The cost is $O(k \cdot d_{key})$.

Step Complexity: $O(k \cdot \log k + k \cdot d_{key})$.

Total Complexity. Summing the components, the total time complexity T_{total} is:

$$T_{total} = O(k^2 \cdot d_{key}) + O(k^2) + O(k \cdot \log k + k \cdot d_{key}) \tag{8}$$

Since the number of keywords k is usually much smaller than the keyword frequency d_{key}, and $\log k$ is negligible, the overall complexity simplifies to:

$$T_{total} \approx O(k^2 \cdot d_{key}) \tag{9}$$

This analysis shows that the complexity grows linearly with respect to the average number of chunks associated with a keyword and quadratically with respect to the query length, which is usually very small in practice. This theoretical advantage is directly reflected in our experiments, where we observe a up to 15 $\times$ QA speedup over baselines (Section 5.2).

C More Details of Experiments

C.1 Datasets

PopQA focuses on factual knowledge, especially long-tail entities. The queries in this dataset involve entities that are less common, making it particularly well-suited for evaluating model performance on simple QA tasks that require precise retrieval of infrequent information.

MultiHop-RAG is specifically designed to evaluate multi-document reasoning ability. It is built from real-world news articles and includes four types of queries—Inference, Comparison, Temporal, and Null—reflecting real-world complexities in RAG applications.

LiHua-World captures a year-long virtual user's communication across digital platforms, integrating personal experiences, temporal dynamics, and contextually rich interactions. The questions in this dataset span single-hop, multi-hop, and summarization tasks, requiring models to reason over fragmented and evolving information.

The statistics of these three datasets are summarized in Table 4.

C.2 Hyperparameters

We adopt the official default hyperparameter settings for all baseline methods. Details are provided in Table 5.

Table 4: Statistics of the datasets used in our experiments.

Dataset	LiHua-World	MultiHop-RAG	PopQA
Num of documents	442	609	8,676
Num of total characters	957,793	6,418,348	4,725,700
Num of questions	637	2,556	1,000

Table 5: Summary of hyperparameters for LightRAG, MiniRAG, and HippoRAG 2

Hyperparameter	LightRAG	MiniRAG	HippoRAG 2
Mode	hybrid	–	–
Top-k Phrases for QA	60	60	5
Chunk Token Size	1,200	1,200	whole doc
Chunk Overlap Token Size	100	100	–
Entity Summary Max Tokens	500	500	–
Synonym Threshold	–	–	0.8
Damping Factor of PPR	–	–	0.5
Temperature	–	–	0.0

C.3 Experimental Environment

All experiments were performed on a desktop computer with the following specifications:

- **Hardware:** NVIDIA GeForce RTX 4070 SUPER GPU, Intel Core i5-12600KF CPU, and 32GB RAM.
- **Operating System:** Windows 10 Professional 64-bit.
- **Model Execution:** All local LLMs and embedding models were run via Ollama 0.11.11 (Ollama, 2025). GPT-4o-mini was accessed through the official API.

C.4 Comparison with Classical IR Methods

To further validate the performance superiority of NoLLMRAG, we compare it against three classical information retrieval (IR) methods using Qwen2.5-7B-Instruct: **Sparse Retrieval (BM25)**, **Dense Retrieval (BGE-M3)**, and **Hybrid Retrieval (BM25 + BGE-M3)**.

As shown in Table 6, NoLLMRAG consistently outperforms these baselines across all datasets, especially in complex reasoning tasks, highlighting the advantage of NoLLMRAG's design.

C.5 Graph Statistics

The statistics of the three-layer heterogeneous graphs constructed for the LiHua-World, MultiHop-RAG, and PopQA datasets are summarized in Table 7

C.6 Case Study

To qualitatively illustrate the advantages of NoLLMRAG over all baselines, we present a representative case in Table 8. The selected query, taken from MultiHop-RAG, requires reasoning over multiple document paragraphs to get the correct answer.

In this instance, NoLLMRAG successfully produces the correct answer, Sam Bankman-Fried. In contrast, all other baselines fail. LightRAG outputs Reza Pour, which is a wrong but plausible-sounding individual. MiniRAG outputs William Martel, another wrong entity. HippoRAG2 outputs Google and provides a long, off-topic explanation about antitrust cases, showing a clear topic misalignment.

Table 6: Comparison of NoLLMRAG with classical IR methods on Qwen2.5-7B-Instruct. **Bold** values represent the best performance.

Method	LiHua-World		MultiHop-RAG		PopQA	
	acc	err	acc	err	acc	err
BM25	59.34%	22.29%	58.02%	32.04%	63.30%	5.70%
BGE-M3	42.70%	18.21%	46.64%	20.38%	63.20%	4.60%
BM25 + BGE-M3	61.22%	17.27%	61.19%	17.64%	67.90%	5.00%
NoLLMRAG	**73.16%**	16.17%	**69.05%**	17.49%	**70.50%**	5.00%

Table 7: Statistics of the constructed three-layer heterogeneous graphs

	Metric	LiHua-World	MultiHop-RAG	PopQA
Node Counts	Document Nodes ($\mathcal{V}^D$)	442	609	8,676
	Chunk Nodes ($\mathcal{V}^C$)	2,350	13,491	13,963
	Token Nodes ($\mathcal{V}^T$)	5,491	36,848	59,504
	Total Nodes	8,283	50,948	82,143
Average Degrees	Document Nodes ($\mathcal{V}^D$)	5.32	22.15	1.61
	Chunk Nodes ($\mathcal{V}^C$)	45.14	49.21	31.88
	Token Nodes ($\mathcal{V}^T$)	38.51	42.11	19.10
Edge Counts	Doc-Chunk Edges ($\mathcal{E}^{DC}$)	2,350	13,491	13,963
	Chunk-Token Edges ($\mathcal{E}^{CT}$)	103,727	650,370	431,160
	Token-Token Edges ($\mathcal{E}^{TT}$)	53,894	450,838	353,028
	Total Edges	159,971	1,114,699	798,151
Graph Density		3.88×10^{-3}	6.85×10^{-4}	1.84×10^{-4}
Disk Storage Size		40.3 MB	210 MB	139 MB

This case highlights that NoLLMRAG can effectively retrieve information from multiple documents to generate the correct answer, underlining its superior multi-hop reasoning performance.

C.7 Additional Metric Definitions

Coefficient of Variation (CV). To measure the stability of performance across corpus expansion steps, we compute the Coefficient of Variation (CV) for each method:

$$\mathrm{CV} = \frac{\sigma}{\mu},$$

where σ is the standard deviation and μ is the mean of the evaluation metric (e.g., accuracy) across all expansion steps. A smaller CV indicates more stable performance.

C.8 Statistics of Corpus Expansion

Table 9 shows the statistics of the real-time corpus expansion for the three datasets. We observe that, as similarly sized corpus segments are added, the number of newly introduced tokens generally decreases, leading to shorter indexing times. This reduction is particularly pronounced in the initial expansion stages.

C.9 More Details of Ablation Study

C.9.1 Implementation Details

As decribed in Section 3, NoLLMRAG consists of three tightly integrated components: LLM-free graph indexing, graph-statistics-driven keyword extraction, and clustering-based retrieval via keyword co-occurrence. Among these, LLM-free graph indexing serves as the foundation of the entire

Table 8: A specific case including the query, gold answer, and responses from NoLLMRAG and all baselines, based on Qwen2.5-7B-Instruct.

Query	Who is the individual associated with the cryptocurrency industry facing a criminal trial on fraud and conspiracy charges, as reported by both The Verge and TechCrunch, and is accused by prosecutors of committing fraud for personal gain?
Gold Answer	Sam Bankman-Fried
LightRAG	The individual you are referring to is probably **Reza Pour**, also known as C̈rypto Reza.Ḧe was indicted in October 2021 on multiple counts of conspiracy to commit wire fraud and wire fraud. According to the charges, he allegedly engaged in fraudulent schemes involving cryptocurrency projects, including a Ponzi scheme, for personal financial gain. The case has been widely covered by tech news outlets like The Verge and TechCrunch due to its complexities and the involvement of prominent figures in the crypto industry. However, it's important to note that until there is a verdict, these are accusations made by prosecutors.
MiniRAG	The individual you are referring to is probably **William Martel**, also known as Ẅilli Dix.Ḧe was indicted in 2019 on multiple counts of wire fraud and conspiracy to commit wire fraud. According to the charges, he allegedly engaged in fraudulent schemes related to cryptocurrency. Both The Verge and TechCrunch have reported on this case. However, it's important to note that names and details can sometimes get mixed up when referring to specific cases, so if you need more precise information, double-checking with reliable sources is recommended.
HippoRAG2	Based on the information provided in the articles from The Verge and TechCrunch, the individual associated with the cryptocurrency industry facing a criminal trial on fraud and conspiracy charges is **Google**. However, it seems there might be some confusion as the articles are discussing different cases: 1. **Google Antitrust Case**: The articles discuss an ongoing antitrust case against Google by the U.S. Justice Department, which alleges that Google uses its platforms and deals with partners to block out any competition in search or advertising. 2. **US v Google Trial**: This trial involves various aspects of Google's business practices, including vertical search, pre-installs on devices, and agreements with Android OEMs. The case is about whether Google has engaged in anticompetitive behavior that harms competitors. 3. **Google vs Epic Games**: Another ongoing case where Google is being sued by Fortnite maker Epic Games for alleged anticompetitive behavior related to its app store policies. 4. **Microsoft Case**: The article from The Verge mentions a 1990s Microsoft trial, which is unrelated to the current cryptocurrency industry context. 5. **US v Google (2023)**: This case involves allegations that Google has used its dominant position in search and advertising to harm competitors, particularly by making it difficult for other companies to access user data and compete effectively. Given this information, there isn't a specific individual from the cryptocurrency industry mentioned as facing fraud charges. The articles are discussing broader antitrust issues related to Google's business practices rather than an individual involved in cryptocurrency fraud. If you're referring to someone specifically within the cryptocurrency space, it would be helpful to have more context or details about the case being reported.
NoLLMRAG	The individual associated with the cryptocurrency industry facing a criminal trial on fraud and conspiracy charges, as reported by both The Verge and TechCrunch, is **Sam Bankman-Fried**. He is the founder of FTX, a failed cryptocurrency exchange, and Alameda Research. Prosecutors accuse him of committing wire fraud for personal gain, misappropriating customer funds, and other related offenses.

Table 9: Statistics of corpus expansion for three datasets, including the number of characters in each corpus segment (Chars), the corresponding indexing time (IT), and the number of newly added tokens (New tokens)

Dataset	Metric	Segments									
		1	2	3	4	5	6	7	8	9	10
LiHua-World	Chars	92193	96667	96538	97235	95875	91149	100048	90999	100916	96173
	IT (s)	23.43	18.18	16.00	14.45	16.90	19.02	15.70	15.55	15.76	15.21
	New tokens	1236	742	502	304	510	771	370	368	383	305
MultiHop-RAG	Chars	630658	648040	642011	641612	643932	624367	654248	647674	635903	649903
	IT (s)	127.24	91.52	77.54	74.94	73.58	63.70	67.90	76.72	65.65	66.07
	New tokens	9776	5254	3770	3377	3218	1982	2349	2969	2054	2099
PopQA	Chars	468155	476788	472559	472621	471456	473480	472037	472984	472543	473077
	IT (s)	151.51	98.39	90.40	87.71	82.26	104.88	80.05	73.16	80.50	75.79
	New tokens	12847	6848	5794	5500	4661	7551	4436	3576	4382	3909

framework and is deeply integrated with the other two components. An ablation on this component would inevitably change the downstream processing pipeline of both keyword extraction and retrieval, making a clean ablation infeasible. Moreover, its effectiveness has already been validated in Section 5.2, where we demonstrated substantial improvements in indexing time compared to all baselines. Therefore, we focus our ablation study on the remaining two components.

For graph-statistics-driven keyword extraction, we replace it with a commonly used LLM-based approach without using any graph statistics: the query is first passed to an LLM to extract keywords, which are then matched to the most similar token nodes via vector database. The prompt we used in LLM-based approach is provided in Appendix D.3. **For clustering-based retrieval via keyword co-occurrence**, we remove the keyword clustering step and directly apply the cluster-based chunk retrieval algorithm to the whole keywords which are collectively treated as a single cluster.

Table 10: Comparison of QA Time per Query (QT) of NoLLMRAG and the ablation for graph-statistics-driven keyword extraction on three datasets using Qwen2.5-7B-Instruct (in seconds). **Bold** values represent the fastest time.

Method	LiHua-World	MultiHop-RAG	PopQA
	QT	QT	QT
NoLLMRAG	**2.04**	**5.25**	**2.08**
w/o Graph stats	2.54	5.64	2.94

C.9.2 Efficiency Analysis

Table 10 shows the QA time per query of NoLLMRAG and the ablation for graph-statistics-driven keyword extraction on three datasets using Qwen2.5-7B-Instruct, which indicates that replacing graph-statistics-driven keyword extraction with the LLM-based method increases the QA time with the largest relative increase reaching 41.35%. These results, together with the accuracy drops reported in Section 5.3, demonstrate that the graph-statistics-driven keyword extraction method is crucial not only for improving answer quality but also for maintaining efficient retrieval.

D Prompts

D.1 Prompt for Generation

This prompt is used to guide the LLM in generating answers conditioned on the retrieved contexts and the user query. The complete template is shown in Figure 3.

system:
---Role---
You are a helpful assistant responding to questions based on context provided.

---Goal---
Generate a response that responds to the user's question, summarizing all information in the context, and incorporating any relevant general knowledge.
If you don't know the answer, just say so. Do not make anything up.
Do not include information where the context for it is not provided.

user:
Context:
context1:
{content}

context2:
{content}

……

Question: {query}

Figure 3: LLM prompt for generation

D.2 Prompt for Evaluation

This prompt is employed in the evaluation stage, where the LLM is used as a judge to measure whether a response is semantically equivalent to the gold answer. The complete prompt is given in Figure 4.

system:
Now, I'll give you a question, a gold answer to this question, and four answers provided by different students.

Determine the answer according to the following rules:
If the answer is correct, get 1 point.
If the answer is irrelevant to the question, it will receive 0 points.
If the answer is incorrect, get -1 point.

Return your answer in JSON mode.

For example:

Example1:
Question:
When does Li Hua arrive to the city?

Gold Answer:
20260105

Answer1: LiHua arrived on the afternoon of January 5th
Answer2: Sorry, there is no information about LiHua's arrival in the information you provided
Answer3: There is no accurate answer in the information you provided, but according to the first information found, LiHua arrived on April 17th
Answer4: Jan 5, 2026

output:
{{
"Score1": 1,
"Score2": 0,
"Score3": -1,
"Score4": 1,
}}

Example2:
Question:
When does Li Hua arrive to the city?

Gold Answer:
Insufficient information

Answer1: LiHua arrived on the afternoon of January 5th
Answer2: Sorry, there is no information about LiHua's arrival in the information you provided
Answer3: There is no accurate answer in the information you provided, but according to the first information found, LiHua arrived on April 17th
Answer4: Insufficient information

output:
{{
"Score1": -1,
"Score2": 1,
"Score3": -1,
"Score4": 1,
}}

Real data:

Question:
{question}
Gold Answer:
{ga}

Answer1: {light}
Answer2: {mini}
Answer3: {hippo2}
Answer4: {nollm}

output:

Figure 4: LLM prompt for evaluation

D.3 Prompt for LLM-Based Keyword Extraction

This prompt is used for the LLM-based keyword extraction baseline in the ablation study, where the LLM is asked to extract the most important keywords from a user query. Figure 5 provides the detailed prompt.

system:
You are an assistant that extracts search keywords from user queries for a Retrieval-Augmented Generation (RAG) system.
Given a question, identify the most relevant content-bearing words (entities, concepts, technical terms, domain-specific expressions) that are important for retrieving supporting documents.
Do not include stopwords, filler words, or generic terms.
Return the result only as a JSON object in the following format, without extra text or explanation:

{ "keywords": ["keyword1", "keyword2", "keyword3"] }

Example:

Input:
"What are the main challenges of applying deep learning in medical image analysis?"

Output:
{ "keywords": ["deep learning", "medical image analysis", "challenges"] }

user:
{query}

Figure 5: LLM prompt for LLM-based keyword extraction