PathRAG: Pruning Graph-Based Retrieval Augmented Generation with Relational Paths

Boyu Chen[1], Zirui Guo[2], Zidan Yang[1], Yuluo Chen[1], Junze Chen[1], Zhenghao Liu[3], Chuan Shi[1], Cheng Yang[1†]

[1]Beijing University of Posts and Telecommunications, China
[2]University of Hong Kong, China
[3]Northeastern University, China
chenbys4@bupt.edu.cn, yangcheng@bupt.edu.cn

Abstract

Retrieval-augmented generation (RAG) improves the response quality of large language models (LLMs) by retrieving knowledge from external databases. Typical RAG approaches split the text database into chunks, organizing them in a flat structure for efficient searches. To better capture the inherent dependencies and structured relationships across the text database, researchers propose to organize textual information into an indexing graph, known as *graph-based RAG*. However, we argue that the limitation of current graph-based RAG methods lies in the redundancy of the retrieved information, rather than its insufficiency. Moreover, previous methods use a flat structure to organize retrieved information within the prompts, leading to suboptimal performance. To overcome these limitations, we propose PathRAG, which retrieves key relational paths from the indexing graph, and converts these paths into textual form for prompting LLMs. Specifically, PathRAG effectively reduces redundant information with flow-based pruning, while guiding LLMs to generate more logical and coherent responses with path-based prompting. Experimental results show that PathRAG consistently outperforms state-of-the-art baselines across six datasets and five evaluation dimensions.

Code — https://github.com/BUPT-GAMMA/PathRAG

Introduction

Retrieval-augmented generation (RAG) empowers large language models (LLMs) to access up-to-date or domain-specific knowledge from external databases, enhancing the response quality without additional training (Gao et al. 2023a,b; Fan et al. 2024; Procko and Ochoa 2024; Mavromatis and Karypis 2024; Su et al. 2025; Zhang, Feng, and You 2025). Most approaches divide the text database into chunks, organizing them in a flat structure to facilitate efficient searches (Yepes et al. 2024; Lyu et al. 2025).

To better capture the inherent dependencies and structured relationships across texts in a database, researchers have introduced graph-based RAG (Edge et al. 2024; Guo et al. 2024), which organizes textual information into an indexing graph. In this graph, nodes represent entities extracted from the text, while edges denote the relationships between these entities. Traditional RAG (Liu et al. 2021; Yasunaga et al. 2021; Gao et al. 2022) usually focuses on questions that can be answered with local information about a single entity or relationship. In contrast, graph-based RAG targets on global-level questions that need the information across a database to generate a summary-like response. For example, GraphRAG (Edge et al. 2024) first applies community detection on the graph, and then gradually summarizes the information in each community. The final answer is generated based on the most query-relevant communities. LightRAG (Guo et al. 2024) extracts both local and global keywords from input queries, and retrieves relevant nodes and edges using these keywords. The ego-network information of the retrieved nodes is then used as retrieval results.

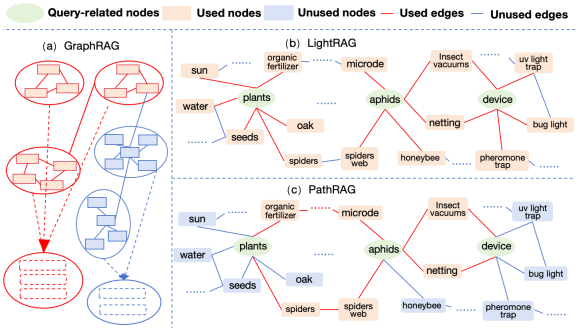

Figure 1: Comparison between different graph-based RAG methods. GraphRAG (Edge et al. 2024) uses all the information within certain communities, while LightRAG (Guo et al. 2024) uses all the immediate neighbors of query-related nodes. In contrast, the proposed PathRAG focuses on key relational paths between query-related nodes to alleviate noise and reduce token consumption.

However, we argue that the information considered in previous graph-based RAG methods is often redundant, which can introduce noise, degrade performance, and increase token consumption. GraphRAG method uses all the information from the nodes and edges within certain communities. Similarly, LightRAG retrieves the immediate neighbors of query-related nodes to generate answers. The redundant information retrieved in these methods may act as noise, and

†Corresponding author.
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

negatively affecting subsequent generation. Moreover, both methods adopt a flat structure to organize retrieved information in the prompts, *e.g.,* directly concatenating the textual information of all retrieved nodes and edges, resulting in answers with suboptimal logicality and coherence.

To overcome the above limitations, we propose PathRAG, which performs key path retrieval among retrieved nodes and converts these paths into textual form for LLM prompting. We focus on the key relational paths between retrieved nodes to alleviate noise and reduce token consumption. Specifically, we first retrieve relevant nodes from the indexing graph based on the keywords in the query. Then we design a flow-based pruning algorithm with distance awareness to identify the key relational paths between each pair of retrieved nodes. The pruning algorithm enjoys low time complexity, and can assign a reliability score to each retrieved path. Afterward, we sequentially concatenate the node and edge information alongside each path as textual relational paths. Considering the "lost in the middle" issue of LLMs (Liu et al. 2024), we place the textual paths into the prompt in ascending order of reliability scores for better answer generation. To comprehensively evaluate the effectiveness of PathRAG, we extend the benchmark datasets from prior work (Guo et al. 2024) with four additional ones (Wang et al. 2022; Chen et al. 2022; Qian et al. 2024) from different domains. Experimental results demonstrate that PathRAG consistently outperforms state-of-the-art baselines across all five evaluation dimensions. In particular, compared to GraphRAG and LightRAG, PathRAG achieves average win rates of 59.93% and 57.09%, respectively. The contributions of this work are as follows:

- We highlight that the limitation of current graph-based RAG methods lies in the redundancy of the retrieved information, rather than its insufficiency. Moreover, previous methods use a flat structure to organize retrieved information within the prompts, leading to suboptimal performance.
- We propose PathRAG, which efficiently retrieves key relational paths from an indexing graph with flow-based pruning, and effectively generates answers with path-based LLM prompting.
- PathRAG outperforms state-of-the-art baselines across six datasets and five evaluation dimensions. Extensive experiments further validate the design of PathRAG.

Related Work

Text-based RAG. To improve text quality (Fang et al. 2024; Xu et al. 2024; Zhu et al. 2024) and mitigate hallucination effects (Lewis et al. 2020; Guu et al. 2020), retrieval-augmented generation (RAG) is widely used in large language models (LLMs) by leveraging external databases. These databases primarily store data in textual form, containing a vast amount of domain knowledge that LLMs can directly retrieve. We refer to such systems as text-based RAG. Based on different retrieval mechanisms (Fan et al. 2024), text-based RAG can be broadly classified into two categories: **sparse vector retrieval** (Alon et al. 2022; Schick et al. 2023; Jiang et al. 2023; Cheng et al. 2023) and **dense vector retrieval** (Lewis et al. 2020; Hofstätter et al. 2023; Li et al. 2024; Zhang et al. 2024). Sparse vector retrieval typically identifies the most representative words in each text segment by word frequency, and retrieves relevant text for a specific query based on keyword matching. In contrast, dense vector retrieval addresses issues like lexical mismatches and synonyms by encoding both query terms and text into vector embeddings. It then retrieves relevant content based on the similarity between these embeddings. However, most text-based RAG methods use a flat organization of text segments, and fail to capture essential relationships between chunks (*e.g.,* the contextual dependencies), limiting the quality of LLM-generated responses (Edge et al. 2024; Guo et al. 2024).

KG-RAG. Besides text databases, researchers have proposed retrieving information from knowledge graphs (KGs), known as KG-RAG (Yasunaga et al. 2021; Gao et al. 2022; Li, Miao, and Li 2024; Procko and Ochoa 2024; He et al. 2025). These methods can utilize existing KGs (Wen, Wang, and Sun 2024; Dehghan et al. 2024) or their optimized versions (Fang, Meng, and Macdonald 2024; Panda et al. 2024), and enable LLMs to retrieve information of relevant entities and their relationships. Specifically, KG-RAG methods typically extract a local subgraph from the KG (Bordes et al. 2015; Talmor and Berant 2018; Gu et al. 2021), such as the immediate neighbors of the entity mentioned in a query. However, most KG-RAG methods focus on addressing questions that can be answered with a single entity or relation in the KG (Joshi et al. 2017; Yang et al. 2018; Kwiatkowski et al. 2019; Ho et al. 2020), narrowing the scope of their applicability.

Graph-based RAG. Instead of utilizing pre-constructed KGs, graph-based RAG (Edge et al. 2024; Guo et al. 2024) typically organizes text databases as text-associated graphs, and focuses on global-level tasks that need the information from multiple segments across a database. The graph construction process often involves extracting entities from the text and identifying relationships between these entities. Also, contextual information is included as descriptive text to minimize the information loss during the text-to-graph conversion. GraphRAG (Edge et al. 2024) first applies community detection algorithms on the graph, and then gradually aggregates the information from sub-communities to form higher-level community information. LightRAG (Guo et al. 2024) adopts a dual-stage retrieval framework to accelerate the retrieval process. First, it extracts both local and global keywords from the question. Then, it retrieves relevant nodes and edges using these keywords, treating the ego-network information of the retrieved nodes as the final retrieval results. This approach simplifies retrieval and effectively handles global-level tasks. However, the retrieved information covers all immediate neighbors of relevant nodes, which may introduce noise harming the answer quality. A recent work MiniRAG (Fan et al. 2025) also considers path information to assist retrieval. But they focus on addressing questions that can be answered by the information of a specific node, and thus explore paths between query-related and answer-related nodes like KG reasoning (Yasunaga et al. 2021; Liu et al. 2021; Tian et al. 2022).

Preliminaries

In this section we will introduce and formalize the workflow of a graph-based RAG system.

Instead of storing text chunks as an unordered collection, graph-based RAG automatically structures a text database into an **indexing graph** as a preprocessing step. Given a text database, the entities and their interrelations within the textual content are identified by LLMs, and utilized to construct the node set $\mathcal{V}$ and edge set $\mathcal{E}$. Specifically, each node $v \in \mathcal{V}$ represents a distinct entity with an identifier k_v (*e.g.,* entity name) and a textual chunk t_v (*e.g.,* associated text snippets), while each edge $e \in \mathcal{E}$ represents the relationship between entity pairs with a descriptive textual chunk t_e to enrich relational context. We denote the indexing graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{K}_\mathcal{V}, \mathcal{T})$, where $\mathcal{K}_\mathcal{V}$ represent the collection of node identifiers and $\mathcal{T}$ is the collection of textual chunks in the indexing graph.

Given a query q, a graph-oriented retriever extracts relevant nodes and edges in the indexing graph. Then the textual chunks of retrieved elements are integrated with query q to obtain the answer by an LLM generator. The above process can be simplified as:

$$\mathcal{A}(q, \mathcal{G}) = \mathcal{F} \circ \mathcal{M}(q; \mathcal{R}(q, \mathcal{G})), \tag{1}$$

where $\mathcal{A}$ denotes the augmented generation with retrieval results, $\mathcal{R}$ means the graph-oriented retriever, $\mathcal{M}$ and $\mathcal{F}$ represent the prompt template and the LLM generator, respectively. In this paper, we primarily focus on designing a more effective graph-oriented retriever and the supporting prompt template to achieve a better graph-based RAG.

Methodology

In this section, we propose a novel graph-based RAG framework with the path-based retriever and a tailored prompt template, formally designated as PathRAG. As illustrated in Figure 2, the proposed framework operates on an indexing graph through three sequential stages: node retrieval, path retrieval, and answer generation.

Node Retrieval

In this stage, we identify keywords from the input query by LLMs, and accordingly extract relevant nodes from the indexing graph. Given a query q, an LLM is utilized to extract keywords from the query text. The collection of keywords extracted from query q is denoted as $\mathcal{K}_q$. Based on the extracted keywords, dense vector matching is employed to retrieve related nodes in the indexing graph $\mathcal{G}$. In dense vector matching, the relevance between a keyword and a node is calculated by their similarity in the semantic embedding space, where the commonly used cosine similarity is adopted in our method. Specifically, we first encode both node identifiers and the extracted keywords using a semantic embedding model $f : \mathcal{K}_q \cup \mathcal{K}_\mathcal{V} \to \mathcal{X}_q \cup \mathcal{X}_\mathcal{V}$, where $\mathcal{X}_\mathcal{V} = \{x_v\}_{v \in \mathcal{V}}$ represents the embeddings of node identifiers, and $\mathcal{X}_q = \{x_{q,i}\}_{i=1}^{|\mathcal{K}_q|}$ denotes the embeddings of the extracted keywords. Based on the obtained embeddings above, we then iterate over $\mathcal{X}_q$ to search the most relevant nodes among $\mathcal{X}_\mathcal{V}$ with the embedding similarity, until a predefined number N of nodes is reached. The resulting subset of retrieved nodes is denoted as $\mathcal{V}_q \subseteq \mathcal{V}$.

Path Retrieval

In this subsection, we introduce the path retrieval module that aggregates textual chunks in the form of relational paths to capture the connections between retrieved nodes.

Given two distinct retrieved nodes $v_{\text{start}}, v_{\text{end}} \in \mathcal{V}_q$, there could be many reachable paths between them. Since not all paths are helpful to the task, further refinement is needed to enhance both effectiveness and efficiency. Inspired by the resource allocation strategy (Lü and Zhou 2011; Lin et al. 2015), we propose a flow-based pruning algorithm with distance awareness to extract key paths.

Formally, we denote the sets of nodes pointing to v_i and nodes pointed by v_i as $\mathcal{N}(\cdot, v_i)$ and $\mathcal{N}(v_i, \cdot)$, respectively. We define the resource of node v_i as $\mathcal{S}(v_i)$. We set $\mathcal{S}(v_{\text{start}}) = 1$ and initialize other resources to 0, followed by propagating the resources through the neighborhood. The resource flowing to v_i is defined as:

$$\mathcal{S}(v_i) = \sum_{v_j \in \mathcal{N}(\cdot, v_i)} \frac{\alpha \cdot \mathcal{S}(v_j)}{|\mathcal{N}(v_j, \cdot)|}, \tag{2}$$

where α represents the decay rate of information propagation along the edges. Based on the assumption that the closer two nodes are in the indexing graph, the stronger their connection will be, we introduce this penalty mechanism to enable the retriever to perceive distance. It is crucial to emphasize that our approach differs from strictly sorting paths with a limited number of hops. Detailed comparative experiments will be presented in subsequent sections.

Notably, due to the decay penalty and neighbor allocation, nodes located far from the initial node are assigned with negligible resources. Therefore, we introduce an early stopping strategy to prune paths in advance when

$$\frac{\mathcal{S}(v_i)}{|\mathcal{N}(v_i, \cdot)|} < \theta, \tag{3}$$

where θ is the pruning threshold. This ensures that the algorithm terminates early for nodes that contribute minimally to the overall propagation. For efficiency concerns, we update the resource of a node at most once.

We denote each path as an ordered sequence $P = v_0 \xrightarrow{e_0} \cdots v_i \xrightarrow{e_i} \cdots = (\mathcal{V}_P, \mathcal{E}_P)$, where v_i and e_i represent the i-th node and directed edge, and $\mathcal{V}_P$ and $\mathcal{E}_P$ represent the set of nodes and edges in the path $\mathcal{P}$, respectively. For each path $P = (\mathcal{V}_P, \mathcal{E}_P)$, we calculate the average resource values flowing through its edges as the measurement of reliability, which can be formulated as:

$$\mathcal{S}(P) = \frac{1}{|\mathcal{E}_P|} \sum_{v_i \in \mathcal{V}_P} S(v_i), \tag{4}$$

where $|\mathcal{E}_P|$ is the number of edges in the path. Then, we sort these paths based on the reliability $\mathcal{S}(P)$ and retain only the most reliable relational paths for this node pair. These paths are added to the global candidate pool in the form of path-reliability pair $(P, \mathcal{S}(P))$. We repeat the above process for

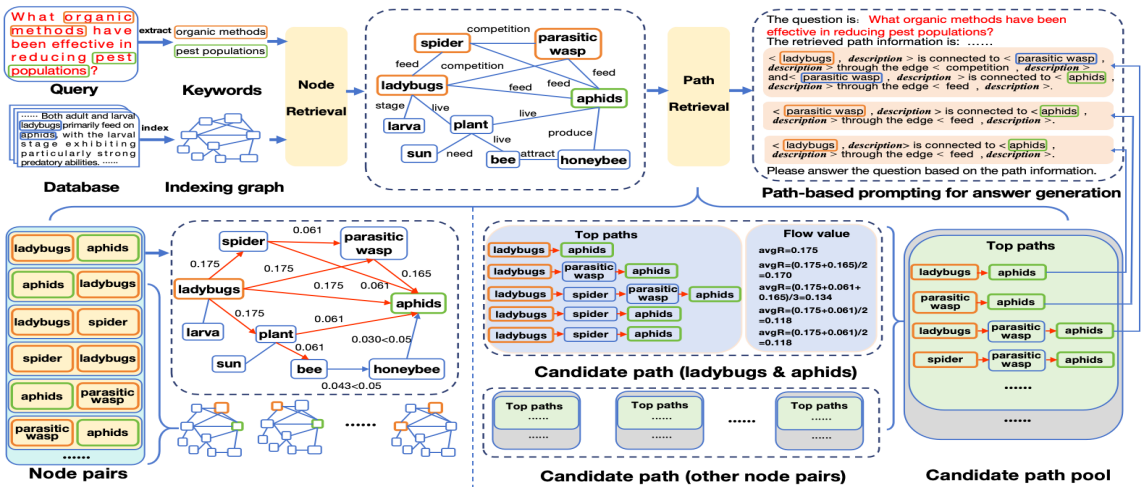

Figure 2: The overall framework of our proposed PathRAG with three main stages. 1) Node Retrieval Stage: Relevant nodes are retrieved from the indexing graph based on the keywords in the query; 2) Path Retrieval Stage: We design a flow-based pruning algorithm to extract key relational paths between each pair of retrieved nodes, and then retrieve paths with the highest reliability scores; 3) Answer Generation Stage: The retrieved paths are placed into prompts in ascending order of reliability scores, and finally fed into an LLM for answer generation.

each distinct node pair, ultimately obtaining all candidate paths. Then the top-K reliable paths can be obtained from the candidate pool to serve as the retrieval information of query q for subsequent generation, which we denote as $\mathcal{P}_q$.

Answer Generation

For better answer generation, we establish path prioritization based on their reliability, then strategically position these paths to align with LLMs' performance patterns (Qin et al. 2023; Liu et al. 2024; Cuconasu et al. 2024).

Formally, for each retrieved relational path, we concatenate the textual chunks of all nodes and edges within the path to obtain a textual relational path, which can be formulated as:

$$t_P = \text{concat}([\cdots; t_{v_i}; t_{e_i}; t_{v_{i+1}}; \cdots]), \tag{5}$$

where $\text{concat}(\cdot)$ denotes the concatenation operation, v_i and e_i are the i-th node and edge in the path P, respectively.

Considering the "lost in the middle" issue (Liu et al. 2024; Cao et al. 2025) for LLMs in long-context scenarios, directly aggregating the query with different relational paths may lead to suboptimal results. Therefore, we position the most critical information at the two ends of the template, which is regarded as the golden memory region for LLM comprehension. Specifically, we place the query at the beginning of the template and organize the textual relational paths in a reliability ascending order, ensuring that the most reliable relational path is positioned at the end of the template. The final prompt can be denoted as:

$$\mathcal{M}(q; \mathcal{R}(q, \mathcal{G})) = \text{concat}([q; t_{P_K}; \cdots; t_{P_1}]), \tag{6}$$

where P_1 is the most reliable path and P_K is the K-th reliable path. This simple prompting strategy can significantly improve the response performance of LLM compared with placing the paths in a random or reliability ascending order in our experiments.

Discussion

Path Selection Algorithms in RAG. While previous methods in KG-RAG enhance model reasoning performance through path selection algorithms (Asai et al. 2019; Sun et al. 2023; Chen et al. 2024), they are primarily designed for tasks that can be answered by the information contained in a single node or edge, making them unsuitable for global-level tasks. Moreover, in prior approaches, paths are typically used as a means to obtain retrieval results rather than being part of the retrieval output itself. When multiple paths lead to retrieval results, no efficient method exists to filter the paths. In contrast, PathRAG employs a flow-based pruning algorithm to evaluate multiple paths between nodes, selecting relevant paths as part of the retrieval result to address global-level tasks.

Complexity Analysis of Path Retrieval. After the i-th step of resource propagation, there are at most $\frac{\alpha^i}{\theta}$ nodes alive due to the decay penalty and early stopping. Hence the total number of nodes involved in this propagation is at most $\sum_{i=0}^{\infty} \alpha^i/\theta = \frac{1}{(1-\alpha)\theta}$. Thus the complexity of extracting candidate paths between all node pairs is $\mathcal{O}(\frac{N^2}{(1-\alpha)\theta})$. In our settings, the number of retrieved nodes $N \in [10, 60]$ is much less than the total number of nodes in the indexing graph $|\mathcal{V}| \sim 10^4$. Thus the time complexity is completely acceptable. Further details are provided in Appendix H.

Necessity of Path-based Prompting. Note that different retrieved paths may have shared nodes or edges. To reduce the prompt length, it is possible to flatten the paths and remove duplications as a set of nodes and edges. However, this conversion will lose the semantic relations between the two endpoints of each path. We also validate the necessity of path-based prompting in the experiments.

Table 1: Performance across six datasets and five evaluation dimensions in terms of win rates.

	Legal		**History**		**Biology**		**Mix**		**SQuALITY**		**SummScreen**	
	NaiveRAG	**PathRAG**	NaiveRAG	**PathRAG**	NaiveRAG	**PathRAG**	NaiveRAG	**PathRAG**	NaiveRAG	**PathRAG**	NaiveRAG	**PathRAG**
Comprehensiveness	31.6%	68.4%	33.2%	66.8%	29.8%	70.2%	26.2%	73.8%	35.2%	64.8%	30.0%	70.0%
Diversity	24.4%	75.6%	38.4%	61.6%	35.2%	64.8%	33.2%	66.8%	29.2%	70.8%	24.2%	75.8%
Logicality	35.2%	64.8%	40.4%	59.6%	34.4%	65.6%	36.2%	63.8%	34.4%	65.6%	30.2%	69.8%
Relevance	27.2%	72.8%	37.2%	62.8%	42.0%	58.0%	38.4%	61.6%	31.4%	68.6%	33.0%	67.0%
Coherence	34.0%	66.0%	42.4%	57.6%	38.4%	61.6%	42.0%	58.0%	37.2%	62.8%	30.2%	69.8%
	HyDE	**PathRAG**	HyDE	**PathRAG**	HyDE	**PathRAG**	HyDE	**PathRAG**	HyDE	**PathRAG**	HyDE	**PathRAG**
Comprehensiveness	38.4%	61.6%	34.8%	65.2%	33.2%	66.8%	42.8%	57.2%	37.2%	62.8%	30.2%	69.8%
Diversity	21.6%	78.4%	35.2%	64.8%	36.0%	64.0%	33.8%	66.2%	33.2%	66.8%	30.0%	70.0%
Logicality	30.2%	69.8%	38.4%	61.6%	45.2%	54.8%	45.6%	54.4%	35.6%	64.4%	35.2%	64.8%
Relevance	35.6%	64.4%	35.6%	64.4%	46.4%	53.6%	43.4%	56.6%	40.4%	59.6%	31.4%	68.6%
Coherence	42.0%	58.0%	40.4%	59.6%	42.4%	57.6%	45.6%	54.4%	40.0%	60.0%	35.6%	64.4%
	G-retriever	**PathRAG**	G-retriever	**PathRAG**	G-retriever	**PathRAG**	G-retriever	**PathRAG**	G-retriever	**PathRAG**	G-retriever	**PathRAG**
Comprehensiveness	33.8%	66.2%	41.2%	58.8%	43.6%	56.4%	27.4%	72.6%	35.2%	64.8%	44.2%	55.8%
Diversity	35.2%	64.8%	43.6%	56.4%	32.0%	68.0%	24.4%	75.6%	38.4%	61.6%	30.0%	70.0%
Logicality	34.4%	65.6%	42.0%	58.0%	40.0%	60.0%	30.2%	69.8%	40.2%	59.8%	44.8%	55.2%
Relevance	35.6%	64.4%	44.0%	56.0%	38.4%	61.6%	36.2%	63.8%	40.0%	60.0%	41.2%	58.8%
Coherence	38.0%	62.0%	46.6%	53.4%	35.2%	64.8%	34.4%	65.6%	37.2%	62.8%	43.6%	56.4%
	HippoRAG	**PathRAG**	HippoRAG	**PathRAG**	HippoRAG	**PathRAG**	HippoRAG	**PathRAG**	HippoRAG	**PathRAG**	HippoRAG	**PathRAG**
Comprehensiveness	34.4%	65.6%	43.6%	56.4%	46.0%	54.0%	35.8%	64.2%	43.0%	57.0%	30.0%	70.0%
Diversity	38.0%	62.0%	38.4%	61.6%	24.4%	75.6%	37.2%	62.8%	27.2%	72.8%	26.4%	73.6%
Logicality	34.4%	65.6%	45.6%	54.4%	41.8%	58.2%	40.2%	59.8%	47.2%	52.8%	33.0%	67.0%
Relevance	40.2%	59.8%	43.2%	56.8%	40.0%	60.0%	44.0%	56.0%	46.6%	53.4%	34.4%	65.6%
Coherence	41.2%	58.8%	44.4%	55.6%	43.6%	56.4%	45.4%	54.6%	42.6%	57.4%	31.8%	68.2%
	GraphRAG	**PathRAG**	GraphRAG	**PathRAG**	GraphRAG	**PathRAG**	GraphRAG	**PathRAG**	GraphRAG	**PathRAG**	GraphRAG	**PathRAG**
Comprehensiveness	33.8%	66.2%	41.0%	59.0%	39.6%	60.4%	41.2%	58.8%	42.0%	58.0%	37.0%	63.0%
Diversity	29.8%	70.2%	36.6%	63.4%	38.2%	61.8%	36.2%	63.8%	38.4%	61.6%	41.2%	58.8%
Logicality	41.6%	58.4%	43.6%	56.4%	34.4%	65.6%	42.0%	58.0%	42.0%	58.0%	40.0%	60.0%
Relevance	40.6%	59.4%	44.0%	56.0%	42.4%	57.6%	40.4%	59.6%	42.4%	57.6%	44.4%	55.6%
Coherence	38.2%	61.8%	40.8%	59.2%	43.6%	56.4%	41.6%	58.4%	41.6%	58.4%	43.6%	56.4%
	LightRAG	**PathRAG**	LightRAG	**PathRAG**	LightRAG	**PathRAG**	LightRAG	**PathRAG**	LightRAG	**PathRAG**	LightRAG	**PathRAG**
Comprehensiveness	36.6%	63.4%	44.0%	56.0%	42.6%	57.4%	40.4%	59.6%	44.0%	56.0%	46.4%	53.6%
Diversity	38.2%	61.8%	43.2%	56.8%	43.6%	56.4%	42.0%	58.0%	43.2%	56.8%	46.4%	53.6%
Logicality	37.2%	62.8%	41.6%	58.4%	45.2%	54.8%	43.6%	56.4%	44.8%	55.2%	44.8%	55.2%
Relevance	40.0%	60.0%	44.0%	56.0%	44.8%	55.2%	44.0%	56.0%	45.6%	54.4%	44.4%	55.6%
Coherence	38.8%	61.2%	44.4%	55.6%	44.4%	55.6%	38.4%	61.6%	44.4%	55.6%	46.4%	53.6%

Experiments

We conduct extensive experiments to answer the following research questions (**RQs**): **RQ1:** How effective is our proposed PathRAG compared to the state-of-the-art baselines? **RQ2:** Has each component of our framework played its role effectively? **RQ3:** How does the model perform with indexing graphs of varying sparsity levels? **RQ4:** How does our framework perform under different LLM backbones? **RQ5:** How much token cost does PathRAG require to achieve the performance of state-of-the art baseline?

Experimental Setup

Datasets. We follow the experimental settings of LightRAG (Guo et al. 2024), and additionally consider four datasets from UltraDomain (Qian et al. 2024), SQuALITY (Wang et al. 2022) and SummScreen (Chen et al. 2022) for a thorough evaluation. These datasets vary significantly in scale, with token counts ranging from $180,000$ to $5,000,000$. We tune hyperparameters on Agriculture and CS datasets, and then test on the other six datasets.

Baselines. We compare PathRAG with six state-of-the-art methods: NaiveRAG (Gao et al. 2023b), HyDE (Gao et al. 2023a), G-retriever (He et al. 2025), HippoRAG (Gutiérrez et al. 2024), GraphRAG (Edge et al. 2024), and LightRAG (Guo et al. 2024). These methods cover cutting-edge text-based, KG-based and graph-based RAG approaches.

Implementation Details. To ensure fairness in the experimental process, we uniformly use "GPT-4o-mini" as the base model for all methods, and adopt the "text-embedding-3-small" model for embedding. In addition, we construct the indexing graph following the method of GraphRAG (Edge et al. 2024), and the retrieved edges corresponding to the global keywords in LightRAG (Guo et al. 2024) are placed after the query. For components involving randomness, we average over ten trials. The maximum input token length for the LLMs is limited to $8,000$ to ensure fair handling of different forms of retrieved information. The hyperparameters of PathRAG are fixed as $N = 40$, $K = 15$, and $\alpha = 0.7$.

Evaluation Metrics. Due to the absence of ground truth answers, we follow the LLM-based evaluation procedures as GraphRAG and LightRAG. Specifically, we utilize "GPT-4o-mini" to evaluate the generated answers across multiple dimensions. The evaluation dimensions are based on those from GraphRAG and LightRAG, including Comprehensiveness and Diversity, while also incorporating three new dimensions from recent advances in LLM-based eval-

Table 2: Ablation study on the path retrieval algorithm of PathRAG.

	Legal		History		Biology		Mix		SQuALITY		SummScreen	
	Random	**Flow-based**	Random	**Flow-based**	Random	**Flow-based**	Random	**Flow-based**	Random	**Flow-based**	Random	**Flow-based**
Comprehensiveness	44.0%	56.0%	46.0%	54.0%	44.0%	56.0%	42.8%	57.2%	45.6%	54.4%	46.6%	53.4%
Diversity	45.2%	54.8%	31.4%	68.6%	29.8%	70.2%	46.0%	54.0%	44.4%	55.6%	42.8%	57.2%
Logicality	46.6%	53.4%	44.0%	56.0%	42.0%	58.0%	46.4%	53.6%	41.8%	58.2%	43.6%	56.4%
Relevance	44.8%	55.2%	46.0%	54.0%	45.8%	54.2%	45.6%	54.4%	45.6%	54.4%	44.8%	55.2%
Coherence	44.6%	55.4%	41.0%	59.0%	41.6%	58.4%	44.0%	56.0%	43.6%	56.4%	46.4%	53.6%
	Hop-first	**Flow-based**	Hop-first	**Flow-based**	Hop-first	**Flow-based**	Hop-first	**Flow-based**	Hop-first	**Flow-based**	Hop-first	**Flow-based**
Comprehensiveness	44.4%	55.6%	45.8%	54.2%	48.8%	51.2%	43.2%	56.8%	45.2%	54.8%	46.4%	53.6%
Diversity	36.0%	64.0%	49.6%	50.4%	46.0%	54.0%	47.6%	52.4%	44.0%	56.0%	45.4%	54.6%
Logicality	45.2%	54.8%	41.2%	58.8%	44.8%	55.2%	43.6%	56.4%	46.0%	54.0%	46.0%	54.0%
Relevance	43.6%	56.4%	46.0%	54.0%	37.4%	62.6%	41.4%	58.6%	44.8%	55.2%	46.8%	53.2%
Coherence	41.0%	59.0%	40.0%	60.0%	42.6%	57.4%	44.8%	55.2%	46.8%	53.2%	46.4%	53.6%

Table 3: Ablation study on the prompt format of PathRAG.

	Legal		History		Biology		Mix		SQuALITY		SummScreen	
	Flat	**Path-based**	Flat	**Path-based**	Flat	**Path-based**	Flat	**Path-based**	Flat	**Path-based**	Flat	**Path-based**
Comprehensiveness	40.0%	60.0%	48.8%	51.2%	45.6%	54.4%	49.6%	50.4%	47.2%	52.8%	46.8%	53.2%
Diversity	42.0%	58.0%	39.6%	60.4%	44.4%	55.6%	43.2%	56.8%	44.4%	55.6%	45.2%	54.8%
Logicality	37.2%	62.8%	45.6%	54.4%	48.0%	52.0%	42.0%	58.0%	47.6%	52.4%	43.2%	56.8%
Relevance	44.8%	55.2%	49.0%	51.0%	47.4%	52.6%	44.8%	55.2%	45.4%	54.6%	46.0%	54.0%
Coherence	39.2%	60.8%	45.6%	54.4%	44.6%	55.4%	42.4%	57.6%	48.0%	52.0%	46.8%	53.2%

uation (Chan et al. 2023), namely Logicality, Relevance, and Coherence. We compare the answers generated by each baseline and our method and conduct win-rate statistics. A higher win rate indicates a greater performance advantage over the other. Note that the presentation order of two answers will be alternated, and the average win rates will be reported. More experimental setup details are provided in Appendices A, B, C, and D.

Main Results (RQ1)

As shown in Table 1, **PathRAG consistently outperforms the baselines across all evaluation dimensions and datasets**. From the perspective of evaluation dimensions, compared to all baselines, PathRAG shows an average win rate of 62.52% in Comprehensiveness, 65.37% in Diversity, 60.68% in Logicality, 59.92% in Relevance, and 59.43% in Coherence on average. These advantages highlight the effectiveness of our proposed path-based retrieval, which contributes to better performance across multiple aspects of the generated responses. From a dataset-level perspective, PathRAG achieves notable average win rates of 64.66%, 58.94%, 60.44%, and 61.59% on the Legal, History, Biology, and Mix datasets, respectively. Furthermore, it demonstrates robust performance on the more challenging and unconventional SQuALITY and SummScreen benchmarks, with average win rates of 60.67% and 63.20%. These results collectively indicate that PathRAG offers superior multi-domain adaptability and consistently outperforms baseline models across diverse evaluation scenarios.

Considering the human-written summaries in the SQuALITY dataset, we further evaluate the alignment between generated answers and references using automated metrics such as BLEU, ROUGE, and METEOR. As shown in Table 4, PathRAG achieves superior performance across all metrics, with a 7.06% average improvement over the best baseline. In future work, we will also integrate human evaluation and other semantic-level assessment methods.

Table 4: Evaluation on the SQuALITY dataset using human-written summaries.

	BLEU-1	**BLEU-2**	**ROUGE-1-F1**	**ROUGE-2-F1**	**METEOR**
NaiveRAG	31.78%	12.31%	13.80%	3.51%	16.90%
HyDE	31.68%	11.84%	13.95%	3.50%	16.95%
G-retriever	32.42%	12.03%	14.02%	3.12%	17.50%
HippoRAG	32.12%	11.89%	14.18%	3.39%	17.61%
GraphRAG	32.98%	12.27%	14.23%	3.59%	17.52%
LightRAG	33.37%	12.42%	14.56%	3.30%	17.66%
PathRAG	35.41%	13.81%	15.35%	3.95%	18.53%

Ablation Study (RQ2)

We conduct ablation experiments to validate the design of PathRAG. A detailed introduction to the variants can be found in Appendix G.

Necessity of Path Ordering. We consider two different strategies to rank the retrieved paths in the prompt, namely random and hop-first. As shown in the Table 2, the average win rates of PathRAG compared to the random and hop-first variants are respectively 56.44% and 55.64%, indicating the necessity of path ordering in the prompts.

Necessity of Path-based Prompting. While retrieval is conducted using paths, the retrieved information in the prompts does not necessarily need to be organized in the same manner. To assess the necessity of path-based organization, we compare prompts structured by paths with those using a flat organization. As shown in Table 3, path-based prompts achieve an average win rate of 55.19%, outperforming the flat format. In PathRAG, node and edge information within a path is inherently interconnected, and separating them can result in information loss. Therefore, after path retrieval, prompts should remain structured to preserve contextual relationships and enhance answer quality.

Graph Sparsity Analysis (RQ3)

To assess the robustness of PathRAG under varying levels of graph sparsity, we conduct experiments on the Agriculture and CS datasets. We simulate different sparsity levels by randomly removing 10%, 20%, 30%, 40%, and 50% of the edges from the original indexing graphs. Subsequently, we perform pairwise comparisons among PathRAG, LightRAG, and NaiveRAG under each sparsity condition. The results of this evaluation are presented in Figure 3.

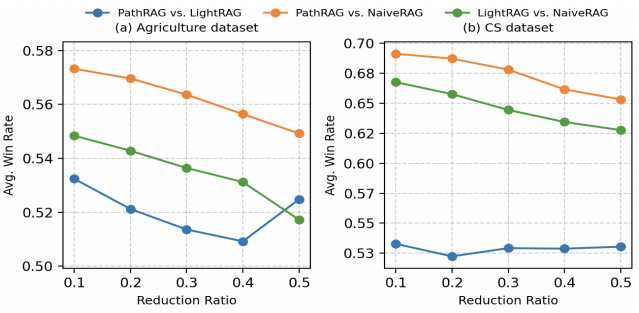

Figure 3: Performance of PathRAG, LightRAG, and NaiveRAG under different levels of graph sparsity on the Agriculture and CS datasets.

Although graph-based methods exhibit a degree of performance degradation when applied to increasingly sparse indexing graphs, PathRAG continues to demonstrate practical robustness. The simulation of sparsity through the random removal of a substantial proportion of edges does lead to a measurable decline in effectiveness. Nevertheless, PathRAG consistently outperforms both LightRAG, which similarly relies on graph structure and NaiveRAG, which operates independently of any graph-based context. Specifically, on the Agriculture and CS datasets, PathRAG achieves average win rates ranging from 54.84% to 57.32% and from 51.72% to 54.92%, respectively, when compared with NaiveRAG. Against LightRAG, PathRAG maintains win rates between 50.92% and 53.24% on Agriculture, and between 52.24% and 53.28% on CS. These findings underscore the robustness and generalizability of PathRAG, even under conditions characterized by severe graph sparsity.

Performance under Different LLMs (RQ4)

Considering that LLMs with different performance levels may affect the overall effectiveness of the framework and the reliability of evaluation, we uniformly replace "GPT-4o-mini" with either "GPT-4o" or "DeepSeek-V3" as the base and evaluation models for comparative experiments. As shown in Table 5, when "GPT-4o-mini" is used as both the base and evaluation model, PathRAG achieves an average win rate of 53.92% on the Agriculture and CS datasets. When the "DeepSeek-V3" model is used, the average win rate increases to 56.48%. With the highest-performing model, "GPT-4o", the win rate reaches a peak of 58.36%. These results indicate that the stronger the LLM used, the better the overall framework performance, and that

Table 5: Performance comparison of PathRAG and LightRAG across different base and evaluation models.

	Agriculture		CS	
GPT-4o-mini	LightRAG	**PathRAG**	LightRAG	**PathRAG**
Comprehensiveness	47.6%	52.4%	47.6%	52.4%
Diversity	44.4%	55.6%	42.6%	57.4%
Logicality	48.0%	52.0%	42.6%	57.4%
Relevance	46.6%	53.4%	45.2%	54.8%
Coherence	45.6%	54.4%	47.2%	52.8%
GPT-4o	LightRAG	**PathRAG**	LightRAG	**PathRAG**
Comprehensiveness	47.4%	52.6%	32.8%	67.2%
Diversity	43.2%	56.8%	34.0%	66.0%
Logicality	45.6%	54.4%	42.4%	57.6%
Relevance	42.0%	58.0%	41.6%	58.4%
Coherence	42.2%	57.8%	45.2%	54.8%
DeepSeek-V3	LightRAG	**PathRAG**	LightRAG	**PathRAG**
Comprehensiveness	47.4%	52.6%	46.0%	54.0%
Diversity	42.0%	58.0%	42.4%	57.6%
Logicality	44.0%	56.0%	42.0%	58.0%
Relevance	44.4%	55.6%	42.2%	57.8%
Coherence	43.2%	56.8%	41.6%	58.4%

PathRAG maintains stable performance across models with varying capabilities.

Token Cost Analysis (RQ5)

For a fair comparison focusing on token consumption, we also consider a lightweight version of PathRAG with $N = 20$ and $K = 5$, dubbed as PathRAG-lt. PathRAG-lt performs on par with LightRAG in overall performance, achieving an average win rate of 50.56%, with detailed results provided in the Appendix I.

Table 6: Comparison of LightRAG, PathRAG-lt and PathRAG in terms of token, time, and monetary cost.

	LightRAG	**PathRAG-lt**	**PathRAG**
token cost	16,728	9,968	14,438
monetary cost	2.51×10^{-3}\$	1.50×10^{-3}\$	2.17×10^{-3}\$

As shown in Table 6, PathRAG achieves significantly better performance while reducing token consumption by 13.69%, with a corresponding cost of only 0.002\$. Meanwhile, PathRAG-lt reduces token usage by 40.41% while maintaining similar performance to LightRAG. These results demonstrate the token efficiency of our method.

Conclusion

In this paper, we propose PathRAG, a novel graph-based RAG method that focuses on retrieving key relational paths from the indexing graph to alleviate noise. PathRAG can efficiently identify key paths with a flow-based pruning algorithm, and effectively generate answers with path-based LLM prompting. Experimental results demonstrate that PathRAG consistently outperforms baseline methods on six datasets. In future work, we will optimize the indexing graph construction process, and consider to collect more human-annotated datasets for graph-based RAG. It is also possible to explore other substructures besides paths to enhance model performance.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No.62192784, 62236003, 62576082), Young Elite Scientists Sponsorship Program (No.2023QNRC001) by CAST, and Beijing Natural Science Foundation(No.L253004).

References

Alon, U.; Xu, F.; He, J.; Sengupta, S.; Roth, D.; and Neubig, G. 2022. Neuro-symbolic language modeling with automaton-augmented retrieval. In *ICML 2023*.

Asai, A.; Hashimoto, K.; Hajishirzi, H.; Socher, R.; and Xiong, C. 2019. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *ICLR 2020*.

Bordes, A.; Usunier, N.; Chopra, S.; and Weston, J. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Cao, Y.; Han, S.; Gao, Z.; Ding, Z.; Xie, X.; and Zhou, S. K. 2025. Graphinsight: Unlocking insights in large language models for graph structure understanding. In *ACL 2025*.

Chan, C.-M.; Chen, W.; Su, Y.; Yu, J.; Xue, W.; Zhang, S.; Fu, J.; and Liu, Z. 2023. ChatEval: Towards Better LLM-based Evaluators through Multi-Agent Debate. In *ICLR 2024*.

Chen, L.; Tong, P.; Jin, Z.; Sun, Y.; Ye, J.; and Xiong, H. 2024. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. In *NeurIPS 2024*.

Chen, M.; Chu, Z.; Wiseman, S.; and Gimpel, K. 2022. SummScreen: A Dataset for Abstractive Screenplay Summarization. In *ACL 2022*.

Cheng, X.; Luo, D.; Chen, X.; Liu, L.; Zhao, D.; and Yan, R. 2023. Lift yourself up: Retrieval-augmented text generation with self-memory. In *NeurIPS 2023*.

Cuconasu, F.; Trappolini, G.; Siciliano, F.; Filice, S.; Campagnano, C.; Maarek, Y.; Tonellotto, N.; and Silvestri, F. 2024. The power of noise: Redefining retrieval for rag systems. In *SIGIR 2024*.

Dehghan, M.; Alomrani, M.; Bagga, S.; Alfonso-Hermelo, D.; Bibi, K.; Ghaddar, A.; Zhang, Y.; Li, X.; Hao, J.; Liu, Q.; et al. 2024. EWEK-QA: Enhanced Web and Efficient Knowledge Graph Retrieval for Citation-based Question Answering Systems. In *ACL 2024*.

Edge, D.; Trinh, H.; Cheng, N.; Bradley, J.; Chao, A.; Mody, A.; Truitt, S.; and Larson, J. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Fan, T.; Wang, J.; Ren, X.; and Huang, C. 2025. MiniRAG: Towards Extremely Simple Retrieval-Augmented Generation. *arXiv preprint arXiv:2501.06713*.

Fan, W.; Ding, Y.; Ning, L.; Wang, S.; Li, H.; Yin, D.; Chua, T.-S.; and Li, Q. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *KDD 2024*.

Fang, F.; Bai, Y.; Ni, S.; Yang, M.; Chen, X.; and Xu, R. 2024. Enhancing Noise Robustness of Retrieval-Augmented Language Models with Adaptive Adversarial Training. In *ACL 2024*.

Fang, J.; Meng, Z.; and Macdonald, C. 2024. Reano: Optimising retrieval-augmented reader models through knowledge graph generation. In *ACL 2024*.

Gao, H.; Wu, L.; Hu, P.; Wei, Z.; Xu, F.; and Long, B. 2022. Graph-augmented learning to rank for querying large-scale knowledge graph. In *AACL 2022*.

Gao, L.; Ma, X.; Lin, J.; and Callan, J. 2023a. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *ACL 2023*.

Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; and Wang, H. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Gu, Y.; Kase, S.; Vanni, M.; Sadler, B.; Liang, P.; Yan, X.; and Su, Y. 2021. Beyond IID: three levels of generalization for question answering on knowledge bases. In *WWW 2021*.

Guo, Z.; Xia, L.; Yu, Y.; Ao, T.; and Huang, C. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

Gutiérrez, B. J.; Shu, Y.; Gu, Y.; Yasunaga, M.; and Su, Y. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *NeurIPS 2024*.

Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; and Chang, M. 2020. Retrieval augmented language model pre-training. In *ICML 2020*.

He, X.; Tian, Y.; Sun, Y.; Chawla, N.; Laurent, T.; LeCun, Y.; Bresson, X.; and Hooi, B. 2025. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *NeurIPS 2024*.

Ho, X.; Nguyen, A.-K. D.; Sugawara, S.; and Aizawa, A. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *COLING 2020*.

Hofstätter, S.; Chen, J.; Raman, K.; and Zamani, H. 2023. Fid-light: Efficient and effective retrieval-augmented text generation. In *SIGIR 2023*.

Jiang, Z.; Xu, F. F.; Gao, L.; Sun, Z.; Liu, Q.; Dwivedi-Yu, J.; Yang, Y.; Callan, J.; and Neubig, G. 2023. Active retrieval augmented generation. In *EMNLP 2023*.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL 2017*.

Kwiatkowski, T.; Palomaki, J.; Redfield, O.; Collins, M.; Parikh, A.; Alberti, C.; Epstein, D.; Polosukhin, I.; Devlin, J.; Lee, K.; et al. 2019. Natural questions: a benchmark for question answering research. *TACL 2019*.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS 2020*.

Li, C.; Liu, Z.; Xiao, S.; Shao, Y.; and Lian, D. 2024. Llama2vec: Unsupervised adaptation of large language models for dense retrieval. In *ACL 2024*.

Li, M.; Miao, S.; and Li, P. 2024. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *ICLR 2025*.

Lin, Y.; Liu, Z.; Luan, H.; Sun, M.; Rao, S.; and Liu, S. 2015. Modeling Relation Paths for Representation Learning of Knowledge Bases. In *EMNLP 2015*.

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2024. Lost in the middle: How language models use long contexts. *TACL 2024*.

Liu, Y.; Wan, Y.; He, L.; Peng, H.; and Philip, S. Y. 2021. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. In *AAAI 2021*.

Lü, L.; and Zhou, T. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6): 1150–1170.

Lyu, Y.; Li, Z.; Niu, S.; Xiong, F.; Tang, B.; Wang, W.; Wu, H.; Liu, H.; Xu, T.; and Chen, E. 2025. Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models. *TOIS 2025*.

Mavromatis, C.; and Karypis, G. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

Panda, P.; Agarwal, A.; Devaguptapu, C.; Kaul, M.; et al. 2024. HOLMES: Hyper-Relational Knowledge Graphs for Multi-hop Question Answering using LLMs. In *ACL 2024*.

Procko, T. T.; and Ochoa, O. 2024. Graph retrieval-augmented generation for large language models: A survey. In *AIxSET 2024*. IEEE.

Qian, H.; Zhang, P.; Liu, Z.; Mao, K.; and Dou, Z. 2024. Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery. *arXiv preprint arXiv:2409.05591*.

Qin, Z.; Jagerman, R.; Hui, K.; Zhuang, H.; Wu, J.; Yan, L.; Shen, J.; Liu, T.; Liu, J.; Metzler, D.; et al. 2023. Large language models are effective text rankers with pairwise ranking prompting. In *NAACL 2024*.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language models can teach themselves to use tools. In *NeurIPS 2023*.

Su, Y.; Fang, Y.; Zhou, Y.; Xu, Q.; and Yang, C. 2025. CluE-RAG: Towards Accurate and Cost-Efficient Graph-based RAG via Multi-Partite Graph and Query-Driven Iterative Retrieval. *arXiv preprint arXiv:2507.08445*.

Sun, J.; Xu, C.; Tang, L.; Wang, S.; Lin, C.; Gong, Y.; Ni, L. M.; Shum, H.-Y.; and Guo, J. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *ICLR 2024*.

Talmor, A.; and Berant, J. 2018. The web as a knowledge-base for answering complex questions. In *NAACL 2018*.

Tian, L.; Zhou, X.; Wu, Y.-P.; Zhou, W.-T.; Zhang, J.-H.; and Zhang, T.-S. 2022. Knowledge graph and knowledge reasoning: A systematic review. *JEST 2022*.

Wang, A.; Pang, R. Y.; Chen, A.; Phang, J.; and Bowman, S. 2022. SQuALITY: Building a Long-Document Summarization Dataset the Hard Way. In *EMNLP 2022*.

Wen, Y.; Wang, Z.; and Sun, J. 2024. MindMap: Knowledge Graph Prompting Sparks Graph of Thoughts in Large Language Models. In *ACL 2024*.

Xu, S.; Pang, L.; Yu, M.; Meng, F.; Shen, H.; Cheng, X.; and Zhou, J. 2024. Unsupervised Information Refinement Training of Large Language Models for Retrieval-Augmented Generation. In *ACL 2024*.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP 2018*.

Yasunaga, M.; Ren, H.; Bosselut, A.; Liang, P.; and Leskovec, J. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *NAACL 2021*.

Yepes, A. J.; You, Y.; Milczek, J.; Laverde, S.; and Li, R. 2024. Financial report chunking for effective retrieval augmented generation. *arXiv preprint arXiv:2402.05131*.

Zhang, H.; Feng, T.; and You, J. 2025. Graph of records: Boosting retrieval augmented generation for long-context summarization with graphs. In *ACL 2025*.

Zhang, L.; Yu, Y.; Wang, K.; and Zhang, C. 2024. Arl2: Aligning retrievers for black-box large language models via self-guided adaptive relevance labeling. In *ACL 2024*.

Zhu, K.; Feng, X.; Du, X.; Gu, Y.; Yu, W.; Wang, H.; Chen, Q.; Chu, Z.; Chen, J.; and Qin, B. 2024. An Information Bottleneck Perspective for Effective Noise Filtering on Retrieval-Augmented Generation. In *ACL 2024*.