

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

Boyu Chen¹, Zirui Guo^{1,2}, Zidan Yang^{1,3}, Yuluo Chen¹, Junze Chen¹,
Zhenghao Liu³, Chuan Shi¹, Cheng Yang¹

¹ Beijing University of Posts and Telecommunications

² University of Hong Kong ³ Northeastern University
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. The code is available at the following link: <https://github.com/BUPT-GAMMA/PathRAG>

1 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., 2022b, 2023; Fan et al., 2024; Procko and Ochoa, 2024). Most RAG approaches divide the text database into chunks, organizing them in a flat structure to facilitate efficient and precise searches (Finardi et al., 2024; Yepes et al., 2024; Lyu et al., 2024).

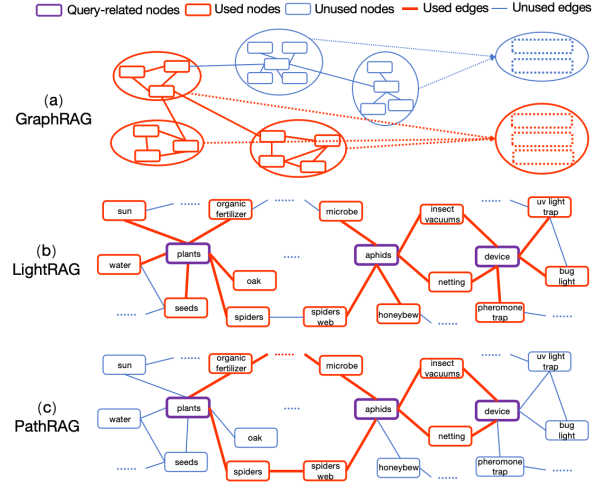


Figure 1: Comparison between different graph-based RAG methods. GraphRAG (Edge et al., 2024) uses all the information within certain communities, while LightRAG (Guo et al., 2024) uses all the immediate neighbors of query-related nodes. In contrast, our PathRAG focuses on key relational paths between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

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., 2022a) 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 com-

what is the difference between graph-rag and traditional rag

munities. 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.

However, we argue that the information considered in previous graph-based RAG methods is often redundant, which can introduce noise, degrade model performance, and increase token consumption. As shown in Figure 1 (a), GraphRAG method uses all the information from the nodes and edges within certain communities. Similarly, as shown in Figure 1 (b), LightRAG retrieves the immediate neighbors of query-related nodes to generate answers. The redundant information retrieved in these two methods may act as noise, and negatively impact the 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 logicity 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. As shown in Figure 1 (c), 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 evaluate the effectiveness of PathRAG, we follow the four benchmark datasets used in previous work (Qian et al., 2024), and additionally explore two larger ones. Experimental results on six datasets show that PathRAG generates better answers across all five evaluation dimensions compared to the state-of-the-art baselines. Compared to GraphRAG and LightRAG, the average win rates of PathRAG are 60.44% and 58.46%, respectively. The advantages of PathRAG are more significant for larger datasets,

making it better aligned with real-world applications. 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 consistently outperforms state-of-the-art baselines across six datasets and five evaluation dimensions. Extensive experiments further validate the design of PathRAG.

2 Related Work

Text-based RAG. To improve text quality (Fang et al., 2024a; 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., 2024) and **dense vector retrieval** (Lewis et al., 2020; Hofstätter et al., 2023; Li et al., 2024a; 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 (Ya-

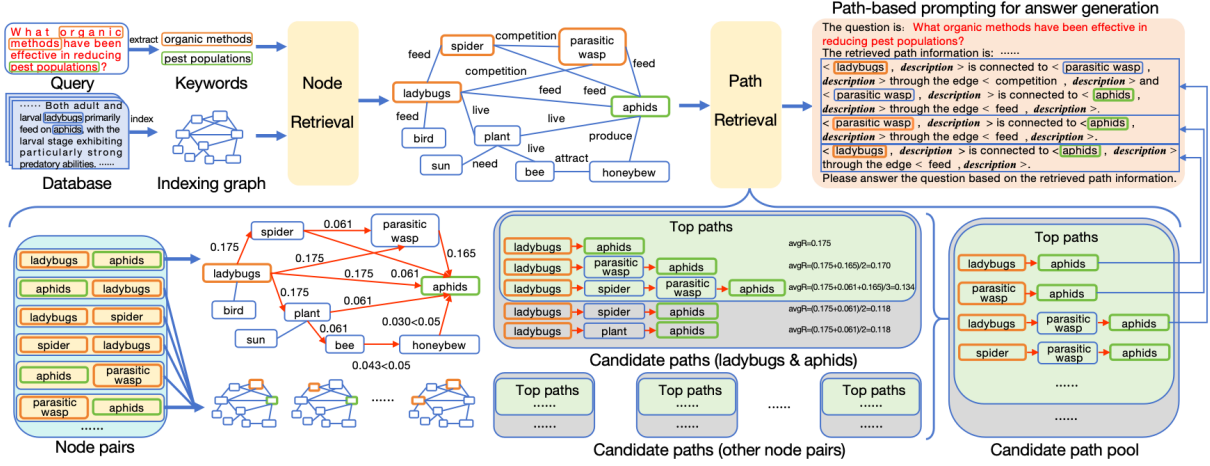


Figure 2: The overall framework of our proposed PathRAG with three main stages. 1) Node Retrieval Stage: Relevant nodes are retrieved from the indexing graph based on the keywords in the query; 2) Path Retrieval Stage: We design a flow-based pruning algorithm to extract key relational paths between each 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.

sunaga et al., 2021; Gao et al., 2022a; Li et al., 2024b; Procko and Ochoa, 2024; He et al., 2025). These methods can utilize existing KGs (Wen et al., 2023; Dehghan et al., 2024) or their optimized versions (Fang et al., 2024b; 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 questions 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 the retrieval process 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. We also notice a concurrent work MiniRAG (Fan et al., 2025) that leverages 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). Their implementation details such as path discovery and integration are also quite different from ours.

3 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:

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

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.

4 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.

4.1 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 $f: \mathcal{K}_q \cup \mathcal{K}_{\mathcal{V}} \rightarrow \mathcal{X}_q \cup \mathcal{X}_{\mathcal{V}}$, 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 $\mathcal{V}_q \subseteq \mathcal{V}$.

4.2 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 $\mathcal{N}(v_i, \cdot)$ and $\mathcal{N}(\cdot, v_i)$, 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:

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

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

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

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 $P = v_0 \xrightarrow{e_0} \dots v_i \xrightarrow{e_i} \dots = (\mathcal{V}_P, \mathcal{E}_P)$, 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 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:

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

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 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 .

4.3 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:

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

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., 2024; Firooz et al., 2024) 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:

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

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.

4.4 Discussion

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.

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.

5 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:** How do different values of key hyperparameters influence the method's performance? **RQ3:** Has each component of our framework played its role effectively? **RQ4:** How much token cost does PathRAG require to achieve the performance of other baselines? **RQ5:** Do the RAG response and its evaluation of PathRAG offer some interpretability?

5.1 Experimental Setup

5.1.1 Datasets

We follow the settings of LightRAG (Guo et al., 2024) and evaluate our model using the UltraDomain benchmark (Qian et al., 2024). The UltraDomain data is sourced from 428 college textbooks

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

	Agriculture		Legal		History		CS		Biology		Mix	
	NaiveRAG	PathRAG	NaiveRAG	PathRAG	NaiveRAG	PathRAG	NaiveRAG	PathRAG	NaiveRAG	PathRAG	NaiveRAG	PathRAG
Comprehensiveness	37.60%	62.40%	31.45%	68.55%	33.87%	66.13%	39.52%	60.48%	35.48%	64.52%	41.60%	58.40%
Diversity	32.26%	67.74%	24.39%	75.61%	36.29%	63.71%	42.40%	57.60%	41.13%	58.87%	33.06%	66.94%
Logicity	35.48%	64.52%	35.20%	64.80%	43.55%	56.45%	36.29%	63.71%	44.35%	55.65%	43.20%	56.80%
Relevance	40.80%	59.20%	26.61%	73.39%	42.40%	57.60%	37.39%	62.61%	34.67%	65.33%	41.94%	58.06%
Coherence	38.21%	61.79%	33.06%	66.94%	44.00%	56.00%	38.71%	61.29%	34.68%	65.32%	37.60%	62.40%
	HyDE	PathRAG	HyDE	PathRAG	HyDE	PathRAG	HyDE	PathRAG	HyDE	PathRAG	HyDE	PathRAG
Comprehensiveness	38.02%	61.98%	38.40%	61.60%	34.68%	65.32%	40.80%	59.20%	33.06%	66.94%	42.74%	57.26%
Diversity	36.29%	63.71%	21.60%	78.40%	34.68%	65.32%	39.52%	60.48%	36.00%	64.00%	33.87%	66.13%
Logicity	44.00%	56.00%	30.33%	69.67%	38.21%	61.79%	38.71%	61.29%	45.08%	54.92%	45.53%	54.47%
Relevance	39.34%	60.66%	35.48%	64.52%	35.77%	64.23%	37.39%	62.61%	46.34%	53.66%	43.55%	56.45%
Coherence	41.46%	58.54%	41.94%	58.06%	40.32%	59.68%	37.60%	62.40%	41.94%	58.06%	45.60%	54.40%
	GraphRAG	PathRAG	GraphRAG	PathRAG	GraphRAG	PathRAG	GraphRAG	PathRAG	GraphRAG	PathRAG	GraphRAG	PathRAG
Comprehensiveness	44.72%	55.28%	33.87%	66.13%	41.13%	58.87%	37.60%	62.40%	39.52%	60.48%	41.13%	58.87%
Diversity	45.97%	54.03%	29.84%	70.16%	36.59%	63.41%	42.74%	57.26%	38.21%	61.79%	36.29%	63.71%
Logicity	32.52%	67.48%	41.60%	58.40%	43.55%	56.45%	37.39%	62.61%	34.45%	65.55%	41.94%	58.06%
Relevance	43.09%	56.91%	40.65%	59.35%	43.55%	56.45%	34.68%	65.32%	42.28%	57.72%	40.32%	59.68%
Coherence	41.13%	58.87%	38.21%	61.79%	40.80%	59.20%	38.02%	61.98%	43.55%	56.45%	41.60%	58.40%
	LightRAG	PathRAG	LightRAG	PathRAG	LightRAG	PathRAG	LightRAG	PathRAG	LightRAG	PathRAG	LightRAG	PathRAG
Comprehensiveness	41.94%	58.06%	36.29%	63.71%	42.74%	57.26%	43.20%	56.80%	44.72%	55.28%	44.80%	55.20%
Diversity	41.46%	58.54%	36.49%	63.51%	43.90%	56.10%	45.16%	54.84%	43.09%	56.91%	42.74%	57.26%
Logicity	43.09%	56.91%	39.84%	60.16%	38.71%	61.29%	44.72%	55.28%	45.60%	54.40%	41.94%	58.06%
Relevance	39.20%	60.80%	37.81%	62.19%	41.13%	58.87%	41.46%	58.54%	42.28%	57.72%	40.65%	59.35%
Coherence	40.80%	59.20%	36.29%	63.71%	41.46%	58.54%	41.60%	58.40%	43.55%	56.45%	39.52%	60.48%

across 18 distinct domains. Besides the four domains used in LightRAG’s evaluation (Agriculture, Legal, Computer Science, and Mix), we extend two more domains (History and Biology), and consider six datasets in total. The token counts of the six datasets range from 600,000 to 5,000,000. We also follow the standardized process from GraphRAG and LightRAG for dataset preprocessing. Detailed information about the datasets can be found in the Appendix A.

5.1.2 Baselines

We compare PathRAG with four state-of-the-art methods: NaiveRAG (Gao et al., 2023), HyDE (Gao et al., 2022b), GraphRAG (Edge et al., 2024), and LightRAG (Guo et al., 2024). These methods cover cutting-edge text-based and graph-based RAG approaches. Detailed descriptions of the baselines can be found in the Appendix B.

5.1.3 Implementation Details

To ensure fairness and consistency across experiments, we uniformly use “GPT-4o-mini” for all LLM-related components across both the baseline methods and our approach. Also, the indexing graphs for different graph-based RAG methods are the same as GraphRAG (Edge et al., 2024). Retrieved edges that correspond to global keywords of LightRAG are placed after the query. For the key hyperparameters of PathRAG, the number of retrieval nodes N is selected from $\{10, 20, 30, 40, 50, 60\}$, the number of paths K is varied within $\{5, 10, 15, 20, 25\}$, the decay rate α is chosen from $\{0.6, 0.7, 0.8, 0.9, 1.0\}$, and the

threshold θ is fixed as 0.05.

5.1.4 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 evaluation (Chan et al., 2023), namely Logicity, 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. Detailed descriptions of these evaluation dimensions can be found in Appendix C.

5.2 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 60.88% in Comprehensiveness, 62.75% in Diversity, 59.78% in Logicity, 60.47% in Relevance, and 59.93% 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

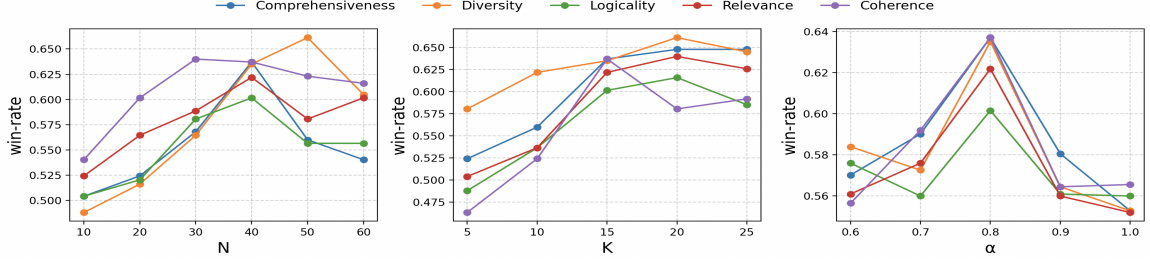


Figure 3: Impact of three hyperparameters in PathRAG on the Legal dataset.

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

	Agriculture		Legal		History		CS		Biology		Mix	
	Random	Flow-based	Random	Flow-based	Random	Flow-based	Random	Flow-based	Random	Flow-based	Random	Flow-based
Comprehensiveness	44.80%	<u>55.20%</u>	46.77%	<u>53.23%</u>	45.97%	<u>54.03%</u>	38.40%	<u>61.60%</u>	44.00%	<u>56.00%</u>	42.74%	<u>57.26%</u>
Diversity	38.40%	<u>61.60%</u>	49.19%	<u>50.81%</u>	31.45%	<u>68.55%</u>	37.70%	<u>62.30%</u>	29.84%	<u>70.16%</u>	47.58%	<u>52.42%</u>
Logicity	47.97%	<u>52.03%</u>	46.77%	<u>53.23%</u>	44.00%	<u>56.00%</u>	44.63%	<u>55.37%</u>	41.94%	<u>58.06%</u>	46.40%	<u>53.60%</u>
Relevance	45.45%	<u>54.55%</u>	44.80%	<u>55.20%</u>	45.97%	<u>54.03%</u>	41.46%	<u>58.54%</u>	45.83%	<u>54.17%</u>	48.39%	<u>51.61%</u>
Coherence	44.35%	<u>55.65%</u>	44.60%	<u>55.40%</u>	40.98%	<u>59.02%</u>	38.40%	<u>61.60%</u>	41.46%	<u>58.54%</u>	47.15%	<u>52.85%</u>
	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	48.78%	<u>51.22%</u>	44.35%	<u>55.65%</u>	45.83%	<u>54.17%</u>	47.15%	<u>52.85%</u>	48.80%	<u>51.20%</u>	43.20%	<u>56.80%</u>
Diversity	42.98%	<u>57.02%</u>	36.00%	<u>64.00%</u>	49.59%	<u>50.41%</u>	43.55%	<u>56.45%</u>	45.97%	<u>54.03%</u>	47.58%	<u>52.42%</u>
Logicity	47.58%	<u>52.42%</u>	45.16%	<u>54.84%</u>	41.13%	<u>58.87%</u>	40.80%	<u>59.20%</u>	44.80%	<u>55.20%</u>	43.44%	<u>56.56%</u>
Relevance	44.72%	<u>55.28%</u>	43.44%	<u>56.56%</u>	45.97%	<u>54.03%</u>	41.46%	<u>58.54%</u>	37.40%	<u>62.60%</u>	41.46%	<u>58.54%</u>
Coherence	39.34%	<u>60.66%</u>	41.13%	<u>58.87%</u>	39.84%	<u>60.16%</u>	48.80%	<u>51.20%</u>	42.74%	<u>57.26%</u>	44.72%	<u>55.28%</u>

generated responses. From the dataset perspective, PathRAG has a win rate of 60.13% in Agriculture, 60.26% in CS and 59.02% in Mix on average. For the larger three datasets, PathRAG shows greater advantages, with an average win rate of 65.53% in Legal, 60.13% in History and 59.50% in Biology. This indicates that our proposed PathRAG effectively reduces the impact of irrelevant information when handling larger datasets, making it more aligned with real-world applications and offering stronger practical significance compared to existing RAG baselines.

5.3 Hyperparameter Analysis (RQ2)

We adjust one hyperparameter at a time on the Legal dataset, and then calculate the win rates compared with LightRAG, the best baseline.

Number of retrieved nodes (N). As shown on the left side of Figure 3, we observe that as N increases, the average win rate gradually improves, peaking at $N = 40$, followed by a slight decline.

This is because the retrieved path information becomes increasingly sufficient as the number of nodes grows. However, as N continues to increase, the retrieved nodes are less relevant to the question and negatively impact the performance.

Number of retrieved paths (K). As shown in the middle of Figure 3,

we observe that as K increases, the average win rate reaches its peak at $K = 15$. When $K = 25$, the average win rate drops, meaning that additional

retrieved paths can not bring further improvement to the model. In practice, larger datasets prefer larger values of K .

Decay rate α . As shown on the right side of Figure 3, when $\alpha = 0.6$, the pruning algorithm prioritizes shorter paths, resulting in an average win rate of only 0.57. As α increases, the average win rate peaks at 0.63 when $\alpha = 0.8$, but then begins to decline. At $\alpha = 1.0$, where the decay rate is completely ignored, the average win rate significantly drops. This suggests that prioritizing shorter paths with a proper α serves as effective prior knowledge for the pruning process.

5.4 Ablation Study (RQ3)

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

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 56.75% and 56.08%, 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

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

	Agriculture		Legal		History		CS		Biology		Mix	
	Flat	Path-based	Flat	Path-based	Flat	Path-based	Flat	Path-based	Flat	Path-based	Flat	Path-based
Comprehensiveness	45.60%	54.40%	39.52%	60.48%	48.80%	51.20%	41.13%	58.87%	45.53%	54.47%	49.59%	50.41%
Diversity	44.72%	55.28%	41.94%	58.06%	39.52%	60.48%	40.80%	59.20%	44.35%	55.65%	43.09%	56.91%
Logicality	46.40%	53.60%	37.19%	62.81%	45.53%	54.47%	43.55%	56.45%	47.97%	52.03%	41.94%	58.06%
Relevance	39.52%	60.48%	44.72%	55.28%	48.39%	51.61%	44.35%	55.65%	47.58%	52.42%	44.80%	55.20%
Coherence	41.13%	58.87%	39.20%	60.80%	45.60%	54.40%	46.34%	53.66%	44.72%	55.28%	42.28%	57.72%

Question is: **What are the common challenges researchers face when measuring the impact of sustainable techniques in agriculture?**

LightRAG response

Researching sustainable agriculture faces challenges from practical, methodological, financial, and systemic constraints, affecting effectiveness and applicability. **1.Data Collection Difficulties**[...] **2.Complex Interactions**[...] **3. Resistance to Change**[...] **4.Economic Constraints**[...] **5.Policy and Regulatory Influences**[...] Conclusion: Measuring the impact of sustainable techniques is complex, requiring collaboration among scientists, policymakers, and farmers. Overcoming these challenges enables research that supports sustainable food production while balancing productivity and ecological health. ...

PathRAG response

Measuring the impact of sustainable agriculture is challenging due to complex systems, diverse sustainability goals, and the need for robust metrics. Below are key issues in this field. **1.Complexity of Agricultural Systems**[...] **2.Definition and Scope of Sustainability**[...] **3.Metrics and Methodologies** **4.Data Limitations**[...] **5.Temporal and Spatial Variability**[...] **6.Socio-Economic Factors**[...] Conclusion: Measuring the impact of sustainable agriculture involves challenges like complex systems and diverse sustainability goals. Effective research requires clear metrics, data collection, and socio-economic context. Overcoming these challenges can lead to better outcomes. This will ultimately support more sustainable agricultural practices.

LLM Decision

Comprehensiveness: Answer 2 provides a broader range of challenges, including metrics, definitions of sustainability, and temporal variability, while Answer 1 focuses more on specific examples without addressing the broader thematic context as thoroughly.

Diversity: Answer 2 covers a greater variety of perspectives related to the challenges of measuring sustainability, including socio-economic factors, the complexity of agricultural systems, and definitional issues. Answer 1 is less diverse in its discussion.

Logicality: Answer 2 presents a structured approach that logically connects different challenges within the context of agricultural systems. Answer 1, while coherent, feels slightly less organized in presenting its points.

Coherence: Answer 2 maintains a clearer thematic structure throughout its points, whereas Answer 1, while coherent, tends to shift focus slightly without a strong connective thread between sections.

Relevance: Answer 2 stays closely aligned with the overarching question of challenges in measuring impact, providing a comprehensive view of relevant factors. Answer 1, while relevant, is slightly narrower in focus.

Figure 4: Case study comparing the answers generated by PathRAG and the best baseline LightRAG.

organization. As shown in Table 3, path-based prompts achieve an average win rate of 56.14%, 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.

5.5 Token Cost Analysis (RQ4)

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 50.69%. The average token consumptions per question for LightRAG, PathRAG and PathRAG-lt are 15, 837, 13, 318 and 8, 869, respectively. Hence PathRAG reduces 16% token cost with much better performance, and the corresponding monetary cost is only 0.002\$. PathRAG-lt reduces 44% tokens while maintaining comparable performance to LightRAG. These results demonstrate the token efficiency of our method.

5.6 Case Study (RQ5)

To provide a more intuitive demonstration of the evaluation process, we present a case study from the Agriculture dataset. Given the same question, both LightRAG and PathRAG generate responses based on the retrieved text. The responses are then evaluated by GPT-4o-mini across five dimensions, with justifications provided, as shown in Figure 4. We highlight the key points in the answers in bold, with LLM justification for winning judgments displayed in blue and losing judgments in purple. The case study demonstrates that our proposed PathRAG provides comprehensive support for answer generation, with clear advantages in all five dimensions.

6 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 opti-

mize 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.

7 Limitations

This work focuses on how to retrieve relevant information from an indexing graph for answering questions. For a fair comparison with previous methods, the indexing graph construction process is not explored. Also, we prioritize simplicity in our proposed PathRAG, and thus the path retrieval algorithm involves no deep neural networks or parameter training, which may limit the performance. Besides, we follow the evaluation protocol of previous graph-based RAG methods, and the metrics are relative rather than absolute. We will consider to collect more datasets and design new metrics for graph-based RAG in future work.

References

- Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. 2022. Neuro-symbolic language modeling with automaton-augmented retrieval. In *International Conference on Machine Learning*, pages 468–485. PMLR.
- Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.
- Yukun Cao, Shuo Han, Zengyi Gao, Zezhong Ding, Xike Xie, and S Kevin Zhou. 2024. Graphinsight: Unlocking insights in large language models for graph structure understanding. *arXiv preprint arXiv:2409.03258*.
- Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*.
- Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2024. Lift yourself up: Retrieval-augmented text generation with self-memory. *Advances in Neural Information Processing Systems*, 36.
- Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729.
- Mohammad Dehghan, Mohammad Ali Alomrani, Sunyam Bagga, David Alfonso-Hermelo, Khalil Bibi, Abbas Ghaddar, Yingxue Zhang, Xiaoguang Li, Jianye Hao, Qun Liu, et al. 2024. Ewek-qa: Enhanced web and efficient knowledge graph retrieval for citation-based question answering systems. *arXiv preprint arXiv:2406.10393*.
- Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.
- Tianyu Fan, Jingyuan Wang, Xubin Ren, and Chao Huang. 2025. Minirag: Towards extremely simple retrieval-augmented generation. *arXiv preprint arXiv:2501.06713*.
- Wenqi Fan, Yajuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.
- Feiteng Fang, Yuelin Bai, Shiwen Ni, Min Yang, Xiaojun Chen, and Ruifeng Xu. 2024a. Enhancing noise robustness of retrieval-augmented language models with adaptive adversarial training. *arXiv preprint arXiv:2405.20978*.
- Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. 2024b. Reano: Optimising retrieval-augmented reader models through knowledge graph generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2094–2112.
- Paulo Finardi, Leonardo Avila, Rodrigo Castaldoni, Pedro Gengo, Celio Larcher, Marcos Piau, Pablo Costa, and Vinicius Caridá. 2024. The chronicles of rag: The retriever, the chunk and the generator. *arXiv preprint arXiv:2401.07883*.
- Hamed Firooz, Maziar Sanjabi, Wenlong Jiang, and Xiaoling Zhai. 2024. Lost-in-distance: Impact of contextual proximity on llm performance in graph tasks. *arXiv preprint arXiv:2410.01985*.
- Hanning Gao, Lingfei Wu, Po Hu, Zhihua Wei, Fangli Xu, and Bo Long. 2022a. Graph-augmented learning to rank for querying large-scale knowledge graph. *AACL 2022*.
- Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022b. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

- Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.
- Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.
- Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2025. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *COLING 2020*.
- Sebastian Hofstätter, Jiecao Chen, Karthik Raman, and Hamed Zamani. 2023. Fid-light: Efficient and effective retrieval-augmented text generation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1437–1447.
- Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *EMNLP 2023*.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.
- Chaofan Li, Zheng Liu, Shitao Xiao, Yingxia Shao, and Defu Lian. 2024a. Llama2vec: Unsupervised adaptation of large language models for dense retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3490–3500.
- Mufei Li, Siqi Miao, and Pan Li. 2024b. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *ICLR 2025*.
- Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paran-jape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Ye Liu, Yao Wan, Lifang He, Hao Peng, and S Yu Philip. 2021. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6418–6425.
- Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170.
- Yuanjie Lyu, Zhiyu Li, Simin Niu, Feiyu Xiong, Bo Tang, Wenjin Wang, Hao Wu, Huanyong Liu, Tong Xu, and Enhong Chen. 2024. Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models. *ACM Transactions on Information Systems*.
- Pranoy Panda, Ankush Agarwal, Chaitanya Devaguptapu, Manohar Kaul, et al. 2024. Holmes: Hyper-relational knowledge graphs for multi-hop question answering using llms. *ACL 2024*.
- Tyler Thomas Procko and Omar Ochoa. 2024. Graph retrieval-augmented generation for large language models: A survey. In *2024 Conference on AI, Science, Engineering, and Technology (AIxSET)*, pages 166–169. IEEE.
- Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. 2024. Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery. *arXiv preprint arXiv:2409.05591*.
- Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, et al. 2023. Large language models are effective text rankers with pairwise ranking prompting. *NAACL 2024*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.
- Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *NAACL 2018*.

- Ling Tian, Xue Zhou, Yan-Ping Wu, Wang-Tao Zhou, Jin-Hao Zhang, and Tian-Shu Zhang. 2022. Knowledge graph and knowledge reasoning: A systematic review. *Journal of Electronic Science and Technology*, 20(2):100159.
- Yilin Wen, Zifeng Wang, and Jimeng Sun. 2023. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv preprint arXiv:2308.09729*.
- Shicheng Xu, Liang Pang, Mo Yu, Fandong Meng, Huawei Shen, Xueqi Cheng, and Jie Zhou. 2024. Un-supervised information refinement training of large language models for retrieval-augmented generation. *ACL 2024*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *EMNLP 2018*.
- Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *NAACL 2021*.
- Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li. 2024. Financial report chunking for effective retrieval augmented generation. *arXiv preprint arXiv:2402.05131*.
- Lingxi Zhang, Yue Yu, Kuan Wang, and Chao Zhang. 2024. Arl2: Aligning retrievers for black-box large language models via self-guided adaptive relevance labeling. *ACL 2024*.
- Kun Zhu, Xiaocheng Feng, Xiyuan Du, Yuxuan Gu, Weijiang Yu, Haotian Wang, Qianglong Chen, Zheng Chu, Jingchang Chen, and Bing Qin. 2024. An information bottleneck perspective for effective noise filtering on retrieval-augmented generation. *ACL 2024*.

A Dataset Descriptions

We conduct experiments on the following six datasets, and the statistics of each dataset and corresponding indexing graph are shown in Table 4.

- **Agriculture dataset:** This dataset focuses on the agricultural domain, covering various aspects of agricultural practices, such as beekeeping, crop cultivation, and farm management.
- **Legal dataset:** This dataset focuses on the legal domain, covering various aspects of legal practices, such as case law, legal regulations, and judicial procedures.
- **History dataset:** This dataset focuses on the field of history, covering various periods, events, and figures throughout time. It includes historical texts, articles, and documents related to world history, significant historical movements, and important historical figures from different regions and cultures.
- **CS dataset:** This dataset focuses on the field of computer science, covering multiple subfields such as algorithms, data structures, artificial intelligence, machine learning, and computer networks. It particularly provides various practical application examples in the areas of machine learning and big data.
- **Biology dataset:** This dataset focuses on the field of biology, covering a wide range of topics such as plants, animals, insects, and more. It provides detailed information about the physical characteristics, behaviors, ecosystems, and other aspects of various organisms.
- **Mix dataset:** This dataset contains a variety of literary classics, including essays, poetry, and biographies, covering multiple fields such as philosophy, history, and literature.

B Baseline Descriptions

The detailed baseline descriptions are as follows:

- **NaiveRAG:** This method is mainly used for retrieving information from text databases by splitting the text into chunks for storage. During the storage process, the chunks are embedded using text embeddings. For a query, the question is converted into a text embedding, and retrieval is performed based on maximum similarity between the query embedding and the text chunks, enabling efficient and direct access to answers.
- **HyDE:** This model shares a similar storage framework with NaiveRAG. However, during the

Table 4: Dataset statistics.

Datasets	Agriculture	Legal	History	CS	Biology	Mix
Number of documents	12	94	26	10	27	61
Number of tokens	1,923,163	4,719,555	5,088,196	2,039,199	3,234,487	602,537
Number of nodes in the indexing graph	22,973	20,772	63,051	20,286	41,968	10,657

Table 5: Comparison between PathRAG-It and LightRAG in terms of win rates.

	Agriculture		Legal		History		CS		Biology		Mix	
	LightRAG	PathRAG-It	LightRAG	PathRAG-It	LightRAG	PathRAG-It	LightRAG	PathRAG-It	LightRAG	PathRAG-It	LightRAG	PathRAG-It
Comprehensiveness	<u>56.45%</u>	43.55%	47.58%	<u>52.42%</u>	<u>57.72%</u>	42.28%	<u>52.89%</u>	47.11%	49.60%	<u>50.40%</u>	41.46%	<u>58.54%</u>
Diversity	<u>52.00%</u>	48.00%	<u>56.10%</u>	43.90%	<u>54.03%</u>	45.97%	48.80%	<u>51.20%</u>	<u>52.89%</u>	47.11%	<u>52.42%</u>	47.58%
Logicity	45.16%	<u>54.84%</u>	43.09%	<u>56.91%</u>	48.80%	<u>51.20%</u>	45.60%	<u>54.40%</u>	48.78%	<u>51.22%</u>	41.94%	<u>58.06%</u>
Relevance	49.60%	<u>50.40%</u>	47.58%	<u>52.42%</u>	45.53%	<u>54.47%</u>	<u>52.89%</u>	47.11%	<u>53.66%</u>	46.34%	35.48%	<u>64.52%</u>
Coherence	<u>52.89%</u>	47.11%	47.15%	<u>52.85%</u>	<u>52.42%</u>	47.58%	<u>51.20%</u>	48.80%	<u>52.89%</u>	47.11%	42.74%	<u>57.26%</u>

query phase, it uses an LLM to generate a hypothetical document based on the question, which is then used to retrieve relevant text chunks and generate the final answer.

- **GraphRAG**: This is a graph-based RAG. It uses an LLM to extract entities and relationships from the text, representing them as nodes and edges, with descriptions from the original text attached as features to reduce information loss. For each question, a community detection algorithm is applied to summarize and generalize the information contained in the nodes from the bottom up, forming new community descriptions. Finally, the results of the community detection are used to answer global summarization questions.

- **LightRAG**: This is also a graph-based RAG, inheriting the graph construction method mentioned in GraphRAG. However, considering the high cost of retrieval in GraphRAG, LightRAG cleverly employs a dual-level retrieval framework, performing more detailed and precise searches in the graph at both local and global levels, significantly reducing token and time consumption.

C Evaluation Dimensions

LLM will evaluate RAG responses based on the following five dimensions:

- **Comprehensiveness**: How much detail does the answer provide to cover all aspects and details of the question?
- **Diversity**: How varied and rich is the answer in providing different perspectives and insights on the question?
- **Logicity**: How logically does the answer respond to all parts of the question?
- **Relevance**: How relevant is the answer to the question, staying focused and addressing the in-

tended topic or issue?

- **Coherence**: How well does the answer maintain internal logical connections between its parts, ensuring a smooth and consistent structure?

D Details of Ablated Variants

D.1 Path Ordering

- **Random ordering**. We randomly select K paths and place them into the prompt.

- **Hop-first ordering**. Paths are sorted based on the number of hops. Paths with fewer hops are considered to have more direct relevance. Within the same hop count, paths are randomly ordered. Finally, K paths are selected and arranged in ascending order, placing the most important paths at the end of the prompt to enhance memory retention.

D.2 Prompt Format

- **Flat organization**. In this setting, the retrieved paths are decomposed into individual nodes and edges. The order of nodes and edges is randomized and not structured based on their original paths.

E Detailed Comparison between PathRAG-It and LightRAG

Table 5 presents the win rates of PathRAG-It against LightRAG on six datasets. PathRAG-It has an overall win rate of 50.69%.

F Additional Case study

We also provide an additional case study comparing PathRAG and LightRAG on the CS dataset. Given the question, “What derived features should be considered to enhance the dataset’s predictive power? ”, both LightRAG and PathRAG generate responses based on the retrieved text. These responses are then evaluated by GPT-4o-mini across

Question is: **What derived features should be considered to enhance the dataset's predictive power?**

LightRAG response

Derived features, computed from existing attributes, are crucial for enhancing predictive power in machine learning by offering additional insights and improving model performance. Key derived features to consider include: **1.Polynomial Features[...]** **2.Interaction Features[...]** **3.Binning[...]** **4.Aggreated Features[...]** **5.Ratios and Differences[...]** **6.Date and Time Features[...]** **7.Text-derived Features[...]** **8.Normalization and Standardization[...]** **Conclusion:** Adding derived features can boost a model's predictive power. Evaluating their value based on domain knowledge and context improves performance and accuracy. ...

PathRAG response

Derived features, created from existing data, can greatly enhance model performance and improve predictive power in machine learning. Here are potential derived features to consider: **1.Feature Interactions[...]** **2.Aggreated Features[...]** **3.Temporal Features[...]** **4.Categorical Transformations[...]** **5.Normalization or Standardization[...]** **6.Statistical Features[...]** **7.Text Feature Extraction[...]** **8.Polynomial Features[...]** **9.Dimensionality Reduction Components[...]** **10. Domain-Specific Features[...]** **Conclusion:** Carefully selecting and engineering derived features improves both predictive capability and interpretability. Experimenting with feature combinations can identify those with significant predictive power, especially in complex models like decision trees and ensembles. Always validate feature engineering efforts with data and cross-validation to assess their impact on model performance. ...

LLM Decision

Comprehensiveness: Answer 2 provides a more extensive list of derived features, covering 10 distinct types and explaining their importance in enhancing predictive power. In contrast, Answer 1 lists only 8 features with less depth on some topics, making it less comprehensive.

Diversity: Answer 2 covers a broader spectrum of derived feature techniques, including complex methods like dimensionality reduction and domain-specific features, while Answer 1 primarily focuses on more common methods without exploring other advanced techniques.

Logicity: Answer 2 presents a more logical and structured approach, clearly associating derived features with their potential impacts on predictive power. The rationale for each feature is well-articulated, whereas Answer 1 has a less cohesive reasoning structure.

Coherence: Answer 2 maintains a clear and coherent structure, logically progressing through the different types of derived features. Each point builds upon the previous one, whereas Answer 1, while coherent, is less fluid in its transitions.

Relevance: Both answers are relevant, but Answer 2 is slightly more focused on the question by incorporating a wider range of applicable derived features that enhance predictive power, while Answer 1 sticks more closely to common techniques without exploring deeper.

Figure 5: Case study comparing the answers generated by PathRAG and the best baseline LightRAG on the CS dataset.

five dimensions, with justifications provided, as shown in Figure 5. We highlight the key points in the answers in bold, with LLM justification for winning judgments displayed in blue and losing judgments in purple. The case study demonstrates that our proposed path information retrieval method provides comprehensive support for answer generation. PathRAG exhibits clear advantages in all five dimensions.