HopRAG: Multi-Hop Reasoning for Logic-Aware Retrieval-Augmented Generation

Hao Liu[1†], **Zhengren Wang**[1,2†], **Xi Chen**[3], **Zhiyu Li**[2*],
Feiyu Xiong[2], **Qinhan Yu**[1], **Wentao Zhang**[1,4*]
[1]Peking University [2]Center for LLM, Institute for Advanced Algorithms Research, Shanghai
[3]Huazhong University of Science and Technology [4]Zhongguancun Academy
{liuhao_2002,wzr,yuqinhan}@stu.pku.edu.cn xichenai@hust.edu.cn
{lizy, xiongfy}@iaar.ac.cn wentao.zhang@pku.edu.cn

Abstract

Retrieval-Augmented Generation (RAG) systems often struggle with imperfect retrieval, as traditional retrievers focus on lexical or semantic similarity rather than logical relevance. To address this, we propose **HopRAG**, a novel RAG framework that augments retrieval with logical reasoning through graph-structured knowledge exploration. During indexing, HopRAG constructs a passage graph, with text chunks as vertices and logical connections established via LLM-generated pseudo-queries as edges. During retrieval, it employs a *retrieve-reason-prune* mechanism: starting with lexically or semantically similar passages, the system explores multi-hop neighbors guided by pseudo-queries and LLM reasoning to identify truly relevant ones. Experiments on multiple multi-hop benchmarks demonstrate that HopRAG's *retrieve-reason-prune* mechanism can expand the retrieval scope based on logical connections and improve final answer quality.

1 Introduction

> *"Everyone and everything is six or fewer steps away, by way of introduction, from any other person in the world."*
>
> — Six Degrees of Separation

Retrieval-augmented generation (RAG) has become the standard approach for large language models (LLMs) to tackle knowledge-intensive tasks (Guu et al., 2020a; Lewis et al., 2020a; Izacard et al., 2022; Min et al., 2023; Ram et al., 2023; Liang et al., 2025). Not only can it effectively address the inherent knowledge limitations and hallucination issues (Zhang et al., 2023), but it can also enable easy interpretability and provenance tracking (Akyurek et al., 2022). Especially, the efficacy of RAG hinges on its retrieval module for identifying relevant documents from a vast corpus.

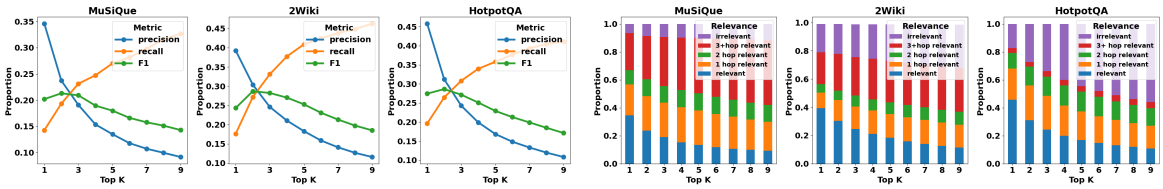
Currently, there are two mainstream types of retrievers: sparse retrievers (Jones, 1973; Robertson and Zaragoza, 2009b) and dense retrievers (Xiao et al., 2024; Wang et al., 2024b; Sturua et al., 2024; Wang et al., 2024c), which focus on lexical similarity and semantic similarity respectively, and are often combined for better retrieval performance (Sawarkar et al., 2024). Despite advancements, the ultimate goal of information retrieval extends beyond lexical and semantic similarity, striving instead for *logical relevance*. Due to the lack of logic-aware mechanism, the imperfect retrieval remains prominent (Wang et al., 2024a; Shao et al., 2024; Dai et al., 2024; Su et al., 2024a,b). For precision, the retrieval system may return lexically and semantically similar but indirectly relevant passages; regarding recall, it may fail to retrieve all the necessary passages for the user query.

Both cases eventually lead to inaccurate or incomplete LLM responses (Chen et al., 2024; Xiang et al., 2024; Zou et al., 2024), especially for multi-hop or multi-document QA tasks requiring multiple relevant passages for the final answer. In contrast, the reasoning capability of generative models is rapidly advancing, with notable examples such as OpenAI-o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025). Therefore, a natural research question arises: *"Is it possible to introduce reasoning capability into the retrieval module for more advanced RAG systems?"*

From a logical structure perspective, existing RAG systems can be mainly categorized into three types: **Non-structured RAG** simply adopts sparse or dense retrievers. The retrieval is only based on keyword matching or semantic vector similarity, but fails to capture the logical relations between user queries and passages. **Tree-structured RAG** (Sarthi et al., 2024; Chen et al., 2023; Fatehkia et al., 2024) focuses on the hierarchical logic of

† Equal contribution; * Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2025, pages 1897–1913



(a) Precision, recall and F1 score

(b) Proportions of passages on relevance

Figure 1: (a) Precision, recall and F1 score of BGE dense retrievers on MuSiQue, 2WikiMultiHopQA and HotpotQA with different top_k parameters, revealing the severe imperfect retrieval phenomenon. The highest recall reaches saturation at 0.45 in our settings. (b) We categorize retrieved passages into **relevant**, **indirectly relevant** and **irrelevant** according to the logical relevance to the query. The relevant passages are exactly the supporting facts, and indirectly relevant passages can hop to the supporting facts via HopRAG while irrelevant passages cannot. A large proportion of retrieved passages are indirectly relevant.

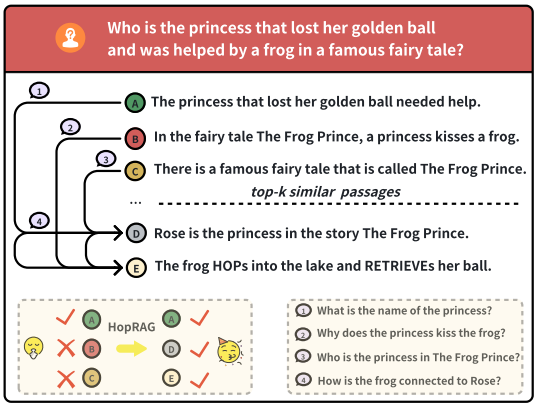


Figure 2: Demonstration of hopping between passages. For the user query, BGE dense retriever can only return one of the three supporting facts within top_k budget. However, lexically or semantically similar passages complement each other. Hopping between passages, by questions as pathways, improves the retrieval accuracy and completeness.

passages within a single document, but ignores relations beyond the hierarchical structure or across documents. Further, it introduces redundant information across different levels. **Graph-structured RAG** (Soman et al., 2024; Kang et al., 2023; Edge et al., 2024a; Guo et al., 2024) models logical relations in the most ideal form by constructing knowledge graphs (KGs) to represent documents, where entities are vertices and their relations are edges. However, the reliance on predefined schemas limits the flexible expressive capability (Li et al., 2024); constructing and updating knowledge graphs are challenging and prone to errors or omissions (Edge et al., 2024a); the triplet format of knowledge necessitates extra textualization or fine-tuning to improve LLMs' understanding (He et al., 2024).

Motivation As reported by (Wang et al., 2024a), even with advanced real-world search engines, roughly 70% retrieved passages do not directly contain true answers in their settings. We confirm the severity of imperfect retrieval in terms of both precision and recall, as illustrated in Figure 1(a). Inspired by the small-world theory (Kleinberg, 2000) or six degrees of separation (Guare, 2016), we propose that, *although lexically and semantically similar passages could be indirectly relevant or even distracting, they can serve as helpful starting points to reach truly relevant ones*. As shown in Figure 1(b), considering a graph composed of passages with logical relations as edges, a large proportion of retrieved passages fall within several hops of the ground truths.

Based on these observations, we propose **HopRAG**, an innovative graph-structured RAG system. At indexing phase, we construct a graph-structured knowledge index with passages as vertices and logic relations as directed edges. Specifically, the passages are connected by pseudo-queries generated by *query simulation* and *edge merging* operations. For example, as demonstrated in Figure 2, the pseudo-query "Why does the princess kiss the frog?" connects the raiser passage and the solver passage, as the pivot for logical hops. During retrieval, we employ reasoning-augmented graph traversal, following a three-step paradigm of retrieval, reasoning, and pruning. This process searches for truly relevant passages within the multi-hop neighborhood of indirectly relevant passages, guided by both the index structure and LLM reasoning.

Contributions Our contributions are as follows:

- We reveal the severe imperfect retrieval phe-

nomenon for multi-hop QA tasks. The results quantify that currently over 60% of retrieved passages are indirectly relevant or irrelevant. To turn "trash" into "treasure", we further employ indirectly relevant passages as stepping stones to reach truly relevant ones.

- We propose HopRAG, a novel RAG system with logic-aware retrieval mechanism. As lexically or semantically similar passages complement each other, HopRAG connects the raiser and solver passages with pseudo-queries. Beyond similarity-based retrieval, it reasons and prunes along the queries during retrieval. It also features flexible logical modeling, cross-document organization, efficient construction and updating.

- Extensive experiments confirm the effectiveness of HopRAG. The retrieve-reason-prune mechanism achieves over 36.25% higher answer metric and 20.97% higher retrieval F1 score compared to conventional information retrieval approaches. Several ablation studies provide more valuable insights.

2 Related Work

Retrieval-Augmented Generation Retrieval-augmented generation significantly improves large language models by incorporating a retrieval module that fetches relevant information from external knowledge sources (Févry et al., 2020; Guu et al., 2020b; Izacard and Grave, 2021; Zhao et al., 2024; Yu et al., 2025). Retrieval models have evolved from early sparse retrievers, such as TF-IDF (Jones, 1973) and BM25 (Robertson and Zaragoza, 2009b), which rely on word statistics and inverted indices, to dense retrievers (Lewis et al., 2020b) that utilize neural representations for semantic matching. Advanced methods, such as Self-RAG (Asai et al., 2023) and FLARE (Jiang et al., 2023) which determine the necessity and timing of retrieval, represent significant developments. However, the knowledge index remains logically unstructured, with each round of search considering only lexical or semantic similarity.

Tree&Graph-structured RAG Tree and graph are both effective structures for modeling logical relations. RAPTOR (Sarthi et al., 2023) recursively embeds, clusters, and summarizes passages, constructing a tree with differing levels of summarization from the bottom up. MemWalker (Chen et al., 2023) treats the LLM as an interactive agent walking on the tree of summarization. SiReRAG (Zhang et al., 2024) explicitly considers both similar and related information by constructing both similarity tree and relatedness tree. PG-RAG (Liang et al., 2024) prompts LLMs to organize document knowledge into mindmaps, and unifies them for multiple documents. GNN-RAG (Mavromatis and Karypis, 2024) reasons over dense KG subgraphs with learned GNNs to retrieve answer candidates. For query-focused summarization, GraphRAG (Edge et al., 2024b) builds a hierarchical graph index with knowledge graph construction and recursive summarization. Despite advancements, tree-structured RAG only focuses on the hierarchical logic within a single document; graph-structured RAG is costly, time-consuming, and returns triplets instead of plain text. In contrast, HopRAG offers a more lightweight and downstream task friendly alternative, with flexible logical modeling, cross-document organization, efficient construction and updating.

3 Method

In this section, we introduce our logic-aware RAG system, named HopRAG. An overview of this system is illustrated in Figure 3.

3.1 Problem Formulation

Given a passage corpus $P = \{p_1, p_2, ..., p_N\}$ and a query q which requires the information from multiple passages in P, the task is to design (1) a graph-structured RAG knowledge base that not only stores all the passages in corpus P but also models the similarity and logic between passages; (2) a corresponding retrieval strategy that can hop from indirectly relevant passages to truly relevant passages for better retrieval. Finally, with the query q and k passages as context $C = \{p_{i_1}, p_{i_2}, ..., p_{i_k}\}$, the LLM generates the response $\mathcal{O} \sim \mathcal{P}(\mathcal{O}|q, C)$.

3.2 Graph-Structured Index

We construct a graph-structured index $G = (\mathcal{V}, \mathcal{E})$ where the vertex set $\mathcal{V}$ consists of vertices storing all the passages and the directed edge set $\mathcal{E} = \{\langle v_i, e_{i,j}, v_j \rangle | v_i, v_j \in \mathcal{V}\} \subset \mathcal{V} \times \mathcal{V}$ is established based on the logical relations between passages for multi-hop reasoning. To establish G, we utilize **Query Simulation** to identify the logical relations and leverage textual similarity for efficient **Edge Merging**.

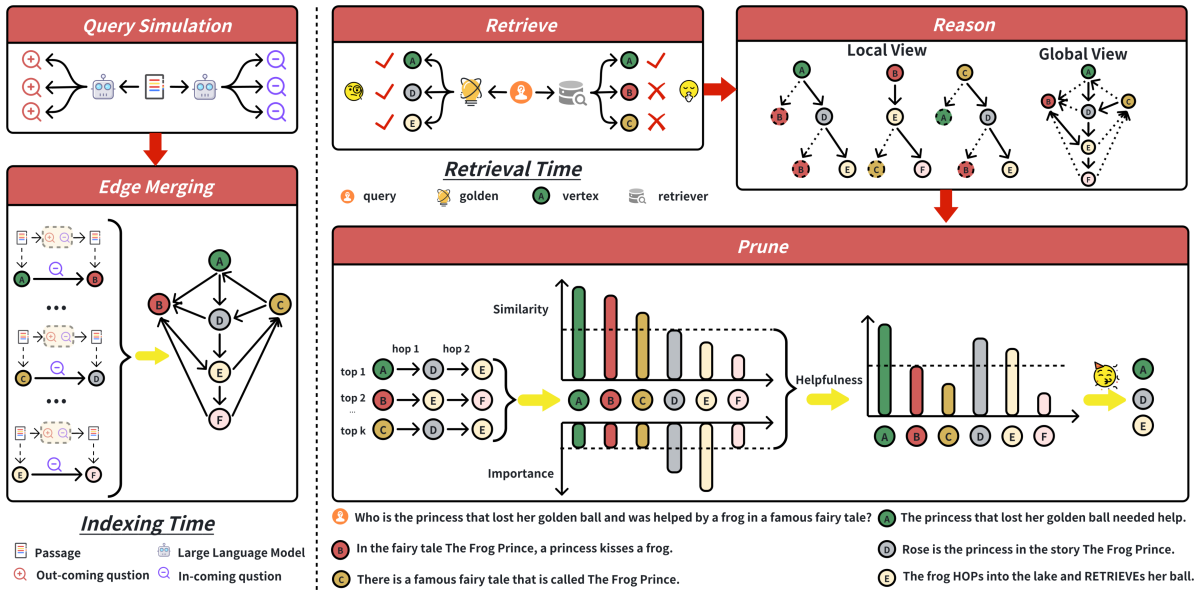


Figure 3: The workflow of HopRAG. **Left:** At indexing time, we first utilize *Query Simulation* to generate pseudo-queries for each passage and then apply *Edge Merging* to connect passages with directed logical edges. **Right:** At retrieval time, we employ a *Retrieve-Reason-Prune* pipeline. We first retrieve through purely similarity-based retrieval, then run reasoning-augmented graph traversal to explore the neighborhood, and finally prune the search by a novel metric *Helpfulness* considering both textual similarity and logical importance.

Query Simulation To identify the logical relations between passages, we generate a series of pseudo-queries for each passage, and use them to explore the passage's relations with the others and bridge the inherent gap between user-queries and passages (Wang et al., 2024c). Specifically, we adopt LLM to generate two groups of pseudo-queries for each passage p_i: (1) m out-coming questions $Q_i^+ = \bigcup_{1\le j\le m}\{q_{i,j}^+\}$ that originate from this passage but cannot be answered by itself; (2) n in-coming questions $Q_i^- = \bigcup_{1\le j\le n}\{q_{i,j}^-\}$ whose answers are within the passage. As demonstrated in Figure 2, for the toy passage "Rose is the princess in the story The Frog Prince", one in-coming question might be "What is the name of the princess?" and one out-coming question might be "How is the frog connected to Rose?". The prompts are in Appendix A.5.

We extract keywords from Q_i^+ and Q_i^- using named entity recognition NER$(\cdot)$ for sparse representation, and embed these questions into semantic vectors using an embedding model EMB$(\cdot)$ for dense representation. This results in sparse representations $K_i^+ = \bigcup_{1\le j\le m}\{k_{i,j}^+\}$ and $K_i^- = \bigcup_{1\le j\le n}\{k_{i,j}^-\}$, and dense representations $V_i^+ = \bigcup_{1\le j\le m}\{\mathbf{v}_{i,j}^+\}$ and $V_i^- = \bigcup_{1\le j\le n}\{\mathbf{v}_{i,j}^-\}$. We further define out-coming triplets as $r_{i,j}^+ := (q_{i,j}^+, k_{i,j}^+, \mathbf{v}_{i,j}^+)$ and in-coming triplets $r_{i,j}^- := (q_{i,j}^-, k_{i,j}^-, \mathbf{v}_{i,j}^-)$. Each passage p_i is stored inside a vertex v_i, featured with its out-coming triplets $R_i^+ = \bigcup_{1\le j\le m}\{r_{i,j}^+\}$ and in-coming triplets $R_i^- = \bigcup_{1\le j\le n}\{r_{i,j}^-\}$.

Edge Merging Given the out-coming and in-coming triplets, we match paired triplets via hybrid retrieval and establish directed edges between the corresponding passages. For each out-coming triplet $r_{s,i}^+$ of source vertex v_s, the most matching in-coming triplet r_{t^*,j^*}^- is determined as follows:

$$\mathrm{SIM}(r_{s,i}^+, r_{t,j}^-) = \frac{\frac{|k_{s,i}^+ \cap k_{t,j}^-|}{|k_{s,i}^+ \cup k_{t,j}^-|} + \frac{\mathbf{v}_{s,i}^+ \cdot \mathbf{v}_{t,j}^-}{||\mathbf{v}_{s,i}^+|| \cdot ||\mathbf{v}_{t,j}^-||}}{2}$$
$$r_{t^*,j^*}^- = \arg\max_{r_{t,j}^-} \mathrm{SIM}(r_{s,i}^+, r_{t,j}^-) \quad (1)$$

We then build the directed edge $\langle v_s, e_{s,t^*}, v_{t^*}\rangle$ with aggregated features, where $e_{s,t^*} := (q_{t^*,j^*}^-, k_{t^*,j^*}^- \cup k_{s,i}^+, \mathbf{v}_{t^*,j^*}^-)$.

3.3 Reasoning-Augmented Graph Traversal

For more accurate and complete responses, HopRAG's retrieval strategy leverages the reasoning ability of LLM to explore the neighborhood of probably indirectly relevant passages and hop to relevant ones based on the logical relations in the graph structure. As shown in Algorithm 1, by reasoning over the questions on out edges $e_{i,j}$ of a

current vertex v_i and then choosing to hop to the most promising vertex v_j, we realize reasoning-augmented graph traversal for better retrieval performance.

Retrieval Phase To start the local search over the graph for query q, we first use NER$(\cdot)$ and EMB$(\cdot)$ to get the keywords k_q and vector $\mathbf{v}_q$ of q, which will be used for hybrid retrieval to match top_k similar edges $\langle v_i, e_{i,j}, v_j \rangle$, following Equation 1. With each vertex v_j from these edges we initialize a context queue C_{queue} for breadth-first local search (Voudouris et al., 2010).

Reasoning Phase To fully exploit the logical relations over the graph and hop from indirectly relevant vertices to relevant ones, we introduce breadth-first local search which utilizes the LLM to choose the most appropriate neighbor for each v_j in C_{queue} to append to the tail of the queue. Specifically, for each v_j in C_{queue} in each round of hop, we leverage LLM to reason over all the questions from its out edges to choose one $e_{j,k}$ with the question which the LLM regards as the most helpful for answering q and append vertex v_k to C_{queue}. After hopping from each vertex in the current C_{queue} we can expand the context with at most top_k new vertices. From these new vertices we continue the next round of hop. Since different vertices may hop to the same vertex, we believe the vertices with more visits are more important for answering q, and use a counter C_{count} to track the number of visits for each vertex and measure its importance. By conducting n_{hop} rounds of hop, we realize reasoning-augmented graph traversal and expand the context length to at most $(n_{hop} + 1) \times top_k$.

Pruning Phase To avoid including too many intermediate vertices during the traversal, we introduce a novel metric Helpfulness $H(\cdot)$ that integrates similarity and logic to re-rank and then prune the traversal counter C_{count}. We calculate H_i following Equation 2 for each v_i in C_{count} and keep the top_k vertices with the highest H_i, where hybrid textual similarity SIM(v_i, q) calculates the average lexical and semantic similarity between the passage in v_i and query q following Equation 1; and IMP(v_i, C_{count}) is defined as the normalized number of visits of v_i in C_{count} during traversal following Equation 3. We prune C_{count} by retaining top_k vertices with the highest H value, resulting in the final context C.

$$H_i = \frac{\text{SIM}(v_i, q) + \text{IMP}(v_i, C_{count})}{2} \quad (2)$$

$$\text{IMP}(v_i, C_{count}) = \frac{C_{count}[v_i]}{\sum_{v_j \in C_{count}} C_{count}[v_j]} \quad (3)$$

Algorithm 1: Reasoning-Augmented Graph Traversal

Input: q, top_k, n_{hop}, G
Output: C

$\mathbf{v}_q \leftarrow$ EMB(q);
$k_q \leftarrow$ NER(q);
$C_{queue} \leftarrow$ Retrieve$(\mathbf{v}_q, k_q, G)$;
$C_{count} \leftarrow$Counter(C_{queue});
for $i \leftarrow 1, 2, ..., n_{hop}$ **do**
  for $j \leftarrow 1, 2, ..., \mid C_{queue} \mid$ **do**
    $v_j \leftarrow C_{queue}$.dequeue();
    $v_k \leftarrow$ Reason$(\{\langle v_j, e_{j,k}, v_k \rangle\})$;
    if v_k *not in* C_{count} **then**
      C_{queue}.enqueue(v_k);
      $C_{count}[v_k] \leftarrow 1$;
    else
      $C_{count}[v_k] + +$;
    end
  end
end
$C \leftarrow$Prune$(C_{count}, \mathbf{v}_q, k_q, top_k)$;
return C

4 Experiments

4.1 Experimental Setups

Datasets We collect several multi-hop QA datasets to evaluate the performance of HopRAG. We use HotpotQA dataset (Yang et al., 2018), 2WikiMultiHopQA dataset (Ho et al., 2020) and MuSiQue dataset (Trivedi et al., 2022). Following the same procedure as (Zhang et al., 2024), we obtain 1000 questions from each validation set of these three datasets. See Appendix A.2 for details.

Baselines We compare HopRAG with a variety of baselines: (1) unstructured RAG - sparse retriever BM25 (Robertson and Zaragoza, 2009a) (2) unstructured RAG - dense retriever BGE (Xiao et al., 2024; Karpukhin et al., 2020) (3) unstructured RAG - dense retriever BGE with query decomposition (Min et al., 2019) (4) unstructured RAG - dense retriever BGE with reranking

Method	MuSiQue		2Wiki		HotpotQA		Average	
	EM	F1	EM	F1	EM	F1	EM	F1
BM25	5.80	11.00	27.00	31.55	33.40	44.30	22.07	28.95
BGE	11.80	18.60	27.90	30.80	38.40	50.56	26.03	33.32
Query Decomposition	21.50	31.40	43.90	47.06	43.60	58.94	31.10	40.01
Reranking	24.50	34.53	46.70	50.89	47.70	62.95	34.67	43.60
HippoRAG	32.60	43.78	**66.40**	**74.01**	59.90	74.29	52.97	64.03
RAPTOR	35.30	47.47	54.90	61.20	58.10	72.48	49.43	60.38
SiReRAG	38.90	52.08	60.40	68.20	**62.50**	77.36	53.93	65.88
HopRAG	**39.10**	**53.00**	61.60	68.93	61.30	**78.34**	**54.00**	**66.76**

Table 1: We test our HopRAG against a series of baselines on multiple datasets using GPT-4o and GPT-3.5-turbo as the inference model with top 20 passages. We report the QA performance metrics EM and F1 score with GPT-3.5-turbo here and GPT-4o in Table 2, where the best score is in **bold** and the second best is underlined.

Method	MuSiQue		2Wiki		HotpotQA		Average	
	EM	F1	EM	F1	EM	F1	EM	F1
BM25	13.80	21.50	40.30	44.83	41.20	53.23	31.77	39.85
BGE	20.80	30.10	40.10	44.96	47.60	60.36	36.17	45.14
Query Decomposition	29.00	38.50	55.70	60.57	52.80	68.67	47.46	55.91
Reranking	32.00	40.29	53.70	58.44	55.40	70.03	48.61	56.25
GraphRAG	12.10	20.22	22.50	27.49	31.70	42.74	22.10	30.15
RAPTOR	36.40	49.09	53.80	61.45	58.00	73.08	49.40	61.21
SiReRAG	40.50	53.08	59.60	67.94	61.70	**76.48**	53.93	65.83
HopRAG	**42.20**	**54.90**	**61.10**	**68.26**	**62.00**	76.06	**55.10**	**66.40**

Table 2: We report the QA performance metrics EM and F1 score with GPT-4o and top 20 passages here, where the best score is in **bold** and the second best is underlined.

(Nogueira and Cho, 2020) (5) tree-structured RAG - RAPTOR (Sarthi et al., 2024) (6) tree-structured RAG - SiReRAG (Zhang et al., 2024) (7) graph-structured RAG - GraphRAG (Edge et al., 2024a) with the local search function (8) graph-structured RAG - HippoRAG (Gutiérrez et al., 2025). For structured RAG baselines, we follow the same setting as previous work (Zhang et al., 2024).

Metrics To measure the answer quality of different methods, we adopt exact match (EM) and F1 score which focus on the accuracy between a generated answer and the corresponding ground truth. We also use retrieval metrics to compare graph-based methods. Since tree-based methods like SiReRAG (Zhang et al., 2024) and RAPTOR (Sarthi et al., 2024) create new candidates (e.g., summary nodes) in the retrieval pool, it would be unfair to use retrieval metrics to compare them with others. We report both the answer and retrieval metrics in the ablations and discussion on HopRAG. See Appendix A.3 for more metric details.

Settings We use BGE embedding model for semantic vectors at 768 dimensions. To avoid the loss of semantic information caused by chunking at a fixed size, we adopt the same chunking methods utilized in the original datasets respectively. GPT-4o-mini serves as both the model generating in-coming and out-coming questions when constructing the graph index, and the reasoning model for graph traversal. We use two reader models GPT-4o and GPT-3.5-turbo to generate the response given the context with 20 retrieval candidates and $n_{hop} = 4$. See Appendix A.4 for more setting details.

4.2 Main Results

The main results are presented in Table 1 and 2. We observe that almost in all the settings HopRAG gives the best performance, with exceptions on HotpotQA when compared against SiReRAG and 2WikiMultiHopQA against HippoRAG. Overall, HopRAG achieves approximately 76.78% higher than dense retriever (BGE), 48.62% higher than query decomposition, 36.25% higher than rerank-

	MuSiQue			2Wiki			HotpotQA			Average		
	Answer		Retrieval	Answer		Retrieval	Answer		Retrieval	Answer		Retrieval
top_k	EM	F1	F1	EM	F1	F1	EM	F1	F1	EM	F1	F1
2	32.50	46.31	**37.83**	47.80	53.91	**36.77**	52.00	67.78	**50.23**	44.10	56.00	**41.61**
4	36.50	49.53	35.02	54.50	59.35	33.22	55.60	71.10	46.45	48.87	59.99	38.23
8	38.50	50.81	26.36	56.10	61.81	23.90	58.20	75.05	34.14	50.93	62.56	28.13
12	37.50	51.47	20.38	57.70	64.33	18.54	59.50	75.54	26.34	51.57	63.78	21.75
16	37.50	51.44	16.47	60.00	67.52	15.02	59.50	76.45	21.75	52.33	65.14	17.75
20	**39.10**	**53.00**	13.89	**61.60**	**68.93**	12.51	**61.30**	**78.34**	18.48	**54.00**	**66.76**	14.96

Table 3: We test the robustness w.r.t hyperparameter top_k on HopRAG using GPT-3.5-turbo on multiple datasets. We vary top_k from 2 to 20 and report both the answer and retrieval metrics, where the best score is in **bold** and the second best is underlined.

	MuSiQue		2Wiki		HotpotQA		Average	
n_{hop}	Retrieval F1	LLM Cost	Retrieval F1	LLM Cost	Retrieval F1	LLM Cost	Retrieval F1	LLM Cost
1	8.78	**20.00**	8.68	**19.86**	6.78	**19.91**	8.08	**19.92**
2	11.86	30.32	11.42	31.52	15.13	29.39	12.80	30.41
3	12.67	37.28	11.97	37.15	16.76	33.35	13.80	35.93
4	**13.89**	40.32	**12.51**	40.12	**18.48**	35.14	**14.96**	38.53

Table 4: We test the effect of hyperparameter n_{hop} on HopRAG using GPT-3.5-turbo on multiple datasets with top 20 passages. We vary n_{hop} from 1 to 4 and report both the answer and retrieval metrics in Table 8, and report the retrieval metrics here. For retrieval metrics, we calculate the retrieval F1 score and also the average number of calling LLM during traversal to measure the cost (the lower, the better). The best score is in **bold** and the second best is underlined.

ing (BGE), 9.94% higher than RAPTOR, 3.08% higher than HippoRAG, 1.11% higher than SiReRAG. This illustrates the strengths of HopRAG in capturing both textual similarity and logical relations for handling multi-hop QA.

Specifically, BM25, BGE and BGE with query decomposition yield unsatisfactory results since they rely solely on similarity, and BGE with reranking cannot capture logical relevance among candidates. Since GraphRAG considers relevance among entities instead of similarity for graph search, and RAPTOR focuses on the hierarchical logical relations among passages but cannot capture other kinds of relevance, both of them are more suitable for query-focused summarization but not the most competitive method for multi-hop QA tasks, as also reported in (Zhang et al., 2024).

In terms of HippoRAG, it prioritizes relevance signals such as vertices with the most edges and does not explicitly model similarity while our design HopRAG directly integrates similarity with logical relations when constructing edges. Although HopRAG only outperforms SiReRAG by a small margin in the scenario with top 20 candidate passages, our general graph structure does not introduce additional summary and proposition aggregate nodes and can facilitate efficient graph traversal for faster retrieval compared with SiReRAG. In the discussion, we will show that HopRAG can achieve competitive results with a smaller context length. Besides quantitative scores, we also demonstrate a case study in Appendix A.6 comparing HopRAG and GraphRAG.

4.3 Ablations and Discussion

To confirm the robustness of HopRAG and provide more insights, we vary top_k, n_{hop} and conduct ablation studies on traversal model.

Effects of top_k To show our efficiency in faster hop from indirectly relevant passages to truly relevant ones, we test the robustness by evaluating both the QA and retrieval performance on GPT-3.5-turbo with smaller top_k, as is shown in Table 3. From the results, we find that even with top 12 candidates, the QA performance of HopRAG is still comparable to that of HippoRAG or RAPTOR with 20 candidates, which highlights the effectiveness of our graph traversal design in efficiently retrieving more information within a limited context length. Meanwhile, we also observe that as top_k increases, the retrieval F1 score gradually decreases due to

Method (Traversal Model)	MuSiQue			2Wiki			HotpotQA			Average		
	Answer		Retrieval	Answer		Retrieval	Answer		Retrieval	Answer		Retrieval
	EM	F1	F1	EM	F1	F1	EM	F1	F1	EM	F1	F1
BM25	5.80	11.00	5.79	27.00	31.55	9.25	33.40	44.30	8.75	22.07	28.95	7.93
BGE	11.80	18.60	8.76	27.90	30.80	7.60	38.40	50.56	11.10	26.03	33.32	9.16
HopRAG (non-LLM)	19.00	27.68	8.27	42.20	46.72	8.09	46.90	61.17	11.73	36.04	45.19	9.36
HopRAG (Qwen2.5-1.5B-Instruct)	38.00	46.73	11.91	58.40	64.78	11.82	58.20	74.74	18.22	51.53	62.08	13.98
HopRAG (GPT-4o-mini)	**39.10**	**53.00**	**13.89**	**61.60**	**68.93**	**12.51**	**61.30**	**78.34**	**18.48**	**54.00**	**66.76**	**14.96**

Table 5: We conduct an ablation study on the reasoning model during traversal with GPT-3.5-turbo as the inference model and top 20 passages. We compare 5 scenarios including sparse retriever (BM25), dense retriever (BGE), HopRAG (non-LLM), HopRAG (Qwen2.5-1.5B-Instruct) and HopRAG (GPT-4o-mini) and report both the answer and the retrieval metrics, where the best score is in **bold** and the second best is underlined.

the inclusion of excessive redundant information. Conversely, the answer quality generally improves, attributed to GPT-3.5-turbo's strong capability in processing and reasoning over extended contexts, with only one exception in the MuSiQue dataset.

Effects of n_{hop} To assess the effects of the hyperparameter n_{hop} on reasoning-augmented graph traversal, we vary n_{hop} from 1 to 4 and evaluate the corresponding retrieval performance and cost, which is measured by the total number of LLM calls during graph traversal. The results shown in Table 4 indicate that as n_{hop} increases, retrieval performance tends to improve, as more vertices are visited during traversal for reasoning and pruning. However, the expense and latency from calling LLM also increase with n_{hop}, creating a trade-off between performance and cost. We notice that as n_{hop} increases, the number of new vertices in C_{queue} requiring LLM reasoning decays rapidly. Since different vertices may hop to the same important vertex, the actual queue length in each round of hop is less than top_k. Specifically, the average queue length is 2.60 in the fourth round and 1.23 in the fifth round, suggesting that for the three datasets, the local area in the graph structure can be largely explored within four rounds of hop, eliminating the need for an additional hop. We set n_{hop} as 4 in Table 1 and 2. We also evaluate the answer performances as n_{hop} varies and show the overall results in Appendix A.8.

Ablation on Traversal Model In order to generalize HopRAG to scenarios with less computational overhead during retrieval, we supplement results from (1) HopRAG with traversal model Qwen2.5-1.5B-Instruct (2) HopRAG with non-LLM graph traversal that replaces the reasoning phase in Algorithm 1 with similarity matching. Table 5 shows that even without using the reasoning ability of LLM in the graph traversal, HopRAG can achieve 45.84% higher than BM25 and 25.43% higher than dense retriever (BGE), which proves the effectiveness of HopRAG in capturing textual similarity and logical relations for logic-aware retrieval. The introduction of reasoning ability from LLM (GPT-4o-mini) can achieve about 45.78% higher average score than the non-LLM version, and Qwen2.5-1.5B-Instruct as traversal model produces comparable results with less cost and higher efficiency. We analyze the retrieval efficiency in Appendix A.7.

5 Conclusion

In this paper, we introduced HopRAG, a novel RAG system with a logic-aware retrieval mechanism. HopRAG connects related passages through pseudo-queries, which allows identifying truly relevant passages within multi-hop neighborhoods of indirectly relevant ones, significantly enhancing both the precision and recall of retrieval.

Extensive experiments on multi-hop QA benchmarks, i.e. MuSiQue, 2WikiMultiHopQA, and HotpotQA, demonstrate that HopRAG outperforms conventional RAG systems and state-of-the-art baselines. Specifically, HopRAG achieved over 36.25% higher answer accuracy and 20.97% improved retrieval F1 score compared to conventional information retrieval approaches. It highlights the effectiveness of integrating logical reasoning into the retrieval module. Moreover, ablation studies provide insights into the sensitivity of hyperparameters and models, revealing trade-offs between retrieval performance and computational costs.

HopRAG paves the way toward reasoning-driven knowledge retrieval. **Future work** involves scaling HopRAG to broader domains beyond QA tasks; optimizing indexing and traversal strategies for more complex scenarios with lower computation costs.

Acknowledgments

This work is supported by the National Key R&D Program of China (2024YFA1014003), National Natural Science Foundation of China (92470121, 62402016), CAAI-Ant Group Research Fund, and High-performance Computing Platform of Peking University.

Limitations

Despite the benefits of HopRAG, the current evaluation focuses on multi-hop or multi-document QA tasks. In order to mitigate the risk of performance fluctuations when applying HopRAG to other datasets, we should explore its generalization capabilities across a broader range of domains. Besides, more sophisticated query simulation and edge merging strategies may lead to further improvements. Finally, though we were inspired by the theories of six degrees of separation and small-world networks, the degree distribution of our passage graph vertices does not exhibit the power-law characteristic. On the one hand, these theories only serve as the motivation and intuitive analogy; on the other hand, exploring more appropriate degree distribution strategies is an interesting research topic. We leave these research problems for future work.

Ethics Impact

In our work, we acknowledge two key ethical considerations. First, we utilized AI assistant to enhance the writing process of our paper and code. We ensure that the AI assistant was used as a tool to improve clarity and conciseness, while the final content and ideas were developed and reviewed by human authors. Second, we employed multiple open source datasets and one open source tool Neo4j Community Edition (Webber, 2012) under GPL v3 license in our experiments. We are transparent about their origin and limitations, and we respect data ownership and user privacy.

References

Ekin Akyurek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. 2022. Towards tracing knowledge in language models back to the training data. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2429–2446, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Akari Asai, Zeqiu Wu, Yizhong Wang, et al. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arxiv:2310.11511*.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023. Walking Down the Memory Maze: Beyond Context Limit through Interactive Reading. *arXiv preprint*. ArXiv:2310.05029.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Unifying bias and unfairness in information retrieval: A survey of challenges and opportunities with large language models. *arXiv preprint arXiv:2404.11457*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024a. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *arXiv preprint*. ArXiv:2404.16130 [cs].

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

Masoomali Fatehkia, Ji Kim Lucas, and Sanjay Chawla. 2024. T-rag: Lessons from the llm trenches. *Preprint*, arXiv:2402.07483.

Thibault Févry, Livio Baldini Soares, et al. 2020. Entities as experts: Sparse memory access with entity supervision. In *EMNLP*.

John Guare. 2016. Six degrees of separation. In *The Contemporary Monologue: Men*, pages 89–93. Routledge.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. LightRAG: Simple and Fast Retrieval-Augmented Generation. *arXiv preprint*. ArXiv:2410.05779.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2025. Hipporag: Neurobiologically inspired long-term memory for large language models. *Preprint*, arXiv:2405.14831.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020a. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Kelvin Guu, Kenton Lee, Zora Tung, et al. 2020b. REALM: retrieval-augmented language model pre-training. *ICML*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering. *arXiv preprint*. ArXiv:2402.07630.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *Preprint*, arXiv:2011.01060.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, et al. 2023. Active retrieval augmented generation. *arXiv:2305.06983*.

Karen Sparck Jones. 1973. Index term weighting. *Information storage and retrieval*, 9(11):619–633.

Minki Kang, Jin Myung Kwak, Jinheon Baek, and Sung Ju Hwang. 2023. Knowledge graph-augmented language models for knowledge-grounded dialogue generation. *Preprint*, arXiv:2305.18846.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2004.04906.

Jon Kleinberg. 2000. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020a. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, et al. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*.

Zijian Li, Qingyan Guo, Jiawei Shao, Lei Song, Jiang Bian, Jun Zhang, and Rui Wang. 2024. Graph Neural Network Enhanced Retrieval for Question Answering of LLMs. *arXiv preprint*. ArXiv:2406.06572.

Xun Liang, Simin Niu, Zhiyu Li, Sensen Zhang, Hanyu Wang, Feiyu Xiong, Jason Zhaoxin Fan, Bo Tang, Shichao Song, Mengwei Wang, et al. 2025. Saferag: Benchmarking security in retrieval-augmented generation of large language model. *arXiv preprint arXiv:2501.18636*.

Xun Liang, Simin Niu, Sensen Zhang, Shichao Song, Hanyu Wang, Jiawei Yang, Feiyu Xiong, Bo Tang, Chenyang Xi, et al. 2024. Empowering large language models to set up a knowledge retrieval indexer via self-learning. *arXiv preprint arXiv:2405.16933*.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2023. Nonparametric masked language modeling. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2097–2118, Toronto, Canada. Association for Computational Linguistics.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy. Association for Computational Linguistics.

Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage re-ranking with bert. *Preprint*, arXiv:1901.04085.

Qwen Development Team. 2025. Qwen 2.5 Speed Benchmark. `https://qwen.readthedocs.io/en/stable/benchmark/speed_benchmark.html`. Accessed: 2025-5-4.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083*.

Stephen Robertson and Hugo Zaragoza. 2009a. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Stephen E. Robertson and Hugo Zaragoza. 2009b. The probabilistic relevance framework: BM25 and beyond. *FTIR*, 3(4):333–389.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. *arXiv preprint*. ArXiv:2401.18059 [cs].

Parth Sarthi, Salman Abdullah, Aditi Tuli, et al. 2023. Raptor: Recursive abstractive processing for tree-organized retrieval. In *ICLR*.

Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. 2024. Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. *arXiv preprint arXiv:2404.07220*.

Rulin Shao, Jacqueline He, Akari Asai, Weijia Shi, Tim Dettmers, Sewon Min, Luke Zettlemoyer, and Pang Wei Koh. 2024. Scaling retrieval-based language models with a trillion-token datastore. *arXiv preprint arXiv:2407.12854*.

Karthik Soman, Peter W Rose, John H Morris, Rabia E Akbas, Brett Smith, Braian Peetoom, Catalina Villouta-Reyes, Gabriel Cerono, Yongmei Shi, Angela Rizk-Jackson, Sharat Israni, Charlotte A Nelson, Sui Huang, and Sergio E Baranzini. 2024. Biomedical knowledge graph-optimized prompt generation for large language models. *Preprint*, arXiv:2311.17330.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, et al. 2024. jina-embeddings-v3: Multilingual embeddings with task lora. *arXiv preprint arXiv:2409.10173*.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S Siegel, Michael Tang, et al. 2024a. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *arXiv preprint arXiv:2407.12883*.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. 2024b. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *Preprint*, arXiv:2407.12883.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Preprint*, arXiv:2108.00573.

Christos Voudouris, Edward PK Tsang, and Abdullah Alsheddy. 2010. Guided local search. In *Handbook of metaheuristics*, pages 321–361. Springer.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö Arık. 2024a. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. *arXiv preprint arXiv:2410.07176*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024b. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.

Zhengren Wang, Qinhan Yu, Shida Wei, Zhiyu Li, Feiyu Xiong, Xiaoxing Wang, Simin Niu, Hao Liang, and Wentao Zhang. 2024c. QAEncoder: Towards Aligned Representation Learning in Question Answering System. *arXiv preprint*. ArXiv:2409.20434 [cs].

Jim Webber. 2012. A programmatic introduction to neo4j. In *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*, SPLASH '12, page 217–218, New York, NY, USA. Association for Computing Machinery.

Chong Xiang, Tong Wu, Zexuan Zhong, David Wagner, Danqi Chen, and Prateek Mittal. 2024. Certifiably robust rag against retrieval corruption. *arXiv preprint arXiv:2405.15556*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

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. *Preprint*, arXiv:1809.09600.

Qinhan Yu, Zhiyou Xiao, Binghui Li, Zhengren Wang, Chong Chen, and Wentao Zhang. 2025. Mramg-bench: A beyondtext benchmark for multimodal retrieval-augmented multimodal generation. *arXiv preprint arXiv:2502.04176*.

Nan Zhang, Prafulla Kumar Choubey, Alexander Fabbri, Gabriel Bernadett-Shapiro, Rui Zhang, Prasenjit Mitra, Caiming Xiong, and Chien-Sheng Wu. 2024. Sirerag: Indexing similar and related information for multihop reasoning. *Preprint*, arXiv:2412.06206.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *arXiv preprint*. ArXiv:2309.01219 [cs].

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

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2024. Poisonedrag: Knowledge poisoning attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867*.

A Appendix

Contents

A.1 Symbols

The symbols and their corresponding meanings are listed in Table 6.

A.2 Datasets

Table 7 shows the basic statistics of our datasets with their corresponding passage pool. Compared with knowledge graph, our graph-structured index is less dense and more efficient to construct. Since we put each passage text in the vertex, we can use fewer vertices to cover all the passages, which lowers the space complexity of the database. The average number of directed edges for each vertex is only 5.87, which lowers the time complexity for graph traversal.

A.3 Metrics

In our experiment, we mainly report the answer exact match (EM) and F1 score to compare all the methods.

Exact Match (EM) The Exact Match (EM) metric measures the percentage of predictions that match any one of the ground truth answers exactly. It is defined as:

$$EM = \frac{|\{p \mid p = g\}|}{|P|}$$

where p denotes a predicted answer, g denotes the corresponding ground truth answer, and P is the set of all the predictions.

F1 Score The F1 score is the harmonic mean of precision and recall, which measures the average overlap between the prediction and ground truth answer. Precision P and recall R are defined as:

$$\text{P} = \frac{|A \cap \hat{A}|}{|\hat{A}|}, \quad \text{R} = \frac{|A \cap \hat{A}|}{|A|}$$

where $|A \cap \hat{A}|$ refers to the number of matching tokens between the prediction $\hat{A}$ and the ground truth A, and $|\hat{A}|$, $|A|$ denote the number of tokens in the predicted and ground truth answers, respectively.

The F1 score is then computed as:

$$F1 = \frac{2 \cdot \text{P} \cdot \text{R}}{\text{P} + \text{R}}$$

In our ablation study, we also report the retrieval F1 score to test the sensitivity of HopRAG, which is calculated as follows.

The Precision (P) and Recall (R) for retrieval are computed as:

$$P = \frac{|\text{Ret} \cap \text{Rel}|}{|\text{Ret}|}, \quad R = \frac{|\text{Ret} \cap \text{Rel}|}{|\text{Rel}|}$$

where Ret represents the set of passages retrieved during retrieval, and Rel denotes the set of relevant passages that support the ground truth answer.

The Retrieval F1 score is then calculated as the harmonic mean of precision and recall:

$$F1_{\text{retrieval}} = \frac{2 \cdot P \cdot R}{P + R}$$

A.4 Settings

To avoid semantic loss by chunking the documents at a fixed size, we chunk each document in a way corresponding to the supporting facts of each dataset. Specifically, we chunk each document in HotpotQA and 2WikiMultiHopQA by sentence since the smallest unit of these two datasets' supporting facts is a sentence. To get embedding representation for each chunk, we use bge-base model. To extract keywords, we use the part-of-speech tagging function of Python package PaddleNLP to extract and filter entities. In our method, we use the Neo4j graph database to store vertices and build edges. When building edges we employ prompt engineering technique to instruct the LLM to generate an appropriate number of questions for each vertex to cover its information, with a minimum requirement of at least 2 in-coming questions and 4 out-coming questions. To prevent the graph structure from becoming overly complex and dense, we retain only $O(n \cdot \log(n))$ edges, where n is the number of vertices. We use GPT-4o-mini for reasoning-augmented graph traversal, GPT-4o and GPT-3.5-turbo for inference with 2048 max tokens and 0.1 temperature in our main experiments.

For sparse and dense retrievers, we use the Neo4j database to conduct retrieval on the vertices. With this setting, we align the retrieval engine for unstructured baselines with HopRAG to fairly demonstrate the effectiveness of our graph structure index.

symbol	meaning	symbol	meaning
P	passage corpus	Q_i^+	set of out-coming questions for p_i
p	passage	$q_{i,j}^+$	the j-th out-coming question for p_i
q	query	Q_i^-	set of in-coming questions for p_i
C	retrieval context	$q_{i,j}^-$	the j-th in-coming question for p_i
$\mathcal{P}(\cdot\|\cdot)$	LLM distribution	K_i^+	set of keywords for Q_i^+
$\mathcal{O}$	response for q	$k_{i,j}^+$	keywords for $q_{i,j}^+$
G	graph	K_i^-	set of keywords for Q_i^-
$\mathcal{V}$	set of vertices	$k_{i,j}^-$	keywords for $q_{i,j}^-$
v	vertex	V_i^+	set of embeddings for Q_i^+
$\mathcal{E}$	set of directed edges	$\mathbf{v}_{i,j}^+$	embedding for $q_{i,j}^+$
$e_{i,j}$	directed edge from v_i to v_j	V_i^-	set of embeddings for Q_i^-
k_q	keywords for q	$\mathbf{v}_{i,j}^-$	embedding for $q_{i,j}^-$
$\mathbf{v}_q$	embedding for q	R_i^+	set of out-coming triplets
C_{count}	counter of vertices during traversal	$r_{i,j}^+$	out-coming triplet for $q_{i,j}^+$
C_{queue}	queue of vertices during traversal	R_i^-	set of in-coming triplets
H	helpfulness metric	$r_{i,j}^-$	in-coming triplet for $q_{i,j}^-$
top_k	context budget	n_{hop}	number of hop

Table 6: Table of symbols and meanings.

dataset	number	docs	supporting facts	vertices	edges	avg text length	avg edge number
MuSiQue	1000	19990	2800	13086	81348	489.52	6.22
2Wiki	1000	10000	2388	23360	167068	116.07	7.15
HotpotQA	1000	9942	2458	40534	171946	132.44	4.24
Average	1000	13311	2549	25660	140121	246.01	5.87

Table 7: Dataset Statistics. We report the basic statistics of the graph structure on different datasets and demonstrate that our efficient graph structure is traversal-friendly.

For query decomposition, we use GPT-4o-mini to break down the query into multiple sub-queries, each of which should be a single-hop query. With m sub-queries we conduct dense retrieval with BGE for each sub-query to get top_k/m candidates independently and combine them all for final context. For reranking baseline, we use bge-reranker-base to rank $2*top_k$ candidates from dense retriever BGE and keep the top_k ones as the final context. The structured baseline methods rely on specific open-source projects according to their papers.

A.5 Prompts

The prompt used for generating in-coming questions is shown in Figure 4. The prompt used for generating out-coming questions is shown in Figure 5. The prompt for reasoning-augmented graph traversal is shown in Figure 8.

A.6 Case Study

We demonstrate the graph structure in Figure 7(a), one example edge with two vertices in Figure 6(a). Using the query "Donnie Smith who plays as a left back for New England Revolution belongs to what league featuring 22 teams?" as an example we conduct a qualitative analysis. For this multi-hop question (correct answer: Major League Soccer), the HotpotQA corpus contains three relevant sentences: (1) "Donald W. Donnie Smith (born December 7, 1990 in Detroit, Michigan) is an American soccer player who plays as a left back for New England Revolution in Major League Soccer."; (2) "Major League Soccer (MLS) is a men's professional soccer league, sanctioned by U.S. Soccer, that represents the sport's highest level in both the United States and Canada." and (3) "The league comprises 22 teams in the U.S. and 3 in Canada."

With dense retriever (BGE), we can easily retrieve the first sentence but the last two facts can't be retrieved in the context even with a top_k of 30. However, in our graph-structured index, these three vertices are logically connected, as is shown in Figure 6(b). During the traversal, LLM starts from the

Prompt for Generating In-Coming Quesitons

You are a journalist who is good at asking questions and proficient in both Chinese and English. Your task is to generate questions based on a few consecutive sentences from a news article or a biographical text. However, the answers to your questions should only come from these specific sentences, i.e., you should reverse-generate questions from a few sentences of the text. You will only have access to a few sentences, not the entire document. Focus on these consecutive sentences and ask relevant questions, ensuring that the answers come exclusively from these sentences.

Requirements:
1. Each question must include specific news elements (time, place, person) or other key characteristics to reduce ambiguity, clarify the context, and ensure self-containment.
2. You can try to omit or leave blanks in important parts of the sentence and form questions, but do not ask multiple questions about the same part of the sentence. You do not need to ask a question for every part of the sentence.
3. When asking about a part that has been omitted, the non-omitted information should be included in the question, as long as it does not affect the coherence of the question.
4. Different questions should focus on different aspects of the information in these sentences, ensuring diversity and representativeness.
5. All questions combined should cover all key points of the provided sentences, and the phrasing should be standardized.
6. Questions should be objective, fact-based, and detail-oriented. For example, ask about the time an event occurred, personal details of the subject, etc. Ensure that the answers to the questions come solely from these sentences.
7. If a part of the sentence has already been mentioned in a previous question, you should not ask about it again. That is, if the information from a sentence has already been covered in earlier questions, it should not be repeated. However, all information from the sentences must be covered by the questions, and if the sentences are long, the number of questions should increase to accommodate all information. There is no upper limit to the number of questions, but avoid repetition.

Example of Sentence List
["Their eighth studio album, \"(How to Live) As Ghosts\", is scheduled for release on October 27, 2017."]
Example of Answer
```json{{"Question List":["What's the name of their eighth album?","When was the album '(How to Live) As Ghosts' scheduled to be released?"]}}```

Sentence List
{sentences}

Your response must strictly follow the JSON format, avoiding unnecessary escapes, line breaks, or spaces. You should also pay extra attention to ensure that, except for the JSON and list formats themselves using double quotes ("), other instances of double quotes should be replaced with single quotes. For example, use '(How to Live) As Ghosts' instead of "(How to Live) As Ghosts".
```json{{"Question List":["<Question 1>","<Question 2>",.....]}}```

Figure 4: Prompt for generating in-coming questions.

semantically similar but indirectly relevant vertices and can reach all the supporting facts within only a maximum of 3 hops.

Using this query and its passages for demonstration, we provide the visualizations of HopRAG's graph index against GraphRAG to help readers grasp the differences and innovations. As shown in Figure 7 , the main differences between HopRAG and GraphRAG are listed as follows.

- **Vertices**. GraphRAG uses LLM to summarize the information from text chunks and then creates additional vertices (e.g., event, organization and person) in the graph, while HopRAG directly stores the original chunks in the vertices and thus avoids LLM hallucination during summarization, information loss during entity extraction and overly dense graph structure from redundant vertices.

- **Edges**. GraphRAG connects vertices with pre-defined relationships like "part of" or "related", while HopRAG flexibly stores incoming questions on the edges along with their keywords and embeddings, which can not only guide reasoning-augmented graph traversal but also facilitate edge retrieval.

- **Index**. GraphRAG creates and stores embeddings for the summarizations from LLM, while HopRAG creates sparse and dense indexes for both the vertices and the edges, which leads to more precise and efficient information retrieval.

In summary, the graph structure of HopRAG not only excavates logical relationships without creating additional vertices, but also paves the way for reasoning-driven knowledge retrieval.

A.7 Retrieval Efficiency

We supplement more comprehensive analysis of retrieval efficiency, along with optimization strategies for further speedup. The main latency in HopRAG's retrieval comes from the LLM inference time during retrieval-augmented graph traversal. Since HopRAG with locally deployed Qwen2.5-1.5B-Instruct as the traversal model also showcases competitive performances, we focus on the retrieval efficiency in this scenario. Following the hyper-

Prompt for Generating Out-Coming Quesitons

You are a journalist skilled in asking insightful questions and proficient in two languages. Your task is to generate follow-up questions based on a few consecutive sentences from a news article or biographical text. A follow-up question refers to a question whose answer is not found within the given sentences, but the answer may be inferred from the context before or after the given sentences, from related documents covering the same event, or from logical, causal, or temporal extensions of keywords within the given sentences.

You will only have access to a few sentences, not the entire document. After reading the consecutive sentences, generate related questions ensuring that the answer is not contained within these specific sentences. You can try to predict what the reader might ask next after reading these sentences, but the answers to your questions should be as concise as possible, so it is better to focus on objective questions.

Requirements:
1. Each question must include specific news elements (time, place, person) or other key features to reduce ambiguity and ensure self-containment.
2. Different follow-up questions should focus on diverse, objective aspects of the overall event represented by these sentences, ensuring variety and representativeness. Prioritize objective questions.
3. Based on the given sentences, generate questions about details that involve causal relationships, parallelism, sequencing, progression, connections, and other logical aspects. Possible areas to explore include, but are not limited to: the background of the event, information, reasons, impacts, significance, development trends, or perspectives of the individuals involved.
4. Questions should be objective, factual, and detail-oriented. For example, inquire about the time an event occurred, or ask for personal information about the subject. However, ensure that the answers to your questions are *not* contained in these specific sentences.
5. Aim to generate as many questions as possible without repetition, but ensure that the answers to the questions do not appear in these sentences. There is no upper limit to the number of questions, but please avoid duplicating questions.

Example of Sentence
"Their eighth studio album, \"(How to Live) As Ghosts\", is scheduled for release on October 27, 2017."
Example of Answer
```json{{" Question List ":["Whose eighth studio album is '(How to Live) As Ghosts'?","How did the album '(How to Live) As Ghosts' perform?","How long did it take to make the album '(How to Live) As Ghosts'?"]}}```

Sentences of News
{sentences}

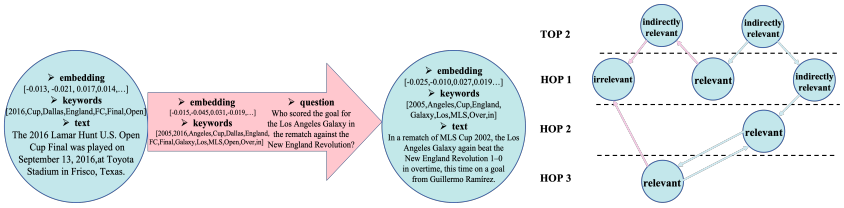
Your response must strictly follow the JSON format, avoiding unnecessary escapes, line breaks, or spaces. You should also pay extra attention to ensure that, except for the JSON and list formats themselves using double quotes ("), other instances of double quotes should be replaced with single quotes. For example, use '(How to Live) As Ghosts' instead of "(How to Live) As Ghosts".
```json{{"Question List":["<Question 1>","<Question 2>","<Question 3>",.....]}}```

Figure 5: Prompt for generating out-coming questions.

parameters $n_{hop} = 4$ and $top_k = 20$ from our main experiments, each question requires calling LLM 38.53 times, where each LLM call involves selecting one edge from an average of 5.87 edges, with an input of around 500 tokens and an output of 20 tokens. According to (Qwen Development Team, 2025), the output token speed for Qwen2.5-1.5B-Instruct is about 40.86 token/s using BF16 and Transformer. Therefore, the additional latency for each question from LLM inference will be $38.53 * 20/40.86 = 18.86$ seconds. However, there are many optimization strategies to improve the retrieval efficiency. Using vLLM and GPTQ-Int4 techniques, the additional latency for each question can be reduced to $38.53 * 20/174.04 = 4.43$ seconds. Moreover, parallelism techniques like multithreading can further reduce the total execution time for all the queries.

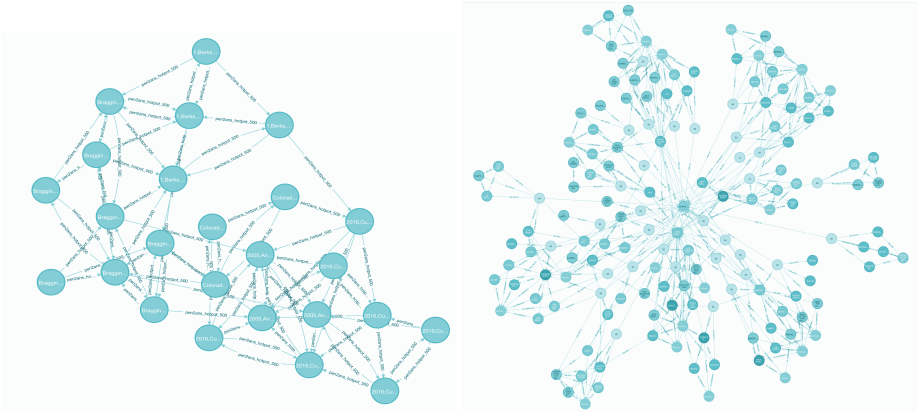
A.8 Discussion Results on n_{hop}

In the discussion we notice that as the hyper-parameter n_{hop} varies from 1 to 4, the answer and retrieval performance both increase, along with the retrieval cost of calling LLM during traversal. Since the average queue length in the fifth hop is only as small as 1.23, we believe 4 is the ideal n_{hop}. The overall results are shown in Table 8.



(a) Demonstration of one edge between two vertices. (b) Demonstration of reasoning-augmented graph traversal.

Figure 6: Demonstration of the traversal in the graph structure



(a) Demonstration of HopRAG's graph. (b) Demonstration of GraphRAG's graph.

Figure 7: Visualizations of HopRAG and GraphRAG

	MuSiQue				2Wiki				HotpotQA				Average			
	Answer		Retrieval		Answer		Retrieval		Answer		Retrieval		Answer		Retrieval	
$nhop$	EM	F1	F1	Cost	EM	F1	F1	Cost	EM	F1	F1	Cost	EM	F1	F1	Cost
1	21.50	30.77	8.78	**20.00**	48.60	52.44	8.68	**19.86**	47.90	62.92	6.78	**19.91**	39.33	48.71	8.08	**19.92**
2	32.00	43.75	11.86	30.32	54.90	60.37	11.42	31.52	55.90	71.26	15.13	29.39	47.60	58.46	12.80	30.41
3	32.50	44.50	12.67	37.28	52.90	59.16	11.97	37.15	57.40	73.86	16.76	33.35	47.60	59.17	13.80	35.93
4	**39.10**	**53.00**	**13.89**	40.32	**61.60**	**68.93**	**12.51**	40.12	**61.30**	**78.34**	**18.48**	35.14	**54.00**	**66.76**	**14.96**	38.53

Table 8: We test the effect of hyperparameter n_{hop} on HopRAG using GPT-3.5-turbo with top 20 passages. We vary n_{hop} from 1 to 4 and report both the answer and retrieval metrics. For answer metrics, we report the answer EM and F1 score; For retrieval metrics, we report the F1 score and average number of calling LLM during traversal to measure the cost (the lower, the better). The best score is in **bold** and the second best is underlined.

Prompt for Reasoning Augmented Graph Traversal

You are a question-answering robot. I will provide you with a main question that involves multiple pieces of information, as well as an additional auxiliary question. Your task is to answer the main question, but since the main question involves a lot of information that you may not know, you have the opportunity to use the auxiliary question to gather the information you need. However, the auxiliary question may not always be useful, so you need to assess the relationship between the auxiliary and the main question to determine whether or not to use it.

You need to assess whether the auxiliary question is completely irrelevant, indirectly relevant, or relevant and necessary for answering the main question. You can only return one of these three outcomes.

Please note that the main question will involve multiple background sentences, meaning that answering the main question requires the combination and reasoning of several pieces of information. However, you do not know which specific sentences are necessary to answer the main question. Your task is to assess whether the given auxiliary question is relevant and necessary, Indirectly relevant, or completely irrelevant in answering the main question.

Result 1: [Completely Irrelevant]. In this case, you determine that even without the information from the auxiliary question, you can still answer the main question, or the information in the auxiliary question is completely unrelated to the answer of the main question.
Result 2: [Indirectly relevant]. In this case, you find that the auxiliary question is related to the main question, but its answer is not part of the multiple pieces of information needed to answer the main question. The auxiliary question focuses on a related topic but does not provide critical information necessary for answering the main question.
Result 3: [Relevant and Necessary]. In this case, you find that the auxiliary question is a sub-question of the main question, meaning that without answering the auxiliary question, you will not be able to answer the main question. The information provided by the auxiliary question is necessary to answer the main question.

Example of Result 1:
Main Question: Donnie Smith who plays as a left back for New England Revolution belongs to what league featuring 22 teams?
Auxiliary Question: What is the purpose of the State of the Union address presented by the President of the United States?
In this case, after careful consideration, you find that the auxiliary question does not help answer the main question. The auxiliary question is completely unrelated to the main question. Your response should be:
```json{{" Decision ":"Completely Irrelevant"}}```

Example of Result 2:
Main Question: Donnie Smith who plays as a left back for New England Revolution belongs to what league featuring 22 teams?
Auxiliary Question: What is the significance of this league for second teams in the region?
In this case, you notice that both the main and auxiliary questions involve similar topics, but the auxiliary question focuses on the significance of the league, while the main question asks about the league to which a specific player belongs. Upon careful consideration, you find that the auxiliary question is related, but its answer does not provide any critical information to answer the main question. Your response should be: ```json{{" Decision ":" Indirectly relevant "}}```

Example of Result 3:
Main Question: Donnie Smith who plays as a left back for New England Revolution belongs to what league featuring 22 teams?
Auxiliary Question: Which team does Donald W. 'Donnie' Smith play for in Major League Soccer?
In this case, after careful consideration, you find that the auxiliary question is indeed related to the main question. The auxiliary question is a sub-question of the main question and provides necessary information to answer the main question. Without the answer to the auxiliary question, you will not be able to answer the main question. Your response should be:
```json{{" Decision ":" Relevant and Necessary "}}```

Now please strictly follow the JSON format in your response, avoiding unnecessary escapes, line breaks, or spaces. Additionally, please note that, except for the JSON and list formats, you should replace all double quotes with single quotes. For example, use '(How to Live) As Ghosts'
Main Question, Auxiliary Question as follows:
Main Question: {query}
Auxiliary Question: {question}

Figure 8: Prompt for reasoning-augmented graph traversal.