You Don't Need Pre-built Graphs for RAG: Retrieval Augmented Generation with Adaptive Reasoning Structures

Shengyuan Chen*, Chuang Zhou*, Zheng Yuan, Qinggang Zhang†, Zeyang Cui, Hao Chen, Yilin Xiao, Jiannong Cao, Xiao Huang

The Hong Kong Polytechnic University

{sheng-yuan.chen, qinggang.zhang, jiannong.cao, xiao.huang}@polyu.edu.hk,

{chuang-qgzj, yzheng.yuan, ze-yang.cui, yilin.xiao}@connect.polyu.hk, sundaychenhao@gmail.com

Abstract

Retrieval-Augmented Generation (RAG) is widely used to mitigate hallucinations of Large Language Models (LLMs) by leveraging external knowledge. Recent advances leverage pre-constructed graphs to capture the relational connections among distributed documents, showing remarkable performance in integrating fragmented information for complex tasks. However, existing Graph-based RAG (GraphRAG) methods rely on a costly process to transform the corpus into a graph, introducing overwhelming token cost and update latency. Moreover, real-world queries vary in type and complexity, requiring different logic structures for accurate reasoning. The pre-built graph may not align with these required structures, resulting in ineffective knowledge retrieval. To this end, we propose a **Logic**-aware **R**etrieval-**A**ugmented **G**eneration framework (**LogicRAG**) that dynamically extracts reasoning structures at inference time to guide adaptive retrieval without any pre-built graph. LogicRAG begins by decomposing the input query into a set of subproblems and constructing a directed acyclic graph to model the logical dependencies among them. To support coherent multi-step reasoning, LogicRAG then lin- earizes the graph using topological sort, so that subproblems can be addressed in a logically consistent order. Besides, LogicRAG applies graph pruning to reduce redundant retrieval and uses context pruning to filter irrelevant context, significantly reducing the overall token cost. Extensive experiments demonstrate that LogicRAG achieves both superior performance and efficiency compared to state-of-the-art baselines.

Code — https://github.com/chensyCN/LogicRAG

Extended version — https://arxiv.org/abs/2508.06105

Introduction

Large language models (LLMs), like Claude (Anthropic 2024) and ChatGPT (OpenAI 2023), have shown remarkable ability in a wide range of tasks like complex reasoning (Chen et al. 2024; Zhou et al. 2025a; Hong et al. 2024, 2025), question answering (Khashabi et al. 2020), and social analysis (Lu et al. 2021). However, these foundation models frequently struggle with domain-specific tasks, often producing hallucinations or inaccurate responses when handling knowledge-intensive queries (Huang et al. 2023). To mitigate these issues, retrieval-augmented generation (RAG) (Gao et al. 2023; Lewis et al. 2020; Zhang et al. 2025a) emerges as a promising framework that enhances LLMs by retrieving query-relevant contexts from external knowledge bases.

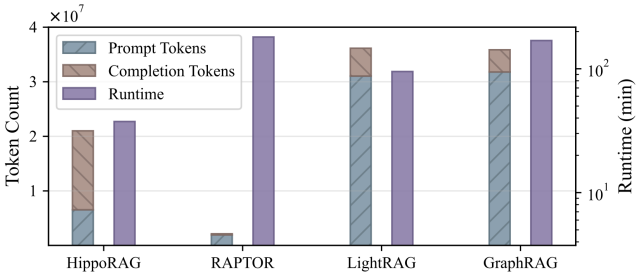

Figure 1: Token and runtime cost of the graph construction process of graph-based RAG methods on 2WikiMQA.

Real-world RAG systems often face significant challenges when handling large-scale, unstructured domain corpora (Peng et al. 2024; Zhang et al. 2025b). Documents sourced from research papers, textbooks, or technical reports vary widely in reliability and completeness (Guo et al. 2025; Zhong et al. 2024; Wu et al. 2025), and the retrieved information is often complex and disorganized, as domain knowledge is typically scattered across multiple sources without clear dependencies (Sun et al. 2024; Ma et al. 2024). To manage this complexity, RAG systems commonly segment documents into smaller chunks for indexing (Borgeaud et al. 2022; Izacard et al. 2023; Jiang et al. 2023). This approach, however, sacrifices critical contextual information, leading to reduced retrieval accuracy and limited capability for complex reasoning tasks, particularly those requiring multi-hop reasoning across interconnected concepts.

To address this, recent advances (Zhang et al. 2024; Xiang et al. 2025; Li et al. 2023; Zhou et al. 2025b) leverage pre-constructed graphs to capture the relational connections among distributed documents, showing remarkable performance in complex tasks. The typical graph-based RAG (GraphRAG) approaches like Microsoft GraphRAG (Edge et al. 2024) utilize hierarchical community-based search

*Equal contribution.

†Corresponding author.

and combine local and global querying to enhance response quality. Similarly HippoRAG (Gutiérrez et al. 2024) leverages PageRank-inspired algorithms to prioritize highly relevant nodes in memory-augmented retrieval for enhanced contextual coherence, RAPTOR (Sarthi et al. 2024) uses recursive abstractive processing for hierarchical text representation. LinearRAG (Zhuang et al. 2025) explores relation-free graph construction, eliminating LLM token costs and improve robust graphRAG with elegant bipartite graphs. These GraphRAG methods excel in handling multi-hop queries by providing structured contextual depth.

Despite recent advances, GraphRAG systems still face critical limitations in real-world scenarios. (i) Efficiency issues. Existing GraphRAG models rely on a costly process to transform the corpus into a graph, introducing overwhelming token cost and update latency as shown in Figure 1. It is hard to generalize to practical scenarios where knowledge bases are large-scale or dynamically evolving (Edge et al. 2024). (ii) Low quality of the pre-built graph. Existing methods leverage LLMs to automatically build the graph without any guidance, which may introduce irrelevant or redundant information, leading to inefficiencies in both retrieval and reasoning (Guo et al. 2024). (iii) Lack of flexibility. Real-world queries vary in type and complexity, requiring different logic structures for accurate reasoning (Peng et al. 2024). The pre-built graph may not align with these required structures, resulting in ineffective knowledge retrieval. These challenges highlight the need for a more adaptive and efficient approach.

To this end, we propose LogicRAG, which dynamically extracts reasoning structures at inference time to guide adaptive retrieval without any pre-constructed graph. Specifically, LogicRAG begins by decomposing the input query into a set of subproblems and constructing a directed acyclic graph to model the logical dependencies among them. This structured representation enables adaptive planning of the retrieval process by identifying which evidence chunks are logically connected to each subproblem. To support coherent multi-step reasoning, LogicRAG linearizes the graph using topological sort, so that subproblems can be addressed in a logically consistent order. To further improve efficiency without compromising performance, the model applies graph pruning to reduce redundant retrieval and uses context pruning to filter irrelevant context.

Our contributions are summarized as follows:

- We identify the limitations of pre-built graphs used in existing GraphRAG models, and propose LogicRAG, a novel framework that dynamically extracts reasoning structures at inference time to guide adaptive retrieval without any pre-built graph.
- We model query logical dependencies with a directed acyclic graph, offering a principled and universal modeling of reasoning structure.
- We enable efficient reasoning by *graph reasoning linearization* and *context and graph pruning*.
- Extensive experiments on benchmark datasets show that LogicRAG achieves both superior performance and efficiency compared to state-of-the-art baselines.

Problem Statement

Given an input query Q, retrieval-augmented generation aims to produce an answer A by retrieving relevant context $\mathcal{C}$ from an external knowledge corpus $\mathcal{K}$ and generating a response using a language model. Formally:

$$A = f_{\mathrm{RAG}}(Q, \mathcal{C}), \quad \mathcal{C} = \mathcal{R}(Q).$$

The key in a RAG system is the relevance of the retrieved context to the query. While direct semantic matching is often sufficient for simple fact retrieval tasks, it becomes inadequate for complex queries that require assembling multiple pieces of supporting knowledge. These pieces are often logically interconnected and must be retrieved and composed in a way that supports coherent reasoning. As such, the problem is to identify and retrieve a structured set of relevant contexts from $\mathcal{K}$ that collectively address the query's underlying information needs. This requires reasoning over both the content and the relationships among the retrieved evidence to guide the generation model toward producing accurate, complete, and logically consistent answers.

The Framework of LogicRAG

To handle complex queries in RAG, we propose a structured inference framework that decomposes complex queries into interdependent subproblems and resolves them via a logic-guided retrieval and generation process. The core of our method is the construction and utilization of a Query Logic Dependency Graph, a directed acyclic graph (DAG) that models the logical structure underlying the query. Each node in the DAG represents a subproblem, while edges encode the directional dependencies required for reasoning.

The framework operates in three sequential stages. First, the input query is decomposed into subproblems, and a DAG is constructed to capture their logical relationships. This graph is dynamically adapted during inference to reflect evolving retrieval needs. Second, the DAG is topologically sorted to produce a linear execution order that respects the dependencies among subproblems. Each subproblem is then resolved in a greedy, forward-pass manner, wherein retrieval is conditioned on the outputs of previously resolved subproblems. This process ensures context-aware retrieval and avoids recursive dependencies that hinder efficiency. Finally, to enhance scalability, we apply a two-dimensional pruning strategy that reduces context redundancy and merges semantically similar subproblems. Context pruning uses LLM-based summarization to maintain a rolling memory of relevant information, while graph pruning consolidates loosely coupled subproblems for unified resolution.

This logic-aware RAG pipeline transforms the traditionally flat retrieval paradigm into a dependency-sensitive inference mechanism. By aligning retrieval operations with the query's internal reasoning structure, the framework enables efficient, accurate, and scalable multi-step reasoning over complex information needs.

Query Logic Dependency Graph Construction

To model the interdependencies among sub-problems during inference, we construct a logic dependency graph that adaptively captures the structure of complex queries. This graph

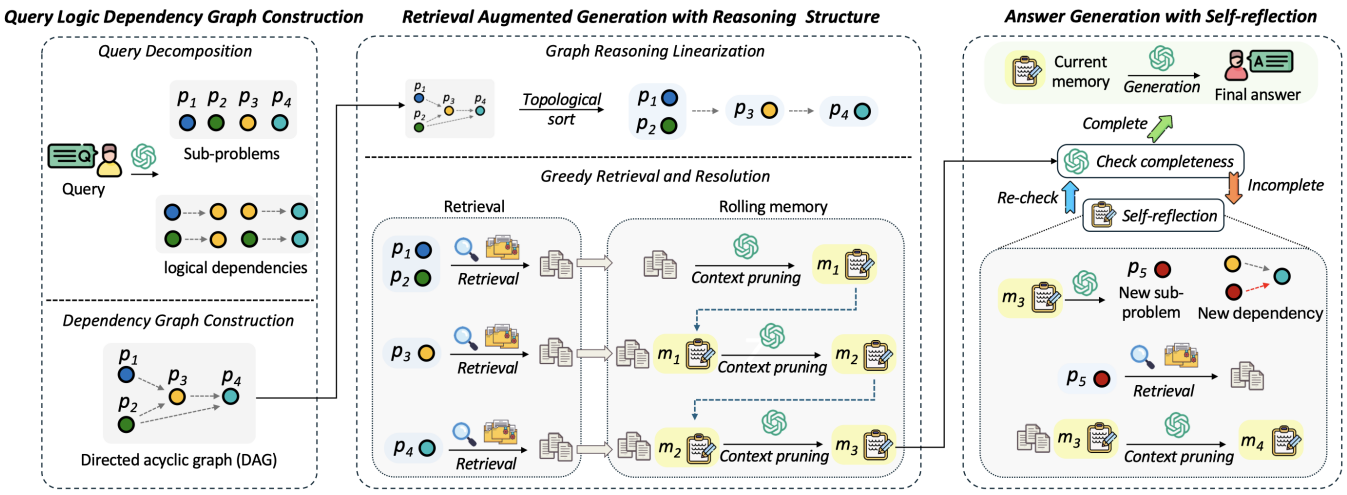

Figure 2: Illustration of the proposed LogicRAG.

is instantiated as a directed acyclic graph (DAG), which guides the retrieval process in a manner aligned with the query's reasoning structure. We discuss the details of the graph modeling and graph construction below:

Graph modeling. Let Q denote the input query, decomposed into a set of subproblems $P = \{p_1, p_2, \ldots, p_n\}$, where each p_i corresponds to a node v_i in a DAG $G = (V, E)$. The node set $V = \{v_1, v_2, \ldots, v_n\}$ represents the subproblems, and the directed edge set $E \subseteq V \times V$ encodes logical dependencies among them. The acyclic nature of G ensures a well-defined ordering for retrieval and reasoning.

Graph construction. The DAG construction process is guided by three key considerations: ❶ *Decomposition Accuracy*: The query Q is first segmented into subproblems via LLM-based reasoning, using few-shot prompting to ensure precise task separation. ❷ *Dependency Modeling*: Edges are inferred by the LLM based on logical precedence among subproblems, and the resulting graph is verified via topological sorting to enforce acyclicity. ❸ *Dynamic Adaptation*: If retrieval for a node yields insufficient context, the LLM dynamically augments the DAG by adding new subproblems and updating dependencies.

This DAG-based representation enables a principled representation of the query structures by explicit modeling of conditional dependencies across subproblems, supporting query decomposition and prioritization of retrieval. By aligning retrieval with the query's internal logic, the model is better equipped for complex reasoning.

Graph Reasoning Linearization

Although query logic dependency graphs provide a structured approach to modeling complex queries, integrating RAG with logical reasoning remains challenging due to their inherent operational asymmetry: RAG assumes that queries are independent and self-contained, whereas reasoning requires the sequential processing of interdependent subproblems, with intermediate results composed and passed along logical dependencies. A naïve integration of the two paradigms results in significant inefficiencies and semantic drift: each sub-query must be resolved in context, yet its formulation may depend on unresolved predecessors, creating a circular dependency in retrieval and generation.

Formally, let $f_{\text{RAG}}(p_i, \mathcal{C}_i)$ denote the retrieved answer for subproblem p_i, given context $\mathcal{C}_i$, and let $\mathcal{R}(p_i)$ be a retrieval function producing $\mathcal{C}_i$ from an external corpus $\mathcal{K}$. Then the complete query resolution requires computing:

$$A = \text{Compose}\left(\{f_{\text{RAG}}(p_i, \mathcal{R}(p_i))\}_{i=1}^{n}\right),$$

where *Compose* denotes the reasoning function that aggregates sub-results according to the logic encoded in the DAG G.

However, due to the dependencies in G, not all p_i can be processed in parallel. Specifically, the retrieval for p_i may depend on the answers to its parent nodes $\text{Pa}(v_i)$. This introduces a local directed constraint:

$$\mathcal{C}_i = \mathcal{R}\left(p_i \,\middle|\, \{f_{\text{RAG}}(p_j, \mathcal{C}_j)\}_{v_j \in \text{Pa}(v_i)}\right),$$

which prevents direct batching of retrievals. Without careful scheduling, this leads to recursive resolution that is inefficient and hard to optimize.

To address this, we exploit the acyclic nature of the dependency graph G and reduce the query resolution process to a two-step solution:

Step 1: Topological Sort. To efficiently schedule the resolution process, we perform a topological sort over the DAG G, using a depth-first search traversal. This yields an ordered sequence of subproblems $\langle p_{(1)}, p_{(2)}, \ldots, p_{(n)} \rangle$, in which every subproblem appears after all its dependencies. This linearization respects the logical flow of reasoning and allows sequential resolution of subproblems in a single forward pass. The DFS-based topological sort has $O(V + E)$ time and space complexity.

Step 2: Greedy Retrieval and Resolution. We then iterate over the sorted subproblems, resolving each $p_{(i)}$ greedily

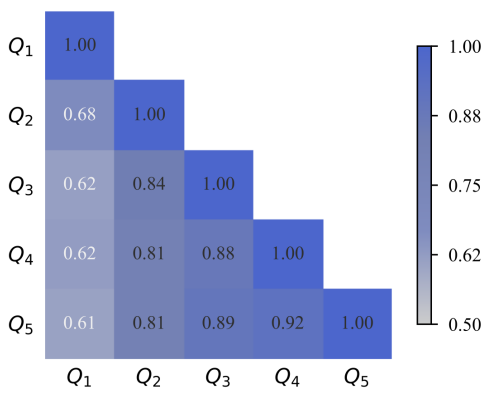

Figure 3: Word-level Jaccard similarity between subqueries across rounds in the agentic RAG process on 2WikiMQA.

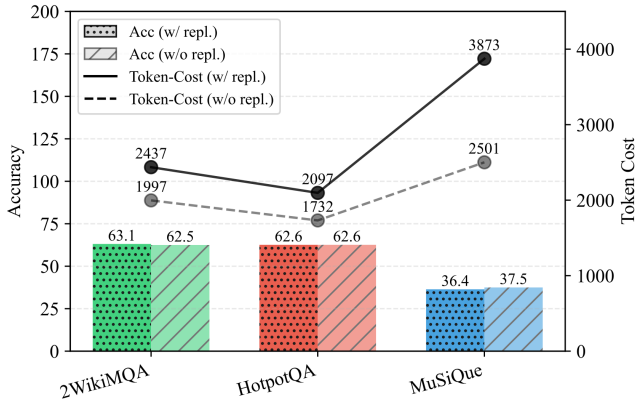

Figure 4: Comparison between sampling two strategies: w/ and w/o replacement.

using the current context:

$$\mathcal{C}_{(i)} = \mathcal{R}\left(p_{(i)} \,\middle|\, \{f_{\text{RAG}}(p_{(j)}, \mathcal{C}_{(j)})\}_{j<i,\, v_{(j)} \in \text{Pa}(v_{(i)})}\right), \quad (1)$$

$$a_{(i)} = f_{\text{RAG}}(p_{(i)}, \mathcal{C}_{(i)}). \quad (2)$$

This greedy approach eliminates recursion while maintaining logical consistency. Each context $\mathcal{C}_{(i)}$ can be dynamically adapted based on previously retrieved and inferred sub-answers, enabling *context-aware retrieval* and *incremental reasoning* with linear complexity.

Overall, this DAG-guided reasoning framework transforms the complex interplay between retrieval and logic into a tractable pipeline. The explicit decomposition via topological sort and greedy resolution ensures efficiency, scalability, and faithfulness to the original query's reasoning structure.

Graph and Context Pruning

While the topological sort linearizes the reasoning process, the overall inference cost remains non-trivial due to two factors: the growing size of accumulated context as subproblems are resolved, and the potential redundancy among subproblems with similar semantics or overlapping dependencies. To address this, we introduce a two-dimensional pruning mechanism that improves both computational efficiency and retrieval quality by reducing unnecessary context propagation and subproblem duplication.

Context pruning via rolling memory. As the RAG progresses through the topologically sorted subproblems, each retrieval step accumulates more contexts—text chunks retrieved from the knowledge base—which can overwhelm the LLM and introduce noise that impairs generation. To mitigate this, we maintain a compressed *memory state*—a single text string that serves as a rolling summary of the most salient facts retrieved so far. After each subproblem $p_{(i)}$ is resolved, its retrieved context and answer $a_{(i)}$ are distilled via LLM-based summarization, and the resulting summary is incorporated into the memory. At each subsequent step, newly retrieved context is summarized oriented by the query and merged with the existing memory to form an updated memory:

$$\text{Mem}_{(i)} = \text{Summarize}(\text{Mem}_{(i-1)} \cup \mathcal{R}(p_{(i)})), \quad (3)$$

where $\mathcal{R}(p_{(i)})$ is the set of retrieved text chunks for subproblem $p_{(i)}$. This summarization-based pruning prevents context bloat while preserving only the most relevant information for downstream reasoning.

Graph pruning via unified subquery generation. Multiple subproblems in the DAG often share similar priorities or loosely coupled dependencies, such as leaf or sibling nodes in the topological order, representing parallelizable factoid sub-tasks. To reduce redundant retrievals and improve efficiency, we merge subproblems with the same topological rank into a single *unified query*. For each step in the graph reasoning process (Section), let $S_{(i)} = \{p_{(j)} \mid \text{rank}(p_{(j)}) = \text{rank}(p_{(i)})\}_{j=i}^{n}$ denote the set of subproblems with the same rank as $p_{(i)}$. We construct a unified query:

$$q_{(i)}^{\text{uni}} = \text{Merge}(S_{(i)}), \quad (4)$$

and perform a single retrieval to obtain a unified context $\mathcal{C}_{(i)}^{\text{uni}}$ and a response set $\{a_{(j)}\}_{p_{(j)} \in S_{(i)}}$. The process then advances to the next topological rank, skipping individual subproblem iterations. This graph pruning reduces retrieval operations, shortens the reasoning chain, and enhances consistency among semantically related subproblems.

By jointly applying context pruning and graph pruning, our framework achieves more efficient inference without compromising reasoning quality. The pruning process is tightly guided by the DAG and dynamically adapts to the evolving retrieval state, offering a scalable and context-aware solution for complex query answering.

Sampling Strategy

We have developed a logic-guided RAG model that operates in an iterative, agentic manner. However, such systems often encounter a "hesitation" issue, wherein the language model, faced with uncertainty, repeatedly generates similar subqueries. This behavior impedes progress in resolving downstream subproblems and leads to inefficiencies in both computation and information gathering.

During the subquery generation step in Equation 4, we consider two strategies for iterating over subproblems: (1)

Type	Model	HotpotQA		2WikiMQA		MuSiQue	
		Str-Acc.	LLM-Acc.	Str-Acc.	LLM-Acc.	Str-Acc.	LLM-Acc.
Direct Zero-shot LLM	Llama3 (8b)	17.1	11.1	22.3	4.7	2.3	2.0
	Llama3 (13b)	23.7	20.1	33.8	15.4	6.4	6.0
	GPT-3.5-turbo	31.5	35.4	24.0	22.0	7.9	10.9
	GPT-4o-mini	38.7	36.3	26.4	24.3	17.6	14.0
Vanilla RAG	VanillaRAG (Top-1)	38.4	48.6	34.8	37.3	13.2	18.5
	VanillaRAG (Top-3)	43.2	53.1	43.0	42.0	20.3	23.6
	VanillaRAG (Top-5)	44.1	53.9	46.7	45.6	21.0	23.6
Graph-based RAG	KGP	46.4	57.1	47.5	43.7	23.3	27.5
	G-retriever	28.5	40.9	26.7	35.7	9.1	15.6
	RAPTOR	48.1	57.8	47.7	45.9	25.2	29.1
	GraphRAG	39.6	45.2	46.3	43.3	16.5	23.1
	LightRAG	47.8	57.7	43.1	36.3	18.1	19.4
	HippoRAG	53.7	55.6	47.7	47.2	24.9	30.1
	HippoRAG2	**56.7**	61.9	50.0	47.1	27.0	32.6
Ours	LogicRAG	54.8	**62.6**	**64.7**	**62.5**	**30.4**	**37.5**

Table 1: Question answering accuracy across benchmark datasets.

sampling with replacement, where the current batch of subproblems $S_{(i)}$ is retained after generating the unified subquery $q^{\text{uni}}_{(i)}$, allowing the language model to decide whether to proceed to the next batch $S_{(i+1)}$; and (2) *sampling without replacement*, where $S_{(i)}$ is removed after generating $q^{\text{uni}}_{(i)}$, enforcing forward progression to $S_{(i+1)}$. The first strategy, commonly adopted in iterative frameworks (Fu et al. 2023; Yang et al. 2024; Li, Li, and Nie 2021), is prone to stalling at intermediate stages, producing near-duplicate subqueries until the maximum number of retrieval iterations is reached.

Figure 3 presents the similarity matrix of subqueries generated under the first strategy. The heatmap indicates an increasing tendency toward repetition as iterations progress. Figure 4 compares token consumption and answer accuracy under both strategies. Sampling without replacement consistently yields lower per-question token cost across three datasets while maintaining comparable answer quality. We attribute this to two factors: (1) forcing progression prevents the model from lingering on uncertain subproblems and encourages the assimilation of broader knowledge, and (2) avoiding redundant subquery generation leads to more efficient inference.

Motivated by these findings, we adopt sampling without replacement as the default strategy to ensure efficient and effective multi-step retrieval and reasoning.

Experiments

Experiment Setup

Datasets. We evaluate our method's RAG capabilities using three multi-hop question-answering benchmarks: MuSiQue (answerable) (Trivedi et al. 2022), 2WikiMultiHopQA (abbreviated as 2WikiMQA) (Ho et al. 2020), and HotpotQA (Yang et al. 2018). To manage experimental costs, we follow HippoRAG (Gutiérrez et al. 2024) by extracting 1,000 questions from each dataset's validation set. Additionally, we adopt the approach of IRCOT (Trivedi et al. 2023) and HippoRAG (Gutiérrez et al. 2024) to gather all candidate passages, including both supporting and distractor passages, from the selected questions to create a retrieval corpus for each dataset. Dataset details are presented in Table 2.

Baselines. We compare against several baselines: (i) Zero-shot LLM Inference (llama3(8b), llama3(13b) (LlamaIndex 2024), gpt3.5-turbo, gpt-4o-mini (Achiam et al. 2023)), (ii) vanilla retrieval augmented generation (top1, top3, top5), (iii) graph-based retrieval augmented generation (KGP (Wang et al. 2024), RAPTOR (Sarthi et al. 2024), G-retriever (He et al. 2024), GraphRAG (Edge et al. 2024), LightRAG (Guo et al. 2024), HippoRAG (Gutiérrez et al. 2024), HippoRAG2 (Gutiérrez et al. 2025)). For those who provide both single-step and multi-step options, we use their multi-step version for its general superior performance.

Metrics. We evaluate end-to-end QA performance on the datasets using two metrics: *string-based accuracy*, which computes whether the gold answer is included in the generated answer after normalizing them to lowercase words; and *LLM-based accuracy*, which lets an LLM decide whether the generated answer correctly matches the gold answer.

Implementation details. To ensure fair comparison, we use the same embedding models (sentence-transformers/all-MiniLM-L6-v2) for all algorithms. The k is set to 3 for each top-k retrieval. All the RAG methods use the same large language models used for generation and evaluation, which is gpt-4o-mini. Experiments are conducted on a server with one NVIDIA 3090 GPU and one Intel Xeon CPU.

	MuSiQue	2WikiMQA	HotpotQA
# of Passage	11,656	6,119	9,221
# of Query type	3	4	2

Table 2: Dataset statistics.

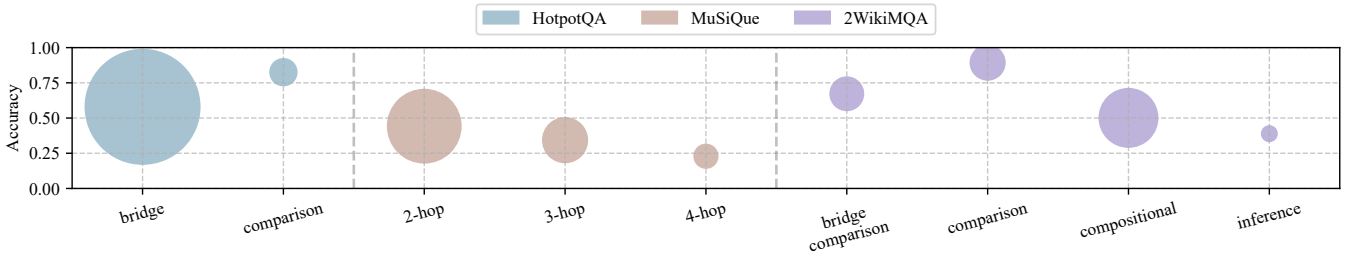

Figure 5: Distribution of accuracy across question types. Each ball represents a question type, with the y-axis position indicating its accuracy and the radius reflecting its proportion in the dataset.

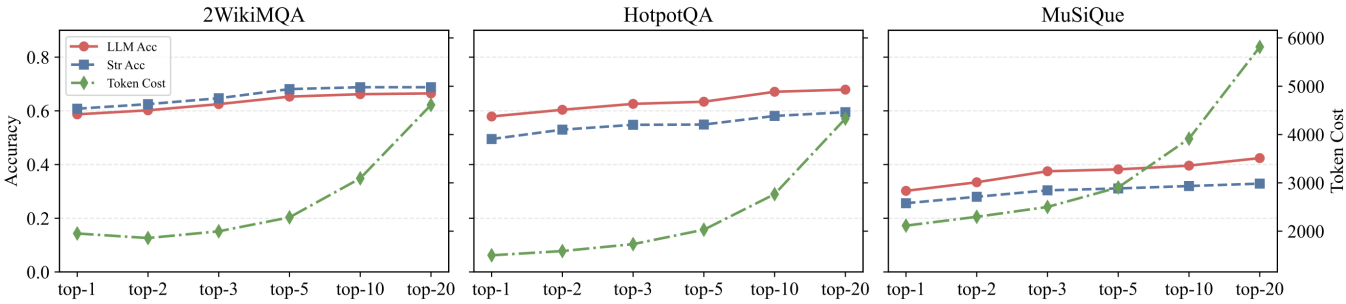

Figure 6: Pareto frontier of efficiency versus effectiveness for different k values.

Main Results

Table 1 reports the end-to-end question answering accuracy across three benchmarks, comparing LogicRAG against baselines. Metrics include *string-based accuracy* and *LLM-based accuracy*. The results lead to several key observations:

Zero-shot LLM inference. Direct prompting of LLMs without any retrieval yields poor performance across all datasets. Even strong models like GPT-4o-mini achieve only 38.7% string accuracy on HotpotQA and significantly lower on 2WikiMQA and MuSiQue, underscoring the necessity of retrieval augmentation for knowledge-intensive questions.

Vanilla RAG. Integrating corpus retrieval substantially enhances performance. Increasing k consistently improves accuracy across all datasets. However, the performance gains from larger k diminish at higher values, revealing that vanilla RAG's limitation lies in its bottleneck: reasoning over multi-hop dependencies.

Graph-based RAG. Methods that explicitly model relationships among text chunks via graph structures further enhance performance. HippoRAG2 achieves the best results among existing methods, reaching 56.7% string accuracy on HotpotQA and 27.0% on MuSiQue. However, these methods remain limited in integrating logic constraints or leveraging structural reasoning during retrieval.

LogicRAG outperforms all baselines by a significant margin across all datasets. Notably, LogicRAG achieves 64.7% string accuracy on 2WikiMQA, a +14.7% absolute improvement over the next best baseline (HippoRAG2), and boosts GPT-based accuracy to 62.5%. Similarly, on MuSiQue, LogicRAG reaches 30.4% string accuracy and 37.5% LLM accuracy, outperforming HippoRAG2 by +3.4% and +4.9%, respectively. These results highlight the effectiveness of structured, logic-guided reasoning in enhancing retrieval and QA quality in multi-hop settings.

Performance Breakdown by Question Type

Figure 5 presents the distribution of LogicRAG's accuracy across question types with a ball plot. This visualization allows us to assess the proportion of query types and their impact on model performance.

On HotpotQA, LogicRAG demonstrates strong performance on `comparison` questions, achieving an accuracy of 83%, significantly higher than the 58% accuracy on the more prevalent `bridge` questions. This suggests that LogicRAG is particularly effective in tasks that require identifying and comparing specific facts, while performance on bridge-type questions—often requiring entity chaining—is more susceptible to retrieval errors or reasoning drift.

In MuSiQue and 2WikiMQA, we observe more diverse reasoning patterns. For MuSiQue, accuracy consistently declines as the number of reasoning hops increases. This trend underscores the growing complexity of multi-hop reasoning and the challenge of maintaining coherent retrieval chains. In 2WikiMQA, LogicRAG excels at `comparison` (89%) and `bridge-comparison` (67%) questions. In contrast, it shows lower performance on `inference` questions (39%) and moderate performance on `compositional` questions (50%), which represent the largest category (44.4%). Given both the prevalence and the difficulty of the compositional type, improving model capability in handling compositional reasoning would likely yield substantial gains in overall performance on 2WikiMQA and represents a valuable direction for future research.

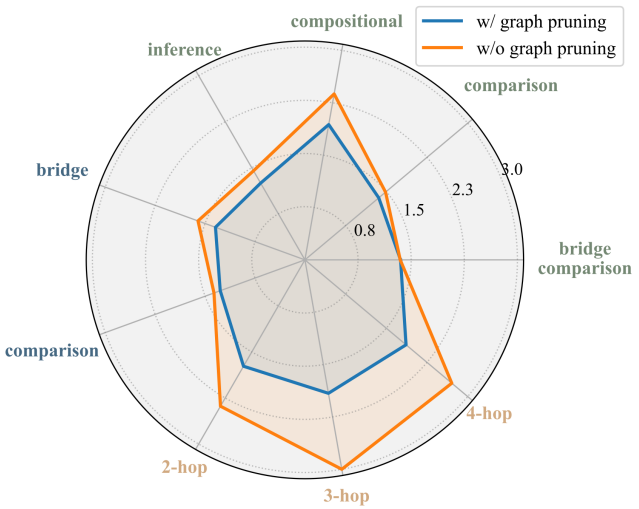

Figure 7: Impact of graph pruning, by comparing the average number of retrieval rounds with and without graph pruning.

Efficiency-Effectiveness Trade-off of Top-k

Figure 6 serves as a Pareto frontier analysis, highlighting how k impacts answer quality and computational overhead.

Across all datasets, both accuracy metrics generally improve as k increases, but with diminishing returns beyond top-3 or top-5. For example, on 2WikiMQA, increasing k from 1 to 5 leads to a notable gain in string accuracy, while gains from top-10 to top-20 are marginal. A similar trend holds for HotpotQA, where LLM accuracy saturates near top-10. On MuSiQue, improvements are more gradual, reflecting the inherent difficulty and more dispersed knowledge distribution in this dataset.

However, increasing k incurs a steep rise in token cost, particularly in MuSiQue, where the average cost exceeds 6000 tokens at top-20. This growing cost makes high k values less practical in real-world applications where latency or budget constraints are critical. Taken together, these results suggest that moderate values (e.g., $k = 3$ or $k = 5$) offer the best balance between effectiveness and efficiency. This insight also motivates our context pruning with rolling memory that summarizes salient facts during iterative RAG to avoid incremental context accumulation.

Impact of Graph Pruning

We evaluate the impact of graph pruning by comparing the average number of retrieval rounds across different query types, as shown in Figure 7. Overall, graph pruning consistently reduces the number of retrieval steps across all categories, confirming its effectiveness in eliminating redundant subproblem resolution and promoting efficient inference. Notably, compositional and multi-hop questions—typically requiring deeper reasoning—benefit most from pruning due to their higher potential for semantic overlap across subqueries.

Interestingly, while multi-hop questions (2-hop to 4-hop) generally demand more retrieval rounds, the increase is not strictly monotonic. In particular, 4-hop questions exhibit fewer retrieval rounds on average than 3-hop questions. Our investigation reveals that this arises from a behavioral tendency of the LLM: *when faced with long, complex queries, it sometimes becomes prematurely confident and produces an answer based on partial information.* This mirrors a well-known human cognitive bias—those with less insight may display higher confidence—highlighting the challenge of trust calibration in LLM-driven reasoning.

Method	Avg. Time (s)	Avg. Token
ZeroShot	5.88	216.2
VanillaRAG	4.28	489.7
G-retriever	12.50	1000.0
KGP	70.72	11097.8
Raptor	5.79	2568.0
GraphRAG	13.05	4699.8
LightRAG	35.14	5730.6
HippoRAG	6.30	2608.8
HippoRAG2	5.89	2809.2
LogicRAG	9.83	1777.9

Table 3: Query answering efficiency on 2WikiMQA.

Query-time Efficiency Comparison

Table 3 reports the query-time efficiency of various multi-hop QA models on 2WikiMQA, measured by average latency and token usage during the retrieval and generation stages. Notably, this comparison excludes the cost of graph construction, which is required by all graph-based baselines. These methods typically rely on costly preprocessing to build knowledge graphs, including steps like named entity recognition, open information extraction, and offline graph generation. The constructed graphs are then used to summarize and index the corpus. This preprocessing often takes tens to hundreds of minutes and can incur substantial additional token usage, as shown in Table 1.

By contrast, LogicRAG eliminates the need for any offline graph construction, yet still maintains competitive query-time performance with a modest latency and moderate token usage. While flat retrieval baselines like VanillaRAG and ZeroShot achieve slightly faster response times, they lack the structured reasoning capabilities enabled by LogicRAG. Taken together, these results suggest that LogicRAG offers a cost-effective and practical solution for multi-hop QA, particularly in deployment scenarios where preprocessing and knowledge update overhead is a bottleneck.

Conclusion

In this work, we present LogicRAG, a novel retrieval-augmented generation framework that dynamically leverages reasoning structures for complex query resolution. By decomposing queries into subproblems and modeling their dependencies as a directed acyclic graph, LogicRAG enables logic-aware and efficient multi-step retrieval. This principled framework offers a practical and scalable solution for knowledge-intensive complex question answering with large language models.

Acknowledgements

The work described in this paper was fully supported by a grant from the Innovation and Technology Commission of the Hong Kong Special Administrative Region, China (Project No. ITS/263/24FP).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. *Claude-3 Model Card*.

Borgeaud, S.; Mensch, A.; Hoffmann, J.; Cai, T.; Rutherford, E.; Millican, K.; Van Den Driessche, G. B.; Lespiau, J.-B.; Damoc, B.; Clark, A.; et al. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning (ICML)*.

Chen, S.; Zhang, Q.; Dong, J.; Hua, W.; Li, Q.; and Huang, X. 2024. Entity Alignment with Noisy Annotations from Large Language Models. *arXiv preprint arXiv:2405.16806*.

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

Fu, Y.; Peng, H.; Ou, L.; Sabharwal, A.; and Khot, T. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, 10421–10430. PMLR.

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

Guo, K.; Shomer, H.; Zeng, S.; Han, H.; Wang, Y.; and Tang, J. 2025. Empowering GraphRAG with Knowledge Filtering and Integration. *arXiv preprint arXiv:2503.13804*.

Guo, Z.; Xia, L.; Yu, Y.; Ao, T.; and Huang, C. 2024. LightRAG: Simple and Fast Retrieval-Augmented Generation. *arXiv preprint arXiv:2410.05779*.

Gutiérrez, B. J.; Shu, Y.; Gu, Y.; Yasunaga, M.; and Su, Y. 2024. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Gutiérrez, B. J.; Shu, Y.; Qi, W.; Zhou, S.; and Su, Y. 2025. From RAG to Memory: Non-Parametric Continual Learning for Large Language Models. In *Forty-second International Conference on Machine Learning*.

He, X.; Tian, Y.; Sun, Y.; Chawla, N. V.; Laurent, T.; LeCun, Y.; Bresson, X.; and Hooi, B. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Ho, X.; Nguyen, A.-K. D.; Sugawara, S.; and Aizawa, A. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Hong, Z.; Yuan, Z.; Chen, H.; Zhang, Q.; Huang, F.; and Huang, X. 2024. Knowledge-to-SQL: Enhancing SQL Generation with Data Expert LLM. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Hong, Z.; Yuan, Z.; Zhang, Q.; Chen, H.; Dong, J.; Huang, F.; and Huang, X. 2025. Next-generation database interfaces: A survey of llm-based text-to-sql. *IEEE Transactions on Knowledge and Data Engineering*.

Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems (TOIS)*.

Izacard, G.; Lewis, P.; Lomeli, M.; Hosseini, L.; Petroni, F.; Schick, T.; Dwivedi-Yu, J.; Joulin, A.; Riedel, S.; and Grave, E. 2023. Atlas: Few-shot learning with retrieval augmented language models. *The Journal of Machine Learning Research (JMLR)*.

Jiang, Z.; Xu, F. F.; Gao, L.; Sun, Z.; Liu, Q.; Dwivedi-Yu, J.; Yang, Y.; Callan, J.; and Neubig, G. 2023. Active retrieval augmented generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Khashabi, D.; Min, S.; Khot, T.; Sabharwal, A.; Tafjord, O.; Clark, P.; and Hajishirzi, H. 2020. Unifiedqa: Crossing format boundaries with a single qa system. In *Findings of Empirical Methods in Natural Language Processing (EMNLP)*.

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

Li, Y.; Li, W.; and Nie, L. 2021. A graph-guided multi-round retrieval method for conversational open-domain question answering. *arXiv preprint arXiv:2104.08443*.

Li, Y.; Li, Z.; Wang, P.; Li, J.; Sun, X.; Cheng, H.; and Yu, J. X. 2023. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*.

LlamaIndex. 2024. LlamaIndex - Build Knowledge Assistants over your Enterprise Data. *LlamaIndex Blog*.

Lu, Z.; Ding, K.; Zhang, Y.; Li, J.; Peng, B.; and Liu, L. 2021. Engage the public: Poll question generation for social media posts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 29–40.

Ma, S.; Xu, C.; Jiang, X.; Li, M.; Qu, H.; and Guo, J. 2024. Think-on-graph 2.0: Deep and interpretable large language model reasoning with knowledge graph-guided retrieval. In *International Conference on Learning Representations (ICLR)*.

OpenAI. 2023. GPT-4 Technical Report. *OpenAI Blog*.

Peng, B.; Zhu, Y.; Liu, Y.; Bo, X.; Shi, H.; Hong, C.; Zhang, Y.; and Tang, S. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Sarthi, P.; Abdullah, S.; Tuli, A.; Khanna, S.; Goldie, A.; and Manning, C. D. 2024. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. In *International Conference on Learning Representations (ICLR)*.

Sun, J.; Xu, C.; Tang, L.; Wang, S.; Lin, C.; Gong, Y.; Ni, L.; Shum, H.-Y.; and Guo, J. 2024. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. In *International Conference on Learning Representations (ICLR)*.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2022. ♪ MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10: 539–554.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Wang, Y.; Lipka, N.; Rossi, R. A.; Siu, A.; Zhang, R.; and Derr, T. 2024. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19206–19214.

Wu, K.; Chen, H.; Wang, C.; Karray, F.; Li, Z.; Wang, Y.; and Zhong, F. 2025. Hierarchical Instruction-aware Embodied Visual Tracking. *arXiv preprint arXiv:2505.20710*.

Xiang, Z.; Wu, C.; Zhang, Q.; Chen, S.; Hong, Z.; Huang, X.; and Su, J. 2025. When to use graphs in rag: A comprehensive analysis for graph retrieval-augmented generation. *arXiv preprint arXiv:2506.05690*.

Yang, D.; Rao, J.; Chen, K.; Guo, X.; Zhang, Y.; Yang, J.; and Zhang, Y. 2024. Im-rag: Multi-round retrieval-augmented generation through learning inner monologues. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 730–740.

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

Zhang, F.; Huang, Z.; Zhou, Y.; Guo, Q.; Li, Z.; Luo, W.; Jiang, D.; Fang, Y.; and Zhou, X. 2025a. EraRAG: Efficient and Incremental Retrieval Augmented Generation for Growing Corpora. *arXiv preprint arXiv:2506.20963*.

Zhang, Q.; Dong, J.; Chen, H.; Zha, D.; Yu, Z.; and Huang, X. 2024. Knowgpt: Knowledge graph based prompting for large language models. *Advances in Neural Information Processing Systems*, 37: 6052–6080.

Zhang, Q.; Xiang, Z.; Xiao, Y.; Wang, L.; Li, J.; Wang, X.; and Su, J. 2025b. FaithfulRAG: Fact-Level Conflict Modeling for Context-Faithful Retrieval-Augmented Generation. *arXiv preprint arXiv:2506.08938*.

Zhong, F.; Wu, K.; Wang, C.; Chen, H.; Ci, H.; Li, Z.; and Wang, Y. 2024. Unrealzoo: Enriching photo-realistic virtual worlds for embodied ai. *arXiv preprint arXiv:2412.20977*.

Zhou, H.; Du, J.; Zhou, C.; Yang, C.; Xiao, Y.; Xie, Y.; and Huang, X. 2025a. Each Graph is a New Language: Graph Learning with LLMs. *arXiv preprint arXiv:2501.11478*.

Zhou, Y.; Su, Y.; Sun, Y.; Wang, S.; Wang, T.; He, R.; Zhang, Y.; Liang, S.; Liu, X.; Ma, Y.; et al. 2025b. In-depth Analysis of Graph-based RAG in a Unified Framework. *arXiv preprint arXiv:2503.04338*.

Zhuang, L.; Chen, S.; Xiao, Y.; Zhou, H.; Zhang, Y.; Chen, H.; Zhang, Q.; and Huang, X. 2025. LinearRAG: Linear Graph Retrieval Augmented Generation on Large-scale Corpora. *arXiv preprint arXiv:2510.10114*.