QA-GraphRAG: Query-Adaptive Plug-and-Play Retrieval Integration for Graph-based Retrieval-Augmented Generation

Zeang Sheng[◇]
Peking University
shengzeang18@pku.edu.cn

Ruihong Sun
Tencent Inc
rachelsun@tencent.com

Jiahao Xu
Tencent Inc
jettexu@tencent.com

Hanmei Luo
Tencent Inc
mavisluo@tencent.com

Peng Chen
Tencent Inc
pengchen@tencent.com

Wentao Zhang[◇‡]
Peking University
wentao.zhang@pku.edu.cn

Bin Cui[◇†]
Peking University
bin.cui@pku.edu.cn

ABSTRACT

Large Language Models (LLMs) have demonstrated remarkable capabilities, yet they often suffer from hallucinations and lack up-to-date knowledge. Retrieval-Augmented Generation (RAG) addresses these limitations by grounding LLMs in external knowledge. While vector-based RAG is effective for simple queries, it struggles with complex queries that require multi-hop reasoning. Graph-based RAG frameworks have emerged to solve this by constructing knowledge graphs that capture global relationships and enable multi-hop reasoning. However, these graph-based approaches frequently underperform on simple fact-based queries compared to their vector-based counterparts, as they may lose detailed entity information. In this paper, we conduct dataset-level and framework-level analysis targeting graph-based RAG approaches. We find that existing QA benchmark datasets can be split to "Local" and "Global" queries that have different properties; and different RAG frameworks perform differently on these two kinds of queries. Concretely, existing graph-based RAG frameworks, including recent dual-branch ones, cannot consistently outperform vector-based RAG on "Local" queries. We attribute this phenomenon to the fact that graph-based RAG often employs a fixed retrieval strategy, leading to redundant information retrieval and unnecessary cost for simple queries. Based on the analysis, we propose QA-GraphRAG, a new query-adaptive plug-and-play retrieval integration for graph-based RAG frameworks. QA-GraphRAG incorporates a pre-trained router that predicts the optimal knowledge hierarchy from which to start retrieval based on the characteristics of the input query. Extensive experiments on KGQA datasets and GraphRAG-Bench demonstrate that equipping existing graph-based RAG frameworks with our QA-GraphRAG leads to substantial performance improvements.

PVLDB Reference Format:
Zeang Sheng, Ruihong Sun, Jiahao Xu, Hanmei Luo, Peng Chen, Wentao Zhang, Bin Cui. QA-GraphRAG: Query-Adaptive Plug-and-Play Retrieval Integration for Graph-based Retrieval-Augmented Generation. PVLDB, 19(9): 2224 - 2233, 2026.
doi:10.14778/3819518.3819546

[◇]School of Computer Science & Key Lab of High Confidence Software Technologies (MOE) & Beijing Key Laboratory of Software and Hardware Cooperative Artificial Intelligence Systems, Peking University

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/PKU-DAIR/Query-Adaptive-GraphRAG.

1 INTRODUCTION

Large Language Models (LLMs) have attracted numerous attention for their impressive performance on diverse real-world tasks [4, 19, 20, 35, 36, 38, 52]. To further improve LLM's capability, Retrieval-Augmented Generation (RAG) [7, 18, 21, 53] has been introduced to relieve LLMs' hallucination issue [13, 46] and provide up-to-date knowledge. Vanilla vector-based RAG retrieves relevant raw text chunks from its knowledge base. However, vector-based RAG fails to handle queries that require multi-hop knowledge or synthesized analysis [6] since its retrieved text chunks are mostly independent and fragmented. Borrowing ideas from graph machine learning [9, 16, 27, 31, 34, 47, 50], graph-based RAG frameworks [6, 8, 26, 39, 51] are proposed to address this issue. Graph-based RAG constructs knowledge graph as its knowledge base based on the raw text chunks. In this way, graph-based RAG can capture multi-hop knowledge and have a more global understanding of the stored knowledge.

Although graph-based RAG has achieved significant performance gain in many tasks, some recent researches have found that graph-based RAG performs poorly on simple fact-based queries [10, 40, 41, 55], even outperformed by vanilla vector-based RAG. They attribute this phenomenon to graph-based RAG's weakness on capturing the detailed entity information. Some recent graph-based RAG frameworks [2, 8, 12] try to resolve this issue by adopting a

[◇]Center for Machine Learning Research & Beijing Key Laboratory of Data Intelligence and Security, Peking University
[‡]Zhongguancun Academy
[†]Institute of Computational Social Science, Peking University (Qingdao)

Proceedings of the VLDB Endowment, Vol. 19, No. 9 ISSN 2150-8097.
doi:10.14778/3819518.3819546

dual-branch retrieval strategy. One branch is responsible for retrieving local-level knowledge, which is based on keyword [8] or vector similarity [2] search; and the other branch is responsible for retrieving global-level knowledge, which is similar to early graph-based RAG frameworks. Another framework HiRAG [12] additionally retrieves bridging paths between local and global knowledge. By retrieving local and global knowledge simultaneously, these frameworks achieve decent performance on both simple fact-based queries and complex queries.

However, regardlessly retrieving from two branches may introduce unnecessary cost and redundant information for many queries. In this paper, we conduct dataset-level and framework-level analysis on graph-based RAG frameworks. We find that the queries within existing QA benchmark datasets can be categorized into "Local" queries and "Global" queries, and they have diverse characteristics. Further, we find that graph-based RAG frameworks and vector-based RAG frameworks perform differently on these two kinds of queries (E.g., graph-based RAG frameworks cannot consistently outperform vector-based RAG frameworks on "Local" queries). We also noticed that most existing graph-based RAG frameworks [2, 6, 12, 26, 49] orchestrate their knowledge base in a hierarchical manner. We find that low knowledge hierarchies are proper for answering simple fact-based queries; while high knowledge hierarchies are proper for answering complex queries.

Based on the analysis results and this observation, we propose a new query-adaptive plug-and-play retrieval integration called QA-GraphRAG, which is compatible with most existing graph-based RAG frameworks. We design a flexible router module that can adaptively alter retrieval strategy according to query characteristics. The deployment of QA-GraphRAG can be split to two stages: 1) offline pre-train and 2) online inference. During offline pre-train, we implement two router deployment strategies with different performance-cost trade-offs, which produces *Generalist* router and *Specialist* router, respectively. The *Generalist* router is trained by question-answer pairs from public benchmark datasets [11, 33, 43] and preprocessed LLM performance scores to learn to predict the optimal knowledge retrieval hierarchy. The *Specialist* router is further finetuned based on the *Generalist* router when a small number of queries from the target dataset is available. During online inference, the router directly predicts the appropriate retrieval strategy for the current query based on its embeddings. Experiments on Knowledge Graph Question Answer (KGQA) datasets and GraphRAG-Bench [41] show that existing graph-based RAG frameworks achieve significant performance improvements when equipped with QA-GraphRAG.

Our contributions in this paper can be summarized as follows:

- (1) *New Finding*. We find that existing dual-branch graph-based RAG frameworks cannot well handle different kinds of queries and further introduce additional noise and cost.
- (2) *Plug-and-play Integration*. We propose QA-GraphRAG, a query-adaptive retrieval integration that is compatible with most existing graph-based RAG frameworks.
- (3) *Strong Performance*. We integrate QA-GraphRAG into many graph-based RAG frameworks and find that they achieve great performance gains on various public benchmark datasets.

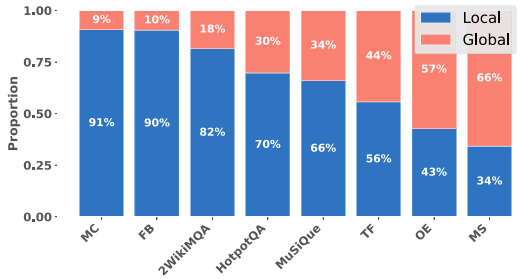


Figure 1: "Local" vs."Global" query ratio per dataset.

2 PRELIMINARIES

2.1 Problem Formulation

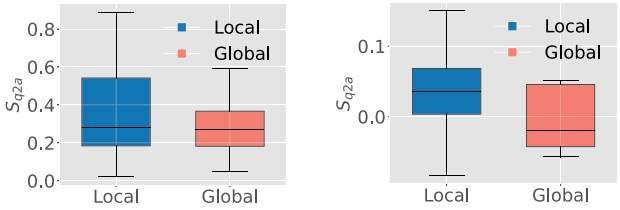
In this paper, we focus on the Knowledge Graph Question Answering (KGQA) task [5, 30, 30, 44, 56]. In conventional KGQA datasets, each question-answer pair is equipped with its own context. When evaluating RAG frameworks on these datasets, we rebuild the peripheral knowledge base for each question-answer pair. In contrast, the evaluation setting of new graph-based RAG specific KGQA datasets [40, 41] is more akin to real-world scenarios. In these datasets, all the question-answer pairs share the same context. Thus, when evaluating on these datasets, we build the shared peripheral knowledge base only once when evaluating on these datasets. In this paper, we use both these two kinds of KGQA datasets for evaluation.

2.2 Graph-based RAG

Retrieval-Augmented Generation (RAG) [7, 53, 54] is a powerful approach that helps the LLMs to reduce hallucination and gain up-to-date knowledge when answering questions. Graph-based RAG frameworks [3, 6] are proposed to enhance the multi-hop reasoning and synthesizing capability of conventional RAG approaches. GraphRAG [6] by Microsoft first identifies entities and relations from raw text chunks using LLMs and constructs the first-level knowledge graph. Then, GraphRAG conducts the Leiden algorithm [32] iteratively and finally produces a hierarchical knowledge base. When encountered with a query, GraphRAG retrieves textual descriptions of the relevant entities from the hierarchical knowledge base and feed them to LLMs for answer generation. This way, the LLM is exposed to more coarse-grained information; and the relations inside the hierarchical knowledge base act as shortcuts connecting related concepts. RAPTOR [26] is another graph-based RAG framework that adopts a hierarchical clustering algorithm implemented by Gaussian Mixture Model clustering [42] to build tree-like hierarchical knowledge base.

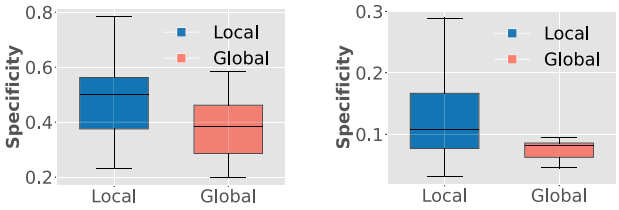
3 DATASET-LEVEL ANALYSIS

Recently, some researchers have found through experimental analysis that some graph-based RAG frameworks underperform vanilla vector-based RAG frameworks on many KGQA datasets [10, 40, 41, 55]. In the same time, graph-based RAG frameworks show significantly higher performance on newly-proposed graph-based RAG specific benchmark datasets [40, 41]. In this section, we conduct an empirical analysis comparing KGQA datasets and graph-based RAG



(a) 2WikiMQA (b) MC in GraphRAG-Bench

Figure 2: S_{q2a} comparison of "Local" and "Global".



(a) 2WikiMQA (b) MC in GraphRAG-Bench

Figure 3: *Specificity* comparison of "Local" and "Global".

specific benchmark datasets and try to identify their intrinsic differences. Concretely, we choose MuSiQue [33], 2WikiMQA [11] and HotpotQA [43] for KGQA datasets and the five tasks in GraphRAG-Bench [41] for graph-based RAG specific datasets.

3.1 Inter-dataset Analysis

Existing experimental analysis found that many graph-based RAG frameworks perform poorly on KGQA datasets. They attributed this phenomenon to that graph-based RAG frameworks introduce redundant global knowledge when only local detailed knowledge is required for answering simple fact-based queries.

Here, we use Qwen2.5-32B-Instruct [22] as the classifier to categorize each query in the QA datasets to either "Local" or "Global". The "Local" category means that this query only needs simple fact related knowledge and require no multi-hop or synthetic capability. The "Global" category means that this query needs more high-level summarized knowledge and possibly require multi-hop or synthetic capability. To ensure the reliability of the LLM-based classifier, we randomly sample 50 queries from each dataset (400 in total) and have three human annotators independently label them as "Local" or "Global" following the same criteria defined above. The majority vote of the annotators shows strong agreement with the LLM predictions, which achieves 90.75% accuracy (363/400). The validation results confirm that our LLM-based classifier reliably distinguishes between "Local" and "Global" queries.

We report the "Local" vs. "Global" ratio of each dataset in Figure 1. "FB", "MC", "MS", "TF" and "OE" each denotes a unique task in GraphRAG-Bench, which is short for "Fill-in-Blank", "Multi-Choice", "Multi-Select", "True-or-False", "Open-Ended", respectively. Figure 1 shows that KGQA datasets possess significantly more "Local" queries than "Global" queries. For example, 82% of queries in the 2WikiMQA dataset are "Local" queries. As for GraphRAG-Bench, different tasks show diverse trends. For example, nearly all the queries in "MC" and "FB" tasks are "Local" queries; while most queries in "OE" and "MS" tasks are "Global" queries. This empirical analysis reveals that "Local" queries dominate in KGQA datasets and "Global" queries constitute a significantly larger proportion in certain tasks of graph-based RAG specific benchmarks.

3.2 Within-dataset Analysis

The empirical analysis in the previous subsection shows that KGQA datasets have significantly more "Local" queries and less "Global" queries than graph-based RAG benchmark datasets. This finding may explain why some graph-based RAG frameworks performs poorly on KGQA datasets. In this subsection, we take a closer look on the 2WikiMQA dataset and the MC task in GraphRAG-Bench, trying to locate the specific properties of "Local" queries and "Global" queries. We propose two quantitative metrics to analyze these two kinds of queries. The first quantitative metric is the *average cosine similarity between query and answer* of each QA pair, which is formulated as follows:

$$S_{q2a} = \frac{1}{N}\sum_{k=1}^{N}\frac{\mathbf{Q}_k \cdot \mathbf{A}_k}{|\mathbf{Q}_k|\,|\mathbf{A}_k|}, \tag{1}$$

where $\mathbf{Q}$ and $\mathbf{A}$ are embedding matrix of paired questions and answers in the KGQA dataset encoded by the same encoding model (*all-MiniLM-L6-v2* [37] in our case) in the RAG frameworks, respectively. The second quantitative metric is called *Specificity*, which measures the fraction of tokens in question that are named entities or capitalized tokens. Intuitively, "Local" queries would mention more specific entities than "Global" queries since they care more about detailed entity information.

Evaluation results of these two metrics are reported in Figure 2 and 3. Figure 2 shows that "Global" queries possess higher average S_{q2a} than "Local" queries on the 2WikiMQA dataset; while the MC task in GraphRAG-Bench shows the opposite trend. This observation illustrates that "Local" and "Global" queries have no substantial difference in terms of question-answer similarity. Figure 3 shows that "Local" queries has significantly higher *Specificity* than "Global" queries on both two datasets. This finding suggests that "Local" queries indeed mentions more specific entities in their questions, which may require more detailed knowledge to tackle.

3.3 Dataset-level Analysis Summary

After conducting the dataset-level empirical analysis of graph-based RAG frameworks, we have the following three findings:

- 1) KGQA datasets are mostly composed of "Local" queries while some tasks in new graph-based RAG specific benchmark datasets contain significantly more "Global" queries.
- 2) The question-answer similarities of "Local" queries and "Global" queries have no substantial difference, and depend more on specific dataset properties.
- 3) "Local" queries mention significantly more specific entities than "Global" queries, which may require more detailed knowledge to deal with.

In the next section, we will characterize the performance of different RAG frameworks on KGQA datasets and graph-based RAG specific benchmark datasets.

Table 1: Graph vs. vector RAG on KGQA datasets.

Frameworks	MuSiQue		2WikiMQA		HotpotQA	
	Acc	Recall	Acc	Recall	Acc	Recall
BM25	17.23	**28.87**	33.29	45.87	51.78	57.74
TF-IDF	17.58	28.11	**33.86**	46.04	51.52	58.69
GraphRAG	15.31	25.57	31.40	43.57	50.02	55.76
RAPTOR	**18.13**	28.59	33.26	**46.29**	**52.29**	**59.42**

4 FRAMEWORK-LEVEL ANALYSIS

In this section, we conduct empirical analysis to profile existing graph-based RAG frameworks. Firstly, we compare graph-based RAG against vector-based RAG in Section 4.1 to see whether graph-based RAG is superior in most scenarios. Secondly, we compare new dual-branch graph-based RAG frameworks [2, 8] against old single-branch ones in Section 4.2 to see if the dual-branch approaches are effective. We adopt Qwen2.5-7B-Instruct [22] as the backbone LLM throughout this empirical analysis.

4.1 Graph-based RAG vs. Vector-based RAG

In this subsection, we compare graph-based RAG against vector-based RAG frameworks. For graph-based RAG, we use GraphRAG [6] and RAPTOR [26] in the comparison. For vector-based RAG, we use TF-IDF [29] and BM25 [24] to retrieve relevant text chunks. We report evaluation results of all the frameworks on MuSiQue [33], 2WikiMQA [11] and HotpotQA [43] in Table 1.

Evaluation results in Table 1 show that simple vector-based RAG methods can sometimes beat complex graph-based RAG frameworks on KGQA datasets. Our empirical analysis results are in accord with many recent experimental analyses on graph-based RAG [10, 40, 41, 55]. They found that graph-based RAG frameworks cannot beat vector-based RAG frameworks in many scenarios where simple fact-based queries dominate. This finding raises the concern that whether we need the complex retrieval pattern in graph-based RAG frameworks to solve simple fact-based queries.

4.2 Dual-branch vs. Single-branch

In this subsection, we compare new dual-branch graph-based RAG frameworks against old single-branch ones. For dual-branch graph-based RAG frameworks, we use TREX [2] that is built on RAPTOR [26] for comparison. For single-branch frameworks, we modify the original TREX to get two new variants: *TREX-local* and *TREX-global*. TREX-local only preserves the local branch and TREX-global only preserves the global branch (i.e., equivalent to RAPTOR). We report evaluation results of TREX and two its variants together with TF-IDF and BM25 on GraphRAG-Bench in Table 2.

Table 2 shows that TREX-local outperforms TREX on the FB and MC task that consist of more "Local" queries as shown in Figure 1. While on other tasks consisting of more "Global" queries like MS and OE, TREX and TREX-global outperform TREX-local. We also report the per-query time of TREX and its two variants in Table 3. Table 3 shows that TREX has longer per-query time than TREX-local and TREX-global, which is in accord with intuition. Comparison between these two kinds of graph-based RAG frameworks illustrates that newly proposed dual-branch approaches introduce additional cost yet cannot outperform old single-branch ones consistently.

Table 2: Accuracy comparison on GraphRAG-Bench.

Frameworks	FB	MC	MS	TF	OE
BM25	50.28	62.80	66.17	74.49	16.32
TF-IDF	**51.71**	61.98	67.52	74.17	16.58
TREX-local	51.16	**63.02**	71.62	74.69	17.15
TREX-global	46.12	62.05	**72.97**	75.66	17.64
TREX	51.41	62.77	71.83	**78.81**	**18.25**

Table 3: Per-query time comparison on GraphRAG-Bench.

Frameworks	FB	MC	MS	TF	OE
TREX-local	**2.86s**	**2.85s**	**2.77s**	**2.83s**	**2.96s**
TREX-global	3.23s	3.26s	3.35s	3.27s	3.37s
TREX	3.50s	3.51s	3.58s	3.53s	3.67s

4.3 Framework-level Analysis Summary

After conducting the framework-level empirical analysis of graph-based RAG frameworks, we have the following two findings:

- 1) Graph-based RAG frameworks have no significant advantage over vector-based RAG or even underperform on KGQA datasets where simple fact-based queries dominate.
- 2) Recent dual-branch graph-based RAG frameworks partially address the performance issue on simple fact-based queries, yet still fall behind under scenarios that require extreme local knowledge.

Together with the analysis results in Section 3, we find that vector-based RAG performs better on "Local" queries and graph-based RAG performs better on "Global" queries. However, for graph-based RAG, we can approximate it to vector-based RAG by setting its retrieval level to 0 (i.e., raw text chunks). If one graph-based RAG framework can alter its retrieval strategy according to query properties, it can achieve satisfying performance on both two kinds of queries. Based on this intuition, we propose QA-GraphRAG that can adaptively alter its retrieval strategy when faced with different queries.

5 FRAMEWORK DESIGN

In this section, we introduce our QA-GraphRAG in detail. Firstly, we provide the overview of QA-GraphRAG in Section 5.1. Then, we describe the details of QA-GraphRAG's two stages — offline pre-train and online inference — in Section 5.2 and 5.3, respectively.

5.1 Framework Overview

The structures of the knowledge bases in vector-based RAG and graph-based RAG are quite different. Vector-based RAG orchestrates its text chunks all at the same semantic level. However, inside the knowledge base of graph-based RAG frameworks, low-level and high-level text chunks are more akin to detailed fact-based knowledge and generalized knowledge, respectively. Ideally, graph-based RAG frameworks should retrieve low-level text chunks for simple fact-based queries, and vice versa. However, existing graph-based RAG frameworks adopt fixed retrieval strategy and cannot satisfy the different knowledge requirement of different queries.

Motivated by this, we propose a new query-adaptive plug-and-play retrieval integration for graph-based RAG called QA-GraphRAG.

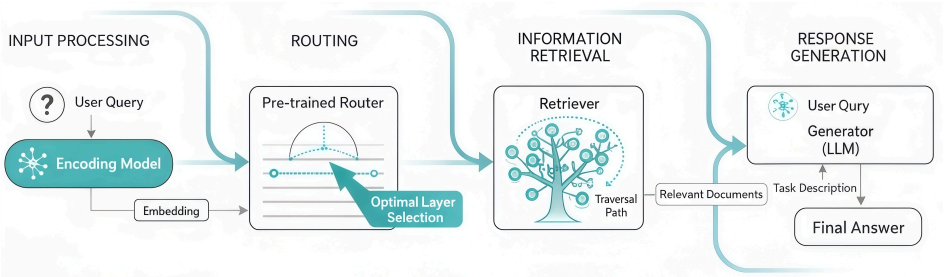


Figure 4: Workflow overview of our proposed QA-GraphRAG.

It can adaptively alter retrieval strategy according to the properties of different queries, which is achieved by a new router module. Any graph-based RAG frameworks that build their knowledge base hierarchically are compatible with QA-GraphRAG. The router module in QA-GraphRAG is responsible for predicting the optimal knowledge level to start retrieval for the current query. For example, if the knowledge base in the current graph-based RAG framework has two levels, then the router module in QA-GraphRAG helps to decide that whether to start retrieval from the second level (i.e., knowledge from both levels) or from the first level (i.e., knowledge only from the first level). The deployment of QA-GraphRAG has two stages: 1) offline pre-train and 2) online inference. In the next two subsections, we will detail these two critical stages of our proposed QA-GraphRAG.

5.2 Offline Pre-train

The router module in QA-GraphRAG is responsible for predicting which knowledge level to start retrieval. To implement this functionality, a straightforward idea is using an LLM to generate the answer. However, it would introduce significant latency when QA-GraphRAG is deployed online for inference, although LLMs can generalize to diverse data and task types. Choosing which knowledge level to start retrieval is similar to choosing the number of model layers, which is a typical Neural Architecture Search (NAS) problem [23] in AutoML. Borrowing the design intuition from the DFG-NAS [48] algorithm that uses a gating mechanism for choosing model layers, we use a 3-layer MLP as the router backbone model. An empirical analysis regarding performance and cost on router backbone model selection can be found in Section 6.3.

We implement two router deployment strategies that have different performance-cost trade-offs:

- Cross-Domain Pre-train (*Generalist* router): We pre-train the router on diverse queries sampled from multiple source datasets. This strategy exposes the router to diverse query patterns and reasoning requirements.
- Cold-Start Adaptation (*Specialist* router): We finetune a dataset-specific router based on the *Generalist* router when a small number of queries from the target dataset is available. This strategy produces a router tailored to the target dataset, which yields better performance with higher cost.

We use the *Generalist* router throughout this paper without further notion. A comparison between these two different deployment strategies can be found in Section 6.5. Next, we will detail the training data generation and the router training processes.

5.2.1 Training Data Generation. To teach the MLP router for predicting the proper knowledge level to start retrieval for queries, we construct the training data from three public Knowledge Graph Question-Answer (KGQA) datasets: HotpotQA [43], Natural Questions (NQ) [17], and TriviaQA [15]. We randomly sample 2,000 question-answer pairs from each dataset (for HotpotQA, we sample from its training set), resulting in a total of 6,000 training samples.

Then, we adopt the graph-based RAG framework inside QA-GraphRAG (RAPTOR in this paper without further notion) to answer these questions when using different knowledge levels to start retrieval. The different answers corresponding to different knowledge levels are evaluated to get their respective accuracy scores. For example, if the knowledge base has K levels, we adopt the graph-based RAG framework to generate answers with the retrieval starting at $1, 2, ..., K$. After evaluation, we get a K-dim accuracy score vector, denoted by $\mathbf{P}$. The query embeddings together with $\mathbf{P}$ are used to train the router in QA-GraphRAG.

5.2.2 Router Training. Suppose the knowledge base has K levels, we initialize the router with its output dim equals to K. The router takes in the encoded query embedding and output its K-dim knowledge level preference score vector, denoted by $\mathbf{Q}$. We use the Gumbel-Softmax [14] function on $\mathbf{Q}$ to approximate the hard-select operation during online inference, and the final output of the router is denoted by $\widetilde{\mathbf{Q}}$. The loss function for training the router is formulated as follows:

$$loss = 1 - (\mathbf{P} * \widetilde{\mathbf{Q}}), \tag{2}$$

where $\mathbf{P}$ and $\widetilde{\mathbf{Q}}$ are multiplied in the element-wise manner. After training, the router in QA-GraphRAG gains the ability to adaptively select the most appropriate knowledge level to start retrieval for the current query based on the query's embedding.

5.3 Online Inference

With the pre-trained router, QA-GraphRAG can now well handle different kinds of queries in a query-adaptive manner. When given a query, the online inference of QA-GraphRAG has following steps:

- 1) The encoding model in QA-GraphRAG encode the textual query to a semantic-rich embedding.
- 2) The pre-trained router in QA-GraphRAG predicts the most appropriate knowledge level to start retrieval based on the query embedding.

Table 4: Evaluation results on KGQA datasets.

Frameworks	MuSiQue		2WikiMQA		HotpotQA	
	Acc	Recall	Acc	Recall	Acc	Recall
BM25	17.23	28.87	33.29	45.87	51.78	57.74
TF-IDF	17.58	28.11	33.86	46.04	51.52	58.69
GraphRAG	15.31	25.57	31.40	43.57	50.02	55.76
GraphRAG-QA	**18.94**	**30.06**	**34.79**	**46.66**	**54.13**	**60.84**
RAPTOR	18.13	28.59	33.26	46.29	52.29	59.42
RAPTOR-QA	**20.92**	**32.17**	**35.77**	**49.30**	**54.81**	**61.69**
LightRAG	15.07	24.38	33.17	46.20	51.58	58.03
LightRAG-QA	**19.08**	**28.95**	**35.71**	**48.33**	**55.31**	**61.79**
TREX	18.34	28.98	33.54	46.58	52.64	59.30
TREX-QA	**21.89**	**33.03**	**35.75**	**49.30**	**54.81**	**62.69**
HiRAG	18.80	29.93	34.62	46.99	52.85	59.98
HiRAG-QA	**22.44**	**33.52**	**36.29**	**50.38**	**55.72**	**62.87**

- 3) The retriever in QA-GraphRAG retrieves relevant text chunks in the hierarchically orchestrated knowledge base starting from the knowledge level predicted by the router.
- 4) The generator in QA-GraphRAG takes in the query and the retrieved text chunks and generates the final answer.

The second step is the most important step where the router chooses the appropriate knowledge level to start retrieval for the current query. The introduced time cost of the MLP router is negligible compared to the LLM inference time. Our proposed QA-GraphRAG is compatible with most existing graph-based RAG frameworks. To gain the query-adaptive retrieval ability, a graph-based RAG framework only needs to incorporate an additional router module and slightly modify its retrieval algorithm to control the retrieval knowledge level.

6 EXPERIMENTS

In this section, we thoroughly compare the performance of our proposed QA-GraphRAG against vector- and graph-based baselines. Code is available at https://github.com/PKU-DAIR/Query-Adaptive-GraphRAG.

6.1 Experiment Settings

6.1.1 Hardware Setup. All the evaluations are conducted on a Linux server with two Intel(R) Xeon(R) Platinum 8255C CPUs and eight Tesla V100s with 32GB GPU memory.

6.1.2 Datasets & Evaluation Metrics. We conduct evaluations on both conventional KGQA datasets and graph-based RAG specific benchmark datasets. For conventional KGQA datasets, we use MuSiQue [33], 2WikiMQA [11] and HotpotQA [43]. For graph-based RAG benchmark datasets, we use GraphRAG-Bench [41] that consists of five tasks. We adopt accuracy and recall to evaluate on the conventional KGQA datasets. While for GraphRAG-Bench, we adopt its original evaluation scripts to obtainscores. We use Qwen2.5-7B-Instruct as the backbone LLM by default.

6.1.3 Baselines. We validate the effectiveness of QA-GraphRAG on the following graph-based RAG baselines: GraphRAG [6], RAPTOR [26], LightRAG [8], TREX [2], and HiRAG [12]. We integrate QA-GraphRAG with them and compare to their original performance. Further, we also report the evaluation results of TF-IDF [29] and BM25 [24] to serve as vector-based RAG representatives for a more comprehensive comparison.

Table 5: Evaluation results on GraphRAG-Bench.

Frameworks	FB	MC	MS	TF	OE
BM25	50.28	62.80	66.17	74.49	16.32
TF-IDF	51.71	61.98	67.52	74.17	16.58
GraphRAG	49.77	65.32	**72.84**	73.27	20.02
GraphRAG-QA	**52.22**	**68.93**	72.79	**81.78**	**21.91**
RAPTOR	51.12	64.05	72.97	70.56	18.64
RAPTOR-QA	**52.74**	**68.66**	**73.23**	**79.61**	**20.96**
LightRAG	45.84	63.84	69.43	72.59	19.86
LightRAG-QA	**51.62**	**68.47**	**72.83**	**79.58**	**20.79**
TREX	51.41	62.77	71.83	78.81	20.25
TREX-QA	**52.69**	**69.22**	**73.29**	**82.21**	**22.05**
HiRAG	52.34	63.85	72.15	79.65	20.77
HiRAG-QA	**53.26**	**69.91**	**72.78**	**82.69**	**21.72**

6.2 Overall Performance Comparison

We report the evaluation results of all the RAG frameworks on conventional KGQA datasets and GraphRAG-Bench in Table 4 and 5, respectively. The "-QA" suffix denotes the variant that is integrated with QA-GraphRAG. Table 4 shows that the "-QA" variants outperform their corresponding vanilla variants on all the three KGQA datasets. Furthermore, on some datasets that the variant originally outperformed by vector-based RAG, integrating QA-GraphRAG helps it beat vector-based RAG. For example, GraphRAG achieves 15.31 accuracy and 25.57 recall on the MuSiQue dataset, while BM25 achieves 17.23 and 28.87, respectively. After integrating with QA-GraphRAG, GraphRAG-QA achieves 18.94 accuracy and 30.06 recall, which exceeds BM25. Evaluation results on GraphRAG-Bench in Table 5 share the same trend as Table 4. All the "-QA" variants outperform their corresponding vanilla variants on all the tasks, except GraphRAG-QA on the MS task. Compared to other datasets, MS task has the highest "Global" query ratio according to Figure 1, yet QA-GraphRAG mainly boosts the performance of existing graph-based RAG frameworks on "Local" queries by limiting their knowledge retrieval to a low level. Overall, the significant performance superiority of "-QA" variants against original variants strongly illustrates the effectiveness of our QA-GraphRAG.

6.3 Router Backbone Selection Analysis

We adopt an MLP as the router in QA-GraphRAG for its balance between effectiveness and efficiency. In this subsection, we compare the overall performance of QA-GraphRAG when utilizing different backbone models for the router, including LLMs and adaptive retrieval methods.

6.3.1 Vs. LLM. We compare the original MLP option against zero-shot Qwen2.5-3B-Instruct and report the evaluation results in Table 6 and 7. Evaluations show that "-QA-LLM" variants has no consistent performance advantage over their corresponding "-QA" variants. Further, Table 7 shows that the "-QA-LLM" variants have significantly higher per-query time than the "-QA" variants, even slower than the original variant most of the time. Thus, we choose to use the MLP as the router in QA-GraphRAG considering both the effectiveness and the query efficiency.

Table 6: Router backbone selection accuracy comparison.

Frameworks	MuSiQue	HotpotQA	FB	MS	TF	OE
GraphRAG-QA	**18.94**	**54.13**	**52.22**	72.79	**81.78**	21.91
GraphRAG-QA-LLM	18.64	53.35	51.45	72.38	81.20	20.87
GraphRAG-Self-RAG	18.73	54.06	51.12	72.23	81.06	20.44
GraphRAG-SeaKR	18.89	53.59	50.69	**73.48**	80.36	**22.29**
RAPTOR-QA	**20.92**	54.81	**52.74**	73.23	79.61	20.96
RAPTOR-QA-LLM	20.01	**54.89**	52.59	**73.65**	80.82	20.76
RAPTOR-Self-RAG	20.59	54.83	52.39	72.66	**81.35**	20.83
RAPTOR-SeaKR	21.03	54.69	51.84	73.21	81.09	**21.49**

Table 7: Router backbone selection cost comparison.

Frameworks	MuSiQue	HotpotQA	FB	MS	TF	OE
GraphRAG	4.66s	4.05s	5.13s	5.05s	4.96s	5.19s
GraphRAG-QA	**4.19s**	**3.82s**	**4.57s**	**4.51s**	**4.47s**	**4.52s**
GraphRAG-QA-LLM	5.46s	5.16s	5.39s	5.45s	5.36s	5.86s
GraphRAG-Self-RAG	11.74s	11.39s	11.53s	12.16s	12.89s	12.65s
GraphRAG-SeaKR	41.93s	42.27s	42.19s	43.27s	43.33s	44.51s
RAPTOR	2.85s	2.48s	3.26s	3.36s	3.26s	3.30s
RAPTOR-QA	**2.57s**	**2.21s**	**2.90s**	**2.98s**	**2.84s**	**2.90s**
RAPTOR-QA-LLM	3.38s	3.33s	3.51s	3.28s	3.32s	3.89s
RAPTOR-Self-RAG	8.82s	9.31s	8.17s	9.52s	8.83s	9.86s
RAPTOR-SeaKR	29.71s	30.45s	29.63s	30.85s	29.38s	31.92s

6.3.2 Vs. Adaptive Retrieval. To further validate the effectiveness of QA-GraphRAG, we compare it against two recent adaptive RAG baselines: Self-RAG [1] and SeaKR [45]. Since these methods are originally designed for vector-based RAG, we adapt them to operate under the unified graph-based RAG environment:

- "-Self-RAG" variant: First retrieves local knowledge (level-0) and generates an answer. If the average token log-probability $< \theta_{self}$, global knowledge (root-level summaries) is retrieved and combined to regenerate the answer.
- "-SeaKR" variant: Generates $N = 5$ answers from local knowledge, computes pairwise embedding similarity, and defines $uncertainty = 1 - avg\_sim$. If $uncertainty > \theta_{sea}$, global knowledge is retrieved and the answer is regenerated; otherwise, select the cluster-center answer.

Both θ_{self} and θ_{sea} are manually tuned on the training set. We integrate these two baselines into GraphRAG and RAPTOR and report their respective performance in Table 6 and 7. Table 6 illustrates that "-QA", "-Self-RAG" and "-SeaKR" variants each achieves varying degrees of success across different datasets. Notably, QA-GraphRAG only requires a single retrieval call per query, whereas Self-RAG and SeaKR occasionally require more retrievals, , which significantly increases the average retrieval cost. This phenomenon is reflected in the time cost comparison in Table 7.

6.4 LLM Backbone Selection Analysis

In this subsection, we substitute the underlying LLM backbone from Qwen2.5-7B-Instruct to Qwen2.5-14B-Instruct and see how LLM backbone selection influences the performance of QA-GraphRAG. Evaluation results in Table 8 show that all the "-14B" variants outperform their corresponding "-7B" variants, which is in accord with intuition. Further, we find that GraphRAG-QA-7B outperforms GraphRAG-14B on the HotpotQA dataset, and the two "-QA-7B" variants outperform their "-14B" variants on the TF task of GraphRAG-Bench. This finding strongly illustrates the effectiveness of our proposed QA-GraphRAG.

Table 8: LLM backbone selection accuracy comparison.

Frameworks	MuSiQue	HotpotQA	FB	MS	TF	OE
GraphRAG-7B	15.31	50.02	49.77	72.84	73.27	20.02
GraphRAG-14B	23.10	53.37	61.93	76.22	73.71	27.88
GraphRAG-QA-7B	18.94	54.13	52.22	72.79	81.78	21.91
GraphRAG-QA-14B	**25.93**	**56.39**	**63.37**	**76.81**	**83.39**	**28.75**
RAPTOR-7B	18.13	52.29	51.12	72.97	70.56	18.64
RAPTOR-14B	25.26	55.73	62.47	**76.57**	75.94	27.35
RAPTOR-QA-7B	20.92	54.81	52.74	73.23	79.61	20.96
RAPTOR-QA-14B	**26.69**	**57.77**	**64.02**	76.61	**82.49**	**28.36**

Table 9: Router deployment strategy comparison.

Methods	HotpotQA	MuSiQue	FB	MS	TF	OE
HotpotQA router	20.62	54.59	52.16	72.62	79.11	20.15
Generalist router	20.92	54.81	52.74	73.23	79.61	20.96
Specialist router	**21.19**	**54.97**	**53.02**	**73.64**	**79.89**	**21.27**

6.5 Router Deployment Strategy Analysis

In this subsection, we compare the performance between the two router deployment strategies inside QA-GraphRAG where the *Specialist* router is trained additionally on 200 samples from each target dataset. We also include the HotpotQA Router in the comparison, which is pre-trained only on the 2,000 samples from the HotpotQA training set. Table 9 shows that the *Generalist* router consistently outperforms the HotpotQA Router, illustrating that pre-training on diverse queries effectively enhances generalization ability. Further, we observe that the *Specialist* Router achieves the highest performance on all the datasets. The modest gap between the two deployment strategies suggests that the *Generalist* router captures general query characteristics effectively. The choice between the two deployment strategies offers a practical trade-off: *Generalist* router requires no target-specific data and delivers robust performance, while *Specialist* router provides a modest accuracy boost at the cost of additionally finetuned on a small set of queries.

6.6 Interpretability Analysis

In this subsection, we split each benchmark dataset to "Local" and "Global" queries the same as in Section 3.1. We report the evaluation results of GraphRAG and RAPTOR on the MuSiQue, HotpotQA dataset and MS, TF task of GraphRAG-Bench in Figure 5. Figure 5 illustrates that GraphRAG-QA and RAPTOR-QA outperform their corresponding original variant on both "Local" and "Global" queries except that GraphRAG-QA outperformed by GraphRAG on the "Global" queries in the MS task. This finding reflects that QA-GraphRAG can improve the performance of graph-based RAG frameworks on both "Local" and "Global" queries, meaning that it can well handle different kinds of queries across datasets.

6.7 Router Decision Analysis

To better understand how our router selects retrieval levels, we randomly select 4 queries from the HotpotQA validation set. For these queries, we report in Table 10 their respective preference scores on

Table 10: Preference scores examples predicted by the *Generalist* router.

Sampled Queries	Level-0	Level-1	Level-2
"When did the director of film Hypocrite die?"	**0.99**	0.01	0.00
"What are the main differences between renewable and non-renewable energy sources?"	0.19	**0.79**	0.02
"Who is the author of the book that won the 2020 Pulitzer Prize for Fiction?"	**0.98**	0.01	0.01
"What is the overall trend in global temperatures over the past century?"	0.01	0.02	**0.97**

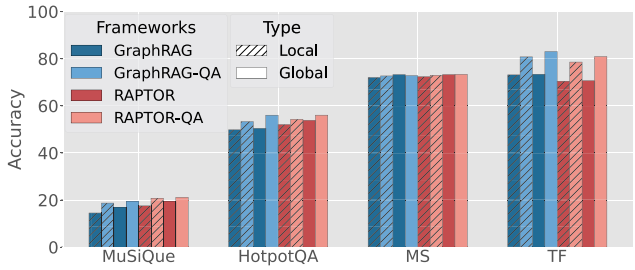


Figure 5: "Local" and "Global" performance comparison.

retrieval levels predicted by the *Generalist* router, where Level-0 means raw text chunks and higher levels means more synthesized knowledge. Table 10 shows that the router's decisions align with human intuition: factoid queries go to lower levels and abstract queries go to higher levels. This case study confirms that our router makes decisions consistent with our intended design.

7 DEPLOYMENT IN TENCENT

The GraphRAG framework in Tencent is deployed in the self-developed Taiji ML platform as an indexing service. The GraphRAG framework in Tencent uses the self-developed EasyGraph graph database for graph data store. To integrate QA-GraphRAG into the existing framework, we add a pre-trained router inside the EasyGraph serving API to ensure that the router is called every time when searching the pre-built GraphRAG index. The router takes in the encoded query embedding returned by the embedding model and outputs the optimal knowledge retrieval level. Then, the original search function is invoked along with the parameter of the predicted optimal knowledge retrieval level. The manual code modification to the existing GraphRAG framework is marginal, illustrating the flexibility of our proposed QA-GraphRAG.

8 RELATED WORKS

8.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) [7, 18, 21, 53] grounds Large Language Models (LLMs) in external and up-to-date knowledge, thereby reduces hallucinations and overcomes knowledge cutoffs. To optimize retrieval efficiency and quality, REPLUG [28] treats the LLM as a black box and prepends retrieved documents to the input. Then, it uses the model's output probability to supervise the retriever. Self-RAG [1] introduces "reflection tokens", which allows the model to adaptively decide when to retrieve and evaluate the relevance of the retrieved passages through self-critique. SeaKR [45] leverages the internal states of LLMs to estimate "self-aware uncertainty". SeaKR triggers retrieval if this uncertainty is high. This uncertainty is also used to re-rank and select the best reasoning path. While these methods improve factuality, they primarily rely on independent text chunks and lack the structural connectivity needed for global context synthesis.

8.2 Graph-based RAG

Graph-based RAG frameworks enhance retrieval by representing text as structured entities and relationships and facilitate multi-hop reasoning. Two representative frameworks are Microsoft's GraphRAG [6] and RAPTOR [26]. GraphRAG utilizes the Leiden community detection algorithm [32] to build a hierarchical knowledge graph, within which summaries are generated at multiple levels. In contrast, RAPTOR builds a tree-like hierarchy through recursive Gaussian Mixture Model clustering [42]. While these systems offer global context, they often suffer from a "granularity gap", where they sometimes provide overly broad summaries for specific fact-based queries through fixed retrieval strategies. LightRAG [8], HybridRAG [25] and TREX [2] try to resolve this issue by retrieving both local knowledge and global knowledge. Moreover, recent works like HiRAG [12] and LeanRAG [49] have explored semantic aggregation to improve this process, yet they frequently overlook the need for query-adaptive retrieval.

9 CONCLUSION

In this paper, we conduct dataset-level and framework-level analysis on existing graph-based RAG frameworks and find that they fail to satisfyingly handle different kinds of queries. To address this issue, we propose QA-GraphRAG, a query-adaptive plug-and-play retrieval integration that helps existing graph-based RAG frameworks to adaptively select the most appropriate knowledge level to start retrieval. Experiments on conventional KGQA datasets and GraphRAG-Bench illustrate that our approach significantly enhances the performance of existing graph-based RAG frameworks, providing a more efficient and effective retrieval mechanism for graph-based RAG frameworks.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (U23B2048, U22B2037, 92470121 and 62402016), Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM113), National Key R&D Program of China (2024YFA1014003), Zhongguancun Academy (C20250204, C20250602), Beijing Major Science and Technology Project (Z251100008125043, Z251100008425023), PKU-Tencent joint research Lab, and High-performance Computing Platform of Peking University. Wentao Zhang and Bin Cui are the corresponding authors.

REFERENCES

[1] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations*.

[2] Joyce Cahoon, Prerna Singh, Nick Litombe, Jonathan Larson, Ha Trinh, Yiwen Zhu, Andreas Mueller, Fotis Psallidas, and Carlo Curino. 2025. Optimizing open-domain question answering with graph-based retrieval augmented generation. In *Proceedings of the 1st workshop connecting academia and industry on Modern Integrated Database and AI Systems*. 1–11.

[3] Yukun Cao, Zengyi Gao, Zhiyang Li, Xike Xie, S. Kevin Zhou, and Jianliang Xu. 2025. LEGO-GraphRAG: Modularizing Graph-Based Retrieval-Augmented Generation for Design Space Exploration. *Proc. VLDB Endow.* 18, 10 (June 2025), 3269–3283.

[4] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology* 15, 3 (2024), 1–45.

[5] Preetam Prabhu Srikar Dammu, Himanshu Naidu, and Chirag Shah. 2025. Dynamic-kgqa: A scalable framework for generating adaptive question answering datasets. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3498–3508.

[6] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).

[7] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997* (2023).

[8] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779* (2024).

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[10] Haoyu Han, Harry Shomer, Yu Wang, Yongjia Lei, Kai Guo, Zhigang Hua, Bo Long, Hui Liu, and Jiliang Tang. 2025. Rag vs. graphrag: A systematic evaluation and key insights. *arXiv preprint arXiv:2502.11371* (2025).

[11] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics*. 6609–6625.

[12] Haoyu Huang, Yongfeng Huang, Yang Junjie, Zhenyu Pan, Yongqiang Chen, Kaili Ma, Hongzhi Chen, and James Cheng. 2025. Retrieval-Augmented Generation with Hierarchical Knowledge. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. Association for Computational Linguistics, 6044–6060.

[13] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems* 43, 2 (2025), 1–55.

[14] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.

[15] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1601–1611.

[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[17] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[18] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110* (2022).

[19] Hao Liang, Zhen Hao Wong, Rui-Tong Liu, Yu-Han Wang, Mei-Yi Qiang, Zheng-Yang Zhao, Cheng-Yu Shen, Cong-Hui He, Wen-Tao Zhang, and Bin Cui. 2026. Data preparation for large language models. *Journal of Computer Science and Technology* 41, 1 (2026), 289–317.

[20] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196* (2024).

[21] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *ACM Transactions on Information Systems* (2024).

[22] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115* (2025).

[23] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2021. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–34.

[24] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. *Okapi at TREC-3*. British Library Research and Development Department.

[25] Bhaskarjit Sarmah, Dhagash Mehta, Benika Hall, Rohan Rao, Sunil Patel, and Stefano Pasquali. 2024. Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance*. 608–616.

[26] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. [n.d.]. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

[27] Bo-Shen Shi, Yong-Qing Wang, Fang-Da Guo, Bing-Bing Xu, Hua-Wei Shen, and Xue-Qi Cheng. 2025. Domain adaptation for graph representation learning: Challenges, progress, and prospects. *Journal of Computer Science and Technology* 40, 2 (2025), 283–300.

[28] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 8371–8384.

[29] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 1 (1972), 11–21.

[30] Nadine Steinmetz and Kai-Uwe Sattler. 2021. What is in the KGQA benchmark datasets? Survey on challenges in datasets for question answering on knowledge graphs. *Journal on Data Semantics* 10, 3 (2021), 241–265.

[31] Yong-Duo Sui, Xiang Wang, Tianlong Chen, Meng Wang, Xiang-Nan He, and Tat-Seng Chua. 2024. Inductive lottery ticket learning for graph neural networks. *Journal of Computer Science and Technology* 39, 6 (2024), 1223–1237.

[32] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports* 9, 1 (2019), 1–12.

[33] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics* 10 (2022), 539–554.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[35] Haishuai Wang, Yang Gao, Xin Zheng, Peng Zhang, Jiajun Bu, and Philip S Yu. 2025. Graph neural architecture search with large language models. *Science China Information Sciences* 68, 12 (2025), 222103.

[36] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[37] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems* 33 (2020), 5776–5788.

[38] Xin Wang, Zeyang Zhang, Linxin Xiao, Haibo Chen, Chendi Ge, and Wenwu Zhu. 2025. Towards multimodal graph large language model. *Science China Information Sciences* 68, 11 (2025), 1–20.

[39] Rukai Wei, Yu Liu, Heng Cui, Yanzhao Xie, and Ke Zhou. 2025. Graph Contrastive-and-Reconstructive Hashing for Unsupervised Cross-Modal Retrieval: R. Wei et al. *Data Science and Engineering* 10, 3 (2025), 411–427.

[40] Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. 2025. When to use graphs in rag: A comprehensive analysis for graph retrieval-augmented generation. *arXiv preprint arXiv:2506.05690* (2025).

[41] Yilin Xiao, Junnan Dong, Chuang Zhou, Su Dong, Qian-wen Zhang, Di Yin, Xing Sun, and Xiao Huang. 2025. GraphRAG-Bench: Challenging Domain-Specific Reasoning for Evaluating Graph Retrieval-Augmented Generation. *arXiv preprint arXiv:2506.02404* (2025).

[42] Miin-Shen Yang, Chien-Yo Lai, and Chih-Ying Lin. 2012. A robust EM clustering algorithm for Gaussian mixture models. *Pattern Recognition* 45, 11 (2012), 3950–3961.

[43] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*. 2369–2380.

[44] Mohammad Yani and Adila Alfa Krisnadhi. 2021. Challenges, techniques, and trends of simple knowledge graph question answering: a survey. *Information* 12, 7 (2021), 271.

[45] Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Liu Weichuan, Lei Hou, and Juanzi Li. 2025. Seakr: Self-aware knowledge retrieval for adaptive retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 27022–27043.

[46] Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. 2023. Cognitive mirage: A review of hallucinations in large language models. *arXiv preprint arXiv:2309.06794* (2023).

[47] Xi Zeng, Fang-Yuan Lei, Chang-Dong Wang, and Qing-Yun Dai. 2024. Multi-view Heterogeneous Graph Neural Networks for Node Classification. *Data Science and Engineering* 9, 3 (2024), 294–308.

[48] Wentao Zhang, Zheyu Lin, Yu Shen, Yang Li, Zhi Yang, and Bin Cui. 2022. Deep and flexible graph neural architecture search. In *International Conference on Machine Learning*. PMLR, 26362–26374.

[49] Yaoze Zhang, Rong Wu, Pinlong Cai, Xiaoman Wang, Guohang Yan, Song Mao, Ding Wang, and Botian Shi. 2026. Leanrag: Knowledge-graph-based generation with semantic aggregation and hierarchical retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 34862–34869.

[50] Yu-Qing Zhang, Chen Pang, Pei Geng, Xue-Quan Lu, and Lei Lyu. 2025. Multi-Scale Adaptive Large Kernel Graph Convolutional Network for Skeleton-Based Action Recognition. *Journal of Computer Science and Technology* 40, 5 (2025), 1285–1300.

[51] Chao Zhao, Yurong Cheng, Boyang Li, Yi Yang, and Xue Wang. 2025. DFedKG: Diffusion-Based Federated Knowledge Graph Completion: C. Zhao et al. *Data Science and Engineering* (2025), 1–14.

[52] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology* 15, 2 (2024), 1–38.

[53] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-Augmented Generation for AI-Generated Content: A Survey. *arXiv preprint arXiv:2402.19473* (2024).

[54] Ruochen Zhao, Hailin Chen, Weishi Wang, Fangkai Jiao, Xuan Long Do, Chengwei Qin, Bosheng Ding, Xiaobao Guo, Minzhi Li, Xingxuan Li, et al. 2023. Retrieving Multimodal Information for Augmented Generation: A Survey. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 4736–4756.

[55] Yingli Zhou, Yaodong Su, Youran Sun, Shu Wang, Taotao Wang, Runyuan He, Yongwei Zhang, Sicong Liang, Xilin Liu, Yuchi Ma, et al. 2025. In-depth Analysis of Graph-based RAG in a Unified Framework. *arXiv preprint arXiv:2503.04338* (2025).

[56] Chen Zirui, Wang Xin, Wang Lin, XU Dawei, and Jia Yongzhe. 2021. Survey of Open-Domain Knowledge Graph Question Answering. *Journal of Frontiers of Computer Science & Technology* 15, 10 (2021).