FRAG: A Flexible Modular Framework for Retrieval-Augmented Generation based on Knowledge Graphs

Zengyi Gao[1,2*], **Yukun Cao**[1,2*], **Hairu Wang**[1,2],
Ao Ke[1,2], **Yuan Feng**[1,2], **Xike Xie**[1,2†], **S Kevin Zhou**[1,3]
[1]University of Science and Technology of China, China
[2]Data Darkness Lab, MIRACLE Center, USTC, China
[3]MIRACLE Center, Suzhou Institute for Advance Research, USTC, China
{gzy02,ykcho,hrwang00,sa21225249,yfung}@mail.ustc.edu.cn
{xkxie,skevinzhou}@ustc.edu.cn

Abstract

To mitigate the hallucination and knowledge deficiency in large language models (LLMs), Knowledge Graph (KG)-based Retrieval-Augmented Generation (RAG) has shown promising potential by utilizing KGs as an external resource to enhance LLM reasoning. However, existing KG-RAG approaches struggle with a trade-off between flexibility and retrieval quality. Modular methods prioritize flexibility by avoiding the use of KG-fine-tuned models during retrieval, leading to fixed retrieval strategies and suboptimal retrieval quality. Conversely, coupled methods embed KG information within models to improve retrieval quality but at the expense of flexibility. In this paper, we propose a novel flexible modular KG-RAG framework, termed FRAG, which synergizes the advantages of both approaches. FRAG estimates the hop range of reasoning paths based solely on the query and classifies it as either simple or complex. To match the complexity of the query, tailored pipelines are applied to ensure efficient and accurate reasoning path retrieval, thus fostering the final reasoning process. By using the query text instead of the KG to infer the structural information of reasoning paths and employing adaptable retrieval strategies, FRAG improves retrieval quality while maintaining flexibility. Moreover, FRAG does not require extra LLM fine-tuning or calls, significantly boosting efficiency and conserving resources. Extensive experiments show that FRAG achieves state-of-the-art performance with high efficiency and low resource consumption. The code for our method is publicly available at https://github.com/gzy02/FRAG.

1 Introduction

Large language models (LLMs) excel in various NLP tasks but are prone to hallucinations (2023) and errors (2023a) when answering questions that require knowledge beyond training data. These limitations undermine the trustworthiness of LLMs and raise security concerns (2024). To mitigate these issues, retrieval-augmented generation (RAG) (2020; 2020; 2023) has been developed, which dynamically retrieves information from external sources during the *retrieval phase*, while the *generation phase* leverages this retrieved data to improve generation quality. Previous RAG methods (2022a; 2023; 2024; 2024) relying on unstructured data often struggle with capturing relevant knowledge and may introduce noise, hindering effective reasoning. In response to these challenges, knowledge graphs (KGs) are increasingly being integrated into RAG (i.e., *KG-RAG*) as external knowledge sources (2022; 2024; 2024; 2024; 2024). KGs offer editable and explicit knowledge in a structured format, clarifying the context and multi-level interrelations among entities. KG-RAG retrieves "reasoning paths" that are relevant to the query, providing concise and structured contextual information to enhance the reasoning ability of LLMs (2018a; 2019; 2021; 2023a; 2024).

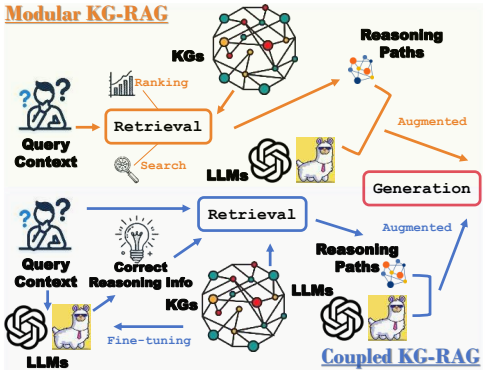

Figure 1: Modular and Coupled KG-RAG Frameworks

There are two lines of research on KG-RAG frameworks, as shown in Figure 1, depending on their ways of integration with LLMs during the retrieval phase, resulting in either *modular* or *coupled* KG-RAG. The former separates the retrieval process from LLMs, prioritizing isolation, flexi-

*Equal contribution
†Corresponding author

Findings of the Association for Computational Linguistics: ACL 2025, pages 6178–6192
July 27 - August 1, 2025

bility, and scalability, while the latter tightly integrates KGs, improving generation quality but at the expense of increased complexity and reduced flexibility and scalability.

Modular KG-RAG frameworks (2023; 2023b; 2023; 2024), aligning with conventional RAG principles, are developed to keep independence between the retrieval phase and KG-fine-tuned LLMs or specific models, while meeting essential criteria of isolation, flexibility, and scalability. This isolation prevents interference with LLMs' internal reasoning. Flexibility is achieved through seamless plug-and-play integration with external KG sources, and scalability allows these frameworks to effectively handle large KGs and complex queries. This is accomplished by using traditional algorithms for reasoning path search (2024; 2024; 2024) and ranking (2024; 2024), without requiring any additional fine-tuning to incorporate KGs information prior to reasoning. Despite the advantages, modular KG-RAG frameworks face challenges, due to their lack of prior knowledge about KGs and correct reasoning paths. This limitation impedes effective adjustment of searching and ranking parameters, leading to inferior generation quality (2023; 2023; 2023b). For example, research like ToG (2024) highlights that fixed search parameters can result in redundancy in simple tasks with short reasoning paths or the omission of critical details in more complex tasks with longer paths, ultimately weakening the reasoning capabilities of LLMs.

In contrast, coupled KG-RAG frameworks, while offering enhanced retrieval of reasoning paths, come with significant overhead due to the extensive fine-tuning of LLMs with embedded KG information (2024b; 2024; 2024a). For instance, RoG (2024b) not only use KGs as an external knowledge base but also fine-tune LLMs or train the specific models with KGs information, enabling the generation of "relation paths" grounded in KGs as retrieval templates tailored to the query context. While this method enhances retrieval and generation quality, it compromises key advantages offered by modular KG-RAG frameworks, such as the isolation of the retrieval process, flexibility in integrating various external sources, and scalability to KGs of different sizes (2023; 2023), not to mention the substantial expense involved in fine-tuning LLMs.

This leads us to our research goal: *Can we develop a solution that combines the strengths of both frameworks*? Specifically, can we implement lightweight pre-computation to obtain

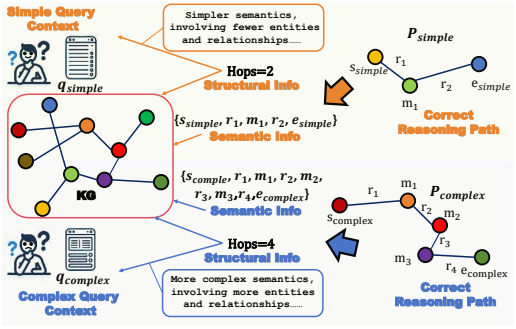

Figure 2: Analysis of Semantic and Structural Information in Reasoning Paths

sufficient information to guide the settings of ranking and searching parameters during the retrieval phase, thereby enhancing generation quality while maintaining isolation, flexibility, and scalability—without the extensive overhead associated with fine-tuning LLMs for specific KGs?

In this work, we propose a novel and flexible modular KG-RAG framework, called the *FRAG*. In a nutshell, FRAG dynamically adapts the retrieval process based on the complexity of the query context to improve reasoning accuracy, without requiring KGs information. As shown in Figure 2, we analyzed the information sources associated with the correct reasoning path P for a query context q on the KG. A path P comprises two types of information: *semantic* (i.e., entities and relations) and *structural* (i.e., number of hops). Semantic information mainly originates from KGs and is hard to perceive and utilize in advance. However, structural information is related to both KGs and the q (2024a). Generally, the more complex the query context q, the greater the number of hops in the path P (i.e., indicating a more difficult reasoning task) (2019; 2020). Thus, within a tolerable margin of error, we can predict the number of hops in P based solely on the query context q. The predicted number of hops can then serve as a key factor in enhancing the non-specific retrieval process.

Starting from the above insights, the FRAG framework is primarily featured by two key modules: "*Reasoning-aware*" and "*Flexible-retrieval*". The reasoning-aware module considers three key aspects of structural information prediction. First, to reduce the impact of inherent prediction errors, it simplifies the prediction task by estimating the coarse-grained hop range, categorizing the reasoning complexity from the query context as either simple or complex based on a hop count threshold. Second, it collects KGs and queries across vari-

ous domains, extracting semantic and statistical features from the query context to train a flexible and generalizable cross-domain classifier. Finally, an optimization strategy utilizing feedback from LLMs is employed to enhance classifier performance on specific KGs as needed.

The flexible retrieval module refines the KG-RAG retrieval process into a "preprocessing, retrieval, and postprocessing" pipeline, facilitating the tailored customization of its three components for both simple and complex reasoning tasks. Each component is built on traditional algorithms and models, ensuring the generality of our framework. For simple reasoning tasks, which typically involve shorter reasoning paths, we employ breadth-first search (BFS) and ranking as the core retrieval strategies, enabling efficient and accurate retrieval. In contrast, for complex reasoning tasks characterized by longer reasoning paths, we advocate for the use of shortest path retrieval and ranking to minimize computational overhead and reduce noise, thereby improving retrieval effectiveness.

The remainder of this paper is organized as follows: In Section 2, we review RAG and KG-RAG frameworks. Section 3 covers basic KG-RAG concepts. Section 4 details the FRAG framework design. Section 5 evaluates FRAG on benchmark datasets against SOTA methods. Finally, Section 6 summarizes our work and future directions.

2 Related Work

Retrieval-augmented generation (RAG). RAG (2021) enhances LLMs by integrating retrieved knowledge during contextual learning, mitigating knowledge gaps and hallucination issues. The NaiveRAG frameworks (2023; 2023c; 2023a) retrieve top-k relevant documents and incorporates them into the prompt for more accurate responses. Later advancements (e.g.modular RAG (2023d; 2023b), advanced RAG (2022b; 2024)) improved retrieval accuracy with additional modules. However, document-based RAG introduces noise and excessive context, impacting reasoning performance (2022a). Recent studies (2024) focus on altering the storage format of external knowledge. Recently, GraphRAG (2024) unifies various knowledge into (knowledge) graph format, transforming retrieval into a fine-grained knowledge path search, enhancing key information extraction.

RAG based on Knowledge Graphs (KG-RAG). KGs, known for dynamic, explicit, and structured knowledge representation, are increasingly used as knowledge bases for RAG. KG-RAG (2024; 2024) retrieves the top-k reasoning paths relevant to the query, providing concise and accurate contextual information for LLMs reasoning. As outlined in Section 1, KG-RAG frameworks are categorized as *Modular* or *Coupled*, based on whether KGs information is fine-tuned into the LLMs. The latest modular approach, like ToG (2024), enhances retrieval accuracy by replacing traditional ranking models with LLMs. However, ToG still struggle due to a lack of prior KG knowledge and the need for frequent LLM calls. For coupled framework, the most recent method, RoG (2024b), fine-tunes LLMs with KG information, allowing them to generate "relation paths" as query templates that directly retrieve the correct reasoning paths from KGs. However, RoG compromises interpretability, efficiency in knowledge updates, and the generality of the RAG process across different domain KGs.

3 Preliminary

Knowledge Graph. A Knowledge Graph (KG), $G = \{(s, r, e) \mid s, e \in V, r \in E\}$, is a structured method to represent entities (V) and their relationships (E). KGs use triples (s, r, e), where s and e are the start and end entity, and r is the relationship, to capture vast domain-specific knowledge.

General Process of KG-RAG. Following existing works (2023b; 2023; 2024b; 2024; 2024; 2024), KG-RAG involves two main stages: *retrieval* and *generation*. Given a query q, the first stage constructs a set of candidate "reasoning paths" by matching q with entities and relationships in the KGs. This is done by searching for relevant triples (s, r, e) from KG (G): $\textit{Retrieve}(q, G) \rightarrow \{P_i\}$. The retrieved reasoning paths set $\{P_i\}$ are then ranked by their relevance to q. A reasoning path P_i can be formally defined as: $P_i = (s, r_1, m_1, r_2, \ldots, r_{k-1}, m_{k-1}, r_l, e)$, where s is the starting entity, e is the answer entity, m_js are the intermediate entities, and r_js are the relationships connecting these entities. The number of hops in a path, equal to the number of relationships, determines the path's length l. For a given query q, a correct reasoning path includes the correct answer entity corresponding to q. In the generation stage, the top-k ranked paths augment the query, forming an enriched query q'. This q' is then fed into LLMs to generate the final output:

Generate$(q', \text{LLM}) \rightarrow$ *output*.

Simple and Complex Reasoning in KG-RAG. Following prior works on reasoning tasks in KGs (2013; 2014; 2019; 2020), KG-RAG reasoning tasks can be categorized into *simple* and *complex* reasoning, based on the minimum number of hops required to find the correct answer from the KG. Given a threshold δ[1], if the minimum hop count $h_{\min}$ of the reasoning path is less than or equal to δ, the problem is categorized as simple; otherwise, it is classified as complex: *Type*(q) = *Simple if* $h_{\min}(q, G) \leq \delta$ *else Complex*. For queries with multiple correct answers, the minimum hops across all correct reasoning paths are considered. This formal classification helps in distinguishing the complexity of reasoning tasks within KG-RAG frameworks.

4 Methodology

4.1 Overview

As illustrated in Figure 3, FRAG consists of three modules. The reasoning-aware module classifies reasoning tasks as either simple or complex based on the query. To promote the module effectiveness, we incorporate an optimization strategy that leverages the feedback from LLMs to refine the original classifier. In the subsequent flexible-retrieval module, we refine the KG-RAG retrieval process into a "preprocessing-retrieval-postprocessing" pipeline, tailored retrieval schemes are applies to identify accurate reasoning paths for both simple and complex tasks. Finally, the identified reasoning paths, together with the questions, are fed to LLMs to generate answers in the reasoning module. FRAG offers two key advantages. First, compared to other modular approaches, FRAG's reasoning-aware module perceives and utilizes the structural information of reasoning paths, thereby enhancing the retrieval process. Second, by decoupling from specific KG information, the modular retrieval module grants FRAG greater generality than coupled methods.

4.2 Reasoning-aware Module

Identifying the complexity of a query is a prerequisite for applying targeted solutions. For a reasoning task, FRAG classifies it as either simple or complex based on the minimum hop count of the correct reasoning paths in KGs. Given the requirement to dissociate from the reasoning-related KG for generality, it is impractical to obtain the precise reasoning paths and, consequently, the exact number of hops.

Extensive research in knowledge graph question answering (KGQA), along with our empirical experiments, has shown that the hop count of the correct inference path is closely linked to specific statistics in the query, such as the number of entities, relations, and clauses (2020; 2022; 2022; 2024b; 2024a). Recognizing the relationship between the structural information of reasoning paths and the query, FRAG seeks to approximate the number of hops based solely on the query, thus categorizing the query accordingly.

To implement this, we train a binary classifier using a set of public KGQA datasets, each consisting of a fundamental KG and a substantial set of queries Q paired with corresponding answers A. Prior to training, for each query $q \in Q$, we identify all of the shortest reasoning paths from the query entities Ent_q to the answer entities in $Ent_a \in A$. The minimum hop $\mathcal{H}$ among all reasoning paths of this query determines the query label $\mathcal{Y}$: $\mathcal{Y} = 1$ (Complex), if $\mathcal{H} \geq \delta$; $\mathcal{Y} = 0$ (Simple), if $\mathcal{H} < \delta$. Here the threshold δ is set to 2 as noted earlier. To capture the query's information, relevant statistics can be extracted and encoded. For simplicity, the entire query is encoded as: $\mathbf{h}_q = \text{QueryEncoder}(q) \in \mathbb{R}^{L_q \times d}$, where *QueryEncoder* can be any encoding mechanism, such as a language model. (e.g., BERT (2019)), word embeddings (e.g., Word2Vec (2013)), or TF-IDF. This way, the classification loss $\mathcal{L}$ of the binary classifier during training is computed as: $\mathcal{L} = -\sum_{q \in Q} \sum_{y_q \in \mathcal{Y}} y_q \cdot \log p(y_q|q)$. Here, $p(y_q|q) = \text{Decoder}(\mathbf{h}_q)$ represents the probability that q is classified as either simple or complex.

Notably, unlike approaches that fine-tune LLMs or train specific models with reasoning-related KGs, which might be domain-specific or proprietary, our method leverages generic, publicly available datasets for training. The use of reasoning-irrelevant KGs maintains the generality of our method while making the reasoning-aware module applicable to other approaches that could benefit from the idea of reasoning task classification.

Since the reasoning-aware task requires only an estimation of the hop range rather than a fine-grained perception of reasoning path, a binary classifier is sufficient for most reasoning tasks. Moreover, we also introduce an optional approach to

[1]In this paper, we set $\delta = 2$, as most real-world reasoning tasks of KG-RAG involve paths within 2 hops (2021a).

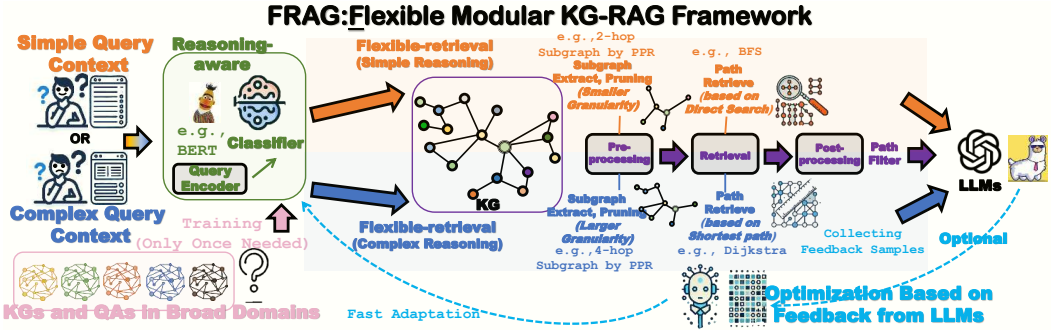

Figure 3: The overall framework of FRAG. 1) The reasoning-aware module classifies incoming queries based on complexity, routing them into either simple or complex reasoning pathways. 2) Subsequently, FRAG leverages tailored *preprocessing–retrieval–postprocessing* pipelines for both simple and complex reasoning, ensuring the retrieval of high-quality and contextually relevant reasoning paths. 3) Finally, the retrieved reasoning paths are provided to the LLM for faithful reasoning, with optional iterative optimization driven by feedback from the LLM, further refining the reasoning-aware module.

further optimize classification performance. This method leverages feedback from large language models (LLMs) to refine query labels, enhancing overall accuracy. Specifically, during the reasoning phase, we prompt LLMs not only to generate standard responses, but also to identify the most relevant reasoning path, from a predefined set of input paths. This approach allows us to derive a more accurate hop count, resulting in a refined label y_{fb} that better represents the complexity of the query. These refined query-label pairs are collected and used to fine-tune the pre-trained binary classifier: $Generate(q', \text{LLM}) \rightarrow answer, y_{fb}$;Classifier = FastAdaptation(Classifier, y_{fb}). Here, q' is the enriched query with reasoning paths, and y_{fb} is the refined label. Such an optimization strategy requires only gathering feedbacks from the LLM, without necessitating additional fine-tuning or calls to LLMs. The prompt used for reasoning is detailed in **Appendix** A.5.

4.3 Flexible-retrieval Module

The retrieval module aims to accurately identify reasoning paths relevant to the query from the KG. To achieve this, we propose a *preprocessing-retrieval-postprocessing* pipeline. The preprocessing step shrinks the retrieval scope, by extracting subgraphs consisting of significant entities and relations from the original KG. During the retrieval step, tailored to the categorization of simple and complex queries, two distinct strategies are applied to search for reasoning paths. In the postprocessing step, redundant reasoning paths are carefully filtered out to prevent the introduction of noise and unnecessary computational cost to the reasoning process. Note that our framework is highly flexible, allowing the use of various method combinations across the three modules of the pipeline. In our implementation, we employ traditional and widely used algorithms known for their effectiveness.

Preprocessing. Given a KG as $G = (V, E)$ and an entity set Ent_q from query q, we extract subgraphs $G_k^s = (V_k^s, E_k^s)$ for each entity $s \in Ent_q$, where each subgraph G_k^s is a k-th order subgraph centered on the entity s. We then take the union of these subgraphs to form the subgraph $G_k = \bigcup_{s \in Ent_q} G_k^s = (V_k, E_k) \subseteq G$. The parameter k, serving as an upper bound on the hop counts among all shortest reasoning paths, is adjusted by the complexity of the queries.

To further prune the subgraph G_k, we remove less relevant entities and edges based on the evaluation of their significance. In the entity-based subgraph pruning, we employ a generalized ranking mechanism (GRM) (e.g., Random Walk with Restart (RWR); Personalized PageRank (PPR); PageRank-Nibble (PRN)), to assess the importance of the entities $v \in V_k$ relative to the query entities Ent_q, and then select the top n entities $\widetilde{V} \subseteq V_k$: $\mathbf{R}(v) = GRM(G_k, Ent_q)$; $\widetilde{V} = \{v_i \mid i \in \text{top-}n(\mathbf{R}(v))\}$, where $\mathbf{R}(v)$ represents the importance score of entity v relative to the query entities Ent_q. The top n entities $\widetilde{V} \subseteq V_k$, along with relations $\widetilde{E} \subseteq E_k$ between $\widetilde{V}$, form a subgraph $\widetilde{G}$. Likewise, in the edge-based subgraph pruning, we apply an edge ranking model (ERM) as a retriever (e.g., BM25; SentenceTransformer), on $\widetilde{G}$ to rank the relations $r \in \widetilde{E}$ based on their semantic similarity to the query q: $\mathbf{S}(r) = ERM(q, \widetilde{E})$; $\widehat{E} = \{r_j \mid j \in \text{top-}m(\mathbf{S}(r))\}$, where $\mathbf{S}(r)$ repre-

sents the similarity score of query q relative to the relations set $\widetilde{E}$ of $\widetilde{G}$. By selecting the top-m relations $\widehat{E} \subseteq \widetilde{E}$ and the corresponding entities $\widehat{V} \subseteq \widetilde{V}$, we construct a more focused subgraph $\widehat{G} = (\widehat{V}, \widehat{E})$.

Retrieval. Upon obtaining the subgraph $\widehat{G}$ in the preprocessing, the retrieval step aims to identify reasoning paths relevant to queries on it. Unlike KGQA tasks, where reasoning paths typically contain the answer (2020; 2022; 2022), in RAG, these paths serve as auxiliary component. They provide value by highlighting intermediate entities and relations that may help supplement missing information in LLMs. However, reasoning paths with excessive intermediate entities and relations introduce redundant information that can burden both retrieval and reasoning. In other words, it introduces a trade-off between acquiring more information and maintaining efficiency. Thus, for simple queries, which typically involve shorter reasoning paths, a broader retrieval approach is essential to minimize information loss. Consequently, the Breadth-First Search (BFS) algorithm is employed, allowing for efficient traversal of all reasoning paths $\mathcal{P}$ between the query entities Ent_q and the entities $\widehat{V}$ within $\widehat{G}$: $\mathcal{P} = \{P_i\} = \text{PathRetrieve}(\widehat{G})$; $P_i = \{s \xrightarrow{r_1} m_1 \xrightarrow{r_2} m_2 \xrightarrow{r_3} \cdots \xrightarrow{r_k} e | s \in Ent_q, m_j \in \widehat{V}\}$.

In contrast, for complex queries with longer reasoning paths, increasing the retrieval paths not only exponentially escalates computational cost but also introduces a large amount of redundant information. This underscores the need for efficiency and pruning. For reasoning paths with the same start and end entities, the shortest path is preferable for directly obtaining answers (2019; 2020; 2023b; 2024) and for reasoning with a shorter prompt. Therefore, we resort to the Dijkstra algorithm to identify efficiently the shortest reasoning paths $\mathcal{P}$ from the query entities Ent_q to entities $\widehat{V}$ within $\widehat{G}$.

Postprocessing. The retrieval process primarily focuses on finding paths from the query entities to potential answer entities, disregarding the semantics of intermediate entities and their relevance to the query. This can lead to an unordered and redundant collection of reasoning paths. Directly incorporating them into the prompt for reasoning lead to several potential issues. First, reasoning paths that contain irrelevant or rare intermediate entities might mislead the reasoning of LLMs. Second, these paths increase the prompt length, which not only adds to the reasoning cost but also risks exceeding the contextual length limit. Last, the reasoning performance is influenced by the placement of these paths within the prompt, with paths positioned at the beginning having a more significant impact (2023; 2024). To address these issues, similar to the previous method, we apply an path ranking model (PRM) (e.g., DPR (2020); ColBERT (2020); BGE (2024)), to rank the reasoning paths $\mathcal{P}$, based on their similarity to the query. Then, the top-u reasoning paths are selected, denoted by $\mathbf{P}$: $\mathbf{T}(p) = PRM(q, \mathcal{P})$; $\mathbf{P} = \{o \mid o \in \text{top-}u(\mathbf{T}(p))\}$, where $\mathbf{T}(p)$ represents the similarity score of query q relative to the reasoning paths $\mathcal{P}$.

This approach effectively filters out a substantial amount of redundant reasoning paths by leveraging the semantic correlation between the query and intermediate entities, thus shortening the prompt length and ensuring that the most relevant and beneficial reasoning paths are favorably positioned.

4.4 Reasoning Module

In the reasoning module, we design a prompt template to augment the question q with the filtered reasoning paths $\mathbf{P}$, forming an enriched prompt q'. This prompt q' guides LLM to conduct reasoning and generate the answer: $q' \leftarrow prompt\,(q, \mathbf{P})$; $answer \leftarrow Generate(q', \text{LLM})$. The reasoning prompt is detailed in **Appendix** A.5.

5 Experiment

5.1 Experimental Design

Datasets and Evaluation Metrics. Our experiments utilize two widely recognized KGQA datasets: WebQSP (2016) and CWQ (2018b), both extensively used in the KGQA and KG-RAG research communities (2023b; 2023b; 2024b; 2024; 2024a; 2024). Table 2 summarizes the distribution of question hops across these datasets, showing that both predominantly feature simple queries. Specifically, WebQSP consists entirely of simple queries, while CWQ includes a small fraction (20.75%) of complex ones. Further dataset details are provided in **Appendix** A.2. Following prior works (2023; 2023; 2024; 2023; 2024; 2024), we employ Hits@1 score as the evaluation metric, assessing the percentage of correct answers ranked first by LLM.

Reasoning LLMs. We evaluate the performance of our approach using six LLMs, which are referenced in this paper as follows: Llama-2-7b-chat-hf (Llama2-7B), Llama-2-70b-chat-hf (Llama2-

Table 1: Performance Comparison with Different Baselines on WebQSP and CWQ

Type	Methods	WebQSP	CWQ	Type	Methods	WebQSP	CWQ
Traditional KGQA	***Without LLMs***			Modular KG-RAG	***Llama-2-7b-chat-hf***		
	KV-Mem (2016)	46.7	21.1		Vanilla LLM	63.4	31.1
	GraftNet (2018)	66.4	36.8		ToG (2024)	10.8	5.2
	PullNet (2019)	68.1	45.9		FRAG (Ours)	76.6	47.3
	EmbedKGQA (2020)	66.6	45.9		FRAG-F (Ours)	**76.7**	**48.9**
	QGG (2020)	73.0	44.1		***Llama-2-70b-chat-hf***		
	NSM (2021a)	68.7	47.6		Vanilla LLM	63.6	37.6
	TransferNet (2021)	71.4	48.6		ToG (2024)	68.9	57.6
	KGT5 (2022)	56.1	36.5		FRAG (Ours)	81.2	60.1
	SR+NSM (2022)	68.9	50.2		FRAG-F (Ours)	**81.3**	**62.2**
	SR+NSM+E2E (2022)	69.5	49.3		***Llama-3-8B-Instruct***		
	HGNet (2022)	70.6	**65.3**		Vanilla LLM	64.0	37.9
	Program Transfer (2022)	74.6	58.1		ToG (2024)	59.8	37.0
	UniKGQA (2022)	**77.2**	51.2		FRAG (Ours)	87.7	64.9
Coupled KG-RAG	***Llama-2-7b-chat-hf (Finetuned)***				FRAG-F (Ours)	**87.8**	**66.1**
	GNN-RAG (2024)	80.6	61.7		***Llama-3-70B-Instruct***		
	RoG (2024b)	**85.7**	**62.6**		Vanilla LLM	73.1	46.1
Modular KG-RAG	***ChatGPT***				FRAG (Ours)	**88.6**	69.4
	Vanilla LLM	66.8	39.9		FRAG-F (Ours)	**88.6**	**70.8**
	KD-CoT (2023b)	73.7	50.5		***GPT-4o-mini***		
	ToG (2024)	76.2	57.1		Vanilla LLM	69.2	43.8
	FRAG (Ours)	**82.3**	61.0		FRAG (Ours)	**86.7**	66.9
	FRAG-F (Ours)	**82.3**	**61.5**		FRAG-F (Ours)	**86.7**	**68.0**

Table 2: Statistics of Question Hops of Datasets

Dataset	1 hop	2 hop	$\geq$ 3 hop
WebQSP	65.49 %	34.51%	0.00%
CWQ	40.91 %	38.34%	20.75%

70B) (2023), Llama-3-8B-Instruct (Llama3-8B), Llama-3-70B-Instruct (Llama3-70B) (2024), GPT-3.5-turbo (ChatGPT) and GPT-4o-mini.

Baselines. Given that FRAG is a modular KG-RAG approach, we primarily compare it with other modular KG-RAG methods, as well as with coupled KG-RAG and traditional KGQA techniques. Among them, ***1) Modular KG-RAG:*** Vanilla LLMs (without RAG), KD-CoT (2023b) and ToG (2024). KD-CoT enhances CoT prompting by integrating KG knowledge. ToG leverages LLMs to iteratively select the most pertinent relations and entities, representing the current SOTA in modular KG-RAG methods. ***2) Coupled KG-RAG:*** GNN-RAG (2024) and RoG (2024b). GNN-RAG leverages GNN to extract useful reasoning paths. RoG utilizes a fine-tuned LLM to generate relation paths for answering questions, representing the SOTA in coupled KG-RAG. ***3) Traditional KGQA:*** 13 traditional KGQA methods, as described in **Appendix** A.3.

Experiment Implementations. ***1) For the reasoning-aware module***, we employ DeBERTaV3 (2021b) as the query encoder and decoder. We construct training datasets for simple and complex reasoning tasks using two large and cross-domain KG databases, Freebase (2008) and Wiki-Movies (2018), and ensuring complete isolation between the training data and the two test datasets. ***2) For the flexible-retrieval module***, we utilize bge-reranker-v2-m3 (2024) as the ERM and PRM during both preprocessing and postprocessing stages. In the preprocessing stage, we use PPR algorithm as GRM, and the hyperparameters as follows: $k = 2$, $n = 2000$, $m = 64$ for simple queries, and $k = 4$, $n = 2000$, $m = 64$ for complex queries. In the postprocessing stage, we set $u = 32$ across all experiments. ***3) For LLMs' reasoning***, we use zero-shot prompting to LLMs generation. ***4) Feedback adjustment is optional***, with the adjustment rate is set to 0.25, indicating that 25% of the samples are selected for fast adaptation of the reasoning-aware module. Detailed settings are provided in **Appendix** A.4.1.

5.2 Results

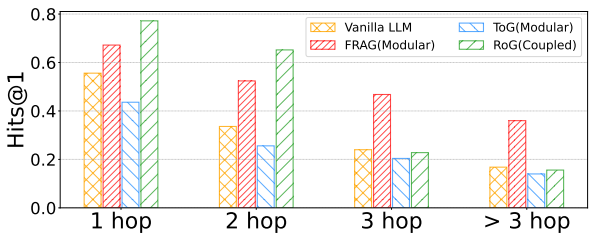

Figure 4: Performance of Different Hops (Llama2-7B)

Table 1 presents a detailed performance comparison of our method against various baselines, while

Figure 4 further illustrate detailed evaluations of sampled queries across different hops, both demonstrating the effectiveness of FRAG. Additionally, Table 3 compare the cost of FRAG with ToG and RoG in terms of training and retrieval operations. The following analysis will highlight our method's advantages in both effectiveness and efficiency.

FRAG Achieves New SOTA as a Modular KG-RAG. As shown in Table 1, our method consistently outperforms the previous SOTA, ToG, across various settings, establishing itself as the most advanced modular KG-RAG method. For example, on the Llama-3-8B-Instruct model, FRAG improves the scores from 59.8 and 37.0 to 87.7 and 64.9, respectively, achieving a 46.7% and 75.4% improvement on two datasets compared to ToG. Furthermore, incorporating feedback in FRAG-F further elevates its performance to 81.3 and 62.2. Notably, ToG exhibits performance degradation on small scales (7B and 8B) LLM bases, such as Llama-2-7b-chat-hf, where the scores on the two datasets drop from 63.4 and 31.1 to 10.8 and 5.2, respectively, compared to the vanilla LLM. In contrast, our method reliably enhances output quality through KG RAG across all LLMs. Additionally, Figure 4 further demonstrates the robust performance of FRAG, which delivers significant improvements over both ToG and vanilla LLMs in varied hop numbers.

Moreover, ToG iteratively selects the most relevant relations and entities, resulting in a significantly higher average number of LLM calls compared to FRAG's zero LLM calls. This underscores the efficiency of our approach, which delivers superior performance with minimal overhead.

FRAG Achieves Comparable Performance with Lower Training Costs. Unlike traditional KGQA methods that rely on specific models to embed KG semantic information, FRAG leverages pretrained LLMs with generalizable retrieval and filtering modules, achieving superior performance with minimal effort. Even with the small-parameter Llama-3-8B-Instruct, our method outperforms all traditional KGQA methods. Compared to the coupled KG-RAG approach, like advanced RoG, FRAG delivers similar performance while drastically reducing training and fine-tuning time. As shown in Table 3, RoG requires 38 hours to fine-tune a 7B LLM, whereas FRAG only considers the structure knowledge of KG in just 306 seconds of training, but reaches approximately 89% of RoG's performance on WebQSP. Moreover,

Table 3: Training and Retrieval Cost Comparison

Method	Training Cost (Time)			Retrieval Cost (Ave. LLM Calls)
	Reasoning-aware		Fine-tune	
	Training	Feedback		
ToG	-	-	-	13.3
RoG	-	-	38h	3
FRAG	306s[2]	-	-	0
FRAG-F	-	7.58s	-	0

Table 4: Ablation Study Results on Two Datasets

Method	Llama2		Llama3		GPT	
	7B	70B	8B	70B	3.5-turbo	4o-mini
	CWQ					
FRAG-F	**48.9**	**62.2**	**66.1**	**70.8**	**61.5**	**68.0**
FRAG	47.3	60.1	64.9	69.4	61.0	66.9
FRAG-Simple	47.1	60.8	63.2	68.6	59.5	67.4
FRAG-Complex	47.0	59.5	62.5	66.6	59.7	64.9
Vanilla LLM	31.1	37.6	37.9	46.1	39.9	43.8
	WebQSP					
FRAG-F	76.7	**81.3**	**87.8**	**88.6**	**82.3**	**86.7**
FRAG	76.6	81.2	87.7	**88.6**	**82.3**	**86.7**
FRAG-Simple	**76.8**	**81.3**	87.7	**88.6**	**82.3**	**86.7**
FRAG-Complex	72.0	76.4	80.4	81.7	76.2	81.5
Vanilla LLM	63.4	63.6	64.0	73.1	66.8	69.2

FRAG's plug-and-play nature allows seamless enhancement with larger-scale LLMs like Llama-3-70B-Instruct, achieving scores of 88.6 and 69.4 on two datasets. This kind of enhancement would be challenging for RoG due to the significantly higher fine-tuning costs. Additionally, Figure 4 highlights FRAG's superior performance in complex reasoning scenarios involving more than two hops, outperforming RoG in these tasks.

5.3 Ablation Study

We conduct ablation experiments to compare our method with those using only the simple reasoning pipeline (FRAG-Simple) or only the complex reasoning pipeline (FRAG-Complex). As shown in Table 4, without the reasoning-aware module that routes simple and complex queries through distinct pipelines, performance slightly declines when a single reasoning approach is applied to all questions. This is partly due to the limited presence of complex queries in the CWQ dataset (20.75%) and their absence in the WebQSP dataset. Notably, although the WebQSP dataset does not contain complex queries, FRAG did not lead to a significant performance decline compared to FRAG-Simple. This further substantiates that our method can intelligently allocate the appropriate pipeline for each

[2]Only once for the Reasoning-aware module training

type of question, ensuring optimal performance. To further explore the impact of our proposed modules under more balanced conditions, we conducts additional ablation experiments on datasets with a more equal distribution of simple and complex queries. These results are provided in the **Appendix** A.4.2.

6 Conclusion

In this paper, we propose FRAG, a modular KG-RAG framework that addresses the challenge of enhancing reasoning accuracy in LLMs without compromising flexibility. By adapting the retrieval process based on the complexity of the query context, FRAG leverages structural information predictions to optimize retrieval strategies. Thus, FRAG comprises two key modules: the reasoning-aware module predicts the complexity of the reasoning tasks (simple or complex) based solely on the query context, while the flexible-retrieval module customizes the retrieval process according to task complexity to enhance retrieval efficiency and effectiveness. Extensive experiments show that FRAG improves retrieval quality while maintaining flexibility, outperforming existing KG-RAG approaches. In the future, we aim to further enhance FRAG's adaptability to more diverse knowledge graph structures and complex reasoning scenarios.

Limitations

FRAG represents a highly flexible framework, yet its potential and adaptability remain underexplored in the current experiments, which primarily rely on widely adopted algorithms and models. For instance, the reasoning-aware module employs DeBERTaV3, the generalized ranking mechanism utilizes PPR, and both the edge and path ranking models adopt bge-reranker-v2-m3. In diverse scenarios, these modules can be substituted with alternative implementations, such as BM25 or large language models (LLMs), to achieve more effective trade-offs between performance and computational efficiency. This adaptability highlights FRAG's capacity to be tailored to varying requirements across different applications.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62472400, Grant 62072428, Grant 62271465, and in part by the Suzhou Basic Research Program under Grant SYG202338. Xike Xie is the corresponding author.

References

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *Preprint*, arXiv:2306.04136.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8128–8140, Dublin, Ireland. Association for Computational Linguistics.

Yukun Cao, Zengyi Gao, Zhiyang Li, Xike Xie, Kevin Zhou, and Jianliang Xu. 2024. Lego-graphrag: Modularizing graph-based retrieval-augmented generation for design space exploration. *Preprint*, arXiv:2411.05844.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Yongrui Chen, Huiying Li, Guilin Qi, Tianxing Wu, and Tenggou Wang. 2022. Outlining and filling: Hierarchical query graph generation for answering complex questions over knowledge graphs. *Preprint*, arXiv:2111.00732.

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. UHop: An Unrestricted-Hop Relation Extraction Framework for Knowledge-Based Question Answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 345–356, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

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

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022a. Precise Zero-Shot Dense Retrieval without Relevance Labels. *Preprint*, arXiv:2212.10496.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022b. Precise zero-shot dense retrieval without relevance labels. *Preprint*, arXiv:2212.10496.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *Preprint*, arXiv:2002.08909.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021a. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 553–561, New York, NY, USA. Association for Computing Machinery.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021b. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *Preprint*, arXiv:2111.09543.

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

Ruixin Hong, Hongming Zhang, Hong Zhao, Dong Yu, and Changshui Zhang. 2023. Faithful question answering with monte-carlo planning. *ACL2023*.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. GRAG: Graph Retrieval-Augmented Generation. *Preprint*, arXiv:2405.16506.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *Preprint*, arXiv:2305.09645.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. 2024. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. *Preprint*, arXiv:2402.11163.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2022. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *Preprint*, arXiv:2004.12832.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6966–6980. Association for Computational Linguistics.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024. Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources. *Preprint*, arXiv:2305.13269.

Haochen Liu, Song Wang, Yaochen Zhu, Yushun Dong, and Jundong Li. 2024. Knowledge Graph-Enhanced Large Language Models via Path Selection. *Preprint*, arXiv:2406.13862.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Luu Anh Tuan. 2024a. ChatKBQA: A Generate-then-Retrieve Framework for Knowledge Base Question Answering with Fine-tuned Large Language Models. *Preprint*, arXiv:2310.08975.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024b. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. *Preprint*, arXiv:2310.01061.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, and Jian Guo. 2024. Think-on-Graph 2.0: Deep and Interpretable Large Language Model Reasoning with Knowledge Graph-guided Retrieval. *Preprint*, arXiv:2407.10805.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query Rewriting for Retrieval-Augmented Large Language Models. *Preprint*, arXiv:2305.14283.

Costas Mavromatis and George Karypis. 2024. GNN-RAG: Graph Neural Retrieval for Large Language Model Reasoning. *Preprint*, arXiv:2405.20139.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Preprint*, arXiv:1301.3781.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *Preprint*, arXiv:2305.14251.

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

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*.

Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. 2024. Large language model based long-tail query rewriting in taobao search. *Preprint*, arXiv:2311.03758.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. Kilt: a benchmark for knowledge intensive language tasks. *Preprint*, arXiv:2009.02252.

Deevashwer Rathee, Dacheng Li, Ion Stoica, Hao Zhang, and Raluca Popa. 2024. Mpc-minimized secure llm inference. *Preprint*, arXiv:2408.03561.

Bhaskarjit Sarmah, Benika Hall, Rohan Rao, Sunil Patel, Stefano Pasquali, and Dhagash Mehta. 2024. Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. *Preprint*, arXiv:2408.04948.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828, Dublin, Ireland. Association for Computational Linguistics.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507, Online. Association for Computational Linguistics.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023a. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *Preprint*, arXiv:2305.15294.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023b. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *Preprint*, arXiv:2305.15294.

Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. Transfernet: An effective and transparent framework for multi-hop question answering over relation graph. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4149–4158.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2380–2390.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. *Preprint*, arXiv:2307.07697.

Alon Talmor and Jonathan Berant. 2018a. The web as a knowledge-base for answering complex questions. *Preprint*, arXiv:1803.06643.

Alon Talmor and Jonathan Berant. 2018b. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *Preprint*, arXiv:1811.00937.

Llama team. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Llama team. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023a. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint arXiv:2308.13259*.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023b. Knowledge-Driven CoT: Exploring Faithful Reasoning in LLMs for Knowledge-intensive Question Answering. *Preprint*, arXiv:2308.13259.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648, Baltimore, Maryland. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2023a. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. *Preprint*, arXiv:2210.00063.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2023b. DecAF: Joint Decoding of Answers and Logical Forms for Question Answering over Knowledge Bases. *Preprint*, arXiv:2210.00063.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023c. Generate rather than retrieve: Large language models are strong context generators. *Preprint*, arXiv:2209.10063.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023d. Generate rather than retrieve: Large language models are strong context generators. *Preprint*, arXiv:2209.10063.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *AAAI*.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models. *Preprint*, arXiv:2310.06117.

A Appendix

A.1 Algorithm for FRAG

Algorithm 1: FRAG Framework

Data: Knowledge graph $G = (V, E)$, Dataset $\mathcal{D}$, Feedback flag $flag$, adjustment rate $ratio$;
Result: Reasoning result A;

$N \leftarrow$ size of $\mathcal{D}$;
$i \leftarrow 0$;
Initialize list of answers $\mathcal{A} \leftarrow [\,]$;
E=extractEntities(q);
foreach *query* q *in Dataset* $\mathcal{D}$ **do**
  Classifier(q);
  $G_k \leftarrow$ SubgraphExtract(G, q);
  if q *is simple query* **then**
    $\widetilde{G} \leftarrow GRM_s\,(G_k, E, n_s, k_s)$;
    $\widehat{G} \leftarrow ERM_s\,(\widetilde{G}, q, m_s)$;
    $\mathcal{P} \leftarrow PathRetrieve_s\,(\widehat{G})$;
    $P \leftarrow PRM_s\,(\mathcal{P}, u_s)$;
  else if q *is complex query* **then**
    $\widetilde{G} \leftarrow GRM_c\,(G, E, n_c, k_c)$;
    $\widehat{G} \leftarrow ERM_c\,(\widetilde{G}, q, m_c)$;
    $\mathcal{P} \leftarrow PathRetrieve_c\,(\widehat{G})$;
    $P \leftarrow PRM_c(\mathcal{P}, u_c)$;
  $ans, y_{fb} \leftarrow$ Generate(LLM, concat(P, q));
  if $flag$ *is True* ***and*** $i < ratio \cdot N$ **then**
    FastAdaptation(Classifier, q, y_{fb});
  $i \leftarrow i + 1$;
  Add ans to $\mathcal{A}$;
return A;

Algorithm 2: Retrieval Algorithm (FRAG-Simple)

Data: Knowledge Graph G, Start node s
Result: All paths $\mathcal{P}$ from s to every other node in G

Initialize queue $Q \leftarrow \{(s, [s])\}$;
Initialize list of paths $\mathcal{P} \leftarrow [\,]$;
while Q *is not empty* **do**
  Dequeue the first element $(v, path)$ from Q;
  foreach *neighbor* u *of* v **do**
    if $u \notin path$ **then**
      Enqueue $(u, path + [u])$ to Q;
      Add $path + [u]$ to $\mathcal{P}$;
    end
  end
end
return $\mathcal{P}$;

A.2 Datasets

We adopt two benchmark KGQA datasets: WebQuestionSP (WebQSP) (Yih et al., 2016) and Complex WebQuestions (CWQ) (Talmor and Berant, 2018b). We follow previous works (Sun et al., 2024; Luo et al., 2024b; Mavromatis and Karypis, 2024) to use the same train and test splits for fair comparison. The statistics of the datasets are shown in Table 5. The distribution of the answer numbers is shown in Table 6.

To ensure a rigorous and balanced evaluation of our proposed method, we construct additional datasets with an equal distribution of simple and complex queries. Specifically, we randomly sample 1,000 instances from the CWQ and WebQSP datasets to form the KG-1000 subsets, where the proportion of queries for each hop was meticulously balanced. This carefully designed sampling strategy allows us to evaluate the effectiveness of the reasoning-aware module under more balanced conditions, thus providing a more precise assessment of its impact on performance across varying query types.

A.3 Baselines

Below, we introduce 13 traditional KGQA methods in order of publication, as they were not covered in detail earlier.

- KV-Mem (Miller et al., 2016) is a key-value structured memory network to retrieve answers from KGs.
- GraftNet (Sun et al., 2018) is a graph convolution-based neural network that reasons over KGs.
- PullNet (Sun et al., 2019) extends GraftNet by iteratively constructing a question-specific subgraph to facilitate reasoning.
- EmbedKGQA (Saxena et al., 2020) models the reasoning on KGs through the embeddings of entities and relations.
- QGG (Lan and Jiang, 2020) proposes a segmented query graph generation method that

Algorithm 3: Retrieval Algorithm (FRAG-Complex)

Data: Knowledge Graph $G = (V, E)$, Start node s $(s \in V)$
Result: Set of shortest paths $\mathcal{P}$ from s to all other nodes in G

Initialize distances: $d[v] \leftarrow \infty$ for all $v \in V$, $d[s] \leftarrow 0$;
Initialize priority queue Q as a min-heap;
Insert s into Q with priority $d[s]$;
Initialize predecessor array $pred[v] \leftarrow$ null for all $v \in V$;
while Q *is not empty* **do**
    Extract node u from the top of Q;
    foreach *neighbor* v *of* u **do**
        if $d[u] + 1 < d[v]$ **then**
            $d[v] \leftarrow d[u] + 1$;
            $pred[v] \leftarrow u$;
            if v *is not in* Q **then**
                Insert v into Q with priority $d[v]$;
            end
            else
                Update priority of v in Q to $d[v]$;
            end
        end
    end
end
Initialize shortest paths set $\mathcal{P} \leftarrow \{\}$;
foreach *node* $t \in V \setminus \{s\}$ **do**
    Initialize path list $path \leftarrow [\,]$;
    $u \leftarrow t$;
    while $u \neq$ *null* **do**
        Prepend u to $path$;
        $u \leftarrow pred[u]$;
    end
    Add $path$ to $\mathcal{P}$;
end
return $\mathcal{P}$;

Table 5: Statistics of datasets.

Datasets	Train	Test	Max hop
WebQSP	2,826	1,628	2
CWQ	27,639	3,531	4

flexibly generates query graphs by simultaneously incorporating constraints and extending relationship paths.

- NSM (He et al., 2021a) introduces a teacher-student framework to simulate the multi-hop reasoning process.
- TransferNet (Shi et al., 2021) implements a graph neural network to effectively capture the relationship between entities and questions, enabling reasoning within a unified framework that handles both label and text relations.
- KGT5 (Saxena et al., 2022) leverages a fine-tuned sequence-to-sequence model on knowledge graphs to generate answers directly from the input question.
- SR+NSM(Zhang et al., 2022) introduces a method for multi-hop reasoning that retrieves relevant subgraphs through a relation-path retrieval mechanism.
- SR+NSM+E2E(Zhang et al., 2022) enhances SR+NSM by employing an end-to-end approach that jointly optimizes both the retrieval and reasoning components.
- HGNet (Chen et al., 2022) introduces a hierarchical approach for generating query graphs, which includes an initial outlining stage to establish structural constraints, followed by a filling stage focused on selecting appropriate instances.
- Program Transfer (Cao et al., 2022) presents a two-stage parsing framework for complex KGQA, utilizing an ontology-guided pruning strategy.
- UniKGQA (Jiang et al., 2022) unifies retrieval and reasoning using a single retriever-reader model.

Table 6: Statistics of the Number of Answers

Dataset	#Ans = 1	2–4	5–9	≥ 10
WebQSP	51.2%	27.4%	8.3%	12.1%
CWQ	70.6%	19.4%	6%	4%

Table 7: Statistics of Sample Size of KG-1000

Datasets	1-hop	2-hop	3-hop	4-hop
KG-1000	250	250	250	250

A.4 Experiment Detail

A.4.1 Detailed Experimental Settings.

All experiments are running on Ubuntu 20.04.6 LTS (Intel(R) Xeon(R) Platinum 8358 CPU@2.60GHz Processor, 4 A100-80G, 400GB memory). The detailed experimental settings are as follows: ***1) Reasoning-aware module.*** The reasoning-aware module underwent training on the designated datasets for 10 epochs with a batch size of 32. The learning rate is set to 1e-5, and the weight decay parameter is set to 0. ***2) Retrieval.*** The damping factor α of PPR algorithm is set to 0.8, and the maximum iteration is set to 1000. ***3) Reasoning.*** The temperature parameter is set to 0.01, and the maximum token length for generation is fixed at 256. We use zero-shot reasoning prompt across all datasets, and the prompt templates are presented in **Appendix** A.5. ***4) Feedback Adjustment.*** After collecting the feedback datasets, the reasoning-aware module is trained for 3 epochs with the batch size of 16. The learning rate is fixed at 1e-4, and the weight decay parameter is set to 0.

A.4.2 Ablation Study on KG-1000.

Table 8: Ablation Study Results on KG-1000

Method	Llama2-7B	Llama3-8B
FRAG	**50.6**	**63.2**
FRAG-Simple	40.3	49.6
FRAG-Complex	46.9	59.8
Vanilla LLM	32.6	38.4

We conduct an ablation study on the KG-1000 dataset to evaluate the effectiveness of the reasoning-aware module with a more equal distribution of simple and complex queries. The experimental results presented in Table 8 clearly demonstrate the critical role of the reasoning-aware module in our proposed framework. Under balanced conditions, FRAG significantly outperforms all baseline variants across both the Llama2-7B and Llama3-8B configurations. Specifically, FRAG achieves performance gains of 10.3 percentage points over the FRAG-Simple baseline for Llama2-7B (50.6 vs. 40.3) and 13.6 percentage points for Llama3-8B (63.2 vs. 49.6). These considerable margins underscore the effectiveness of integrating the reasoning-aware module within the FRAG framework.

A.5 Prompts

The zero-shot reasoning prompt for the reasoning-aware module is as follows:

> **Prompt:** You are an expert reasoner with a deep understanding of logical connections and relationships. Your task is to analyze the given reasoning paths and provide accurate reasoning path to the questions based on these paths. Based on the reasoning paths, please extract the correct reasoning path. If NO correct reasoning path, please just reply NO.
> Reasoning Paths: {paths}
> Question: {question}
> Correct reasoning path:

The zero-shot reasoning prompt for the reasoning module is as follows:

> **Prompt:** You are an expert reasoner with a deep understanding of logical connections and relationships. Your task is to analyze the given reasoning paths and provide clear and accurate answers to the questions based on these paths. Based on the reasoning paths, please answer the given question.
> Reasoning Paths: {paths}
> Question: {question}