KARPA: A Training-free Method of Adapting Knowledge Graph as References for Large Language Model's Reasoning Path Aggregation

Siyuan Fang[1], **Kaijing Ma**[2], **Tianyu Zheng**[3], **Xinrun Du**[3], **Ningxuan Lu**[4], **Ge Zhang**[3], **Qingkun Tang**[5*]

[1]Beijing University of Posts and Telecommunications, [2]Tongji University, [3]Multimodal Art Projection Research Community, [4]Duke University, [5]ZTE Corporation

syfang@bupt.edu.cn, tang.qingkun@zte.com.cn

Abstract

Large language models (LLMs) demonstrate exceptional performance across a variety of tasks, yet they are often affected by hallucinations and the timeliness of knowledge. Leveraging knowledge graphs (KGs) as external knowledge sources has emerged as a viable solution, but existing methods for LLM-based knowledge graph question answering (KGQA) are often limited by step-by-step decision-making on KGs, restricting the global planning and reasoning capabilities of LLMs, or they require fine-tuning or pre-training on specific KGs. To address these challenges, we propose **Knowledge graph Assisted Reasoning Path Aggregation (KARPA)**, a novel framework that harnesses the global planning abilities of LLMs for efficient and accurate KG reasoning. KARPA operates in three steps: pre-planning relation paths using the LLM's global planning capabilities, matching semantically relevant paths via an embedding model, and reasoning over these paths to generate answers. Unlike existing KGQA methods, KARPA avoids stepwise traversal, requires no additional training, and is adaptable to various LLM architectures. Extensive experimental results show that KARPA achieves state-of-the-art performance in KGQA tasks, delivering both high efficiency and accuracy. Our code is available on GitHub.[1]

1 Introduction

In recent years, large language models (LLMs) (Touvron et al., 2023a,b; Achiam et al., 2023; Bai et al., 2023) have revolutionized natural language processing, demonstrating impressive performance in areas such as information extraction (Xu et al., 2023), summarization (Jin et al., 2024), and question answering (Louis et al., 2024). However, despite these advancements, LLMs face notable challenges, particularly in maintaining up-to-date knowledge, domain-specific knowledge (Zhang et al., 2024), and dealing with hallucinations (Zhang et al., 2023; Huang et al., 2023) where LLMs produce incorrect or nonsensical outputs.

Knowledge graphs (KGs) enhance the reasoning capabilities of LLMs by providing structured, reliable external knowledge (Zhu et al., 2024; Pan et al., 2024). Existing approaches to integrating LLMs with KGs fall into two categories: (1) Direct interaction between LLMs and KGs, where the LLM explores the KG step-by-step (Sun et al., 2023; Jiang et al., 2023), often relying on local search strategies like beam search. These methods can produce suboptimal answers by overlooking the LLM's global planning and reasoning potential. Additionally, they require numerous interactions between LLMs and KGs, as shown in Figure 1(b). (2) Training-based methods, such as reasoning on graphs (RoG) (Luo et al., 2023), generate retrieval information for KGQA. However, they often require fine-tuning or pre-training on specific KG data (Li et al., 2023b; Huang et al., 2024). These methods struggle with unseen KGs, necessitate retraining, and are prone to hallucinations during information generation, as illustrated in Figure 1(a).

To address these limitations, we propose **Knowledge graph Assisted Reasoning Path Aggregation (KARPA)**, an innovative framework that leverages the global planning capabilities of LLMs alongside semantic embedding models for efficient and accurate KG reasoning. Our approach consists of three key steps: pre-planning, matching, and reasoning, as shown in Figure 2. In the pre-planning phase, KARPA enables the LLM to generate initial relation paths for the provided question using LLM's inherent reasoning and planning capabilities. With these inital relation paths, KARPA employs a semantic embedding model (Ruder et al., 2019) to identify candidate relations that are semantically similar to the relations within the initial paths. The LLM can then create coherent relation paths that logically connect the topic entity

*Corresponding Author.

[1]https://github.com/Icamd/KARPA

Findings of the Association for Computational Linguistics: ACL 2025, pages 24724–24746
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

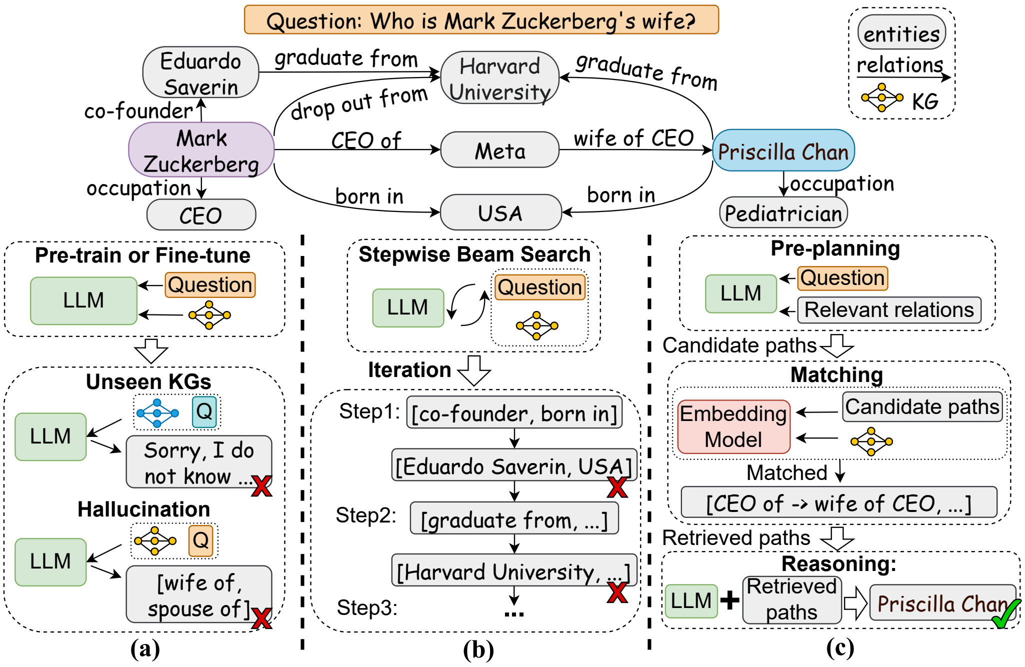

Figure 1: Comparison of different LLM-based KGQA methods: (a) Pre-training or fine-tuning the LLM on specific KG data. (b) Direct reasoning over KGs via stepwise interactions with the LLM. (c) Our KARPA framework, which combines the global planning and reasoning abilities of LLMs with embedding-based matching techniques. KARPA enables comprehensive path planning while avoiding local optima and hallucinations observed in previous methods.

to potential answer entities using these candidate relations. During the matching phase, KARPA employs an embedding model to identify relation paths within the KG that exhibit the highest similarity to the paths generated by the LLM in the pre-planning phase. This avoids locally optimal issues encountered in previous methods. Finally, during the reasoning step, the matched relation paths and their corresponding entities are provided to the LLM to formulate final answers. The detail of our framework is shown in Figure 2.

KARPA offers several key advantages over existing LLM-based KGQA methods: (1) KARPA fully leverages the global planning and reasoning capabilities of LLMs to generate logically coherent paths, which aligns better with human-like reasoning processes. Unlike methods limited to adjacent relations or requiring iterative traversal within the KG, KARPA selects from all potential relations within the KG, significantly reducing interactions between LLMs and KGs. (2) Our embedding-based matching strategy avoids the locally optimal solution that arises from the stepwise interactions between LLMs and KGs, ensuring more effective exploration of the KGs. (3) KARPA is training-free, making it adaptable to various LLMs while enhancing reasoning capabilities with techniques such as chain-of-thought (CoT) (Wei et al., 2022). Our contributions can be summarized as follows:

- We propose KARPA, which combines the global planning and reasoning capabilities of LLMs with embedding models to improve both accuracy and efficiency of KGQA tasks.
- By enabling LLMs to generate initial relation paths across all potential relations within the KG and integrating a semantic embedding model for path matching, KARPA mitigates the risk of local optima and minimizes interactions with KGs. Techniques such as CoT can also be incorporated to further enhance the LLM's reasoning abilities over KGs.
- KARPA operates in a training-free manner and is compatible with various LLMs, providing a plug-and-play solution that achieves state-of-the-art performance on several KGQA benchmark datasets.

2 Related Work

Prompt-Based Reasoning with LLMs. LLMs such as LLaMA (Touvron et al., 2023a,b), Qwen (Bai et al., 2023), and GPT-4 (Achiam et al., 2023)have advanced reasoning by leveraging extensive internal knowledge. Various prompt-based methods further enhance these capabilities. For instance, Chain-of-Thought (CoT) (Wei et al., 2022) improves reasoning by decomposing complex tasks into manageable steps, excelling in domains like mathematical reasoning (Jie et al., 2023) and logical inference (Zhao et al., 2023). Variants such as Auto-CoT (Zhang et al., 2022), Zero-Shot-CoT (Kojima et al., 2022), and Complex-CoT (Fu et al., 2022) further optimize this approach. Frameworks like Tree of Thoughts (ToT) (Yao et al., 2024) and Graph of Thoughts (GoT) (Besta et al., 2024) have expanded the scope of LLM reasoning. Lately, OpenAI o1 series models represent a significant advancement in LLM reasoning. These methods underscore the role of tailored prompts in maximizing LLM reasoning potential.

LLM-Based KGQA. Integrating KGs with LLMs enhances reasoning and mitigates hallucinations. Unlike methods such as CoT that rely solely on the internal knowledge of LLMs, incorporating KGs provides access to structured external knowledge (He et al., 2022; Wang et al., 2023). Approaches like Think-on-Graph (ToG) (Sun et al., 2023), Interactive-KBQA (Xiong et al., 2024) and StructGPT (Jiang et al., 2023) enable stepwise interactions between LLMs and KGs. Methods such as Reasoning on Graphs (RoG) (Luo et al., 2023), chain of knowledge (Li et al., 2023c) and other techniques (Huang et al., 2024; Pan et al., 2024; Li et al., 2023b) utilize pre-trained or fine-tuned LLMs to generate retrieval information for KGQA. Furthermore, methods like UniKGQA (Jiang et al., 2022) and KG-CoT (Zhao et al., 2024) require training specific models for KG information retrieval, further complicating their implementation.

3 Preliminary

Knowledge Graphs (KGs). KGs represent structured information as $G = (E, R)$, where E is the set of entities and R denotes the set of relations. Each relation $r \in R$ connects two entities (e_i, e_j), with $e_i, e_j \in E$.

Relation Paths and Reasoning Paths. A relation path P connects a topic entity e_t to an answer entity e_a via a sequence of relations: $P = (r_1, r_2, \ldots, r_n)$, where $r_i \in R$. Reasoning paths further include intermediate entities along the path, represented as $P_r = \left\{e_t \xrightarrow{r_1} e_1 \xrightarrow{r_2} \ldots \xrightarrow{r_n} e_a\right\}$.

Knowledge Graph Question Answering (KGQA). KGQA aims to answer questions using information from KGs. Given a query Q, the goal of KGQA is to generate an answer A using a function f: $A = f(Q, G)$, where f extracts the answer from the KG G based on Q.

Embedding Models and Semantic Similarity. Embedding models represent text in a continuous vector space, enabling semantic similarity measurements. A function $\Phi : R \to \mathbb{R}^d$ maps a sentence R to a d-dimensional vector. Similarity between embeddings is computed using cosine similarity:

$$sim(r_i, r_j) = \frac{\Phi(r_i) \cdot \Phi(r_j)}{\|\Phi(r_i)\| \|\Phi(r_j)\|}, \tag{1}$$

where $\cdot$ is the dot product and $\|\cdot\|$ is the Euclidean norm. This metric aids in comparing semantic information for retrieval tasks.

4 Approach

In this section, we present KARPA, a framework that leverages the strengths of LLMs and embedding models to enhance KGQA. Our approach is composed of three key steps: pre-planning, matching, and reasoning, as illustrated in Figure 2.

4.1 Pre-Planning with LLM

The pre-planning phase leverages the global planning capabilities of LLMs to generate initial paths $P_{initial}$ and candidate paths P_{cand}. This phase initiates the reasoning process by allowing the LLM to analyze the input question Q and the associated topic entity e_t. By leveraging the reasoning capability of LLM, KARPA is able to propose paths that are not only logically coherent but also have the potential to lead to the answer entities E_a.

Initial Planning Using LLM KARPA starts by using the LLM to generate a set of initial relation paths based on the provided question Q, as shown in Figure 2. The LLM outputs a set of potential relation paths P as follows:

$$\begin{aligned} P &= \{p_1, p_2, \ldots, p_m\}, \\ \text{where } p_i &= (r_1^i, r_2^i, \ldots, r_{n_i}^i). \end{aligned} \tag{2}$$

Here, each p_i is a path of n_i relations r_j^i that could logically connect a topic entity e_t to the potential answer entity e_a. The relations within these paths serve as candidates for relations extraction.

Relation Extraction Strategy With the initial relation paths P, we decompose each path p_i into its constituent relations $R_i = \{r_1^i, r_2^i, \ldots, r_{n_i}^i\}$. For each relation $r_j^i \in R_i$, we utilize an embedding

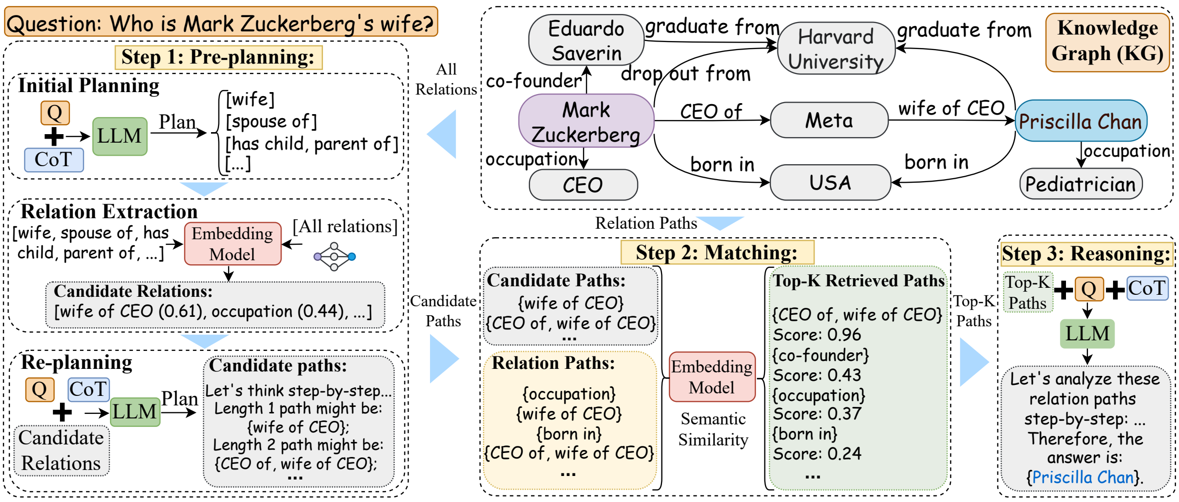

Figure 2: The framework of our KARPA. Our framework consists of three main steps: (1) Pre-planning: The LLM generates initial relation paths based on the given question, decomposes them for relation extraction, and re-plans coherent candidate paths that connect the topic and answer entities with relevant relations. (2) Matching: Relation paths are extracted based on their similarity to the re-planned candidate paths using an embedding model. Our matching method accommodates paths of varying lengths. (3) Reasoning: The selected top-K paths, combined with the question and corresponding entities, form a prompt for the LLM to enable accurate KG question answering.

model to identify top-K semantically similar relations from the entire KG:

$$R_j^i = \{r_{j1}, r_{j2}, \ldots, r_{jk}\} = \text{Top-K}(\text{sim}(\mathbf{r_j^i}, \mathbf{r})), \tag{3}$$

where $\text{sim}(\cdot)$ calculates the semantic similarity function (e.g., cosine similarity) between the embedding of relation r_j^i and all relations $r \in KG$ using Equation 1. The resulting set R_j^i contains the relations that best align with the initial relations, ensuring that LLM has access to all relevant relations beyond the immediate neighbors in the KG.

Re-planning Relation Paths with LLM Using the candidate relations R_j^i from the previous step, the LLM constructs refined relation paths P_{cand} that potentially connect the topic entity e_t to the answer entity e_a:

$$P_{cand} = \text{LLM}(Q, R_j^i), \text{ each } r_j^i \in R_j^i \subset R. \tag{4}$$

Given the question Q and candidate relations R_j^i, the LLM utilizes its reasoning capabilities to produce coherent candidate paths P_{cand}, as shown in Figure 2. During this phase, techniques like CoT can be incorporated to strengthen LLM's logical reasoning, ensuring the construction of semantically meaningful paths.

By extracting relations from the entire KG rather than limiting to adjacent neighbors, KARPA avoids stepwise interactions, reducing the risk of local optima and unnecessary interactions with the KG.

The pre-planning phase sets the foundation for efficient and accurate matching and reasoning in the subsequent steps.

4.2 Relation Paths Matching

The matching step in KARPA extracts relevant relation paths from KGs based on the LLM-generated candidate paths P_{cand}, as shown in Figure 2. This process systematically explores and scores potential relation paths for reasoning step.

4.2.1 Conventional Relation Paths Matching

Conventional LLM-based KG exploration methods, such as ToG(Sun et al., 2023), typically involve the LLM selecting top-K promising relations R_t from the adjacent relations of the current entity e at each step. This strategy resembles greedy algorithms, such as beam search. Formally, let $R(e)$ denote the set of relations available for the current entity e. The selection process can be defined as:

$$R_{\text{selected}} = \text{argmax}_{r \in R(e)} f(r), \; r \in KG. \tag{5}$$

In Equation 5, $f(r)$ is a scoring function indicating the potential of relation r. Since embedding similarity represents the similarity between two relations, we use $1 - sim(r_i, r_j)$ as the cost function for beam search. However, this approach does not guarantee finding the optimal path, as it may overlook globally optimal solutions.

To enhance relation path matching, we employ traditional pathfinding algorithms like Dijkstra's, which can be expressed as:

$$cost(v) = \min\{cost(v), cost(v') + cost(v', v)\}. \tag{6}$$

In Equation 6, the cost to reach node v is determined by either its current known cost or the cost of reaching one of its predecessors v' plus $cost(v', v)$, the cost of the edge connecting v' to v.

In KARPA, we begin from the topic entity e_t and compute the semantic similarity $sim(r_i, r_j)$ using Equation 1 for relations at each step, scoring the relations based on their similarity to the corresponding relations in the candidate relation paths P_{cand}. The cost for each step is defined as: $cost(r) = 1 - sim(r_i, r_j)$. This modification ensures that higher similarity scores correspond to lower costs, facilitating optimal path discovery. Since similarity scores range from 0 to 1, we average the total cost of relation paths of different lengths so that shorter paths can be fairly compared with longer paths. The path matching function based on Dijkstra's algorithm can be defined as:

$$\begin{aligned} cost(e) = \min\Big\{&\frac{1}{n_e} cost(e), \\ &\frac{1}{n_{e'}+1}\left[cost(e') + sim(r_{(e',e)}, r_{cand})\right]\Big\}, \end{aligned} \tag{7}$$

where the cost of entity e is compared between $cost(e)$ averaged by the number of relations n_e to reach entity e, and the cost of its predecessor $cost(e')$ plus the current cost $sim(r_{(e',e)}), r_{cand})$, averaged by number of relations $n_{e'}$ plus one.

4.2.2 Heuristic Value-Based Paths Matching

Since the conventional relation paths matching methods require the cost of each relations alone the paths, the similarity between initial relation paths and current paths within the KG can only be calculated when current paths have the same length as candidate paths P_{cand}. Inspired by the heuristic value in A* algorithm, we design a heuristic value-based relation paths matching method. In the traditional A* algorithm, the heuristic value serves as a guiding function that indicates the distance between current node and target node. In KARPA, the heuristic value h indicate the semantic similarity between the candidate relation paths P_{cand} and current path within the KG. By using heuristic value h as an indicator, we are able to compute the similarity between paths of differing lengths, such as $A \xrightarrow{father} \xrightarrow{father} B$ and $A \xrightarrow{grandfather} B$, as shown in Figure 2. For paths P_a and P_b, we concatenate all relations into one sentence and use the embedding model to calculate their similarity:

$$sim(P_a, P_b) = sim(\text{concat}(R_{Pa}), \text{concat}(R_{Pb})). \tag{8}$$

In Equation 8, the similarity between path P_a and P_b can be calculated using the concatenation of their internal relations R_P with Equation 1. Since the heuristic value represents the semantic distance between P_a and P_b, it can be defined as $h = 1 - sim(P_a, P_b)$. The top-K relation paths P_K with lowest heuristic value can be extracted as:

$$P_K = \text{argmax}_{P \in P_{all}} sim(P, P_{cand}),\ P_{all} \in KG. \tag{9}$$

Through Equation 9, we are able to identify the top-K relevant paths from a diverse range of lengths as retrieved paths P_K for further reasoning.

The relation paths matching method in KARPA broadens the search space and mitigates the risk of missing potentially optimal paths that traditional methods might overlook. By dynamically adapting to paths of varying lengths, KARPA identifies top-K paths for LLM reasoning, ensuring robust and comprehensive path matching.

4.3 Reasoning with LLM

In the reasoning step, we integrate the top-K candidate paths P_K with their corresponding entities e into a prompt, enabling the LLM to generate answers for question Q, as shown in Figure 2. The reasoning process can be expressed as:

$$\begin{aligned} Answer &= \text{LLM}(Q, P_K, e), \\ P_K &= \{p_1, p_2, \ldots, p_n\}. \end{aligned} \tag{10}$$

If the top-K candidate paths do not yield a valid answer, we leverage the LLM's inherent knowledge to provide an appropriate response. The KARPA framework facilitates the LLM's ability to evaluate multiple reasoning paths in parallel, thereby enhancing the overall efficiency of LLM-based KGQA tasks.

5 Experiments

In this section, we detail the experimental setup, present our main results, and conduct further analysis to evaluate the performance of KARPA.

5.1 Experimental Settings

Datasets and Evaluation Metrics We evaluate KARPA on two widely used multi-hop KGQA

Type of Model	Method	WebQSP Accuracy	WebQSP Hit@1	WebQSP F1	CWQ Accuracy	CWQ Hit@1	CWQ F1
Answering with Internal Knowledge							
GPT-4	IO prompt	-	62.5	-	-	44.3	-
GPT-4	CoT* (Sun et al., 2023)	-	67.3	-	-	46.0	-
Training-based Methods							
LLaMA2-7B (Fine-tune)	KD-CoT* (Wang et al., 2023)	-	68.6	52.5	-	55.7	-
Graph Reasoning Model	KG-CoT* (Zhao et al., 2024)	-	84.9	-	-	62.3	-
FiD-3B	DECAF* (Yu et al., 2022)	-	82.1	**78.8**	-	70.4	-
PLM (Pretrain)	UniKGQA* (Jiang et al., 2022)	-	77.2	72.2	-	51.2	49.0
LLaMA2-7B (Fine-tune)	RoG	80.4	84.6	70.1	60.5	61.3	54.2
Direct Inference over KGs with LLMs							
GPT-4o	ToG	58.6	78.5	50.9	53.3	56.8	41.9
GPT-4	ToG* (Sun et al., 2023)	-	82.6	-	-	69.5	-
GPT-4	Interactive-KBQA*	-	-	71.2	-	-	49.1
GPT-4o	KARPA	76.1	87.7	69.2	69.8	75.3	58.4
GPT-4	KARPA	**80.9**	**91.2**	72.1	**73.6**	**78.4**	**61.5**

Table 1: Performance comparison of KARPA with three method categories: (1) Answering with internal knowledge of LLMs, (2) Training-based methods, which require constant re-train for unseen KGs, and (3) Direct inference over KGs with LLMs. *Results are cited from corresponding publications. **Bold** represents the best result, underline represents the second best, and fbox represents the third best.

Method	**Dataset**	**Accuracy**	**Hit@1**	**F1**
RoG	Original	63.5	77.8	64.8
	Anonymized	51.4	64.3	52.9
	Variation	-12.1	-13.5	-11.9
ToG	Original	53.1	73.6	50.3
	Anonymized	45.8	64.2	44.1
	Variation	-7.3	-9.4	-6.2
KARPA	Original	72.3	86.4	67.2
	Anonymized	71.8	82.3	68.7
	Variation	-0.5	-4.1	+1.5

Table 2: Performance variation of different methods between original and anonymized WebQSP datasets.

datasets: WebQuestionSP (WebQSP) (Yih et al., 2016) and Complex WebQuestions (CWQ) (Talmor, 2018), as well as our newly anonymized version of the WebQSP dataset with placeholders replacing specific details. Evaluation metrics include Accuracy, Hit@1 and F1 score.

Baselines for Comparison We compare KARPA against several baselines: (1) LLM-only baselines: IO Prompt (Brown et al., 2020) and CoT (Wei et al., 2022) to evaluate LLM reasoning without external knowledge; (2) Training-based methods: KD-CoT (Wang et al., 2023), KG-CoT (Zhao et al., 2024), UniKGQA (Jiang et al., 2022), DECAF (Yu et al., 2022), and RoG (Luo et al., 2023), highlighting KARPA's performance without extra training; (3) Direct inference over KGs: ToG (Sun et al., 2023) and Interactive-KBQA (Xiong et al., 2024), representing the training-free state-of-the-art methods.

Experimental Details We test KARPA with various LLMs via API calls. We employ all-MiniLM-L6-v2(Reimers, 2019) as our embedding model. For each LLM, we randomly select 300 KGs from each datasets to evaluate KARPA's performance, aiming to reduce computational costs. In matching step, we extract 16 top-K paths with the highest semantic similarity for each candidate paths.

5.2 Main Results

5.2.1 Comparison between Baselines

We compare KARPA with other approaches in Table 1.The results show that KARPA significantly outperforms existing baselines across most metrics, achieving state-of-the-art performance. When comparing to the direct answering methods, we demonstrate that leveraging KGs as external knowledge sources enables the LLM to yield superior answers.

In contrast to training-based methods, KARPA is plug-and-play, requiring no additional training while maintaining effective KG-based reasoning. When comparing with inference-based method, which also utilizes LLMs for reasoning over KGs without additional training, KARPA achieves superior results by leveraging LLM's global planning

Model	Method	WebQSP			CWQ		
		Accuracy	Hit@1	F1	Accuracy	Hit@1	F1
GPT-4o-mini	CoT	-	61.3	-	-	49.5	-
	ToG	56.4	75.2	51.6	50.2	54.0	34.5
	KARPA-B	67.2	82.3	61.5	66.0	72.1	57.8
	KARPA-P	67.8	82.6	62.4	66.4	71.7	58.7
	KARPA-H	**71.9**	**85.3**	**64.5**	**68.1**	**73.3**	**56.5**
GPT-4o	CoT	-	67.0	-	-	52.3	-
	ToG	58.6	78.5	50.9	53.3	56.8	41.9
	KARPA-B	73.8	85.2	67.3	65.0	70.5	55.8
	KARPA-P	73.7	86.8	**69.7**	69.2	74.1	**59.8**
	KARPA-H	**76.1**	**87.7**	69.2	**69.8**	**75.3**	58.4
GPT-4	CoT	-	66.1	-	-	54.7	-
	ToG* (Sun et al., 2023)	-	82.6	-	-	69.5	-
	KARPA-B	73.5	85.5	68.4	71.2	75.4	61.1
	KARPA-P	74.1	86.8	69.3	73.4	77.9	**63.0**
	KARPA-H	**80.9**	**91.2**	**72.1**	**73.6**	**78.4**	61.5
Gemini-1.5-Pro	CoT	-	65.3	-	-	52.1	-
	ToG	62.3	78.4	52.5	51.7	57.9	40.5
	KARPA-B	70.1	84.5	65.9	69.1	74.0	57.2
	KARPA-P	73.8	88.0	67.4	69.6	73.5	**57.7**
	KARPA-H	**80.7**	**90.5**	**68.6**	**69.8**	**75.0**	54.8

Table 3: Comparison of KARPA, ToG, and CoT using various LLMs and matching strategies. **KARPA-B**: Beam search-based matching method with fixed beam width. **KARPA-P**: Pathfinding-based matching constrained to fixed-length paths. **KARPA-H**: Heuristic value-based matching allowing similarity calculations across variable-length paths. KARPA consistently outperforms ToG, the prior SOTA for direct KG-based reasoning using LLM.

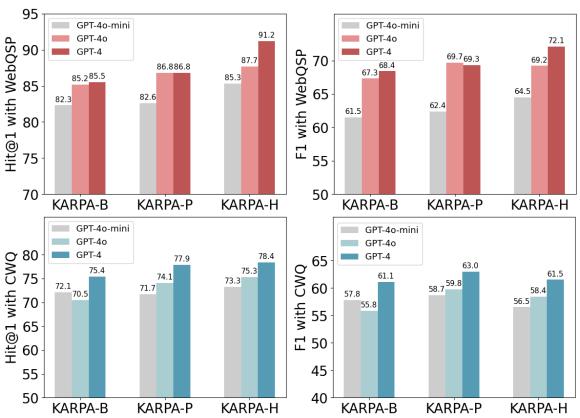

Figure 3: Comparison of different matching strategies across various LLMs on Hit@1 and F1 metrics.

capabilities, enabling the construction of coherent relation paths between topic and answer entities.

To isolate reasoning ability of the LLM from its internal knowledge, we tested KARPA on an anonymized version of the WebQSP dataset, where specific details in questions and answers are replaced with placeholders. For example, question "Where is Jamarcus Russell from? - Mobile." is transformed into "Where is Person A from? - Location A." This ensures that the final results are unaffected by LLM's pre-existing knowledge. In Table 2, RoG utilize its instruction-tuned LLaMA2-7B for planning and GPT-4o-mini for reasoning, while ToG and KARPA employ GPT-4o-mini for entire pipeline. KARPA shows the smallest performance drop, demonstrating its reliance on reasoning rather than LLM's internal knowledge. RoG exhibits larger decline, highlighting the limitations of instruction-tuned LLMs for unseen KGs.

Dataset	Method	Accuracy	Hit@1	F1
WebQSP	CoT	-	41.5	-
	ToG	24.6	30.2	21.9
	KARPA	**65.6**	**79.2**	**58.6**
CWQ	CoT	-	28.3	-
	ToG	22.4	25.8	20.2
	KARPA	**47.6**	**52.7**	**38.8**

Table 4: Performance comparison using Qwen2.5-7B.

5.2.2 Performance Across Different LLMs

We also evaluate ToG and KARPA across various LLMs. As shown in Table 3, KARPA consistently outperforms ToG and CoT, regardless of the LLM, by leveraging global planning to construct more logically sound and complete reasoning chains. In contrast, ToG's reliance on stepwise relation selection limits its effectiveness, as it neglects the

LLM's inherent planning capabilities.

To demonstrate the effectiveness of KARPA on smaller LLMs, we evaluate KARPA with Qwen2.5-7B. Table 4 shows that KARPA outperforms ToG even with smaller LLMs, demonstrating its robustness and reduced reliance on LLMs' planning abilities. More results are provided in Appendix D.

6 Analysis and Discussion

6.1 Interaction Comparison

We compare the average number of interactions required by KARPA and ToG across multiple LLMs and datasets. As shown in Table 5, KARPA reduces interactions by more than half compared to ToG while maintaining higher answer accuracy.

Method	WebQSP	CWQ
ToG*(Sun et al., 2023)	11.2	14.3
KARPA+GPT-4o-mini	5.1	6.2
KARPA+GPT-4o	**4.8**	**5.3**
KARPA+GPT-4	5.5	6.0

Table 5: Comparison of LLM call frequency.

To further illustrate the reduced reasoning complexity, we compare the average number of input and output tokens for both methods using GPT-4o-mini's tokenizer. Table 6 shows KARPA significantly reduces token usage, lowering both reasoning complexity and computational cost.

WebQSP	**Input Tokens/KG**	**Output Tokens/KG**
ToG	6351.5	1836.5
KARPA	2465.9	1492.3
CWQ	**Input Tokens/KG**	**Output Tokens/KG**
ToG	7935.7	2931.6
KARPA	3612.1	2267.1

Table 6: Token usage comparison with GPT-4o-mini.

6.2 Ablation Study

Impact of matching methods. Table 3 shows that KARPA-H achieves the best matching results, demonstrating the advantage of its flexible and robust performance for KGQA. More results are provided in Appendix C.

Influence of different LLMs. Figure 3 shows the impact of LLM capabilities on KARPA's performance. More powerful LLMs, such as GPT-4, generate better relation paths, leading to more accurate answers (Kaplan et al., 2020). With weaker LLMs like GPT-4o-mini, performance declines slightly but still surpasses ToG. This highlights the importance of global planning in KARPA's design.

Influence of Embedding Models. Table 7 evaluates KARPA with different embedding models: (1) all-MiniLM-L6-v2 (86MB): Default model of KARPA. (2) all-mpnet-base-v2 (417MB): More powerful embedding model. (3) paraphrase-multilingual-MiniLM-L12-v2 (448MB): Supports embedding between multiple languages. The results demonstrate that KARPA's performance remains stable across different embedding models.

Embedding Model	**Accuracy**	**Hit@1**	**F1**
all-MiniLM-L6-v2	72.3	**86.4**	67.2
all-mpnet-base-v2	**74.5**	86.1	**68.6**
multilingual-MiniLM-L12-v2	74.1	85.3	68.3

Table 7: KARPA with different embedding models.

6.3 Discussion

KARPA requires fewer interactions and token usage comparing to ToG, while still outperforming ToG even when using smaller LLMs. This efficiency stems from KARPA's ability to generate complete reasoning chains, reducing the need for stepwise interactions in other methods. Methods like ToG impose heavy computational burdens by evaluating hundreds or even thousands of adjacent relations at each step, whereas KARPA's global planning aligns better with human-like reasoning.

Table 7 demonstrates that KARPA performs robustly across embedding models. Its pre-planned paths are distinctive and semantically aligned with correct reasoning paths, making even lightweight embedding models sufficient for path matching.

7 Conclusion

In this paper, we propose KARPA, a novel framework designed to enhance LLM-based KGQA by utilizing the global planning and reasoning capabilities of LLMs. KARPA addresses key limitations of existing methods, achieving superior accuracy and efficiency through its pre-planning, matching and reasoning processes. Our experiments show that KARPA consistently outperforms state-of-the-art methods across multiple datasets. Its training-free design allows seamless integration with various LLMs, making it broadly applicable to different KGQA tasks. By optimizing LLM-KG interactions, KARPA enhances reasoning efficiency and effectiveness, highlighting its potential as a robust approach for future RAG systems.

8 Limitations

Although KARPA effectively reduces the reliance on the capacity of LLMs, its performance is still influenced by the reasoning and planning capabilities of the LLMs themselves. In situations where weaker LLMs are used, KARPA's performance may degrade due to LLMs' limited ability to generate logically coherent paths or perform intricate reasoning tasks. In our future work, we aim to enhance KARPA's performance on weaker LLMs, ensuring that KARPA remains effective across a broader range of LLMs with varying levels of reasoning and planning capabilities.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. Claude 3.5 sonnet model card addendum. Accessed: 2024-09-21.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Tiezheng Guo, Qingwen Yang, Chen Wang, Yanyi Liu, Pan Li, Jiawei Tang, Dapeng Li, and Yingyou Wen. 2024. Knowledgenavigator: Leveraging large language models for enhanced reasoning over knowledge graph. *Complex & Intelligent Systems*, 10(5):7063–7076.

Hangfeng He, Hongming Zhang, and Dan Roth. 2022. Rethinking with retrieval: Faithful large language model inference. *arXiv preprint arXiv:2301.00303*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Rikui Huang, Wei Wei, Xiaoye Qu, Wenfeng Xie, Xianling Mao, and Dangyang Chen. 2024. Joint multi-facts reasoning network for complex temporal question answering over knowledge graph. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10331–10335. IEEE.

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. *arXiv preprint arXiv:2305.09645*.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. *arXiv preprint arXiv:2212.00959*.

Zhanming Jie, Trung Quoc Luong, Xinbo Zhang, Xiaoran Jin, and Hang Li. 2023. Design of chain-of-thought in math problem solving. *arXiv preprint arXiv:2309.11054*.

Hanlei Jin, Yang Zhang, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Shiyang Li, Yifan Gao, Haoming Jiang, Qingyu Yin, Zheng Li, Xifeng Yan, Chao Zhang, and Bing Yin. 2023a. Graph reasoning for question answering with triplet retrieval. *arXiv preprint arXiv:2305.18742*.

Wendi Li, Wei Wei, Xiaoye Qu, Xian-Ling Mao, Ye Yuan, Wenfeng Xie, and Dangyang Chen. 2023b. Trea: Tree-structure reasoning schema for conversational recommendation. *arXiv preprint arXiv:2307.10543*.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Lidong Bing, Shafiq Joty, and Soujanya Poria. 2023c. Chain of knowledge: A framework for grounding large language models with structured knowledge bases. *arXiv preprint arXiv:2305.13269*.

Antoine Louis, Gijs van Dijck, and Gerasimos Spanakis. 2024. Interpretable long-form legal question answering with retrieval-augmented large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22266–22275.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

OpenAI. 2023. Gpt-4 technical report. Technical report, OpenAI.

OpenAI. 2024. Gpt-4o system card. Technical report, OpenAI. `https://www.openai.com/research/gpt-4o`.

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*.

Reimers. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

Talmor. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2022. Large language models are better reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-kbqa: Multi-turn interactions for knowledge base question answering with large language models. *arXiv preprint arXiv:2402.15131*.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, and Enhong Chen. 2023. Large language models for generative information extraction: A survey. *arXiv preprint arXiv:2312.17617*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Wen-tau Yih, Matthew Richardson, Christopher 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.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2021. Kg-fid: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. *arXiv preprint arXiv:2110.04330*.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024. Knowledge graph enhanced large language model editing. *arXiv preprint arXiv:2402.13593*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Ruilin Zhao, Feng Zhao, Long Wang, Xianzhi Wang, and Guandong Xu. 2024. Kg-cot: Chain-of-thought prompting of large language models over knowledge graphs for knowledge-aware question answering.

Xufeng Zhao, Mengdi Li, Wenhao Lu, Cornelius Weber, Jae Hee Lee, Kun Chu, and Stefan Wermter. 2023. Enhancing zero-shot chain-of-thought reasoning in large language models through logic. *arXiv preprint arXiv:2309.13339*.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2024. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *World Wide Web*, 27(5):58.

A Algorithm for KARPA

In this section, we present the pseudo-code for the Knowledge graph Assisted Reasoning Path Aggregation (KARPA) framework, as shown in Algorithm 1. The pseudo-code outlines the key components of our approach, including the pre-planning, matching, and reasoning phases. It demonstrates the interaction between the large language model (LLM) and the embedding model in generating, matching, and refining relation paths, which are crucial for improving LLM-based KGQA tasks.

Algorithm 1: KARPA Framework

Input: Question Q, Topic entity e_t, Knowledge Graph KG
Output: Answers E_a
Pre-Planning Phase:
Generate initial paths $P_i = \{p_1, p_2, \ldots, p_m\}$ using $LLM(Q, e_t)$;
for *each path* $p_i = (r_1^i, r_2^i, \ldots, r_{n_i}^i)$ **do**
  Decompose p_i into relation list $R_i = \{r_1^i, r_2^i, \ldots, r_{n_i}^i\}$;
  for *each relation* r_j^i *in* R_i **do**
    Retrieve top-K similar relations $R_j^i = \text{Top-K}(\text{sim}(\mathbf{r_j^i}, \mathbf{r}))$;
  end
end
Re-plan relation paths $P_{cand} = \text{LLM}(Q, R_j^i)$ based on extracted relations R_j^i;
Matching Phase:
Extract relation paths P_r with length $L \in len(P_{cand})$;
for *each path* p *in* P_{cand} **do**
  Compute similarity between paths using heuristic value $P_{matched} = Heuristic(\text{sim}(p, p_r), p_r \in P_r)$;
  Extract top-K similar paths $P = \text{Top-K}(P_{matched})$;
end
Reasoning Phase:
Combine relation paths $P_{matched} = \{r_1, r_2, \ldots, r_n\}$ with e_t, e_a into prompt;
Predict final answer $E_a = \text{LLM}(Q, P_{matched}, e_t, e_a)$;
return E_a

Model Tpye	Method	WebQSP			
		Accuracy	Hit@1	F1	Precision
GPT-4o-mini	KARPA-B	67.2	82.3	61.5	64.1
	KARPA-P	67.8	82.6	62.4	64.9
	KARPA-H	**71.9**	**85.3**	**64.5**	**65.9**
GPT-4o	KARPA-B	73.8	85.2	67.3	72.3
	KARPA-P	73.7	86.8	**69.7**	70.5
	KARPA-H	**76.1**	**87.7**	69.2	**71.5**
GPT-4	KARPA-B	73.5	85.5	68.4	71.7
	KARPA-P	74.1	86.8	69.3	**73.6**
	KARPA-H	**80.9**	**91.2**	**72.1**	73.1
DeepSeek-V2.5	KARPA-B	71.8	84.0	63.1	65.9
	KARPA-P	73.4	85.3	64.1	66.3
	KARPA-H	**78.1**	**88.4**	**68.7**	**67.6**
Gemini-1.5-Pro	KARPA-B	70.1	84.5	65.9	64.7
	KARPA-P	73.8	88.0	67.4	66.1
	KARPA-H	**80.7**	**90.5**	**68.6**	**67.8**
Claude-3.5-Sonnet	KARPA-B	75.1	85.7	66.0	67.6
	KARPA-P	80.4	89.0	69.7	**70.4**
	KARPA-H	**82.6**	**89.5**	**69.7**	69.1

Table 8: Performance of KARPA with different matching strategies (KARPA-B, KARPA-P, and KARPA-H) and LLMs on the WebQSP dataset.

B Implementation Details

Model Invocation. KARPA is tested with LLMs such as GPT-4 (OpenAI, 2023), GPT-4o (OpenAI, 2024), GPT-4o-mini, Claude-3.5-Sonnet (Anthropic, 2024), Gemini-1.5-pro (Team et al., 2024), and other LLMs through API calls. These LLMs are queried dynamically throughout the experimental pipeline to perform pre-planning, matching, and reasoning steps.

Experimental Setup. During the pre-planning stage, the initial paths generated by the LLM are decomposed and stored, along with the query, into a list. For each element in this list, we extract the top-k relations, where the total number of extracted relations does not exceed 30. These relations are semantically closest to the elements based on the LLM's initial output.

In the matching step, KARPA selects the top 16 relation paths with the highest similarity for each initial relation path. These paths serve as candidate paths for reasoning step. In the reasoning step, we limit the number of candidate paths input to the LLM at one time to a maximum of 8, ensuring that the reasoning process remains manageable and focused on the most relevant paths.

Answer Evaluation. To determine if the LLM correctly answers the question, KARPA enforces a specific output format. The final answer must be enclosed in curly brackets in the LLM's output. We consider an answer correct only when the tail entities of the reasoning paths match the text enclosed within the curly brackets in the LLM's output. For CoT, we consider an answer correct if the LLM's response contains the correct answer entities. This difference reflects the distinct reasoning and output expectations between KARPA and CoT.

C Additional Results

In this section, we present additional experimental results to further evaluate the performance of KARPA when using different matching methods: KARPA-B (beam search-based matching strategy), KARPA-P (pathfinding-based matching strategy), and KARPA-H (heuristic value-based matching strategy). We conduct these experiments across various LLMs, analyzing the effectiveness of each

Model Tpye	Method	CWQ Accuracy	Hit@1	F1	Precision
GPT-4o-mini	KARPA-B	66.0	72.1	57.8	58.6
	KARPA-P	66.4	71.7	**58.7**	**59.8**
	KARPA-H	**68.1**	**73.3**	56.5	55.1
GPT-4o	KARPA-B	65.0	70.5	55.8	57.8
	KARPA-P	69.2	74.1	**59.8**	58.4
	KARPA-H	**69.8**	**75.3**	58.4	**59.5**
GPT-4	KARPA-B	71.2	75.4	61.1	62.7
	KARPA-P	73.4	77.9	**63.0**	62.5
	KARPA-H	**73.6**	**78.4**	61.5	**63.1**
DeepSeek-V2.5	KARPA-B	61.6	63.2	48.4	50.1
	KARPA-P	60.9	63.0	51.8	52.6
	KARPA-H	**62.6**	**64.1**	**51.9**	**53.5**
Gemini-1.5-Pro	KARPA-B	69.1	74.0	57.2	59.5
	KARPA-P	69.6	73.5	**57.7**	**60.3**
	KARPA-H	**69.8**	**75.0**	54.8	55.8
Claude-3.5-Sonnet	KARPA-B	62.8	65.7	49.6	52.1
	KARPA-P	61.5	64.3	52.9	55.5
	KARPA-H	**70.6**	**73.7**	**54.9**	**56.9**

Table 9: Performance of KARPA with different matching strategies (KARPA-B, KARPA-P, and KARPA-H) and LLMs on the CWQ dataset.

matching strategy in conjunction with different LLMs. These results provide a deeper insight into how different matching mechanisms impact the overall performance of KARPA, showcasing the versatility and adaptability of our approach under varying model conditions.

The results presented in Table 8 and Table 9 consistently demonstrate the superior performance of KARPA-H (heuristic value-based matching) compared to the other two matching strategies, KARPA-B (beam search-based) and KARPA-P (pathfinding-based), across different LLMs and datasets (WebQSP and CWQ).

In the majority of LLMs, KARPA-H outperforms the other methods in most metrics. This suggests that KARPA-H is more effective at extracting the correct relation paths, which in turn leads to more accurate and contextually relevant answers. These results highlight KARPA-H as the most robust and reliable matching method among the three, reinforcing its advantage in handling complex KG-based reasoning tasks.

D Additional Experiments

In this section, we provide additional experiments to validate KARPA's performance from different perspectives.

To demonstrate that KARPA has better generalization capabilities than methods based on instruction-tuned LLMs, we conducted an experiment using GPT-4o-mini with a modified version of the WebQSP dataset. Specifically, we slightly alter the questions in WebQSP dataset while preserving their original meaning, using the prompt: "Please revise the question to make it more clear, but the original meaning of the question and the corresponding answers remain unchanged." We test RoG using its instruction-tuned LLaMa2-Chat-7B from in the planning step and GPT-4o-mini for reasoning. In KARPA, we use GPT-4o-mini for both pre-planning and reasoning steps.

The results in Table 10 show that KARPA's performance remains consistent and robust to question modifications, while RoG's performance drops due to path mismatches. This further highlights the advantage of KARPA's training-free framework,

Question	Method	Accuracy	Hit@1	F1	Method	Accuracy	Hit@1	F1
Origin	RoG	67.6	84.1	69.7	KARPA	73.1	85.4	68.1
Revised	RoG	63.5	74.3	64.1	KARPA	72.6	84.5	68.9
Variation	RoG	-4.1	-9.8	-5.6	KARPA	-0.5	-0.9	+0.8

Table 10: Comparison of RoG and KARPA on the WebQSP dataset with original and revised questions.

maintaining superior robustness and adaptability across all KGs.

To demonstrate the effectiveness of KARPA with smaller LLMs, we conduct experiments with Qwen2.5-7B and Qwen2.5-14B as the LLM backbones for KARPA. The results in Table 11 demonstrate that KARPA consistently outperforms stepwise direct inference baselines such as ToG, even when using smaller LLMs. This reinforces the robustness and adaptability of our method across different LLM scales.

Model Type	Method	WebQSP Accuracy	WebQSP Hit@1	WebQSP F1
Qwen2.5-7B	CoT	-	41.5	-
	ToG	24.6	30.2	21.9
	KARPA	**65.6**	**79.2**	**58.6**
Qwen2.5-14B	CoT	-	49.6	-
	ToG	45.0	55.9	42.7
	KARPA	**72.6**	**84.1**	**65.0**
		CWQ		
Qwen2.5-7B	CoT	-	28.3	-
	ToG	22.4	25.8	20.2
	KARPA	**47.6**	**52.7**	**38.8**
Qwen2.5-14B	CoT	-	31.2	-
	ToG	30.2	36.6	29.5
	KARPA	**51.5**	**57.9**	**41.6**

Table 11: Performance comparison of different methods on WebQSP and CWQ datasets using smaller LLMs.

Also, the results in Table 11 show that KARPA can perform well with LLMs that have weaker planning and reasoning capabilities, further highlighting KARPA's robustness and its reduced dependence on the LLM's planning and reasoning abilities compared to other inference-based methods.

In multilingual scenarios, KARPA can effectively address this problem by using multilingual embedding models. For instance, in a multilingual setting, we test KARPA with paraphrase-multilingual-MiniLM-L12-v2, a multilingual embedding model. In the multilingual experiment, we use GPT-4o-mini to generate relation paths in Chinese, and then use the multilingual embedding model to calculate the semantic similarity between the candidate paths and paths in the KG.

These results in Table 12 demonstrate that with a multilingual embedding model, KARPA performs effectively across languages, maintaining its robustness. They also indicate that language variations do not significantly impact KARPA's performance.

To demonstrate the necessity of extending relation paths with different lengths, we restrict the matching step to use only single-relation candidate paths provided by the LLM during re-planning step, and compare the performance of the heuristic value-based matching method (KARPA-H) with the pathfinding-based matching method (KARPA-P) using GPT-4o-mini.

The results in the Table 13 demonstrate that the heuristic value-based matching method outperforms pathfinding-based matching methods in such scenarios, as it effectively addresses the semantic similarity issues that arise from differing path lengths. Moreover, as the questions in the CWQ dataset generally require longer reasoning paths compared to WebQSP, both methods exhibit a more significant decline in various metrics on CWQ. However, the heuristic value-based method shows a less pronounced drop compared to pathfinding-based methods, further demonstrating its superiority.

To validate the performance of KARPA on KGs outside the training scope, we compare KARPA with Chain-of-Thought (CoT) reasoning, where the LLM directly relies on its internal knowledge to answer questions. Using open source LLMs such as Qwen2.5-7B, Qwen2.5-14B and Qwen2.5-72B (with limited stored knowledge), we observe that CoT performance drops significantly on KGQA tasks while KARPA maintains strong performance.

The results in Table 14 highlight KARPA's ability to operate effectively on unseen KGs by focusing on reasoning and planning rather than leveraging the LLM's pre-existing knowledge. The results also show that KARPA maintained strong performance, even as the LLM's stored knowledge was significantly reduced. This means that even if the

Language	WebQSP			CWQ		
	Accuracy	Hit@1	F1	Accuracy	Hit@1	F1
English-English	74.1	85.3	68.3	65.3	69.5	55.4
Chinese-English	74.6	84.5	67.6	63.1	68.0	54.2

Table 12: Performance comparison of different languages using a multilingual embedding model.

Candidate Path	Method	WebQSP			CWQ		
		Accuracy	Hit@1	F1	Accuracy	Hit@1	F1
Original Paths	KARPA-P	66.0	81.2	63.8	61.0	64.5	53.4
Original Paths	KARPA-H	72.3	86.4	67.2	64.6	67.7	55.1
Single-Relation Paths	KARPA-P	63.6	77.3	60.7	40.5	43.9	39.3
Single-Relation Paths	KARPA-H	71.4	85.5	68.9	55.1	59.6	47.4

Table 13: Performance of KARPA-P and KARPA-H using different candidate paths on the WebQSP and CWQ datasets.

LLM does not have ample prior knowledge about a specific domain, KARPA can still leverage the LLM's reasoning and planning capabilities to construct reasoning chains to find the correct answers within the KG.

To demonstrate the effectiveness of KARPA in noisy KGs and specialized domains, we conduct an experiment introducing noise into the KG. For WebQSP and CWQ samples with reasoning paths longer than one, we randomly shuffle the neighboring relations of topic entity and then compared the performance of KARPA and ToG using GPT-4o-mini.

The results in Table 15 show that KARPA experiences a slight drop in performance, demonstrating its resilience to noisy relations. ToG shows a more significant decline, highlighting the limitations of traditional KGQA methods in noisy environments.

E Further Discussion

E.1 Effectiveness Beyond KGQA Tasks

While KARPA is currently designed to address challenges in KGQA tasks, following the settings of prior works such as RoG and ToG, its methodology is generalizable to other knowledge-intensive tasks.

KARPA's core idea lies in letting LLMs generate complete reasoning chains instead of disrupting reasoning continuity with step-by-step searching. This approach mimics human reasoning processes and enhances reasoning efficiency. For example, in knowledge-intensive task such as the retrieval of academic papers, KARPA could generate reasoning chains like "research field → target journal/conference → specific keywords", and then retrieve the corresponding paper using semantic similarity. When extracting information from books, the reasoning chain like "book title → relevant chapter → relevant paragraphs" could streamline the information extraction. This reasoning-chain generation aligns with human thought processes, making it both intuitive and adaptable to diverse knowledge-intensive tasks.

E.2 Incorporating User Feedback Mechanisms

KARPA's architecture is inherently well-suited to incorporating user feedback mechanisms due to its design of generating complete reasoning paths. We provide a potential extension here:

- Initial Path Generation: KARPA generates an initial reasoning path based on the user query.

- Ambiguity Threshold: Using our semantic similarity-based matching method, we match the LLM-generated path with paths within the KG. If the similarity score reaches a certain ambiguity threshold, the query is considered clear; if the similarity score falls below that threshold, we identify the query as potentially ambiguous.

- User Feedback: If the similarity score reaches the threshold, we can provide the user with the retrieved answers. If the score falls below the threshold, we could present the extracted

		WebQSP			CWQ		
Base-Model	**Method**	**Accuracy**	**Hit@1**	**F1**	**Accuracy**	**Hit@1**	**F1**
Qwen2.5-7B	CoT	-	41.5	-	-	28.3	-
	KARPA	65.6	79.2	58.6	47.6	52.7	38.8
	Gain	-	**+37.7**	-	-	+24.4	-
Qwen2.5-14B	CoT	-	49.6	-	-	31.2	-
	KARPA	72.6	84.1	65.0	51.5	57.9	41.6
	Gain	-	+34.5	-	-	**+26.7**	-
Qwen2.5-72B	CoT	-	56.9	-	-	40.5	-
	KARPA	73.2	86.0	64.5	61.1	63.6	52.7
	Gain	-	+29.1	-	-	+23.1	-

Table 14: Performance comparison of CoT and KARPA methods across different base models (Qwen2.5-7B, Qwen2.5-14B, Qwen2.5-72B) on WebQSP and CWQ datasets.

reasoning paths to the user for review and request further clarification or refinement of the query.

- Refinement and Rematching: Based on user feedback, KARPA could adjust the reasoning path and re-run the matching process to generate more accurate results.

Through the steps outlined above, KARPA can establish a comprehensive user feedback mechanism, which enhances the precision of queries based on ongoing user feedback.

F Detailed Related Work

F.1 Prompt-Based Question Answering Using Internal Knowledge

In the field of large language models (LLMs), researchers explore how to combine internal knowledge with external information to enhance reasoning abilities. Existing models utilize a vast internal knowledge base and achieve significant progress in reasoning tasks. To further optimize these capabilities, researchers propose various prompt-based methods, such as Chain of Thought (CoT) (Li et al., 2023c) prompting. This method breaks down complex tasks into manageable steps, promoting structured reasoning and excelling in mathematical and logical reasoning. Building on CoT, researchers also develop variants like Auto-CoT (Zhang et al., 2022), Zero-Shot-CoT (Kojima et al., 2022), Complex-CoT (Fu et al., 2022), and new frameworks such as Tree of Thoughts (ToT) (Yao et al., 2024), which further expand the application range of LLMs.

Additionally, with regard to the "decoding" problem of the reasoning process, Self-consistency CoT (Wang et al., 2022) serves as a representative method. It generates multiple reasoning paths through manually designed prompts and employs a "majority voting" mechanism to identify the "most consistent" path, thereby enhancing CoT performance. CoT verification (Weng et al., 2022) is another important research direction that allows models to self-verify the correctness of their answers through multiple rounds of reasoning. Self-Verification samples multiple candidate reasoning paths and ranks them based on whether the conditions satisfy the conclusions. Recently, OpenAI launches the o1 series models, marking a significant advancement in LLM reasoning abilities, allowing models to develop extensive internal chains of thought and further tap into their reasoning potential.

F.2 Embedding models and Embedding-based methods.

Embedding models have revolutionized how we represent and understand text by converting words and sentences into dense vector representations (Mikolov et al., 2013). These embedding models capture the semantic meaning of the text, enabling models to effectively measure the similarity and relationships between different texts. In recent years, significant progress has been made in the field of text embeddings, largely due to the emergence of pre-trained language models (Vaswani et al., 2017). Models like BERT and its variants have become fundamental tools for efficiently encoding the underlying semantics of data. Key advancements in

Knowledge Graphs	Method	WebQSP Accuracy	WebQSP Hit@1	WebQSP F1	CWQ Accuracy	CWQ Hit@1	CWQ F1
Original KGs	ToG	54.2	72.8	50.3	47.6	52.5	39.1
Shuffled KGs	ToG	32.7	48.2	30.1	23.3	26.7	20.9
Variation	ToG	-21.5	-24.6	-20.2	-24.3	-25.8	-18.2
Original KGs	KARPA	72.3	86.4	67.2	64.6	67.7	55.1
Shuffled KGs	KARPA	70.7	84.1	64.5	56.0	61.3	51.5
Variation	KARPA	-1.6	-2.3	-2.7	-8.6	-6.4	-3.6

Table 15: Comparison of performance between original and shuffled KGs for ToG and KARPA methods on WebQSP and CWQ datasets.

contrastive learning, particularly improvements in negative sampling and knowledge distillation applications, also contribute significantly to the progress in this field. As a result, there is a growing trend to develop universal embedding models that can uniformly support a variety of applications, ranging from information retrieval to natural language processing tasks.

F.3 Knowledge Graphs and Retrieval-Augmented Methods.

Knowledge graphs and retrieval-augmented generation (RAG) (Lewis et al., 2020) play a crucial role in enhancing various downstream tasks, such as question answering, text generation, and information retrieval. Early research (Sun et al., 2018) uses random walk algorithms to retrieve information from knowledge graphs. Subsequent studies (Li et al., 2023a; Yu et al., 2021) employ BM25 and DPR algorithms for knowledge graph-based information retrieval, further improving the performance of LLMs. UniKGQA (Jiang et al., 2022) integrates the retrieval process with LLMs to achieve state-of-the-art performance in knowledge graph question-answering tasks. KELP utilizes an embedding model to filter reasoning paths from the KG. However, it does not leverage the reasoning capabilities of LLMs and is limited to reasoning paths within a 2-hop range, restricting its applicability to more complex queries. KnowledgeNavigator (Guo et al., 2024) employs an iterative process where the LLM retrieves and filters relevant knowledge directly from the KG. GraphRAG (Edge et al., 2024) designs a powerful process that extracts structured data from unstructured text using LLMs. These studies collectively demonstrate that information retrieved from knowledge graphs significantly enhances the reasoning capabilities of LLMs.

G Datasets

We adopt two widely-used multi-hop KGQA datasets in our work. Table 16 below gives detailed statistical information for both datasets.

Statistics	WebQSP	CWQ
Dataset Split		
Train	2,826	27,639
Test	1,628	3,531
Question Hop Distribution		
1 hop	65.49%	40.91%
2 hop	34.51%	38.34%
$\geq$ 3 hop	0.00%	20.75%
Answer Counts Distribution		
Ans = 1	51.2%	70.6%
$2 \leq$ Ans ≤ 4	27.4%	19.4%
$5 \leq$ Ans ≤ 9	8.3%	6.0%
Ans ≥ 10	12.1%	4.0%

Table 16: Comprehensive Statistics of Datasets.

- **WebQuestionsSP (WebQSP)** (Yih et al., 2016) is a knowledge base Q&A dataset containing 4737 questions requiring up to 2-hop reasoning on the KG Freebase (Bollacker et al., 2008), designed to improve the performance of Q&A systems through semantic parsing.

- **Complex WebQuestion (CWQ)** (Talmor, 2018) is extended based on the WebQSP dataset that require up to 4-hop reasoning on the KG Freebase (Bollacker et al., 2008) to solve more complex Q&A tasks.

H Baselines

We consider the following baseline methods for performance comparison:

- **IO Prompt**: Directly query large language models (LLMs) for answers without relying on external sources of information or additional reasoning processes.

- **CoT Prompt**: Utilizing Chain-of-Thought prompting with LLMs to facilitate reasoning involves guiding the LLM through a step-by-step process, where each step reflects the logical sequence of human reasoning.

- **LLM-Based KGQA Methods**:

  KD-CoT (Wang et al., 2023) interacts with external knowledge to verify and amend the reasoning paths within the Chain-of-Thought (CoT), effectively overcoming issues of hallucinations and error propagation. It structures the CoT reasoning process of LLMs into a formatted multi-round QA approach. In each round, LLMs interact with a QA system that retrieves external knowledge, constructing more reliable reasoning paths based on the precise answers retrieved, thereby enhancing the accuracy and credibility of reasoning.

  UniKGQA (Jiang et al., 2022) unifies retrieval and reasoning in both model architecture and parameter learning by designing a shared pre-training task based on question-relation matching and applying fine-tuning strategies to optimize the retrieval and reasoning processes. It includes two main modules: a semantic matching module based on a pre-trained language model (PLM) for question-relation semantic matching, and a matching information propagation module that spreads matching information along directed edges in the knowledge graph (KG).

  DECAF (Yu et al., 2022) arrives at the final answer by co-generating logical forms and direct answers and combining the best of both. Unlike approaches that rely on entity linking tools, DECAF simplifies the process of information retrieval by linearizing the knowledge base into text documents and locating relevant subgraphs using text-based retrieval methods.

  RoG (Luo et al., 2023) is an approach that combines LLMs with KG to achieve reliable and interpretable reasoning. The method first generates knowledge graph-based relational paths that serve as faithful reasoning plans, and then utilizes these plans to retrieve valid reasoning paths from the knowledge graph for accurate reasoning in LLMs. RoG enhances the reasoning capabilities of LLMs by training to distill knowledge from knowledge graphs and allows them to be seamlessly integrated with arbitrary LLMs for reasoning.

  ToG (Sun et al., 2023) proposes a new LLM-KG integration paradigm "LLM $\otimes$ KG" that treats a LLM as an agent that performs a beam search over the knowledge graph iteratively to discover the most promising reasoning paths and return the most possible reasoning results. ToG leverages the reasoning power of LLMs and expert feedback to ensure traceability and correctability of knowledge. The framework is flexible and plug-and-play for different LLMs, knowledge graphs, and cueing strategies with no additional training cost.

I Prompts

Our proposed KARPA framework consists of the following three main steps: (1) Pre-Planning; (2) Matching; (3) Reasoning. Among them, steps (1) and (3) use the Large Language Model (LLM), and Appendix I provides the related Prompts.

I.1 Pre-Planning

I.1.1 Initial-Planning Prompt

In the pre-planning stage, initial planning involves using an LLM to preliminarily generate several relation paths of different lengths. The prompt used for this process is given in Content I.1.1.

Initial-Planning Prompt

In the process of answer retrieval using a knowledge graph, please think step-by-step and generate reasoning paths of lengths 1, 2, and 3 from a given question and the provided head entity (or entities) that could potentially lead to answer entities. If a reasoning path of the specified length does not exist, please explain the reason.

Q:
Name the president of the country whose main spoken language was Brahui in 1980?
Topic Entity: Brahui Language
A:
Length 1 reasoning path: The answer entity cannot be reached within a single step, so the length 1 reasoning path is None: { }.

Length 2 reasoning path: The answer entity may be reached by first finding the corresponding country through the relation "language.human language.main country", and then finding the president of the country through the relation "government.government position held.office holder". So the length 2 reasoning path is: {language.human_language.main_country, government.government_position_held.office_holder}.

Length 3 reasoning path: The answer entity does not require 3 steps to reach, so the length 3 reasoning path is None: {}.

Q:
Who is Tom's wife?
Topic Entity: Tom
A:
Length 1 reasoning path: The answer entity can be reached within a single step by finding Tom's spouse through the relation "people.person.spouse_s". Therefore, the length 1 reasoning path is: {people.person.spouse_s}.

Length 2 reasoning path: The answer entity of the question may be reached if we first find the children through first relation "people.person.children", and then find the parent through second relation "people.person.parent". Therefore, the length 2 reasoning path is: {people.person.children, people.person.parent}.

Length 3 reasoning path: The answer entity of the question does not require 3 steps to reach, so the length 3 reasoning path is None: {}.

Q:
{A Question.}
Topic Entity: *{An Entity}*
A:

I.1.2 Re-Planning Prompt

In the re-planning of pre-planning, the LLM is used to re-plan relation paths based on the extracted relations (specifically the top-K relations), which are then used as retrieval information in the matching step. The prompt used is shown in Content I.1.2.

Re-Planning Prompt

Given a set of relations and a question, please select relevant relations from the provided relation set to form reasoning paths of length 1, 2, and 3 that could lead from the provided topic entity (or entities) to potential answer entities in a knowledge graph. Ensure that the reasoning paths you create logically connect the topic entity and potential answer entities. Only consider length 3 paths if shorter paths are insufficient to reach the answer. If a reasoning path of the specific length cannot be formed, please explain why.

Q:
Name the president of the country whose main spoken language was Brahui in 1980?
Topic Entity: Brahui Language
Relations:
language.human_language.language_family;
language.human_language.main_country;
base.rosetta.languoid.parent;
language.human_language.writing_system;
language.human_language.countries_spoken_in;
kg.object_profile.prominent_type;
A:
Length 1 reasoning path: The provided relations cannot reach the answer entity in one step, so the length 1 reasoning path is None: {}.

Length 2 reasoning path: The answer entity may be reached by first finding the corresponding country through the provided relation "language.human language.main country", and then finding the president of the country through the relation "government.government position held.office holder". So the length 2 reasoning path is: {language.human_language.main_country, government.government_position_held.office_holder}.

Length 3 reasoning path: The answer entity does not require 3 steps to reach, so the length 3 reasoning path is None: {}.

Q:
Who is Tom's wife?
Topic Entity: Tom
Relations:
people.person.profession;
people.marriage.spouse;
people.person.nationality;
award.award_nomination.award_nominee;
people.person.parents;
award.award_nominee.award_nominations;
people.person.children;
A:
Length 1 reasoning path: Tom's wife in knowledge graph could be reached within a single step by finding Tom's spouse through the provided relation "people.person.spouse_s". Therefore, the length 1 reasoning path is: {people.person.spouse_s}.

Length 2 reasoning path: Tom's wife may be reached if we first find the children through the relation "people.person.children", and then find the parent through second relation "people.person.parent". Therefore, the length 2 reasoning path is: {people.person.children, people.person.parent}.

Length 3 reasoning path: The answer entity of the question does not require 3 steps to reach, so the length 3 reasoning path is None: {}.

Q:
{A Question.}
Topic Entity: *{An Entity.}*
Relations: *{A list of Relations.}*
A:

I.2 Reasoning

In the reasoning step, the top-K relation paths retrieved in the matching step, along with their connected topic entity, answer entities, the corresponding question, and all related information are input into the LLM. The prompt used is provided in content I.2 below.

Reasoning Prompt

Given a question and the associated retrieved knowledge graph reasoning paths (topic entity, relation path, tail entity/entities), please think step-by-step and determine whether the tail entity/entities of each provided reasoning paths are the right answer to the question. If any of the tail entities are the correct answer, return those tail entities in the format of answer_1, answer_2, ... If the tail entity of current reasoning path is not the answer of question, explain why. All reasoning paths should be analysed.

Q:
Rift Valley Province is located in a nation that uses which form of currency?
Reasoning Paths:
(Rift Valley Province, location.administrative division.country, Kenya)
(Rift Valley Province, location.location.geo location, UnName Entity)
(Rift Valley Province, location.administrative division.country → location.country.currency used, Kenyan shilling)
A:
Let's analyze the reasoning paths step-by-step to determine the correct answer to the question.
1.(Rift Valley Province, location.administrative division.country, Kenya): Rift Valley Province is located in Kenya. However, it does not provide information about the currency used in Kenya, so Kenya is not the direct answer to the question.
2.(Rift Valley Province, location.location.geo location, UnName Entity): The UnName Entity is not the answer of question.
3.(Rift Valley Province, location.administrative division.country→ location.country.currency used, Kenyan shilling): Kenyan shilling is the form of currency used in the nation where Rift Valley Province is located, so 'Kenyan shilling' is the right answer to the question.
The correct answer to the question is the Kenyan shilling, as identified in the fourth reasoning path. Therefore, the correct tail entity is:
{Kenyan shilling}.

Q:
{A Question.}
Reasoning Paths: *{A list of Reasoning Paths.}*
A:

J Case Study

In this section, we present a detailed case study to illustrate the effectiveness of KARPA in handling complex knowledge graph question answering (KGQA) tasks. KARPA leverages LLMs in both the pre-planning and reasoning steps. For the question "What did James K. Polk do before he was president?", KARPA uses the LLM to generate initial reasoning paths and then further refines the answer by reasoning over the identified relation paths and corresponding entities. The following case study elaborates on the workflow of KARPA in this example, showcasing its ability to utilize external knowledge and LLM planning capabilities to accurately answer the question.

In the pre-planning step, KARPA first utilizes the LLM to generate initial relational paths based on the provided question, as shown in Figure 4. Given the question "What did James K. Polk do before he was president?", the LLM generates paths of varying lengths. Initially, the LLM considers whether the answer entities can be reached within a single relational step. Since the LLM considers the answer entities for this question cannot be reached in one step, the LLM outputs an empty reasoning path of length 1.

When considering a relational path with two associated relations, the LLM infers that the answer entity can be found by first identifying the political positions held by James K. Polk through the relation "government.politician.position_held," and then filtering for the position he held before becoming president using "filter: position before president." Thus, the LLM determines that the answer entities can be reached via the path {government.politician.position_held, filter: position before president}. Additionally, the LLM considers that the answer entities might be accessible through a path involving three relations. This step-by-step reasoning process allows the LLM to initially plan multiple reasoning chains for subsequent relation extraction.

In the third phase of the pre-planning step, KARPA employs the LLM to re-plan the relational paths based on the set of extracted relations. For the question "What did James K. Polk do before he was president?", the LLM is provided with a set of relations, as illustrated in Figure 5. The LLM is tasked with selecting relevant relations from the list and assembling them into complete reasoning chains that potentially connect the topic entity to the answer entities.

In this case, the LLM determines that the answer entities cannot be reached using a single relation from the provided list, and therefore outputs an empty relation path for length 1. When constructing a relation path of length 2, the LLM identifies that "government.politician.government positions held" and "government.government position held.office position or title" form a complete reasoning chain, enabling the extraction of the correct answer entities for the given question. As a result, the LLM outputs the length 2 relation path as {govern-

Initial-Planning

Q:

what did james k polk do before he was president?
Topic Entity: James K. Polk

Response of GPT4:

Length 1 reasoning path: The answer entity cannot be reached with-in a single step, so the length 1 reasoning path is None: {}.

Length 2 reasoning path: The answer entity may be reached by first finding the political positions held by James K. Polk through the relation "government.politician.position_held", and then filtering for the position held before he became president. So the length 2 reasoning path is:
{government.politician.position_held,
filter: position before president}.

Length 3 reasoning path: The answer entity may be reached by first finding the political positions held by James K. Polk through the relation "government.politician.position_held", then finding the specific position such as "Speaker of the House" or "Governor" through the relation "government.government_office_or_title.office_holder", and finally filtering for the position held before he became president.
So the length 3 reasoning path is:
{government.politician.position_held,
government.government_office_or_title.office_holder,
filter: position before president}.

Figure 4: A Case for Relational Path Generation in the Initial-Planning Phase.

ment.politician.government_positions_held, government.government_position_held.office_position or_title}. Since the LLM considers that the answer can be extracted using this two-step reasoning chain, it determines that a three-step reasoning chain is unnecessary and outputs None for the length 3 relation paths.

In the reasoning step of KARPA, several candidate relational paths are provided for the LLM to determine the final answer. Given these candidate paths and their corresponding entities, the LLM analyzes each path step-by-step, enabling more thoughtful and accurate reasoning. An example of KARPA's reasoning process is illustrated in Figure 6.

Re-Planning

Q:

What did james k polk do before he was president?
Topic Entity: James K. Polk
Relations:
government.politician.government_positions_held;
government.government_position_held.office_holder;
government.government_position_held.appointed_by;
government.government_position_held.basic_title;
government.government_position_held.governmental_body;
government.us_president.vice_president;
people.appointee.position;
government.us_vice_president.to_president;
government.government_office_or_title.office_holders;
government.government_position_held.office_position_or_title;
government.government_office_or_title.jurisdiction;
government.government_office_or_title.governmental_body_if_an
y; base.inaugurations.inauguration.president;
military.military_person.participated_in_conflicts;
government.politician.election_campaigns

Response of GPT4:

Length 1 reasoning path: The provided relations cannot reach the answer entity in one step, so the length 1 reasoning path is None: {}.

Length 2 reasoning path: The answer entity may be reached by first finding the government positions held by James K. Polk through the relation "government.politician.government_positions_held", and then identifying the specific position held before becoming president through the relation "government.government_position_held.office_position_or_title".
Therefore, the length 2 reasoning path is:
{government.politician.government_positions_held,
government.government_position_held.office_position_or_title}.

Length 3 reasoning path: The answer entity does not require 3 steps to reach, so the length 3 reasoning path is None: {}.

Figure 5: A Case of Relation Path Assembling in the Re-Planning Phase.

Reasoning

Q:

what did james k polk do before he was president?
Reasoning Paths:
('James K. Polk', 'government.politician.government_positions_held -> government.government_position_held.basic_title', 'President, Speaker')
('James K. Polk', 'government.politician.government_positions_held -> government.government_office_or_title.office_holders', 'Speaker of the United States House of Representatives')
('James K. Polk', 'government.politician.government_positions_held -> government.government_position_held.governmental_body', 'United States House of Representatives')
('James K. Polk', 'government.government_position_held.office_holder -> government.government_position_held.office_position_or_title', 'United States Representative')
('James K. Polk', 'government.politician.government_positions_held -> government.government_position_held.legislative_sessions', '25th United States Congress, 24th United States Congress')

Response of GPT4:

Let's analyze each reasoning path to deduce if it answers the question regarding what James K. Polk did before he became president.

1. ('James K. Polk', 'government.politician.government_positions_held -> government.government_position_held.basic_title', 'President, Speaker'):
   - This path mentions "President, Speaker" indicating he held the position of Speaker possibly before becoming President. "Speaker" could potentially be a valid answer but it lacks the detail of which Speaker position.

2. ('James K. Polk', 'government.politician.government_positions_held -> government.government_office_or_title.office_hold
-ers', 'Speaker of the United States House of Representatives'):
   - This path clearly states that before becoming President, he was the "Speaker of the United States House of Representatives." This directly answers the question about what he did before becoming President.

3. ('James K. Polk', 'government.politician.government_positions_held -> government.government_position_held.government
-tal_body', 'United States House of Representatives'):
   - This indicates he was a member of the United States House of Representatives, which is relevant as one major position held prior to being President but less detailed compared to the Speaker role.

4. ('James K. Polk', 'government.government_position_held.office_holder -
>government.government_position_held.office_po-
sition_or_title', 'United States Representative'):
   - This details his role as a United States Representative, a position held in the legislature before becoming President.

5. ('James K. Polk', 'government.politician.government_positions_held -> government.government_position_held.legislative
_sessions', '25th United States Congress, 24th United States Congress'):
   - This confirms his participation in the 24th and 25th United States Congresses, indicating his legislative roles.

From these reasoning paths, the valid answers to what James K. Polk did before he was president are: {Speaker of the United States House of Representatives, United States Representative}

Figure 6: A Case of Inputting Top-K Relation Paths into LLMs During Reasoning Phase.