ReKG-MCTS: Reinforcing LLM Reasoning on Knowledge Graphs via Training-Free Monte Carlo Tree Search

Xiaozhuang Song
SDS
CUHK-Shenzhen
Shenzhen, China
shawnsxz97@gmail.com

Shufei Zhang[†]
Shanghai AI Laboratory
Shanghai, China
zhangshufei@pjlab.org.cn

Tianshu Yu[†]
SDS
CUHK-Shenzhen
Shenzhen, China
yutianshu@cuhk.edu.cn

Abstract

Recent advancements in combining knowledge graphs (KGs) with large language models (LLMs) have demonstrated promising potential in complex KG reasoning tasks, yet existing approaches face limitations in path exploration strategies or excessive computational overhead. We propose ReKG-MCTS, a novel training-free framework that synergizes Monte Carlo Tree Search (MCTS) with LLM capabilities to enable dynamic reasoning over KGs. The framework conceptualizes KG reasoning as a decision-making process, where MCTS strategically explores paths over KG while LLMs provide semantic guidance for reasoning paths. The framework consists of four phases: (1) UCB-based node selection that balances exploration-exploitation on KG, (2) path expansion with KG structural constraints, (3) LLM-guided MC rollouts for simulation, and (4) value backpropagation. Experimental results on WebQSP and CWQ demonstrate that ReKG-MCTS outperforms existing training-free methods and achieves competitive performance compared to fine-tuned baselines. These findings suggest a new paradigm for leveraging language models in KG reasoning tasks. The code is available at https://github.com/ShawnKS/rekgmcts.

1 Introduction

Large Language Models (LLMs) have brought transformative advancements to various natural language processing (NLP) tasks, exhibiting remarkable capabilities in semantic understanding and inference generation (Chowdhery et al., 2023; Achiam et al., 2023; Jiang et al., 2023a; Yang et al., 2024; DeepSeek-AI et al., 2024). Despite these successes, LLMs still suffer from hallucination and knowledge inconsistency issues (Huang et al., 2024a; Xu et al., 2024b; Li et al., 2024a; Huang et al., 2024b), highlighting the need for grounding their responses in factual knowledge. Knowledge Graphs (KGs) emerge as a promising solution to this challenge, as they provide structured and reliable factual information (Martino et al., 2023; Guan et al., 2024; Zhang et al., 2024b; Agrawal et al., 2024). However, leveraging KGs for reasoning remains challenging, as it requires precise multi-hop traversal and pathfinding capabilities – tasks where LLMs, designed primarily for unstructured text processing, face inherent limitations.

Recently, significant efforts have been made to augment LLMs with KG reasoning mechanisms (Sun et al., 2024; Xu et al., 2024a; Markowitz et al., 2024). To search for relevant paths in knowledge graphs, these methods typically rely on pre-defined heuristics (e.g., beam search (Luo et al., 2024b) or LLM-based relation pruning (Sun et al., 2024)), or delegate the path evaluation process to an LLM (Markowitz et al., 2024). While yielding promising initial results, these approaches often face limitations in effectively utilizing path quality feedback during search (Lambert et al., 2024; Setlur et al., 2025; Xue and Zou, 2022). Specifically, greedy or random strategies cannot adaptively adjust their search priorities based on the "goodness" of intermediate states, leading to either local optima or inefficient exploration of the solution space, particularly in complex multi-hop inference scenarios (Megiddo et al., 1988; Ramalingam and Reps, 1996; Zhao and Han, 2010).

Monte Carlo Tree Search (MCTS) presents a compelling solution to these limitations by naturally incorporating path quality feedback into the search process (Browne et al., 2012; Silver et al., 2016; Świechowski et al., 2023). Unlike static heuristics or one-shot sampling approaches, MCTS dynamically evaluates and adjusts search priorities through its four fundamental steps: selection, expansion, simulation, and backpropagation. This iterative process enables the algorithm to learn from

[†] Corresponding authors

previous search attempts and adaptively balance between exploring new promising paths and exploiting known successful routes (Vinyals et al., 2019; Zhao et al., 2024; Zhang et al., 2024a).

Building upon these intuitions, we propose ReKG-MCTS, a training-free framework that unifies MCTS with LLM-guided heuristics for efficient and interpretable KG reasoning. By viewing reasoning as a sequential decision-making process, our method adaptively refines search paths while balancing exploration and exploitation via MCTS. In this paradigm, the LLM simultaneously acts as a *policy model*, suggesting plausible entities and relations exploration, and as a *value method*, assessing the correctness of KG reasoning paths. This synergy enables effective KG exploration without fine-tuning or dedicated reward model engineering. The experiments on CWQ and WebQSP demonstrate that ReKG-MCTS achieves state-of-the-art performance among training-free methods and shows competitive performance with fine-tuned methods.

In summary, our contributions are threefold:

- We introduce a novel training-free framework, ReKG-MCTS, that incorporates MCTS to guide LLM-based reasoning on KGs, offering a flexible alternative to static or predefined search strategies.
- We demonstrate how LLMs can serve as both policy model and value method within the MCTS process for KG reasoning, thereby enhancing multi-hop search efficiency without additional fine-tuning or reward engineering.
- We conduct extensive evaluations on two benchmark datasets (CWQ and WebQSP), where ReKG-MCTS achieves state-of-the-art results among training-free approaches, providing interpretable and robust multi-hop reasoning paths.

2 Methods

2.1 Preliminaries

Notations We denote a knowledge graph as $\mathcal{G}$, which comprises triples of the form (e_s, r, e_o), representing subject entity e_s, relation r, and object entity e_o. We use E^D and R^D to represent the sets of entities and relations explored at depth D during the search process, respectively. Specifically, E^0 denotes the initial set of entities extracted from the question, and E^D (for $D \geq 1$) represents the set of entities reached after D reasoning steps. Individual reasoning paths are denoted as p_n, and the components of these paths at depth d are indexed as $e_{s,n}^d$, r_n^d, and $e_{o,n}^d$. The beam width used in exploration is denoted by N, indicating the top-N entities or relations considered at each step. We utilize a policy model π_ϕ to guide the search, and the reward value derived from the evaluation module is represented as v_k. T_q represents the search tree for a given question q, and nodes within this tree are denoted as C, with associated visit counts n_C and reward values v_C. D_{max} indicates the maximum tree search depth.

Problem Formulations The central problem addressed in this work is enhancing the knowledge graph reasoning capabilities of LLMs to answer complex questions (Pan et al., 2024). Formally, given a knowledge graph $\mathcal{G}$ consisting of triples (e_s, r, e_o) and a question x, the objective is to find a set of reasoning paths $P = \{(e_s^1, r^1, e_o^1), (e_s^2, r^2, e_o^2), \ldots, (e_s^D, r^D, e_o^D)\}$ within $\mathcal{G}$ that effectively answers x. Traditional approaches often rely on heuristic search methods or manual annotations to guide the search for such paths (Xiong et al., 2017; Xian et al., 2019; Wan et al., 2021; Zhong et al., 2023). However, these methods typically require extensive training data or carefully designed reward functions, which limits their scalability and adaptability. Given the recent advances in LLMs and their strong semantic understanding capabilities, we explore the possibility of leveraging LLMs to dynamically evaluate the reward value of reasoning paths, aiming to enable efficient and accurate knowledge graph reasoning without the aforementioned requirements (Shen et al., 2018; Zhu et al., 2021; Valmeekam et al., 2024).

2.2 Reasoning on Knowledge Graphs

KG represent structured information in the form of triples (e_s, r, e_o). Reasoning over a KG involves finding a sequence of triples, known as a reasoning path, that connects entities to answer a specific question. Formally, given a question x, the goal is to find reasoning paths on the KG that lead to the answer. Generally, reasoning on knowledge graphs to answer questions can be summarized into the following steps: initialization of search, graph exploration, and reasoning.

Initialization Given a question x, the process typically starts by identifying initial entities rele-

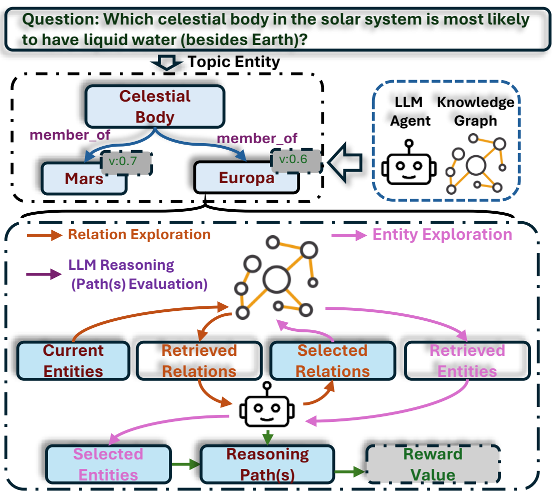


Figure 1: Visualization of graph exploration.

vant to the question to begin reasoning paths on the KG. This initialization step aims to identify a set of top-N entities as starting points for exploration: $E^0 = \{e_1^0, e_2^0, \ldots, e_N^0\}$. It's important to note that the number of initially extracted entities might be less than N if fewer relevant entities are identified.

Graph Exploration In each iteration of exploration, reasoning paths are extended step by step. At the beginning of the D-th iteration, each reasoning path p_n is composed of a sequence of $(D-1)$ triples. We can represent the d-th triple in the n-th reasoning path as $(e_{s,n}^d,\ r_n^d,\ e_{o,n}^d)$. Thus, the reasoning path p_n can be viewed as: $p_n = \left\{\left(e_{s,n}^d,\ r_n^d,\ e_{o,n}^d\right)\right\}_{d=1}^{D-1}$. The sets of entities and relations at the end of these paths at depth $D-1$ are represented as E^{D-1} and R^{D-1} respectively. The exploration phase aims to discover relevant entities E^D from the neighbors of the current entity set E^{D-1}, based on the question x, and to extend the reasoning paths. To manage the complexity, exploration is often structured into relation exploration and entity exploration (Sun et al., 2024; Markowitz et al., 2024). Following these steps, each reasoning path p_n is updated by appending the new relation r_n^D and entity e_n^D, thus further extending the reasoning paths. As the example illustrated in Figure 1, this typically involves two main steps at each depth D:

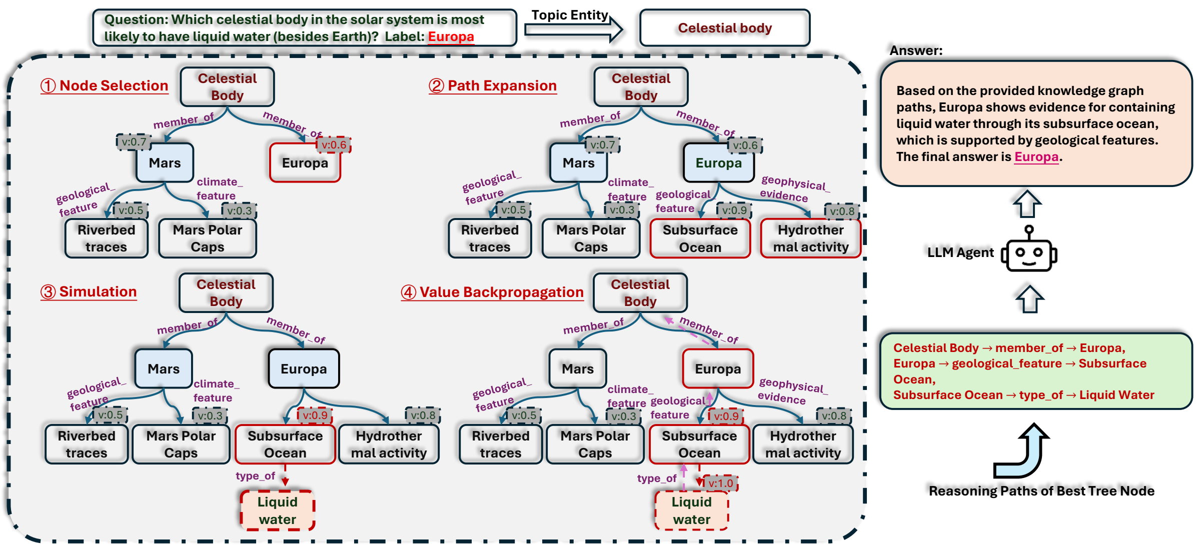
Relation Exploration: A beam search approach is often employed to transition from the set of entities E^{D-1} to a set of relations R^D. For each reasoning path p_n, candidate relations $R_{\text{cand},n}^D$ connected to the entity e_n^{D-1} are considered. From the aggregated candidate relations R_{cand}^D, a selection process is applied to identify the top-N relevant relations $R^D = \{r_1^D,\ r_2^D,\ \ldots,\ r_N^D\}$, based on their relevance to x and the current reasoning paths;

Entity Exploration: For each selected relation $r_n^D \in R^D$ and corresponding entity e_n^{D-1}, the KG is queried to find entities connected via r_n^D. Candidate entities are aggregated into the set E_{cand}^D by querying patterns like $(e_n^{D-1},\ r_n^D,\ ?)$ or $(?,\ r_n^D,\ e_n^{D-1})$. A selection process is then used to choose the top-N entities $E^D = \{e_1^D,\ e_2^D,\ \ldots,\ e_N^D\}$, based on their relevance to the question x and the extended reasoning paths. Following these steps, each reasoning path p_n is updated by appending the new relation r_n^D and entity e_n^D, thus further extending the reasoning paths.

Reasoning The reasoning process aims to find an answer to question x through iterative path evaluation and extension. At each depth D, we maintain a set of N most promising paths $\{p_1, ..., p_N\}$, where each path p_n consists of a sequence of triples $p_n = \{(e_{s,n}^1, r_n^1, e_{o,n}^1), ..., (e_{s,n}^D, r_n^D, e_{o,n}^D)\}$. For each path p_n, we compute a reward value $v_k = \mathit{Evaluate}(x, p_n)$ using the evaluation module, which leverages the LLM to assess both path relevance and answer sufficiency. Based on these evaluations, paths are extended according to the value method $\pi_\phi(e^{d+1}, r^{d+1}|x, p_n) \propto \exp(f_\phi(e^{d+1}, r^{d+1}, x, p_n))$, where $f_\phi(\cdot)$ is the scoring function implemented by the LLM. This process continues until either a path achieves a reward value above threshold τ or the maximum depth $D_{\max}$ is reached, at which point the final answer is further generated as $\mathit{answer} = \mathit{Reasoning}(x, p^*)$ using the best path p^*, thereby enabling dynamic exploration of the knowledge graph guided by the LLM's understanding.

2.3 ReKG-MCTS

In this section, we introduce ReKG-MCTS, a novel framework that integrates Monte Carlo Tree Search (MCTS) with Large Language Model (LLM) heuristics to efficiently reason over knowledge graphs. Our approach redefines question answering as a sequential decision-making problem. Given a knowledge graph $\mathcal{G}$ and a query x, the algorithm seeks an optimal reasoning trajectory p^* by guiding an LLM agent through a series of entity-to-entity transitions. Formally, we can define the objective as: $p^* = \arg\max_{p \in \mathcal{P}} \pi(p \mid x, \mathcal{G})$, where


Question: Which celestial body in the solar system is most likely to have liquid water (besides Earth)? Label: Europa

Answer:

Based on the provided knowledge graph paths, Europa shows evidence for containing liquid water through its subsurface ocean, which is supported by geological features. The final answer is Europa.

Celestial Body → member_of → Europa,
Europa → geological_feature → Subsurface Ocean,
Subsurface Ocean → type_of → Liquid Water

Figure 2: Illustration of the ReKG-MCTS reasoning process on a knowledge graph for answering the question "Which celestial body in the solar system is most likely to have liquid water?". The figure demonstrates a complete MCTS iteration. After MCTS completes, the LLM agent uses the best reasoning paths to generate the final answer.

$\mathcal{P}$ is the set of all plausible reasoning paths, and $\pi(p \mid x, \mathcal{G})$ denotes the LLM-derived evaluation of path p with respect to the query and knowledge graph context. To enhance the search process in knowledge graph reasoning, we leverage the classical Monte Carlo Tree Search (MCTS) framework, which naturally addresses several key challenges in path finding through its four phases: *Node Selection*, *Path Expansion*, *Simulation*, and *Backpropagation*. This integration is particularly beneficial as MCTS's systematic exploration-exploitation balance helps navigate the vast reasoning space, while its simulation-based evaluation provides a robust mechanism for assessing partial paths avoiding full traversal. In the subsequent sections, we detail each of these phases and describe how their integration enables effective KG-based reasoning.

Node Selection Node selection process initiates from the root node C_0 of the search tree T_q, which contains the initial entity set E^0 identified from question x. At each step, we traverse the tree by selecting nodes according to the Upper Confidence Bound (UCB) criterion. For a node C_i representing a partial reasoning path p_i at depth d, the UCB score combines both exploitation and exploration terms, formally expressed as $\text{UCB}(C_i) = \frac{v_{C_i}}{n_{C_i}} + c\sqrt{\frac{\ln n_p}{n_{C_i}}}$, where v_{C_i} denotes the cumulative reward value obtained from the evaluation module, n_{C_i} represents the visit count for node C_i, n_p is the visit count of the parent node, and c is the exploration coefficient. Through this formula, the selection strategy maintains a crucial balance between exploiting known high-reward paths and exploring less-visited branches of the tree. The node selection can be formally described as $C^* = argmax_{C_i \in \text{children}(C)} \text{UCB}(C_i)$, where the selected node C^* represents a partial reasoning path $p^* = \{(e^d_{s,*}, r^d_*, e^d_{o,*})\}^D_{d=1}$ that will be further expanded in the subsequent exploration phase, with $D \leq D_{\text{max}}$. This selection mechanism ensures efficient traversal of the search space while adaptively focusing on promising reasoning paths.

Path Expansion Path expansion phase is triggered when the selection process reaches a non-terminal leaf node C_l in the search tree T_q. At this stage, we systematically expand the tree by generating new child nodes that represent potential reasoning steps in the knowledge graph $\mathcal{G}$. The expansion process begins by examining the current entity e_{C_l} represented by node C_l and identifying all valid outgoing relations $\mathcal{R}(C_l) = \{r | (e_{C_l}, r, e') \in \mathcal{G}\}$. For each relation $r \in \mathcal{R}(C_l)$, we identify the set of target entities $\mathcal{E}(e_{C_l}, r) = \{e' | (e_{C_l}, r, e') \in \mathcal{G}\}$ that can be reached through r. This exploration can be formally expressed as the generation of new child nodes C_{new} for each valid triple (e_{C_l}, r, e'), where $e' \in \mathcal{E}(e_{C_l}, r)$. The complete path to each new node $\mathcal{P}_{C_{\text{new}}}$ is constructed by appending the new triple to the parent path: $\mathcal{P}_{C_{\text{new}}} = \mathcal{P}_{C_l} \cup \{(e_{C_l}, r, e')\}$. To evaluate the reward value of these newly expanded paths, we employ the LLM-based evaluator π_{eval}, which assesses the semantic alignment between the complete path $\mathcal{P}_{C_{\text{new}}}$ and the original question q. This evaluation is formalized as $v_{C_{\text{new}}} = \pi_{\text{eval}}(\mathcal{P}_{C_{\text{new}}}, q)$, where $v_{C_{\text{new}}}$ represents the initial reward value assigned to the new node. Additionally, we initialize the visit count $n_{C_{\text{new}}} = 1$ for each new node, establishing the foundation for future UCB-based selection decisions. This expansion mechanism ensures a systematic exploration of the knowledge graph while maintaining a balance between path relevance and search efficiency, as guided by the LLM's semantic understanding of the question-path relevance.

Simulation and Backpropagation Following the expansion of leaf node C_l, the simulation phase commences by employing a greedy MC rollout strategy that combines UCB-guided node selection with exploration. The process begins at C_l and iteratively selects subsequent nodes according to the UCB criterion until reaching a state where further selection is not possible. At this point, REKG-MCTS transitions into a greedy MC rollout phase guided by the policy module π_ϕ, which is also realized through LLM reasoning with task-specific prompts. For each step t in the rollout, given the current entity e_t and the accumulated path p_t, the policy model further generates a probability distribution over possible relations: $\pi_\phi(r_t|e_t, p_t, q, \mathcal{G})$. The next relation r_t is sampled from this distribution, and the corresponding target entity e_{t+1} is selected from the set $\mathcal{E}(e_t, r_t) = \{e'|(e_t, r_t, e') \in \mathcal{G}\}$. This process continues until either reaching a terminal state or the maximum depth $D_{\max}$, resulting in a complete path $p_{\text{sim}} = \{(e_t, r_t, e_{t+1})\}_{t=1}^{T}$, where $T \leq D_{\max}$. The reward value of this simulated path is then evaluated using the LLM-based evaluator: $v_k = \pi_{\text{eval}}(p_{\text{sim}}, q)$. The backpropagation phase updates the statistics of all nodes in the traversal path from root C_0 to leaf C_l. For each node C in this path, we update its cumulative value according to $v_C \leftarrow (v_C \cdot n_C + v_k)/(n_C + 1)$ and increment its visit count $n_C \leftarrow n_C + 1$.

Overview and Implementation Details Figure 2 illustrates an example of the REKG-MCTS reasoning process over a knowledge graph for answering a query. This recursive update mechanism ensures that the search tree's value estimates converge toward the true path qualities while maintaining exploration-exploitation balance through the UCB selection criterion. The complete algorithm iteratively applies these four phases - node selection, path expansion, simulation, and value backpropagation - until either a satisfactory reasoning path is found or the computational budget is exhausted (e.g., maximum iterations or tree depth). Relevant prompts, and detailed operators with the Knowledge Graph are described further in Appendix E. The pseudocode for the REKG-MCTS algorithm is formalized in Algorithm 1.

Algorithm 1 REKG-MCTS Algorithm

```
Require: KG $\mathcal{G}$, question $x$, max depth $D_{\max}$
Ensure: Optimal reasoning path $p^*$
 1: Initialize root node $C_0$ with entities from $x$
 2: while budget not exhausted do
 3:     $C_l \leftarrow \text{SelectNode}(C_0)$
 4:     if $C_l$ is non-terminal then
 5:         for each valid $(e_{C_l}, r, e')$ in $\mathcal{G}$ do
 6:             Expand $C_{\text{new}}$ with triple $(e_{C_l}, r, e')$
 7:             $p_{C_{\text{new}}} \leftarrow p_{C_l} \cup \{(e_{C_l}, r, e')\}$
 8:             $v_{C_{\text{new}}} \leftarrow \text{Evaluate}(x, p_{C_{\text{new}}})$
 9:         end for
10:     end if
11:     $p_{\text{sim}} \leftarrow \text{Simulate}(C_l, D_{\max})$
12:     $v_k \leftarrow \text{Evaluate}(x, p_{\text{sim}})$
13:     Backpropagate $v_k$ from $C_l$ to root
14: end while
15: return best path $p^*$ in search tree
16: function SELECTNODE($C$)
17:     while $C$ has children do
18:         $C \leftarrow \arg\max_{C_i} \frac{v_{C_i}}{n_{C_i}} + c\sqrt{\frac{\ln n_{\text{parent}}}{n_{C_i}}}$
19:     end while
20:     return $C$
21: end function
22: function SIMULATE($C_l, D_{\max}$)
23:     Initialize $p_{\text{sim}}$ with $p_{C_l}$
24:     while $d < D_{\max}$ do
25:         $r \sim \pi_\phi(r|e_s^d, p_{\text{sim}}, x, \mathcal{G})$
26:         Append $(e_s^d, r, e_o^{d+1})$ to $p_{\text{sim}}$
27:     end while
28:     return $p_{\text{sim}}$
29: end function
```

3 Experiments

3.1 Experimental Setup

Evaluation Metrics Following previous works (Li et al., 2024b; Jiang et al., 2023b; Sun et al., 2024), we use Hits@1 as our evaluation metric, which measures the proportion of questions whose top-1 predicted answer is correct.

Baselines We compare our approach with three main categories of baselines. The first category consists of fundamental LLM prompting techniques: standard input-output prompting (IO) (Brown et al., 2020), Chain-of-Thought (CoT) (Wei et al., 2022), and Self-Consistency (SC) (Wang et al., 2023). The second category focuses on LLM-enhanced KG reasoning methods that explicitly model graph structures, including RoG (Luo et al., 2024b), ToG (Sun et al., 2024), GoG (Xu et al., 2024a), and Tree-of-Traversals (Markowitz et al., 2024). The third category comprises retrieval-enhanced methods: KB-BINDER (Li et al., 2023), StructGPT (Jiang et al., 2023b), ChatKBQA (Luo et al., 2024a), and GNN-RAG (Mavromatis and Karypis, 2024), which employ different strategies for knowledge retrieval and integration.

LLMs We evaluate six LLMs across different scales: closed-source GPT-4 (Achiam et al., 2023) and open-source models including LLaMA-3-70B/8B (Dubey et al., 2024), Mistral-7B (Jiang et al., 2023a), Qwen-7B (Yang et al., 2024), and DeepSeek-7B (DeepSeek-AI et al., 2024). The GPT-4 model is accessed through OpenAI API. All open-source models are deployed using the official HuggingFace implementations with identical generation constraints.

Experimental Configurations The experiments are conducted on two complex question answering benchmarks: WebQSP (Talmor and Berant, 2018) and CWQ (Yih et al., 2016). Freebase (Bollacker et al., 2008) serves as the basic knowledge graph for both datasets. Each LLM is utilized in conjunction with different prompt designs as part of the π_{eval} and π_{ϕ} modules. Detailed setup regarding the experimental environment, datasets, specific parameter configurations for models' implementations are provided in Appendix A.

3.2 Main Results

As shown in Table 1, our experimental results across the two benchmark datasets clearly demonstrate that the proposed REKG-MCTS framework achieves superior performance compared to other baselines. In particular, REKG-MCTS attains superior performance in a training-free setting, and can effectively match or even surpass some of the performance of fine-tuned KG approaches. When compared to KG-based LLM reasoning approaches, REKG-MCTS exhibits a noticeable advantage. This result underscores the benefit of incorporating MCTS for graph exploration while integrating semantic cues provided by LLMs regarding path plausibility. Additionally, REKG-MCTS demonstrates promising progress in narrowing the gap with fine-tuned architectures on KG tasks. On the other hand, REKG-MCTS narrows the gap with fine-tuned models, outperforming GNN-RAG and approaching ChatKBQA, this further highlights how strategic KG exploration can compensate for the absence of model fine-tuning. By dynamically traversing the KG and harnessing evaluations from LLMs, REKG-MCTS enables more in-depth and precise multi-hop inference, ultimately validating its effectiveness.

Table 1: The Hits@1 scores of different models over two datasets under different settings (%).

Method	**CWQ**	**WebQSP**
w.t. Knowledge Graph / Fine-tuned		
RoG	66.1	<u>88.6</u>
GNN-RAG	66.8	85.7
ChatKBQA	<u>76.5</u>	78.1
w.t. Knowledge Graph / Training-Free		
KB-BINDER	-	50.7
StructGPT	-	76.4
ToG	71.0	80.3
GoG	75.2	84.4
Tree-of-Traversals	74.4	83.8
REKG-MCTS	**76.3**	**86.0**

3.3 Ablation Studies

We perform ablation studies to evaluate the individual components of the REKG-MCTS. We investigate the impact of different backbone models, the inclusion of generated knowledge graph (KG) paths, the choice of value methods, and various search tree configurations. Additionally, we provide a detailed error analysis in the Appendix D.

Performance of Different LLMs Table 2 presents several critical insights into both model scalability and method robustness, underscoring the effectiveness of our REKG-MCTS framework. Our experiments on WebQSP and CWQ reveal that the proposed approach consistently outperforms competitive baselines—such as GoG and Tree-of-Traversals—across various LLM sizes. Notably, while high-capacity models achieve the best absolute scores, the relative improvements delivered by REKG-MCTS are especially pronounced in smaller, 7B settings. In several cases, smaller models empowered by REKG-MCTS are even able to match or even outperform larger models. The architectural strengths of our framework, including its

Table 2: Hits@1 scores (%) of different methods with different LLM backbones on WebQSP and CWQ benchmarks.

	WebQSP (↑)					
Method	**GPT4**	**Llama3-70B**	**Llama3-8B**	**Mistral-7B**	**Qwen-7B**	**DeepSeek-7B**
IO prompt	68.5	61.4	55.2	55.9	54.1	52.8
CoT	74.6	63.1	57.8	59.0	56.3	55.1
CoT+SC	78.5	67.2	63.3	63.5	61.5	60.7
ToG	80.3	73.8	68.1	69.3	67.2	65.9
GoG	84.4	77.4	71.6	72.1	70.3	68.8
Tree-of-Traversals	83.8	75.5	69.0	70.5	68.5	67.1
ReKG-MCTS	**86.0**	**78.9**	**72.2**	**73.4**	**70.9**	**69.6**
	CWQ (↑)					
Method	**GPT4**	**Llama3-70B**	**Llama3-8B**	**Mistral-7B**	**Qwen-7B**	**DeepSeek-7B**
IO prompt	40.6	36.3	30.5	32.0	30.0	29.3
CoT	49.3	44.7	33.8	34.3	33.1	32.7
CoT+SC	53.9	46.4	36.9	38.7	36.4	35.2
ToG	71.0	58.5	51.8	53.1	51.9	50.9
GoG	75.2	62.6	56.5	58.2	56.9	55.6
Tree-of-Traversals	74.4	61.0	56.0	57.7	55.7	55.1
ReKG-MCTS	**76.3**	**64.4**	**59.8**	**61.0**	**60.1**	**58.3**

Table 3: Hits@1 scores w.t./w.o. Generated paths (%).

	WebQSP	**CWQ**
GoG w.o. Gen.	83.7	74.2
GoG w.t. Gen.	84.4	75.2
ReKG-MCTS w.o. Gen.	85.1	75.4
ReKG-MCTS w.t. Gen.	**86.0**	**76.3**

Table 4: Comparison of value methods in ReKG-MCTS with Llama3-8B for final answer reasoning.

Method	**CWQ**	**WebQSP**
BM25	52.5	60.3
SentenceBERT	53.7	65.9
Llama3-8B	**59.8**	**72.2**

MCTS-guided graph exploration and LLM-based path evaluation mechanisms, play a pivotal role in this model-agnostic performance. Moreover, on the more challenging CWQ dataset, the performance gap between ReKG-MCTS and ToG increases as the base model shrinks. This further demonstrates that ReKG-MCTS enhances LLM performance by effectively leveraging knowledge graph search to support multi-hop reasoning, even with smaller LLMs. Overall, these observations confirm that our strategy not only enhances multi-hop reasoning performance but also maintains a stable performance with varying model size.

Impact of Generated KG Paths Previous work in GoG validated the effectiveness of incorporating generated paths into the reasoning process (Xu et al., 2024a). Motivated by these findings, we further evaluated the generated path module within our framework. As evidenced in Table 3, while GoG gains modest improvements through generated paths, our framework, ReKG-MCTS, demonstrates even more favorable performance with improvements of 0.9% on both datasets. These results illustrate that the strategy of using generated paths is not only beneficial in the context of GoG, but also remains effective when integrated into our MCTS-based reasoning framework. These updated results indicate that augmenting the graph-based search with a generation module consistently enhances performance, albeit with modest gains. The generation mechanism appears to compensate for the missing edges in the knowledge graph, thereby improving the reasoning performance.

Impact of Value Methods Table 4 presents a comparative analysis of different value methods within the ReKG-MCTS framework. Traditional information retrieval approach BM25 (Robertson et al., 2009) serves as our baseline. The neural embedding-based SentenceBERT (Reimers, 2019) shows moderate improvements over BM25, with gains of 1.2% and 5.6% on CWQ and WebQSP respectively. Notably, Llama3-8B demonstrates substantial improvements, outperforming BM25 by

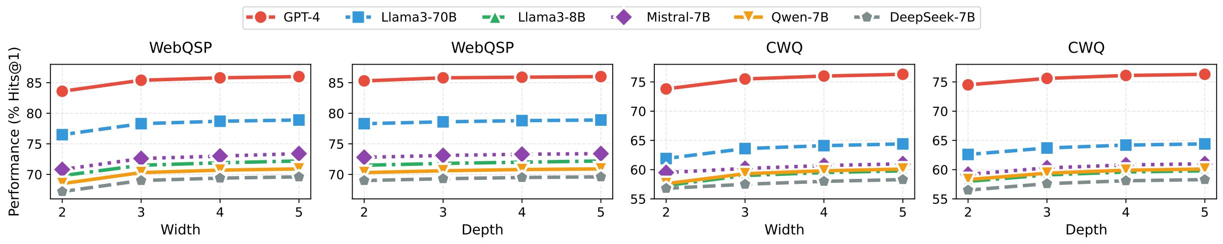


Figure 3: Impact of Search Tree Depth and Width on Performance across WebQSP and CWQ.

significant margins of 7.3% on CWQ and 11.9% on WebQSP. These results suggest that LLMs' sophisticated semantic understanding capabilities provide more effective value estimation compared to conventional retrieval or embedding-based methods.

Impact of Tree Configurations We vary the maximum depth and beam width of the search tree from 2 to 5 and report the resulting performance with various LLM backbones on both WebQSP and CWQ (Figure 3). Across all tested models (GPT-4, Llama3-70B, Llama3-8B, Mistral-7B, Qwen-7B, and DeepSeek-7B), the performance consistently improves as either the depth or width increases, reflecting the benefits of more extensive exploration. The gain is most pronounced when increasing these parameters from 2 to 3, while further enlarging the search yields diminishing returns. These findings underscore that carefully tuning the search tree's maximum depth and width is crucial for balancing computational overhead against the performance gains of multi-hop knowledge graph reasoning.

4 Related Work

4.1 Knowledge Base Question Answering

Knowledge Base Question Answering (KBQA) has progressed from Information Retrieval (IR) methods (Miller et al., 2016; Saxena et al., 2020; Lan et al., 2022), which extract facts from knowledge bases, to Semantic Parsing (SP) approaches that translate questions into logical forms like SPARQL queries. SP methods include step-wise query methods (Yih et al., 2015, 2016; Chen et al., 2019) and sequence-to-sequence models (Das et al., 2021; Ye et al., 2022) for directly generating logical forms. The advent of large language models (LLMs) has introduced both end-to-end and step-by-step methods. End-to-end approaches leverage in-context learning (Li et al., 2023) or fine-tuned LLMs (Luo et al., 2024b,a) to generate queries directly, while step-by-step methods (Sun et al., 2024; Xu et al., 2024a; Markowitz et al., 2024) follow a reasoning process to gradually find answers.

4.2 LLMs Reasoning with KGs

Large Language Models (LLMs) have demonstrated remarkable success in natural language processing tasks (Team et al., 2024; Achiam et al., 2023; Yang et al., 2024; DeepSeek-AI et al., 2024). While LLMs show strong reasoning capabilities through Chain-of-Thought prompting (Wei et al., 2022) and domain adaptation (Tan et al., 2024), their structured knowledge reasoning abilities remain an active research area. The integration of Knowledge Graphs (KGs) with LLMs has emerged as a promising direction to enhance structured reasoning capabilities (Pan et al., 2024; Hu et al., 2023). Recent work has particularly focused on developing efficient and effective KG-LLM integration methods. Notable frameworks including ToG, RoG, Tree-of-Traversals pioneered the combination of reasoning with external knowledge graphs, introduces structured exploration strategies for complex multi-hop reasoning (Sun et al., 2024; Luo et al., 2024b; Markowitz et al., 2024).

5 Conclusion

In this work, we introduce REKG-MCTS, a novel training-free framework that combines Monte Carlo Tree Search (MCTS) with LLMs to solve multi-hop reasoning tasks over KG. This approach frames KG reasoning as a decision-making process, where MCTS explores promising paths and LLMs provide semantic guidance on knowledge graphs. By leveraging this framework, REKG-MCTS enables effective multi-hop reasoning without the need for additional training or fine-tuning. Extensive experiments on WebQSP and CWQ demonstrate that REKG-MCTS not only outperforms existing training-free methods but also narrows the performance gap with fine-tuned approaches, indicating that structured search paradigms can effectively enhance LLMs' KG reasoning capabilities while preserving their semantic strengths.

Limitation

ReKG-MCTS demonstrates competitive performance on KG reasoning tasks but has several limitations. First, the iterative search process is computationally expensive compared to simpler retrieval methods, resulting in higher token usage. Second, the LLM's context window constrains our ability to handle complex queries with numerous entities or lengthy reasoning chains. Additionally, performance depends on both KG coverage and the LLM's parametric knowledge—incomplete graph information or LLM-generated spurious triples can lead to inaccuracies in the reasoning process.

Acknowledgements

This work was supported by National Science and Technology Major Project under Grant 2022ZD0116408, the Guangdong Provincial Key Laboratory of Mathematical Foundations for Artificial Intelligence (2023B1212010001), and a locally commissioned task from the Shanghai Municipal Government.

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*.

Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. 2024. Can knowledge graphs reduce hallucinations in llms?: A survey. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3947–3960.

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.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

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.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay-Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew Mccallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models.

Xinyan Guan, Yanjiang Liu, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. 2024. Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):18126–18134.

Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2023. A survey of knowledge enhanced pre-trained language models. *IEEE Transactions on Knowledge and Data Engineering*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2024a. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.

Yue Huang, Lichao Sun, Haoran Wang, Siyuan Wu, Qihui Zhang, Yuan Li, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, et al. 2024b. Position: Trustllm: Trustworthiness in large language models. In *International Conference on Machine Learning*, pages 20166–20270. PMLR.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023b. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. 2024. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11196–11215.

Ningke Li, Yuekang Li, Yi Liu, Ling Shi, Kailong Wang, and Haoyu Wang. 2024a. Drowzee: Metamorphic testing for fact-conflicting hallucination detection in large language models. *Proceedings of the ACM on Programming Languages*, 8(OOPSLA2):1843–1872.

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)*, pages 6966–6980.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024b. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *International Conference on Learning Representations*.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Anh Tuan Luu. 2024a. ChatKBQA: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2039–2056, Bangkok, Thailand. Association for Computational Linguistics.

Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024b. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Elan Markowitz, Anil Ramakrishna, Jwala Dhamala, Ninareh Mehrabi, Charith Peris, Rahul Gupta, Kai-Wei Chang, and Aram Galstyan. 2024. Tree-of-traversals: A zero-shot reasoning algorithm for augmenting black-box language models with knowledge graphs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12302–12319.

Ariana Martino, Michael Iannelli, and Coleen Truong. 2023. Knowledge injection to counter large language model (llm) hallucination. In *European Semantic Web Conference*, pages 182–185. Springer.

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

Nimrod Megiddo, S Louis Hakimi, Michael R Garey, David S Johnson, and Christos H Papadimitriou. 1988. The complexity of searching a graph. *Journal of the ACM (JACM)*, 35(1):18–44.

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 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

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*.

Ganesan Ramalingam and Thomas Reps. 1996. On the computational complexity of dynamic graph problems. *Theoretical Computer Science*, 158(1-2):233–277.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

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.

Amrith Setlur, Katie Kang, Aviral Kumar, Feryal Behbahani, Roberta Raileanu, and Rishabh Agarwal. 2025. Self-improving foundation models without human supervision. In *ICLR 2025 Workshop Proposals*.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. M-walk: Learning to walk over graphs using monte carlo tree search. *Advances in Neural Information Processing Systems*, 31.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. 2023. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562.

Alon Talmor and Jonathan Berant. 2018. 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.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 930–957.

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

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2024. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

Guojia Wan, Shirui Pan, Chen Gong, Chuan Zhou, and Gholamreza Haffari. 2021. Reasoning like human: Hierarchical reinforcement learning for knowledge graph reasoning. In *International Joint Conference on Artificial Intelligence*. International Joint Conference on Artificial Intelligence.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.

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.

Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 285–294.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Jun Zhao, and Kang Liu. 2024a. Generate-on-graph: Treat LLM as both agent and KG for incomplete knowledge graph question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18410–18430, Miami, Florida, USA. Association for Computational Linguistics.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024b. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Bingcong Xue and Lei Zou. 2022. Knowledge graph quality management: a comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4969–4988.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

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.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*.

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.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. ReST-MCTS*: LLM self-training via process reward guided tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Qinggang Zhang, Junnan Dong, Hao Chen, Daochen Zha, Zailiang Yu, and Xiao Huang. 2024b. Knowgpt: Knowledge graph based prompting for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Dengwei Zhao, Shikui Tu, and Lei Xu. 2024. Efficient retrosynthetic planning with mcts exploration enhanced a* search. *Communications Chemistry*, 7(1):52.

Peixiang Zhao and Jiawei Han. 2010. On graph query optimization in large networks. *Proceedings of the VLDB Endowment*, 3(1-2):340–351.

Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys*, 56(4):1–62.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34:29476–29490.

A Implementation Details

A.1 Dataset Statistics

In this work, we evaluate our KBQA methods on two widely used benchmarks, WebQSP (Talmor and Berant, 2018) and CWQ (Yih et al., 2016), both based on Freebase (Bollacker et al., 2008). WebQSP consists of natural language questions that require up to 2-hop reasoning, whereas CWQ contains more complex questions that call for up to 4-hop reasoning. In addition to the overall number of questions, both datasets provide rich annotations including logical form skeletons, participating entities, and relations. Table 5 summarizes the main statistics of the 2 datasets. All experiments were conducted on a Linux server equipped with 8 NVIDIA A100 80GB GPUs.

A.2 LLMs

GPT-4 (Achiam et al., 2023) represents the cutting-edge closed-source LLM developed by OpenAI, known for its robust natural language processing capabilities across a wide range of tasks, including text generation, summarization, question answering, and more.

LLaMA3-70B/8B (Dubey et al., 2024) is the 70B/8B-parameter version of the large language model series, Llama3 by Meta-AI. Built on a decoder-only transformer architecture, it features a 128K-token vocabulary and grouped query attention (GQA) to enhance inference efficiency, having been pretrained on over 15 trillion tokens.

Mistral-7B (Jiang et al., 2023a) is the 7B-parameter version of the large language model series, Mistral by Mistral-AI. It incorporates a sliding window attention mechanism with a rolling buffer cache, enabling efficient handling of long-range dependencies in text. This model leverages sparse attention patterns to effectively capture relationships in data while maintaining low inference latency.

Qwen-7B (Yang et al., 2024) is the 7B-parameter version of the large language model series, Qwen by Alibaba Cloud. Qwen-7B is a Transformer-based large language model, which is pretrained on a large volume of data over 2.4 trillion tokens.

DeepSeek-7B (DeepSeek-AI et al., 2024) is the 7B-parameter version of the large language model series, DeepSeek by DeepSeek-AI, it was trained on 2 trillion tokens in both English and Chinese.

B Baseline Details

B.1 Prompting Methods

IO (Brown et al., 2020) employs a straightforward prompting strategy for large language models (LLMs). It presents a query (input) followed directly by the desired response (output), relying on the model's pre-trained knowledge. While simple to implement, it may struggle with complex reasoning or multi-step inference tasks.

CoT (Wei et al., 2022) introduces a Chain-of-Thought prompting procedure that guides the model through intermediate reasoning steps. By revealing explicit chains of reasoning, CoT improves interpretability and enables better performance in tasks requiring multi-hop or compositional logic.

SC (Wang et al., 2023) (Self-Consistency) builds upon Chain-of-Thought by generating multiple reasoning paths and then selecting the most consistent answer. This approach aims to mitigate error propagation from a single chain of thought.

B.2 LLM-Enhanced KG Reasoning Methods

RoG (Luo et al., 2024b) (Reasoning over Graphs) leverages LLMs to generate and navigate the triplet-based structure of a knowledge graph (KG). By dynamically capturing node and relation contexts,

Table 5: Detailed statistics for WebQSP and CWQ datasets.

Statistics	**WebQSP**	**CWQ**
Basic Statistics		
#Questions	4,737	34,689
#Logical Form Skeletons	34	174
#Entities	2,461	11,422
#Relations	628	845
Max #Hop	2	4
Answer Distribution		
Single answer	51.2%	70.6%
2-4 answers	27.4%	19.4%
5-9 answers	8.3%	6.0%
$\geq$ 10 answers	12.1%	4.0%

RoG improves reasoning accuracy in scenarios with complex graph connectivity.

ToG (Sun et al., 2024) (Think-on-Graph) designs a step-by-step reasoning mechanism over KGs by incrementally tracking the entities and relations involved. This approach refines logical consistency at each step, enhancing interpretability for multi-hop queries. Nevertheless, its staged reasoning can increase computational overhead in large-scale graphs.

GoG (Xu et al., 2024a) (Generate on Graph) views KG reasoning as a generative process where intermediate nodes and edges are produced in a sequential manner. By encoding the graph structure within prompt-based generation, GoG aims to reduce ambiguity in entity-relation mapping.

Tree-of-Traversals (Markowitz et al., 2024) extends KG reasoning with a tree-structured expansion. Rather than following a single reasoning path, this method explores multiple branches in parallel.

B.3 Retrieval-Enhanced Methods

KB-BINDER (Li et al., 2023) combines large language models with minimal supervision by performing on-demand retrieval from a knowledge base (KB). It structures in-context learning templates to guide answer generation, striking a balance between efficiency and accuracy.

StructGPT (Jiang et al., 2023b) integrates structured knowledge retrieval with LLMs, focusing on the alignment of textual prompts with KB schema. By enforcing structural consistency during retrieval and generation, it reduces the risk of hallucinations in knowledge-intensive tasks.

ChatKBQA (Luo et al., 2024a) employs multi-turn dialogue to refine both retrieval and answer generation. It iteratively queries the KB to gather relevant facts, clarifying ambiguous or incomplete user inputs.

GNN-RAG (Mavromatis and Karypis, 2024) (Graph Neural Network-Retrieved Augmented Generation) couples a graph neural network-based retrieval module with a generation model, targeting improved discovery of relevant KB elements.

C Hyperparameter Settings

In this appendix, we detail the experimental hyperparameters employed by REKG-MCTS. During the exploration stage, each MCTS rollout is capped at 2 expansions to keep the computational overhead manageable, while the search depth $D_{\max}$ is also set to 5. The reward threshold τ is set to 0.8. The exploration coefficient c is set to 0.5 and 0.7 for the WebQSP and CWQ dataset, respectively. We restrict the number of candidate paths considered at each expansion step by employing a beam size of 2. We also use a TopK filtering strategy—setting TopK to 5 for relations entities when expanding relations and entities. This ensures that only the most relevant candidates are considered. During this stage, the LLM-based evaluator assesses the semantic relevance of each partial path and gives an initial reward score that helps guide backpropagation updates. Once exploration concludes, the final MCTS prediction stage retains many of the same parameters but narrows the beam size to 1, thereby focusing on the single most promising reasoning trajectory per rollout. The number of rollouts remains at 2 to preserve consistency with the exploration phase. All models evaluated in this framework are temperature constrained to 0.5 and have a maximum token length limited to 256.

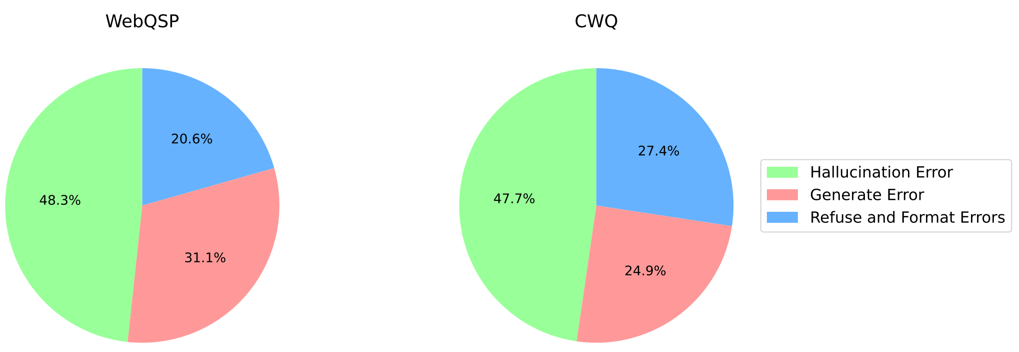


Figure 4: Error distributions.

D Error Analysis

In this section, we present three primary error types observed for proposed methods. These categories are selected based on the issues highlighted in the foregoing discussions regarding incomplete answers, incorrect information, or output misformatting.

Hallucination Error A hallucination error occurs when the Large Language Model (LLM) produces a final answer that lacks proper grounding in the knowledge graph or relevant context. In our experiments, hallucinations remain the most frequent source of incorrect results, as certain constraints and evidence from the intermediate reasoning paths are occasionally overlooked. Strengthening the link between contextual clues from the graph and the LLM's generation process can potentially mitigate such errors.

Refuse and Format Errors Refuse and format errors encompass two distinct but related issues in our framework. Refuse errors occur when the LLM declines to provide any answer, often citing insufficient information.

Format errors, on the other hand, manifest when the output structure deviates from the expected format, such as missing required components or presenting information in an inconsistent manner.

Generate Error Generate errors involve instances where the LLM produces partial or clearly wrong entities during the final answer generation. These errors may appear if the system fails to track the question throughout multiple reasoning hops, leading to a mismatch between the reasons uncovered in the search process and the final predicted entities.

Results Analysis Figure 4 presents a detailed breakdown of the errors observed in our system, revealing that hallucination errors are the most prevalent, followed by refuse and format errors, with generate errors occurring least frequently. The high incidence of hallucination errors suggests that, despite the grounding provided by the knowledge graph, the LLM sometimes produces answers that are insufficiently anchored to the available context. This observation motivates the need for tighter integration between the intermediate reasoning paths and the final answer generation process. In contrast, the moderate frequency of refuse and format errors points to challenges in the LLM's adherence to output constraints and its tendency to exercise undue caution, indicating potential benefits from refining the prompt design and output formatting guidelines. Finally, the relatively low occurrence of generate errors demonstrates that the system generally maintains context across multiple reasoning hops; however, occasional lapses in tracking the query details can still lead to incomplete or erroneous outputs.

E Prompts and SPARQL Operators

In this appendix, we document the prompt templates and SPARQL operators employed in our knowledge graph-based reasoning pipeline, drawing inspiration from best practices in large language model prompting and structured retrieval strategies.

E.1 SPARQL Operators

In this section, we introduce several SPARQL queries that enable relation discovery, entity retrieval, and label conversion for entities within the Freebase knowledge graph (Bollacker et al., 2008). These queries serve as building blocks for the datasets' knowledge graph traversal and entity resolution tasks. Below, we provide each query in a code block alongside brief descriptions.

Relations Extraction

```
# Extract head relations
sparql_head_relations = """
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT ?relation
WHERE {
  ns:%s ?relation ?x .
}
"""

# Extract tail relations
sparql_tail_relations = """
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT ?relation
WHERE {
  ?x ?relation ns:%s .
}
"""
```

These queries enable relation extraction in both directions. The head relations query retrieves all predicates where a given entity acts as the subject, while the tail relations query finds predicates where the entity appears as the object.

Entities Extraction

```
# Extract tail entities
sparql_tail_entities_extract = """
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT ?tailEntity
WHERE {
  ns:%s ns:%s ?tailEntity .
}
"""

# Extract head entities
sparql_head_entities_extract = """
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT ?tailEntity
WHERE {
  ?tailEntity ns:%s ns:%s  .
}
"""
```

These queries facilitate entity extraction in both directions of a relation.

Resolving Names and Aliases

```
sparql_ids = """
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT DISTINCT ?name
WHERE {
  {
    ns:%s ns:type.object.name ?name .  #
        Direct label retrieval
  }
  UNION
  {
    ns:%s <http://www.w3.org/2002/07/owl
        #sameAs> ?alias .  # Alias
        retrieval
    ?alias ns:type.object.name ?name .
  }
}
"""
```

In some cases, an entity may have multiple names or aliases. This query unifies those labels by exploring owl:sameAs relationships and retrieving the corresponding human-readable names.

E.2 Prompts

Path Value Evaluation Template

The `EVALUATE_STATE_PROMPT`, as shown in Figure 5, is designed to rigorously assess whether a selected knowledge graph triplet path contributes to resolving the original query. It provides a structured rubric to score the relevance and usefulness of the triplet, factoring in direct or indirect references to key question entities, the specificity of the presented facts, and the potential for bridging knowledge gaps. A final numeric rating, alongside a concise explanatory note, is returned to signal how effectively the triplet aids in progressing toward the solution.

Extract Relation Prompt

When identifying which knowledge graph relations matter most to the user query, `EXTRACT_RELATION_PROMPT`, as shown in Figure 6 ranks candidate edges by their potential contribution. This helps allocate search resources efficiently in complex graphs or when domain knowledge is sparse. The system prompts for a user-facing explanation of how the chosen relations connect, assigning scores to each that sum to 1.

Entity Extract Prompt

The `EXTRACT_ENTITY_PROMPT`, as shown in Figure 7 helps identify which entities are worth investigating further. When given a user's question, the current discussion context, and a list of potential entities to explore next, this template creates a clear ranking to help the system focus on the most relevant and important options. This hierarchical selection echoes typical beam-search–style expansions in knowledge graph traversal, limiting combinatorial explosion in large graphs.

Answer Prompt

The `ANSWER_PROMPT` is designed for straightforward question answering from both local knowledge graph data and any relevant external knowledge. It outlines a template for structured reasoning and the formulation of a final response. When information from the knowledge graph is insufficient, the system is permitted to incorporate general background knowledge to fill in gaps.

E.3 LLM-integrated KG Query

This section details the integration of Large Language Models (LLMs) with SPARQL queries for entity and relation selection, leveraging predefined prompting templates and SPARQL operators.

Relation Selection

The `relation_select` function refines which relations to explore once entities have been chosen. It achieves this in two distinct steps:

1. **SPARQL Retrieval:** The function runs two SPARQL queries, `sparql_head_relations` and `sparql_tail_relations`, to collect all potential predicates where the entities appear as either the subject or the object. This step generates a comprehensive list of candidate relations for graph traversal.

2. **LLM-based Relation Selection:** Once the SPARQL queries are executed, the next step involves using `EXTRACT_RELATION_PROMPT` in conjunction with a large language model (LLM) to evaluate and rank the relevance of the extracted relations. The LLM assigns relevance scores to each relation, and these scores are used to select the most promising candidates for further exploration. This ranking process ensures that the top N most contextually relevant relations are prioritized in subsequent SPARQL queries.

This process combines SPARQL queries with LLM's understanding to effectively search the knowledge graph, helping find relevant relations.

Entity Selection

The `entity_select` function refines which entities to explore once relations have been chosen. It achieves this in two distinct steps:

1. **SPARQL Retrieval:** The function begins by executing SPARQL queries to extract all connected candidate entities based on the chosen relations from the previous step. This process retrieves a comprehensive set of entities from the knowledge graph that are linked to the current entities.

2. **LLM-based Entity Selection:** Next, the function utilizes the `EXTRACT_ENTITY_PROMPT` in combination with a large language model (LLM) to evaluate and rank the candidate entities. By incorporating the current question context, candidate entities, and path history into the prompt, the LLM assesses the relevance of each entity and selects the top N most promising candidates.

This process combines SPARQL queries with LLM's understanding to effectively search the knowledge graph, helping find relevant entities.

F Future Work

Future work could address these challenges by integrating advanced aggregation functions or specialized operators, allowing the framework to handle queries involving complex numeric reasoning (e.g., "How many mountains are above 3500 meters?"). Incorporating domain-specific knowledge bases—such as medical or legal corpora—would further extend the utility of our approach to specialized, high-stakes applications.

Reinforcement learning (RL) techniques could also be explored to continually refine the search policy module. By using a LLM model as a search agent through RL to discover optimal reasoning paths in knowledge graphs ($e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} ... \xrightarrow{r_n} e_{target}$), we can obtain high-reward reasoning trajectories. These trajectories can then be utilized for supervised fine-tuning of the language model, establishing a synergistic cycle where the RL agent explores enhanced reasoning strategies while the fine-tuned LLM generates more accurate candidate triples.

G Broader Impact

The proposed REKG-MCTS framework presents a novel framework on reasoning over knowledge graphs using Large Language Models (LLMs) and Monte Carlo Tree Search (MCTS). Its positive impacts include improving question answering systems, enabling effective knowledge graph exploration, and empowering smaller models to perform complex tasks. However, challenges such as computational overhead, dependency on the quality of knowledge graphs, and potential biases must be carefully managed. Future developments could focus on expanding the framework to specialized domains and addressing ethical concerns, ensuring that AI systems are transparent, fair, and beneficial.

PATH_VALUE_EVALUATION_PROMPT:

Prompt Template:
Your task is to rigorously evaluate whether the selected triplet from the knowledge graph is useful for reasoning toward answering the given question. Follow these steps carefully:
EVALUATION CRITERIA:
1. Does the triplet directly or indirectly mention entities or relationships from the question? If indirect, is the connection clear and meaningful?

2. Does the triplet provide specific information that helps narrow down or answer the question? Is the information sufficient to make progress toward the answer, or is it too vague or tangential?

3. Do the triplets logically necessary or strongly supportive for constructing a reasoning path to the answer? Does it fill a critical gap in the reasoning process?

SCORING GUIDELINES:
0.0-0.3: The triplet is irrelevant or provides no meaningful contribution to answering the question.
0.4-0.6: The triplet is somewhat relevant but only loosely connected or provides minimal information.
0.7-0.8: The triplet is relevant and contributes to the reasoning process but is not decisive or critical.
0.9-1.0: The triplet provids a clear, unambiguous, and direct answer to the question. It must fully resolve the question or provide a critical piece of information that leaves no room for ambiguity or further reasoning.
Important: The presence of entities from the question alone does not guarantee relevance. The triplet must actively help resolve the question, not just include related triplets' entities.
OUTPUT FORMAT: Provide a score between 0.0 and 1.0 with one decimal place. Include a concise explanation justifying the score based on the evaluation criteria.
EXAMPLES:
Q: The artist nominated for The Long Winter lived where?
T: Laura Ingalls Wilder -> people.person.places_lived -> Unknown
RATING [0.0-1.0]: 0.5
EXPLANATION:
The triplet connects the author (key entity) to places_lived, but "Unknown" value makes it incomplete. While it indicates the KG has author residency data, the lack of concrete location requires supplementing with external knowledge bridging.

Q: What is the official language of the country where Hokkaido is located?
T: Hokkaido -> location.administrative_division.country -> Japan
RATING [0.0-1.0]: 0.8
EXPLANATION:
This triple establishes the direct relationship between the administrative division and its country. Combined with common knowledge that Japanese is the official language of Japan, this connection provides the key information needed to answer the language question.
<Other Examples...>

Q: {question}
T: {triple}
RATING [0.0-1.0]:"""

Figure 5: Path Value Evaluation Prompt.

EXTRACT_RELATION_PROMPT:

Prompt Template:
Please retrieve N relations (separated by semicolon) that contribute to the question and rate their contribution on a scale from 0 to 1 (the sum of the scores of N relations is 1).
Q: Name the president of the country whose main spoken language was Brahui in 1980?
Topic Entity: Brahui Language
Relations: language.human_language.main_country; language.human_language.language_family; language.human_language.iso_639_3_code; base.rosetta.languoid.parent; language.human_language.writing_system; base.rosetta.languoid.languoid_class; language.human_language.countries_spoken_in; kg.object_profile.prominent_type; base.rosetta.languoid.document; base.ontologies.ontology_instance.equivalent_instances; base.rosetta.languoid.local_name; language.human_language.region
A:
1. language.human_language.main_country (Score: 0.4)): ...
2. language.human_language.countries_spoken_in (Score: 0.3): ...
3. base.rosetta.languoid.parent (Score: 0.2): ...

Q: {Question}
Topic Entity: {Topic Entities}
Relations: {Selected Relations}
A:

Figure 6: Relation Extract Prompt.

ENTITY_EXTRACT_PROMPT:

Entity Extraction Prompt:
Based on the question and path history, filter the most relevant `{top_k}` entities from the candidate entities.

Input Format:

- **Question:** `{question}`
- **Current Entity:** `{current_entities}`
- **Current Relation:** `{current_relation}`
- **Path History:** `{path_history}`
- **Candidate Entities:** `{candidate_names}`

Output Requirement:
Please output the entity names in descending order of relevance, separated by commas.

Figure 7: Entity Extract Prompt.

ANSWER_PROMPT:

Answer Prompt:
Given a question and the associated retrieved knowledge graph triplets (entity, relation, entity), you are asked to answer the question using these triplets and your own knowledge. Some triplets may be irrelevant or insufficient, in which case you should rely on your internal knowledge to reason logically.

Answer Format:
Combine the provided triplets with your knowledge to perform comprehensive reasoning.
End your answer with a summary starting with "The final answer is...".

Reasoning Guidance:
First examine if the triplets provide direct relevant information.
If triplets are insufficient or irrelevant, clearly explain how you use your internal knowledge.
Make logical connections based on both given information and commonly known facts.
Provide clear reasoning steps when bridging information gaps.

Q: What is the capital of France?
Knowledge Triplets: France $\rightarrow$ has.city $\rightarrow$ Lyon, France $\rightarrow$ exports.wine $\rightarrow$ Italy
A: While the triplets mention France but don't provide information about its capital, from common knowledge we know that Paris is the capital of France. The given triplets about Lyon and wine exports are not relevant to this question.
The final answer is Paris.

<Other Examples...>

Q: {Question}
A:

Figure 8: Answer Prompt.