RJE: A Retrieval-Judgment-Exploration Framework for Efficient Knowledge Graph Question Answering with LLMs

Can Lin[1]*, Zhengwang Jiang[1]*, Ling Zheng[1], Qi Zhao[1], Yuhang Zhang[1,2], Qi Song[1,3]†, Wangqiu Zhou[4]

[1]University of Science and Technology of China
[2]City University of Hong Kong
[3]Deqing Alpha Innovation Institute, Huzhou, China
[4]Hefei University of Technology

{can.lin, jzw02, lingzheng, zq2021, yhzhang}@mail.ustc.edu.cn
qisong09@ustc.edu.cn, rafazwq@hfut.edu.cn

Abstract

Knowledge graph question answering (KGQA) aims to answer natural language questions using knowledge graphs. Recent research leverages large language models (LLMs) to enhance KGQA reasoning, but faces limitations: retrieval-based methods are constrained by the quality of retrieved information, while agent-based methods rely heavily on proprietary LLMs. To address these limitations, we propose Retrieval-Judgment-Exploration (RJE), a framework that retrieves refined reasoning paths, evaluates their sufficiency, and conditionally explores additional evidence. Moreover, RJE introduces specialized auxiliary modules enabling small-sized LLMs to perform effectively: Reasoning Path Ranking, Question Decomposition, and Retriever-assisted Exploration. Experiments show that our approach with proprietary LLMs (such as GPT-4o-mini) outperforms existing baselines while enabling small open-source LLMs (such as 3B and 8B parameters) to achieve competitive results without fine-tuning LLMs. Additionally, RJE substantially reduces the number of LLM calls and token usage compared to agent-based methods, yielding significant efficiency improvements.[1]

1 Introduction

Knowledge graph question answering (KGQA) aims to find answer entities from knowledge graphs (KGs) in response to natural language questions (Jiang et al., 2023c). With the development of open domain knowledge graphs, such as Freebase (Bollacker et al., 2008) and Wikidata (Pellissier Tanon et al., 2016), KGQA has become an important research topic. Although multi-hop KGQA has been studied (Zhang et al., 2022; Ji et al., 2024; Mavromatis and Karypis, 2024), finding complex multi-hop reasoning edges to infer correct answers remains a challenge.

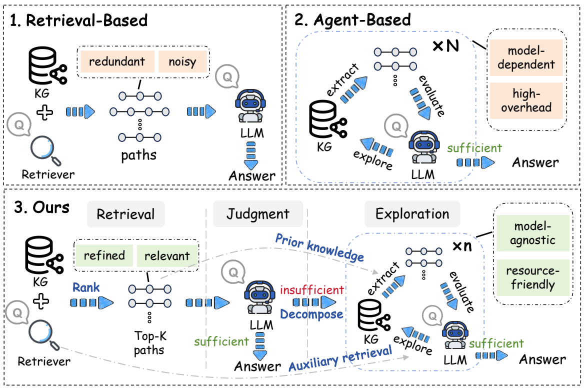


Figure 1: A comparison of three types of methods: retrieval-based methods, agent-based methods, and our proposed RJE that combines precise retrieval with conditional exploration.

Concurrently, large language models (LLMs) have demonstrated remarkable performance across various natural language processing (NLP) tasks (Huang and Chang, 2023; Grattafiori et al., 2024). This convergence has stimulated growing research into methodologies that leverage LLMs to enhance KGQA systems (Gao et al., 2023; Zhang et al., 2025). One straightforward approach is to fine-tune LLMs (Luo et al., 2024b; Jiang et al., 2024), which incurs significant costs and risks catastrophic forgetting (Luo et al., 2023). As illustrated in Figure 1, an alternative class of methods does not require parameter modification of LLMs. They can be categorized as two types:

1) **Retrieval-based methods** incorporate external retrieval mechanisms to extract relevant KG evidence, which is then integrated into textual prompts for LLMs to generate answers grounded in the retrieved knowledge (He et al., 2024; Huang and Zeng, 2024; Li et al., 2025). However, **Constraints of retrieval information** limit the effectiveness of these methods. Retrieved information

[*] Equal contribution.
[†] Corresponding author.
[1] Code and data are available at: https://github.com/Olgird/RJE

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 17277–17294
November 4-9, 2025 ©2025 Association for Computational Linguistics

often struggles to simultaneously accommodate sufficiency and relevance. Insufficient evidence prevents correct inference, while excessive information introduces noise that can derail the reasoning process (Huang and Zeng, 2024).

2) **Agent-based methods** employ LLMs as agents that deliberately navigate the KG. Starting with question topic entities, LLMs systematically select candidate relations and entities, and then evaluate them in an iterative process until they reach an appropriate answer (Jiang et al., 2023a; Sun et al., 2024; Chen et al., 2024). These methods depend heavily on **proprietary LLMs**. They typically operate without initially provided external information, forcing LLMs to handle multi-step exploration independently. Additionally, exploration complexity increases substantially when reasoning over multiple topic entities and navigating extensive entities and relations in KGs. As a result, practical success has predominantly relied on proprietary LLMs like GPT-4, while smaller open-source alternatives demonstrate significantly lower performance (Li et al., 2024).

Above limitations led to a natural question: can we achieve *more accurate reasoning with lower cost*? We answer this question by proposing a three-stage **R**etrieval-**J**udgment-**E**xploration (RJE) framework. As shown in Figure 1, the RJE framework strategically synergizes KG retrieval with reasoning capabilities of LLMs by first retrieving refined reasoning paths from the KG, then employing LLMs to judge information sufficiency and finally executing targeted exploration as needed.

Retrieval: the retrieval stage of RJE prioritizes relevance over sufficiency of retrieved information. The retriever first extracts numerous reasoning paths from the KG based on relevance between the question and relations, where each reasoning path starts from a topic entity and consists of alternating relations and entities. To reduce noise and preserve the relevance, we introduce a lightweight *Reasoning Path Ranking* module to further sort reasoning paths based on relevance scores, and only the top-K reasoning paths are retained.

Judgment: an LLM serves as a judge to evaluate whether the retained reasoning paths provide sufficient evidence to answer the question. If indeed sufficient, the LLM directly formulates a response; otherwise, an exploration phase will be executed to fetch the required information.

Exploration: unlike prior agent-based methods (Sun et al., 2024; Chen et al., 2024), the exploration stage of RJE is dedicated to completing missing evidence and simplifying exploration. When evidence is insufficient, the LLM acts as an exploration agent that first executes *Question Decomposition* which breaks down a complex question into simpler subquestions based on topic entities to focus on specific reasoning paths. RJE then utilizes retrieved reasoning paths as prior knowledge, allowing the LLM to initiate exploration from entities at knowledge gaps rather than topic entities, thereby reducing exploration steps. Subsequently, it iteratively conducts *Retriever-assisted Exploration*, where the retriever pre-filters candidate relations to constrain the search space before the LLM conducts targeted relation and entity exploration to complete the missing evidence chain.

Our contributions can be summarized as follows:

- We introduce a novel framework called RJE that integrates precise retrieval, sufficiency judgment, and conditional exploration for KGQA. This framework simultaneously enhances reasoning capabilities and reduces computational demands.

- We introduce three key auxiliary modules: Reasoning Path Ranking, Question Decomposition, and Retriever-assisted Exploration. These modules reduce the reasoning burden on LLMs, enabling small-sized LLMs to achieve performance comparable to prior work using proprietary models.

- We conduct comprehensive experiments across standard KGQA benchmarks, demonstrating that RJE surpasses existing approaches in both accuracy and efficiency when using proprietary LLMs. Notably, with small open-source LLMs (3B and 8B parameters), RJE outperforms the previous state-of-the-art method PoG by 41.5% and 27.9% on the CWQ dataset under the same model sizes.

2 Related Work

The task of KGQA focuses on answering questions by leveraging information from KGs. We categorize existing approaches as follows.

Semantic parsing methods convert natural language questions into structured queries (e.g., SPARQL) or equivalent forms for execution (Cao et al., 2022; Hu et al., 2022; Zhang et al., 2023; Luo et al., 2024a). Despite their precision, these approaches often require costly logical form annotations and remain vulnerable to execution failures stemming from syntactic or semantic errors.

Retrieval-based methods typically comprise a retrieval module and a reasoning module. Early approaches (Zhang et al., 2022; Jiang et al., 2023c; Baek et al., 2023a; Li et al., 2023; Jiang et al., 2023b; Ding et al., 2024) employed pre-trained models for retrieval paired with lightweight reasoning components. More recent work applies retrieval-augmented generation (RAG) with LLMs, utilizing retrieved subgraphs as contextual prompts (Baek et al., 2023b; He et al., 2024; Zhao et al., 2024; Mavromatis and Karypis, 2024; Huang and Zeng, 2024; Li et al., 2025). However, these methods struggle to balance knowledge sufficiency and relevance, leading to either incomplete evidence for inference or noise that disrupts reasoning.

Agent-based methods extend beyond standard RAG implementations. Some studies explore fine-tuning LLMs to enhance their KG reasoning capabilities, such as RoG (Luo et al., 2024b) and KG-Agent (Jiang et al., 2024), though this strategy incurs significant computational costs and increases the risk of catastrophic forgetting (Luo et al., 2023). Alternatively, iterative prompting methods (Jiang et al., 2023a; Cheng et al., 2024; Markowitz et al., 2024; Sun et al., 2024; Chen et al., 2024; Wang et al., 2025) leverage LLMs as agents to progressively retrieve relevant knowledge, with frameworks like ToG (Sun et al., 2024) and PoG (Chen et al., 2024) incorporating advanced strategies such as beam search, memory mechanisms, and reflection capabilities. However, these approaches often introduce additional system complexity, and small-sized LLMs continue to exhibit significant performance limitations.

3 Preliminary

Knowledge Graph (KG) is a structured semantic knowledge base, denoted as $\mathcal{G} = (E, R)$, where E and R represent the set of entities and relations respectively. A triple $\tau = (e, r, e')$ describes a fact with $e, e' \in E$ and $r \in R$.

Knowledge Graph Question Answering (KGQA) aims to answer natural language questions using KGs. Formally, given a natural language question q and a knowledge graph $\mathcal{G}$, the task of KGQA is to identify the corresponding answer entity subset $A_q \subseteq E$ that satisfies the question by leveraging the structural information in $\mathcal{G}$.

Topic Entities are the main entities mentioned in a question q that serve as starting points for answer retrieval. Following prior work (Sun et al., 2024; Chen et al., 2024; Huang and Zeng, 2024), we assume that the set of topic entities T has been identified from the question and successfully linked to nodes in KGs.

Relation Path is defined as a sequence of relations starting from a topic entity e_t. It is denoted as $p_r = (e_t, r_1, r_2, \ldots, r_h)$, where h represents the length of the relation path.

Reasoning Path refers to a complete path of alternating entities and relations, derived from a given relation path. It is formally represented as $p_e = (e_t, r_1, e_1, r_2, \ldots, r_h, e_h)$, where each entity e_{i-1} is connected to the next entity e_i via relation r_i. Note that a single relation path may derive multiple reasoning paths based on the entity connections within the KG. For instance, $p'_e = (e_t, r_1, e'_1, r_2, \ldots, r_h, e'_h)$ constitutes an alternative reasoning path from the same relation path.

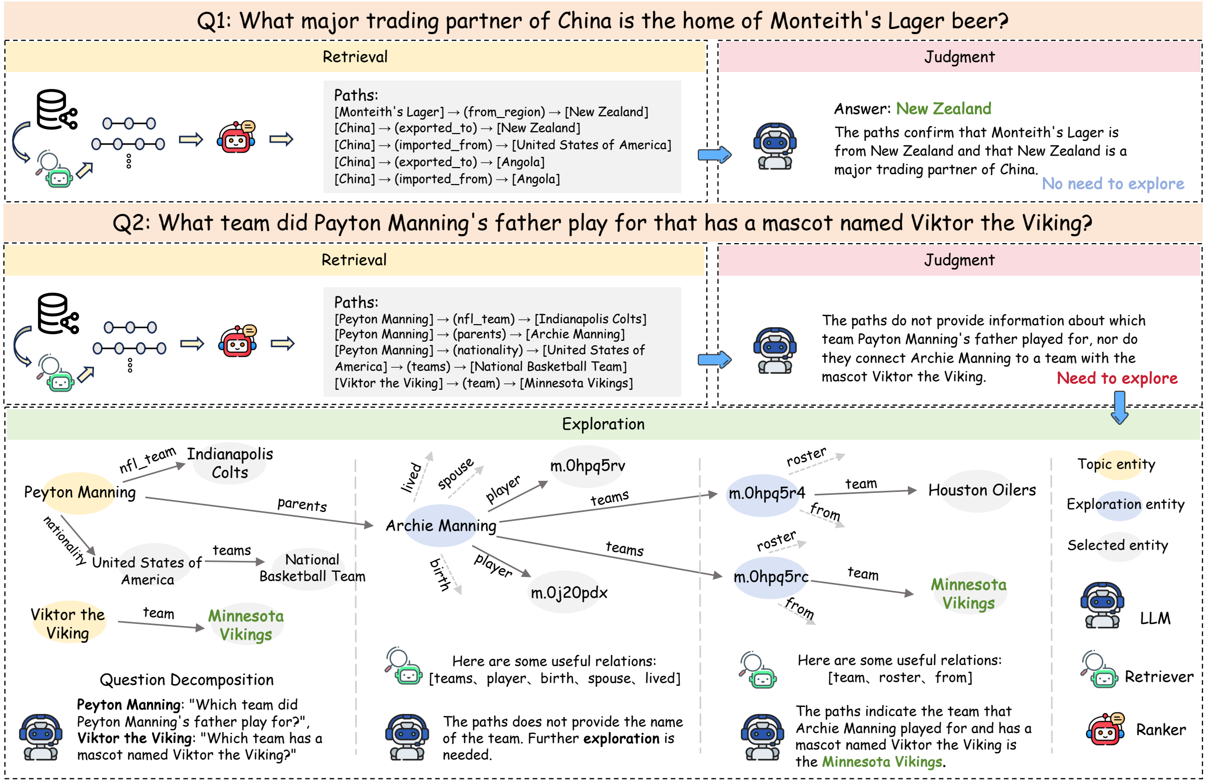
4 Methodology

This section provides a detailed description of each stage in the RJE framework and its auxiliary modules. As illustrated in Figure 2, RJE consists of three stages: Retrieval, Judgment, and Exploration. Retrieval stage: the relation path retriever first identifies relevant relation paths from the KG, which are then refined by the reasoning path ranking module to obtain the top-K reasoning paths. Judgment stage: the top-K reasoning paths are prompted to LLM, which evaluates whether the available evidence is sufficient to answer the question. If deemed sufficient, the LLM proceeds to generate an answer. Exploration stage: when the current paths information is insufficient, further exploration is required. Firstly, the LLM performs question decomposition to guide further exploration. Then through an iterative process of exploration, the LLM gathers additional evidence until it determines sufficient information has been accumulated to generate the final answer.

4.1 Retrieval Stage

4.1.1 Relation Path Retrieval

In order to preliminarily extract information relevant to the question from the KG, we implement the relation path retrieval as the first step of retrieval stage. Following the approach proposed by Zhang et al. (2022) and Huang and Zeng (2024), we fine-



Paths:
[Monteith's Lager] → (from_region) → [New Zealand]
[China] → (exported_to) → [New Zealand]
[China] → (imported_from) → [United States of America]
[China] → (exported_to) → [Angola]
[China] → (imported_from) → [Angola]

The paths confirm that Monteith's Lager is from New Zealand and that New Zealand is a major trading partner of China.

Paths:
[Peyton Manning] → (nfl_team) → [Indianapolis Colts]
[Peyton Manning] → (parents) → [Archie Manning]
[Peyton Manning] → (nationality) → [United States of America] → (teams) → [National Basketball Team]
[Viktor the Viking] → (team) → [Minnesota Vikings]

The paths do not provide information about which team Payton Manning's father played for, nor do they connect Archie Manning to a team with the mascot Viktor the Viking.

Peyton Manning: "Which team did Peyton Manning's father play for?", **Viktor the Viking**: "Which team has a mascot named Viktor the Viking?"

Here are some useful relations: [teams、player、birth、spouse、lived]

The paths does not provide the name of the team. Further **exploration** is needed.

Here are some useful relations: [team、roster、from]

The paths indicate the team that Archie Manning played for and has a mascot named Viktor the Viking is the Minnesota Vikings.

Figure 2: The framework overview of RJE, which flexibly adjusts its strategy based on the sufficiency of path evidence to minimize resource consumption while ensuring answer correctness.

tune a pre-trained language model as relation path retriever based on the question q, topic entities T, and answers A. This enables the model to capture the semantic similarity between the question and relevant relations. During retrieval, the model concatenates the question with the topic entity, employs beam search iteratively to assess semantic similarity between the question and neighboring candidate relations, and produces a set of relation paths $P_r = \{p_{r_0}, p_{r_1}, ..., p_{r_{|P_r|-1}}\}$, each with a corresponding relevance score, where $|P_r|$ is the number of selected relation paths.

4.1.2 Reasoning Path Ranking

Depending on entity connections in the KG, a set of reasoning paths $P_e = \{p_{e_0}, p_{e_1}, ..., p_{e_{|P_e|-1}}\}$ is derived from P_r. To prevent overwhelming downstream LLMs with excessive reasoning paths, we introduce a reasoning path ranking module that employs a relatively small reasoning path ranker based on a pre-trained language model (PLM) to prioritize relevant reasoning paths.

For a specific reasoning path represented as $p_{e_i} = (e_t, r_1, e_1, r_2, e_2, ..., r_h, e_h)$, the ranker constructs an input sequence by concatenating the question and the path:

$$q_i = \{q\ [\mathrm{SEP}]\ e_t \rightarrow r_1 \rightarrow e_1 \rightarrow \cdots \rightarrow e_h\}. \quad (1)$$

Here, [SEP] is the separator token used by the PLM, and the symbol "→" indicates transition within the path. The sequence q_i is fed into the PLM to obtain a relevance score:

$$s = \mathrm{MLP}(E(q_i)), \quad (2)$$

where $E(\cdot)$ denotes the embedding obtained from the PLM by extracting the [CLS] token representation, and MLP is a multi-layer perceptron applied on top of the embedding. To ensure the relevance of extracted information and suppress irrelevant noise, we only select the top-K reasoning paths based on relevance scores. As shown in Figure 2, for the question Q2 *"What team did Payton Manning's father play for that has a mascot named Viktor the Viking?"*, only a few paths most relevant to the topic entities *"Peyton Manning"* and *"Viktor the Viking"* are selected.

The ranker is trained with weak supervision, requiring only the question q and the corresponding answer set A. For a reasoning path p_{e_i}, if there exists an answer $a \in A$ along that path, it is labeled as a positive sample. Otherwise, it is treated as a negative sample. The training objective adopts the margin ranking loss, defined as:

$$\mathcal{L} = \max(0, s_{\text{neg}} - s_{\text{pos}} + \text{margin}), \quad (3)$$

where s_{neg} and s_{pos} denote the scores for negative and positive samples, respectively, and margin is a hyperparameter enforcing a minimum separation between them.

4.2 Judgment Stage

After extracting the top-K reasoning paths, we employ an LLM-based judgment stage to assess information sufficiency. RJE feeds the paths into the LLM, which then evaluates whether these paths contain sufficient information to answer the question. If deemed sufficient, the LLM proceeds to generate answer. Otherwise, RJE identifies the need for further exploration. As shown in Figure 2, for the question Q1 *"What major trading partner of China is the home of Monteith's Lager beer?"*, RJE successfully infers the correct answer by judging that the paths information is sufficient. In contrast, for the question Q2 *"What team did Payton Manning's father play for that has a mascot named Viktor the Viking?"*, RJE identifies the insufficiency of the available information and proceeds with further exploration. These two examples demonstrate the flexibility and adaptability of the RJE framework. The prompt is provided in Appendix J.1.

4.3 Exploration Stage

4.3.1 Question Decomposition

Given a question q and its corresponding set of topic entities $T = \{e_{t_1}, e_{t_2}, \dots, e_{t_n}\}$, the reasoning paths associated with different topic entities need to be jointly considered to infer the final answer. However, during the exploration phase, these paths are relatively independent of each other. To this end, RJE prompts the LLM to perform question decomposition, leveraging both the original question and each topic entity to generate a set of focused sub-questions. Specifically, for each topic entity e_{t_i}, a corresponding sub-question q_{t_i} is generated, which we define as a topic question. This strategy encourages the LLM to concentrate on reasoning paths specific to each topic entity, thereby enhancing both the efficiency and accuracy of reasoning. The prompt is provided in Appendix J.2.

4.3.2 Path Exploration

To fetch the required information, RJE conducts further exploration over the KG. Initially, based on the question decomposition, we obtain a set of topic questions $Q = \{q_{t_1}, q_{t_2}, \dots, q_{t_n}\}$, where n is the number of topic entities. The reasoning path ranker extracts the top-K reasoning paths, which are categorized as $\mathcal{P}^0 = \{P^0_{t_1}, P^0_{t_2}, \dots, P^0_{t_n}\}$. Here, $P^0_{t_i} = \{p^0_{t_i,1}, p^0_{t_i,2}, \dots\}$ represents the set of paths originating from the topic entity e_{t_i}. We initialize the entity set E^0 as the collection of all entities that appear in the paths of $\mathcal{P}^0$. In the D-th round of exploration, assuming that the entity set E^{D-1} and the paths $\mathcal{P}^{D-1}$ have already been obtained from the previous round, we perform further path exploration based on E^{D-1} and $\mathcal{P}^{D-1}$ to acquire additional useful information from the KG.

Exploration Entities Selection. For different topic entities, the number of hops required to reach the answer entity can vary significantly. As shown in Figure 2, theoretically, for question Q2, the answer entity *"Minnesota Vikings"* can be reached in one hop from the topic entity *"Viktor the Viking"*, whereas it takes three hops from *"Peyton Manning"*. To ensure more efficient exploration, we prompt the LLM using the paths of the previous round $\mathcal{P}^{D-1}$, the original question q, the set of topic questions Q, and the set of topic entities T to select a subset of entities E^D_f. For Q2, RJE selects the entity *"Archie Manning"* in the first round and the entities *"m.0hpq5r4"* and *"m.0hpq5rc"* in the second round. This approach, combined with the path information, identifies which paths require further exploration to answer their corresponding topic questions, thereby avoiding redundant exploration on information-sufficient paths. The prompt is provided in Appendix J.3.

Retriever-assisted Relation Exploration. Relation exploration aims to identify all relations relevant to answering the question. Let $E^D_f = \{e^D_{f,1}, e^D_{f,2}, \dots, e^D_{f,m}\}$ denote the set of exploration entities selected in the D-th round. Each entity $e^D_{f,i}$ corresponds to a topic question q_{t_c}. We use pre-defined SPARQL queries to retrieve all relations connected to $e^D_{f,i}$, obtaining a candidate relation set $R^D_{f,i}$. For more reliable LLM reasoning, we employ the relation path retriever to filter the top-N most relevant relations from $R^D_{f,i}$, resulting in a refined set $R^D_{s,i}$. We then prompt the LLM with the topic question q_{t_c}, the entity $e^D_{f,i}$, and the relation set $R_{s,i}^D$ to select a subset of the most useful relations, denoted as R_i^D. By performing this procedure for all entities in E_f^D, we derive the final set of explored relations in the D-th round $R^D = \{R_1^D, R_2^D, \dots, R_m^D\}$. The prompt and SPARQL queries are provided in Appendix J.4 and Appendix K.1, respectively.

Entity Exploration. Entity exploration aims to further infer useful entities to help answer the question. Given the entity set E_f^D and the corresponding relation set R^D obtained in the D-th round, we execute pre-defined SPARQL queries to retrieve the tail entities connected to each entity-relation pair. Following the approach of Chen et al. (2024), we apply a lightweight, train-free BERT to remove semantically irrelevant entities, resulting in a filtered entity set $E_s^D = \{E_{s,1}^D, E_{s,2}^D, \dots, E_{s,k}^D\}$. Each $E_{s,i}^D$ is a set of tail entities derived from a specific pair $\langle e_{f,j}^D, r\rangle$, where $e_{f,j}^D \in E_f^D$ and $r \in R_j^D$. Let q_{t_c} denote the topic question associated with $e_{f,j}^D$. The LLM then performs entity selection based on the topic question q_{t_c}, the entity $e_{f,j}^D$, the relation r, and the candidate tail entity set $E_{s,i}^D$ to select a minimal subset of entities most valuable for answering the question, denoted as E_i^D. After performing entity selection for all relevant entities and relations, we obtain the final entity set $E^D = \{E_1^D, E_2^D, \dots, E_k^D\}$. The prompt and SPARQL queries are provided in Appendix J.5 and Appendix K.2, respectively.

4.3.3 Answer Generation

After performing relation and entity exploration in the D-th round, we use the relation set R^D and the entity set E^D to update and extend the paths from the previous round $\mathcal{P}^{D-1}$ to obtain $\mathcal{P}^D = \{P_{t_1}^D, P_{t_2}^D, \dots, P_{t_n}^D\}$. Given the original question q, the set of topic questions $Q = \{q_{t_1}, q_{t_2}, \dots, q_{t_n}\}$, and the updated paths $\mathcal{P}^D$, we prompt the LLM to reason over each path $P_{t_i}^D$ to answer the corresponding topic question q_{t_i}. Then, the LLM integrates the answers to the topic questions to infer the answer to the original question q. During this process, if the LLM determines that the current paths provide sufficient information to answer the question, the iteration terminates and the final answer is generated. Otherwise, the LLM proceeds to perform a new round of path exploration. To avoid endless exploration, we define a maximum number of exploration rounds $D_{\max}$. If the LLM is still unable to generate an answer after reaching $D_{\max}$, it will produce an answer based on the accumulated paths and its internal knowledge. The prompt is provided in Appendix J.6.

5 Experiments

In this section, we present our experimental design, empirical results, and comprehensive analyses. Our experiments address the following research questions (RQs): **RQ1**: Does RJE achieve superior performance compared to state-of-the-art approaches on KGQA tasks? **RQ2**: Can small-sized, open-source LLMs deliver competitive results within the RJE framework? **RQ3**: Are the core stages and auxiliary modules of RJE effective in contributing to overall system performance? **RQ4**: Can RJE reduce computational overhead and improve efficiency during reasoning?

5.1 Datasets & Evaluation Metrics

To evaluate our proposed KGQA approach, we conduct extensive experiments on two widely used benchmark datasets that rely on the external knowledge graph Freebase (Bollacker et al., 2008): WebQuestionsSP (WebQSP) (Yih et al., 2016) and Complex WebQuestions (CWQ) (Talmor and Berant, 2018). Detailed statistics of the datasets are provided in Appendix A. Following prior research (Sun et al., 2024; Chen et al., 2024; Li et al., 2025), we adopt Hits@1 (exact match accuracy) as our primary evaluation metric.

5.2 Selected Baselines

For a comprehensive comparison with various methods, we selected five categories of baselines as follows: (1) *LLM-only methods*: standard prompting (IO prompt) (Brown et al., 2020), Chain-of-Thought prompting (CoT) (Wei et al., 2022), and Self-Consistency (SC) (Wang et al., 2023). (2) *Retrieval-Reasoning Methods*: SR (Zhang et al., 2022), UniKGQA (Jiang et al., 2023c), ReasoningLM (Jiang et al., 2023b) and EPR (Ding et al., 2024). (3) *Retrieval-Augmented Generation Methods*: RD-P (Huang and Zeng, 2024), KG-CoT (Zhao et al., 2024) and SubgraphRAG (Li et al., 2025). (4) *Fine-tuned Agent-based Methods*: RoG (Luo et al., 2024b), KG-Agent (Jiang et al., 2024) and GCR (Luo et al., 2024c). (5) *Prompting Agent-based Methods*: StructGPT (Jiang et al., 2023a), ToG (Sun et al., 2024), Interactive-KBQA (Xiong et al., 2024), ReKnoS (Wang et al., 2025) and PoG (Chen et al., 2024). The description of the baselines can be found in Appendix B.

Method	CWQ	WebQSP
LLM-Only methods		
IO Prompt (Brown et al., 2020)	37.6	63.3
CoT (Wei et al., 2022)	38.8	62.2
SC (Wang et al., 2023)	45.4	61.1
Retrieval-Reasoning Methods		
SR (Zhang et al., 2022)	50.2	68.9
UniKGQA (Jiang et al., 2023c)	51.2	77.2
ReasoningLM (Jiang et al., 2023b)	69.0	78.5
EPR (Ding et al., 2024)	60.6	71.2
Retrieval-Augmented Generation Methods		
KG-CoT w/GPT-4 (Zhao et al., 2024)	62.3	84.9
RD-P w/ChatGPT (Huang and Zeng, 2024)	63.5	85.3
SubgraphRAG w/GPT4o (Li et al., 2025)	67.5	90.9
Fine-tuned Agent-based Methods		
RoG (Luo et al., 2024b)	62.6	85.7
KG-Agent (Jiang et al., 2024)	72.2	83.3
GCR (Luo et al., 2024c)	75.8	92.2
Prompting Agent-based Methods		
StructGPT w/ChatGPT (Jiang et al., 2023a)	54.3	72.6
Interactive-KBQA w/GPT-4 (Xiong et al., 2024)	59.2	72.5
ToG w/ChatGPT (Sun et al., 2024)	58.9	76.2
ToG w/GPT-4 (Sun et al., 2024)	69.5	82.6
ReKnoS w/GPT-4o-mini (Wang et al., 2025)	66.8	83.8
PoG w/ChatGPT (Chen et al., 2024)	63.2	82.0
PoG w/GPT-4 (Chen et al., 2024)	75.0	87.3
Ours		
RJE w/Llama3.2-3B	62.9	82.6
RJE w/Llama3.1-8B	71.5	89.2
RJE w/Qwen2.5-14B	71.2	90.3
RJE w/ChatGPT	72.6	91.2
RJE w/GPT-4o-mini	77.1	**92.5**
RJE w/DeepSeek-V3	**78.2**	92.4

Table 1: Results of different methods and models on two datasets. The best results are highlighted in bold.

5.3 Implementation Details

We employ RoBERTa-base (Liu et al., 2019) as our backbone PLM, consistent with prior work. For the reasoning path ranker training, we use a learning rate of 2e-5 with a margin of 1.0 for WebQSP, and 1e-5 with a margin of 0.8 for CWQ. Our framework supports integration with various LLMs, we evaluate with both proprietary models (ChatGPT, GPT-4o-mini, DeepSeek-V3) and open-source models (Llama3.2-3B, Llama3.1-8B, Qwen2.5-14B). Across all experiments, we set the number of reasoning paths to 10 ($K = 10$), the number of filtered relations to 30 ($N = 30$), LLM temperature to 0.3, and maximum exploration rounds $D_{\max}$ to 2 for WebQSP and 4 for CWQ. Proprietary LLMs were accessed through their official APIs[2][3], while open-source models were deployed on 4 NVIDIA A800-80G GPUs.

5.4 Main Results (RQ1 & RQ2)

We compare RJE with various state-of-the-art baseline methods to evaluate its effectiveness in KGQA. As shown in Table 1, RJE consistently delivers performance gains across different LLMs and datasets. When GPT-4o-mini and DeepSeek-V3 are used as backbone LLMs, RJE outperforms all baseline methods, including LLM-Only methods, Retrieval-Reasoning Methods, Retrieval-Augmented Generation Methods, Fine-tuned Agent-based Methods, and Prompting Agent-based Methods. This highlights the superiority of our framework. It is noteworthy that small-sized LLMs also demonstrate competitive performance within our framework. For instance, Llama3.2-3B within RJE achieves performance comparable to ChatGPT used in PoG, the performance gap between the two is merely 0.3% on CWQ. Meanwhile, RJE with Llama3.1-8B performs slightly below PoG with GPT-4 on CWQ but slightly surpasses it on WebQSP. Additionally, we present comprehensive Macro-F1 results in Appendix C.

Model	Method	CWQ	WebQSP
Llama3.2-3B	ToG	17.6	40.2
	PoG	21.4	49.3
	RJE	**62.9**	**82.6**
Llama3.1-8B	ToG	35.5	66.8
	PoG	43.6	75.7
	RJE	**71.5**	**89.2**
Qwen2.5-14B	ToG	39.0	75.3
	PoG	54.3	79.6
	RJE	**71.2**	**90.3**
GPT-4o-mini	ToG	65.4	80.7
	PoG	67.2	82.4
	RJE	**77.1**	**92.5**

Table 2: Results of ToG, PoG and RJE on various backbone models on two datasets.

As shown in Table 2, RJE provides significant performance enhancement for small-sized LLMs. Specifically, the smaller the LLM, the greater the performance boost from RJE. With Llama3.1-8B, RJE achieves a 27.9% improvement over PoG on CWQ and 13.5% on WebQSP. Even more notably, using Llama3.2-3B, RJE achieves a remarkable improvement of 41.5% over PoG on CWQ (62.9% vs. 21.4%) and 33.3% on WebQSP (82.6% vs. 49.3%). Although the improvements are less dramatic with larger and more advanced LLMs, such as Qwen2.5-14B and GPT-4o-mini, RJE still consistently outperforms PoG by at least 9.9% on both datasets. These results suggest that RJE is particularly valuable for enhancing the capabilities of smaller LLMs, effectively narrowing the performance gap between smaller and larger models in KGQA tasks. Additionally, Results Analysis and

[2] https://platform.openai.com/docs/overview

[3] https://api-docs.deepseek.com

Method	CWQ	WebQSP
RJE	**71.5**	**89.2**
w/o Exploration	41.1	73.7
w/o Retrieval	60.7	83.9
w/o Ranking	66.9	88.0
w/o Decomposition	68.8	86.0
w/o Assistance	70.2	87.3

Table 3: Ablation experiment results using Llama3.1-8B of removing each stage and each module, respectively.

Case Study can be found in Appendix F and Appendix G, respectively.

5.5 Ablation Study (RQ3)

We conduct comprehensive ablation experiments to evaluate each component of our RJE framework, using Llama3.1-8B as the backbone LLM across both CWQ and WebQSP datasets. Table 3 presents the results of systematically removing key stages and modules: **w/o Exploration** (removing the exploration stage, using only the retrieval and judgment stages), **w/o Retrieval** (removing the retrieval stage, using only the exploration stage), **w/o Ranking** (removing the reasoning path ranking module), **w/o Decomposition** (removing the question decomposition module), and **w/o Assistance** (removing the retriever-assisted exploration module).

As shown in Table 3, the results demonstrate that each component contributes positively to the overall performance. Specifically, on CWQ, w/o Retrieval results in a 10.8% drop in performance, while w/o Exploration leads to a more substantial 30.4% drop. On WebQSP, performance decreases by 5.3% and 15.5% when w/o Retrieval and w/o Exploration, respectively. These findings highlight the critical importance of the synergistic collaboration between retriever and LLMs within our RJE framework. The auxiliary modules in our framework contribute measurable performance improvements as well. Specifically, on WebQSP, the reasoning path ranking, question decomposition, and retriever-assisted exploration modules improve performance by 1.2%, 3.2%, and 1.5%, respectively. On the more challenging CWQ dataset, the corresponding improvements are 4.6%, 2.7%, and 1.9%. These results demonstrate that the auxiliary modules effectively alleviate the reasoning burden on small-sized LLMs in KGQA tasks, contributing to enhanced overall system performance.

Additional ablation studies on reasoning path ranking and the impact of the number of reasoning paths, filtered relations, and exploration rounds are provided in Appendix D and Appendix E.

Dataset	Method	LLM Call	Input Token	Output Token	Time (s)
CWQ	ToG	22.6	8,182.9	1,486.4	96.5
	PoG	13.3	7,803.0	353.2	23.3
	RJE	**7.9**	**5,769.1**	**247.2**	**16.3**
WebQSP	ToG	15.9	6,031.2	987.7	63.1
	PoG	9.0	5,234.8	282.9	16.8
	RJE	**4.1**	**2763.3**	**148.7**	**10.5**

Table 4: Efficiency comparison between our proposed RJE and baseline methods ToG and PoG.

5.6 Efficiency Study (RQ4)

We evaluate the computational efficiency of RJE against two leading Prompting LLM approaches: ToG and PoG. Table 4 summarizes the efficiency metrics across the CWQ and WebQSP datasets, including the average number of LLM calls, token usage, and time consumption per question. As shown in Table 4, RJE demonstrates superior efficiency in LLM calls, token utilization, and time consumption. On CWQ, RJE reduces LLM calls by 65.0% compared to ToG and 40.6% compared to PoG. This efficiency gain is even more pronounced on WebQSP, where RJE achieves reductions of 74.2% and 54.4% in LLM calls compared to ToG and PoG, respectively. As for token consumption, RJE reduces token usage by approximately 30% compared to PoG on CWQ, and by about 50% on WebQSP. Additionally, RJE exhibits significant time efficiency improvements, performing at least 5.9 times faster than ToG and at least 1.4 times faster than PoG across both datasets.

The efficiency gains of RJE stem from two key design aspects: first, its judgment stage directly answers the question when the paths provide sufficient evidence, without requiring additional exploration; second, RJE retrieves key paths from the KG and begins exploration from strategically selected entities along these paths, rather than starting from topic entities as in ToG and PoG. This early resolution capability and targeted exploration approach significantly reduce computational demands while maintaining high accuracy. Experimental results in Appendix H demonstrate that initiating exploration from selected entities along retrieval paths reduces exploration rounds, thereby alleviating computational burden.

6 Conclusion

In this paper, we introduce the RJE framework, which addresses core limitations in existing KGQA methods by integrating precise retrieval, sufficiency judgment, and conditional exploration. Our specialized auxiliary modules, including Reasoning Path Ranking, Question Decomposition, and Retriever-assisted Exploration, significantly enhance the reasoning efficacy of smaller open-source LLMs in knowledge-intensive tasks. Empirical evaluations across standard KGQA benchmarks demonstrate that RJE not only outperforms existing approaches in both accuracy and efficiency, but also enables small-sized open-source LLMs to achieve comparable performance, advancing the development of more efficient and accessible KGQA systems.

Limitations

While our approach demonstrates significant improvements in KGQA, there are several limitations that suggest directions for future work. First, existing KGs, which are primarily constructed from internet corpora, often contain noisy triples and outdated information. Such noisy knowledge can mislead LLMs into making incorrect responses, even when using our proposed framework. In future work, we plan to investigate methods for detecting and filtering unreliable knowledge in KGs to reduce noise and enhance KGQA system reliability. Second, our experimental evaluation is limited to English language datasets. To assess the cross-lingual capabilities of our approach, we intend to extend our evaluation to multiple languages.

Acknowledgments

The research was partially supported by "Pioneer" and "Leading Goose" R&D Program of Zhejiang 2023C01029, the Plans for Major Provincial Science&Technology Projects No.:202303a07020006, and the China National Natural Science Foundation with no. 62132018.

References

Jinheon Baek, Alham Fikri Aji, Jens Lehmann, and Sung Ju Hwang. 2023a. Direct fact retrieval from knowledge graphs without entity linking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10038–10055.

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

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, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Sitao Cheng, Ziyuan Zhuang, Yong Xu, Fangkai Yang, Chaoyun Zhang, Xiaoting Qin, Xiang Huang, Ling Chen, Qingwei Lin, Dongmei Zhang, and 1 others. 2024. Call me when necessary: Llms can efficiently and faithfully reason over structured environments. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 4275–4295.

Wentao Ding, Jinmao Li, Liangchuan Luo, and Yuzhong Qu. 2024. Enhancing complex question answering over knowledge graphs through evidence pattern retrieval. In *Proceedings of the ACM Web Conference 2024*, pages 2106–2115.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2:1.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907.

Xixin Hu, Xuan Wu, Yiheng Shu, and Yuzhong Qu. 2022. Logical form generation via multi-task learning for complex question answering over knowledge bases. In *Proceedings of the 29th international conference on computational linguistics*, pages 1687–1696.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065.

Yubo Huang and Guosun Zeng. 2024. Rd-p: A trustworthy retrieval-augmented prompter with knowledge graphs for llms. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 942–952.

Yixin Ji, Kaixin Wu, Juntao Li, Wei Chen, Mingjie Zhong, Xu Jia, and Min Zhang. 2024. Retrieval and reasoning on kgs: Integrate knowledge graphs into large language models for complex question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7598–7610.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023a. 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.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yaliang Li, and Ji-Rong Wen. 2023b. Reasoninglm: Enabling structural subgraph reasoning in pre-trained language models for question answering over knowledge graph. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3721–3735.

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

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023c. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations*.

Kun Li, Tianhua Zhang, Xixin Wu, Hongyin Luo, James Glass, and Helen Meng. 2024. Decoding on graphs: Faithful and sound reasoning on knowledge graphs through generation of well-formed chains. *arXiv preprint arXiv:2410.18415*.

Mufei Li, Siqi Miao, and Pan Li. 2025. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *The Thirteenth International Conference on Learning Representations*.

Shiyang Li, Yifan Gao, Haoming Jiang, Qingyu Yin, Zheng Li, Xifeng Yan, Chao Zhang, and Bing Yin. 2023. Graph reasoning for question answering with triplet retrieval. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3366–3375.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Haoran Luo, E Haihong, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, and 1 others. 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.

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*.

Linhao Luo, Zicheng Zhao, Chen Gong, Gholamreza Haffari, and Shirui Pan. 2024c. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *arXiv preprint arXiv:2410.13080*.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

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.

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

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From freebase to wikidata: The great migration. In *Proceedings of the 25th international conference on world wide web*, pages 1419–1428.

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 *The Twelfth International Conference on Learning Representations*.

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.

Song Wang, Junhong Lin, Xiaojie Guo, Julian Shun, Jundong Li, and Yada Zhu. 2025. Reasoning of large language models over knowledge graphs with super-relations. In *The Thirteenth International Conference on Learning Representations*.

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 *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-kbqa: Multi-turn interactions for knowledge base question answering with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10561–10582.

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.

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

Lingxi Zhang, Jing Zhang, Yanling Wang, Shulin Cao, Xinmei Huang, Cuiping Li, Hong Chen, and Juanzi Li. 2023. Fc-kbqa: A fine-to-coarse composition framework for knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1002–1017.

Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Junnan Dong, Hao Chen, Yi Chang, and Xiao Huang. 2025. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*.

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. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)*, pages 6642–6650. International Joint Conferences on Artificial Intelligence.

Dataset	KG	Train	Dev	Test	Max hop
WebQSP	Freebase	2,848	250	1,639	2
CWQ	Freebase	27,639	3,519	3,531	4

Table 5: Statistics of different KGQA datasets used in the experiments.

A Datasets

We evaluate the proposed method on the WebQuestionsSP and ComplexWebQuestions datasets. For fair comparison with prior research (Sun et al., 2024; Chen et al., 2024), we maintain identical training and testing splits. Table 5 presents a comprehensive summary of the dataset statistics.

B Baselines

(1) **LLM-only methods**, these approaches rely solely on large language models without explicit knowledge graph integration:

Standard prompting (IO prompt) (Brown et al., 2020) demonstrated that LLMs outperform traditional language models on task-agnostic and few-shot problems.

Chain-of-Thought prompting (CoT) (Wei et al., 2022) incorporates "think step by step" prompts to enhance LLM performance across various natural language processing tasks.

Self-Consistency (SC) (Wang et al., 2023) improves performance by sampling multiple diverse reasoning paths through few-shot CoT prompts and selecting the most consistent answer among the generated outputs.

(2) **Retrieval-Reasoning Methods**, these methods focus on effective subgraph retrieval techniques:

SR (Zhang et al., 2022) proposes a trainable subgraph retriever decoupled from the downstream reasoner, incorporating a PLM to expand paths for subgraph induction with automatic termination criteria.

UniKGQA (Jiang et al., 2023c) integrates graph retrieval and reasoning into a single model that incorporates a PLM.

ReasoningLM (Jiang et al., 2023b) enables effective question understanding and structured reasoning over knowledge graphs by incorporating subgraph-aware self-attention and an adaptation tuning strategy.

EPR (Ding et al., 2024) enhances subgraph extraction for KGQA by modeling and retrieving structural evidence patterns that connect necessary entities and relations.

(3) **Retrieval-Augmented Generation Methods**, these approaches combine retrieval mechanisms with LLM capabilities:

KG-CoT (Zhao et al., 2024) enhances LLMs reasoning by integrating step-by-step graph reasoning over KGs to generate explicit reasoning paths, enabling knowledge-aware, plug-and-play CoT prompting.

RD-P (Huang and Zeng, 2024) integrates KGs with LLMs by retrieving and verifying trustworthy reasoning paths to construct reliable prompts, enhancing reasoning accuracy and efficiency without modifying LLM parameters.

SubgraphRAG (Li et al., 2025) enhances KG-based RAG by efficiently retrieving high-quality subgraphs using a lightweight MLP with structural features, enabling effective and adaptable LLM reasoning without fine-tuning.

(4) **Fine-tuned Agent-based Methods**, these methods involve fine-tuning LLMs with knowledge graph information:

RoG (Luo et al., 2024b) enables faithful and interpretable KG reasoning by fine-tuning LLMs within a planning-retrieval-reasoning framework that distills knowledge into relation path planning and optimized reasoning over retrieved KG paths.

KG-Agent (Jiang et al., 2024) empowers small-sized LLMs to autonomously perform complex KG reasoning by fine-tuning them with code-based instruction data and integrating a multifunctional toolbox for structured operations and iterative decision-making.

GCR (Luo et al., 2024c) ensures faithful KG reasoning by fine-tuning a lightweight KG-specialized LLM to generate constrained decoding paths via a KG-Trie index, then leveraging a general inductive power of LLM to produce accurate final answers.

(5) **Prompting Agent-based Methods**, these approaches utilize sophisticated prompting techniques with KG integration:

StructGPT (Jiang et al., 2023a) defines an interface for accessing and filtering knowledge from KGs data under limited constraints, and leverages LLMs to iteratively infer answers or generate follow-up plans.

Interactive-KBQA (Xiong et al., 2024) Interactive KBQA directly leverages LLMs to interact with KGs, subsequently generating logical forms.

ReKnoS (Wang et al., 2025) enhances knowledge graph reasoning by introducing super-relations, which group domain-specific relations to expand the reasoning space and improve retrieval performance.

Method	**CWQ**		**WebQSP**	
	Hit@1	**Macro-F1**	**Hit@1**	**Macro-F1**
Baselines				
SR (Zhang et al., 2022)	50.2	47.1	68.9	64.1
UniKGQA (Jiang et al., 2023c)	51.2	49.0	77.2	72.2
ReasoningLM (Jiang et al., 2023b)	69.0	64.0	78.5	71.0
EPR (Ding et al., 2024)	60.6	61.2	71.2	70.2
RD-P w/ChatGPT (Huang and Zeng, 2024)	63.5	56.6	85.3	69.7
SubgraphRAG w/GPT4o (Li et al., 2025)	67.5	59.5	90.9	78.2
RoG (Luo et al., 2024b)	62.6	56.2	85.7	70.8
KG-Agent (Jiang et al., 2024)	72.2	69.8	83.3	**81.0**
GCR (Luo et al., 2024c)	75.8	61.7	92.2	74.1
Ours				
RJE w/Llama3.2-3B	62.9	53.7	82.6	65.9
RJE w/Llama3.1-8B	71.5	60.5	89.2	73.8
RJE w/Qwen2.5-14B	71.2	62.8	90.3	76.0
RJE w/ChatGPT	72.6	62.1	91.2	74.8
RJE w/GPT-4o-mini	77.1	65.9	**92.5**	77.6
RJE w/DeepSeek-V3	**78.2**	**70.2**	92.4	78.9

Table 6: Results on CWQ and WebQSP under Hits@1 and Macro-F1; the best results are highlighted in bold.

ToG (Sun et al., 2024) iteratively retrieves relevant triples from the knowledge graph and uses LLMs to evaluate whether the reasoning paths within beam search are sufficient to answer the question or if additional next-hop information is needed.

PoG (Chen et al., 2024) builds upon ToG by incorporating mechanisms such as Adaptive Breadth, Guidance, Memory, and Reflection.

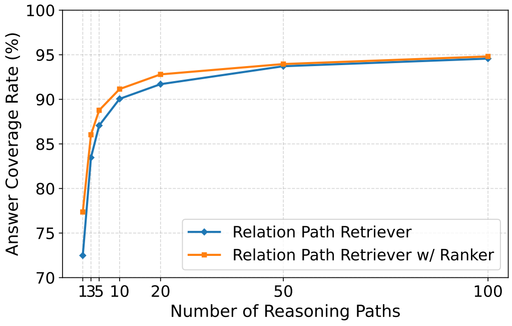
C Analysis of Macro-F1 Metric

To provide a more comprehensive assessment of RJE's performance, we conduct additional evaluations using Macro-F1 across various LLMs. Since Macro F1 is not consistently reported in prior work (e.g., PoG (Chen et al., 2024) and ToG (Sun et al., 2024)), we compare against only those baselines that include this metric.

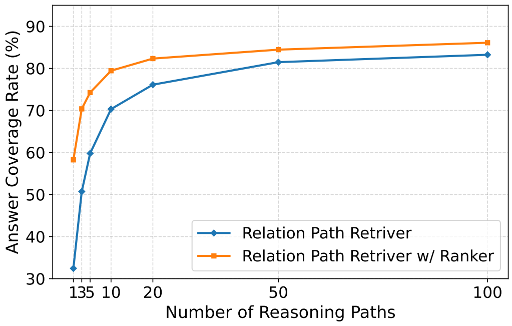
As shown in Table 6 , RJE achieves superior performance compared to most baselines on both Hits@1 and Macro-F1. The only exception occurs on the WebQSP dataset, where KG-agent (a fine-tuned LLM-based method) marginally outperforms RJE with DeepSeek-V3 on Macro-F1. However, RJE with DeepSeek-V3 significantly surpasses KG-agent in Hits@1 performance. The excellent Macro-F1 performance of RJE indicates that it simultaneously achieves high precision and recall in knowledge-graph reasoning tasks.

D The Performance of Reasoning Path Ranking

We conduct a systematic evaluation of the impact of the reasoning path ranking on retrieval performance, using answer coverage at various retrieval scales as the metric. Figure 3 presents a comparative analysis between the standalone relation retriever and its integration with the reasoning path ranker across two datasets. The results demonstrate that our combined approach consistently achieves higher coverage across all candidate reasoning path sizes. Notably, on the more challenging CWQ dataset, the contribution of the ranker yields particularly substantial improvements. Furthermore, as the candidate path count decreases, the performance enhancement provided by the ranker becomes increasingly significant.



(a) WebQSP



(b) CWQ

Figure 3: Comparison of Answer Coverage Across Different Numbers of Candidate Reasoning Paths.

E Addition Ablation Study

The RJE framework contains three manually configurable hyperparameters: the number of reasoning paths to be refined by the reasoning path ranker, the number of relations to be considered by the relation path retriever during retriever-assisted exploration and the maximum number of exploration rounds. To investigate how these parameters influence the performance of RJE, we conduct a series of experiments and derive practical insights.

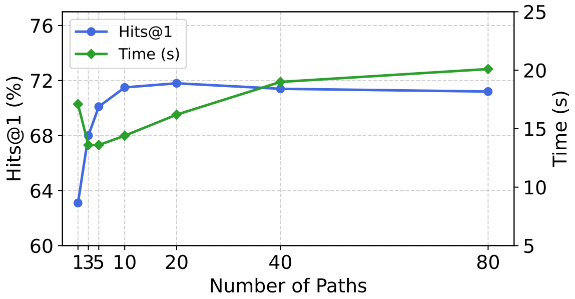


Figure 4: The impact of the number of Reasoning Paths on performance on the CWQ dataset.

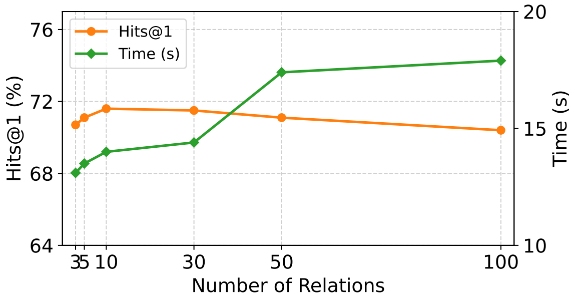


Figure 5: The impact of the number of Relations on performance on the CWQ dataset.

E.1 Impact of the Number of Paths

In the RJE framework, the reasoning path ranker outputs top-K candidate paths for judgment evaluation. The number of paths represents a critical trade-off: too few paths limit available evidence, while too many introduce noise that degrades LLM performance. To examine this balance, we conduct experiments on CWQ with varying path counts.

As shown in Figure 4, performance increases substantially from 1 to 10 paths, confirming the ranker effectively identifies relevant information for accurate judgment. However, performance declines beyond 20 paths, indicating that excessive candidates introduce irrelevant information that misleads the LLM. The relationship between path count and time cost exhibits a U-shaped pattern. Initially, increasing path count reduces overall time as sufficient evidence from multiple paths decreases the need for extensive reasoning exploration. However, when the path count becomes too large, the additional tokens and noise both mislead the LLM and increase processing time.

These results demonstrate that the choice of path count significantly affects performance. An appropriate setting balances the provision of sufficient evidence with noise reduction, improving both reasoning accuracy and computational efficiency.

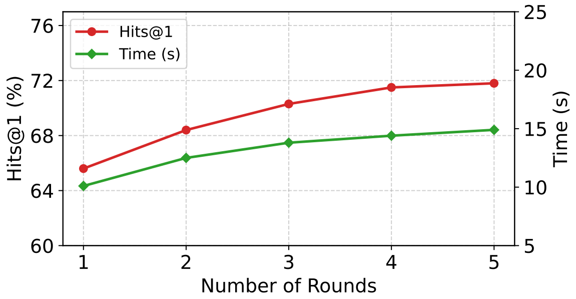


Figure 6: The impact of exploration rounds on performance on the CWQ dataset.

E.2 Impact of the Number of Relations

Previous agent-based approaches typically feed all relations into LLMs during relation exploration. Although effective with powerful proprietary LLMs, this strategy often degrades performance in smaller open-source models due to excessive input length. To address this limitation, we employ the relation path retriever to pre-filter the N most relevant relations, which are then fed to the LLMs for final selection. Notably, N is smaller than the total number of relations. We conduct experiments on CWQ with varying N values to assess the performance impact.

As shown in Figure 5, varying the relation count does not cause large performance fluctuations because the retriever consistently ranks relevant relations at the top. Even a small number of relations can yield strong performance. On closer inspection, increasing the number of relations from 3 to 10 yields gradual gains. However, beyond $N = 10$, performance begins to decline, suggesting that excessive relations introduce noise that impairs the LLM's focus on the most informative candidates. Processing time increases with relation count due to additional tokens and LLM exploration.

These results demonstrate that the relation path retriever effectively narrows the exploration space while enhancing LLM reasoning capabilities through focused candidate provision.

E.3 Impact of Exploration Rounds

To prevent LLMs from engaging in endless exploration, we set an upper limit on the number of exploration rounds. We conduct experiments on CWQ with varying numbers of rounds. As shown in Figure 6, performance consistently improves with additional rounds, suggesting that further exploration enables RJE to discover more relevant information for answering complex questions. However, after four rounds, the performance gain becomes marginal, since the maximum hop count on CWQ is 4.

Processing time increases with the maximum round due to additional exploration cycles. However, the time overhead exhibits diminishing gains at higher values, as fewer questions actually require extensive exploration rounds. To balance computational cost and effectiveness, we set the maximum exploration rounds to 4.

F Results Analysis

In the RJE framework, the final answer can be produced either during the judgment stage or the exploration stage. Figure 7 illustrates the answer accuracy at each stage using DeepSeek-V3 on two datasets. The label **First** denotes answers generated during the judgment stage, while **Second** indicates those generated during the exploration stage.

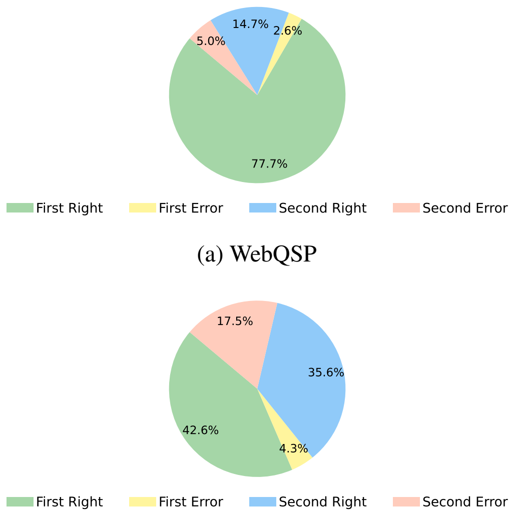
For WebQSP, 77.7% of the questions are correctly answered during the judgment stage, indicating that only a single LLM call is required. This demonstrates the efficiency of RJE in producing accurate answers with minimal computational cost when the path information is sufficient. Among the remaining 19.7% of cases where the initial paths are insufficient, 14.7% of the questions are still answered correctly during the subsequent exploration stage. These findings demonstrate the capacity of RJE to recognize insufficient information and retrieve additional knowledge through exploration, eventually leading to the correct answer.

For CWQ, which is generally more complex than WebQSP, approximately half of the questions are answered during the judgment stage. This outcome reflects the ability of RJE to adapt its strategy according to the complexity of the question. Notably, 35.6% of the questions are correctly answered during the exploration stage, demonstrating the potential of RJE in handling more challenging questions by exploring beyond the initially retrieved paths.

G Case Study

G.1 Case 1

Figure 8 illustrates a case from CWQ that highlights the advantages of the RJE framework. For the question *"What David Slade film starred Taylor Lautner?"*, RJE first retrieves and refines five highly relevant paths. After performing judgment on these paths, RJE identifies that while some of



(a) WebQSP

(b) CWQ

Figure 7: Answer accuracy at different stages in the RJE framework on WebQSP and CWQ.

the paths indicate movies involving *"Taylor Lautner"* and one indicates a movie directed by *"David Slade"*, there is no overlapping entity between them that can support valid reasoning. Therefore, RJE judges that the path information is insufficient and triggers further exploration.

Upon entering the exploration stage, RJE first performs question decomposition based on the topic entities to better facilitate path exploration and reasoning. In the first round of exploration, RJE successfully selects the entities *"The Twilight Saga"*, *"The Twilight Saga: New Moon"*, and *"David Slade"*, aiming to retrieve detailed information about the movie series and its director. After conducting relation and entity exploration, RJE acquires key knowledge: *"Eclipse"* is part of *"The Twilight Saga"*, and *"David Slade"* is its director. By synthesizing this information, RJE is able to accurately infer that the answer to the original question is *"Eclipse"*.

In summary, while three hops would be needed to reach the correct answer theoretically, RJE obtains highly relevant information during the retrieval stage, eliminating the need to start exploration from the topic entity and significantly reducing the number of exploration rounds. This substantially reduces computational cost and inference time. Furthermore, the integration of question decomposition and the retriever-assisted exploration during the exploration process enables RJE to perform path exploration and reasoning more effectively, making it easier to arrive at the correct answer. These strengths contribute to the strong performance of RJE even when using LLMs with relatively small parameter sizes.

G.2 Case 2

Figure 9 presents another case from CWQ that demonstrates the importance of the reasoning path ranker. For the question *"What office position was held from March 19, 1790 and was Abraham Lincoln's political experience?"*, without the reasoning path ranker, RJE retrieves the top five paths with the highest relevance scores. However, none of these paths cover the correct answer. As a result, when the LLM analyzes these paths, it is misled by the weakly relevant and noisy information, leading to an incorrect answer during the judgment stage.

In contrast, with the reasoning path ranker reordering the paths obtained by the relation path retriever, the paths containing the correct answer are successfully selected and ranked at the top. Upon analyzing the reordered paths, the LLM observes the entity *"United States Representative"*, and, based on its internal knowledge that the United States Representative office was held from March 19, 1790, it aligns this information with the question and infers the correct answer.

In summary, the reasoning path ranker effectively filters and prioritizes the most relevant paths, helping LLMs reduce hallucinations and enhancing the probability of answering correctly during the judgment stage without needing further exploration. This substantially reduces computational cost and inference time. Overall, RJE dynamically adjusts its strategy based on question complexity and path relevance, aiming to find the correct answer while reducing resource consumption.

H Analysis of Exploration Round Reduction

To validate that RJE improves exploration efficiency by starting from entities identified in retrieval paths, we conducted experiments on a subset of CWQ questions that explicitly require exploration. This subset excludes questions that RJE can directly answer during the judgment stage, ensuring all questions necessitate exploration. We compared methods using their default maximum number of rounds: RJE (4 rounds), PoG (4 rounds) and ToG (3 rounds). RJE w/o Retrieval is a variant without retrieval paths that initiates exploration

Question: What David Slade film starred Taylor Lautner?

Retrieval	[Taylor Lautner] → (won) → [m.0nf3bp2] → (honored_for) → [The Twilight Saga] [Taylor Lautner] → (nominations) → [m.0c02330] → (nominated_for) → [The Twilight Saga: New Moon] [Taylor Lautner] → (nominations) → [m.0b3tz45] → (nominated_for) → [Twilight] [Taylor Lautner] → (award_nominations) → [m.09tz609] → (nominated_for) → [The Twilight Saga: New Moon] [David Slade] → (film.writer.film) → [Meat Dog: What's fer Dinner]
Judgment	The paths provided only mention Taylor Lautner's involvement in 'The Twilight Saga' series and related films, but do not confirm any film directed by David Slade. Need to explore
Exploration	Question Decomposition: "Taylor Lautner": "Which film starring Taylor Lautner?" "David Slade": "Which film directed by David Slade?" Reasoning Path: [Taylor Lautner] → (won) → [m.0nf3bp2] → (honored_for) → [The Twilight Saga] → (films_in_series) → [The Twilight Saga: New Moon] [Taylor Lautner] → (won) → [m.0nf3bp2] → (honored_for) → [The Twilight Saga] → (films_in_series) → [Eclipse] [Taylor Lautner] → (award_nominations) → [m.09tz609] → (nominated_for) → [The Twilight Saga: New Moon] → (film.director.film) → [Chris Weitz] [David Slade] → (film.director.film) → [Eclipse] [David Slade] → (film.film.directed_by) → [Eclipse] Evaluation: Topic 1 lists various films starring Taylor Lautner, Topic 2 specifically identifies 'Eclipse' as a film directed by David Slade. Since 'Eclipse' is part of the Twilight Saga, which stars Taylor Lautner, we can infer that 'Eclipse' is final answer.

Figure 8: A typical case demonstrates the advantages of the RJE framework. The topic entities, exploration entities, incorrect answers, and correct answers are highlighted in yellow, blue, red, and green, respectively.

Question: What office position was held from March 19, 1790 and was Abraham Lincoln's political experience?

w/o ranker	Retrieval: [Abraham Lincoln] → (presidency_number) → [16] [Abraham Lincoln] → (military_commands) → [m.04h_gx9] [Abraham Lincoln] → (government_positions_held) → [m.0446bdb] → (office_position_or_title) → [President of the United States] [Abraham Lincoln] → (government_positions_held) → [m.0bqspr2] → (office_position_or_title) → [Member of Illinois House of Representatives] [Abraham Lincoln] → (government_positions_held) → [m.0446bdb] → (basic_title) → [President] Judgment: The provided paths confirm both Abraham Lincoln's office position and title, which sufficiently answer the question. The Answer is President of the United States and President.
w ranker	Retrieval: [Abraham Lincoln] → (government_positions_held) → [m.04j60k7] → (office_position_or_title) → [United States Representative] [Abraham Lincoln] → (appointees) → [m.05kq8nh] → (office_position_or_title) → [United States Secretary of State] [Abraham Lincoln] → (government_positions_held) → [m.0446bdb] → (basic_title) → [President] [Abraham Lincoln] → (government_positions_held) → [m.0446bdb] → (office_position_or_title) → [President of the United States] [Abraham Lincoln] → (government_positions_held) → [m.0bqspr2] → (office_position_or_title) → [Member of Illinois House of Representatives] Judgment: From the triplets, we know that Abraham Lincoln held the positions of United States Representative, Member of Illinois House of Representatives, and President of the United States. The position of United States Representative was held from March 19, 1790, which aligns with the original question. Therefore, The Answer is United States Representative.

Figure 9: A typical case illustrates the importance of the reasoning path ranker. With and without the ranker, the LLM infers different answers. The incorrect and correct answers are highlighted in red and green, respectively.

Method	RJE	RJE w/o Retrieval	PoG	ToG
Average Rounds	**1.57**	2.41	2.36	2.58

Table 7: Average number of exploration rounds required by different methods on CWQ subset.

CWQ	1 hop	2 hop	3 hop	4 hop
Hits@1	73.7	80.0	81.5	69.9

Table 8: RJE performance on questions with different numbers of hops using DeepSeek-V3.

from topic entities rather than retrieval path entities.

As shown in Table 7, RJE achieves the lowest average number of rounds (1.57), substantially outperforming RJE w/o Retrieval (2.41 rounds) by 35%. This indicates that starting exploration from retrieval path entities effectively reduces exploration rounds. RJE also outperforms PoG (2.36 rounds) and ToG (2.58 rounds), highlighting the computational efficiency of the RJE framework.

I Performance on Different Hops

To evaluate RJE's performance across questions of varying complexity, we analyze results by hop count on the CWQ dataset. For each question, the hop count is defined as the longest of the shortest paths from topic entities to the answer entity in the provided SPARQL queries.

As shown in Table 8, RJE maintains consistent performance across different hop requirements, achieving Hits@1 scores of 69.9%-81.5%. This consistency highlights RJE's effectiveness on multi-hop reasoning, attributed to its systematic three-stage architecture that mitigates the difficulty of multi-hop questions.

J Prompts

J.1 Judgment

Your task is to infer the answer based on the given question and given triple paths.
{In-Context Few-shot}
Task Requirements:
1. Do not provide explanations or extra text.
2. The output must be in strict JSON format.
Now, based on the following input, determine whether the question can be answered.
Question:
Paths:

J.2 Question Decomposition

You are an expert in question decomposition and knowledge-based reasoning.
Given:
- A complex natural language question.
- A list of topic entities mentioned in the question.
Your tasks are: Generate a sub-question for each entity. The sub-question should reflect the original question's intent, but be scoped only to that specific entity.
{In-Context Few-shot}
Original Question:
Topic Entities:

J.3 Exploration Entities Selection

Your task is to determine which entities should be explored next, based on the Original Question, Topic Question and given triple paths.
{In-Context Few-shot}
Now you need to output the entities from the Entity List, without additional explanations or formatting. Strictly follow the entity names as they appear in the Entity List.
Note: Only include entities that are necessary for answering the Original Question, and ensure the list has no more than 10 entities.
Original Question:
Topic 1:
Topic Question:
Topic Entity:
Triplets:
Topic 2:
. . .

J.4 Relation Exploration

Your task is to select useful relations from a given list based on the current question and connected entity.
{In-Context Few-shot}
Task Requirements:
1. Strictly output only the selected relations

from the provided list.
2. Do not include any additional relations, explanations, reasoning, or extra formatting.
Now, based on the following input, select the useful relations.
Question:
Connected entity:
Relations List:

J.5 Entity Exploration

Your task is to select the minimal set of relevant entities from the given triplets.
{In-Context Few-shot}
Task Requirements:
1. Entities must come strictly from the given list; do not introduce new entities.
2. Strictly output only the selected entities, without explanations or additional formatting.
Now, based on the following input, select the minimal relevant entities.
Question:
Triplets:

J.6 Answer Generation

Your task is to infer the answer to the original question by first reasoning over each Topic Question using the provided triplets and your knowledge, then combining the insights from all Topic Questions to derive the final answer.
Instructions:
1. For each topic entity:
- Read its corresponding topic question.
- Use the associated triplets and your knowledge to infer an answer.
2. After processing all topic entities:
- Analyze how the individual answers relate to the original question.
- If possible, synthesize them to derive the final answer.
{In-Context Few-shot}
Task Requirements:
1. Do not provide explanations or extra text.
2. The output must be in strict JSON format.
Now, based on the following input, determine whether the question can be answered.

Original Question:
Topic 1:
Topic Question:
Topic Entity:
Triplets:
Topic 2:
...

K SPARQL

K.1 Relation Search

```
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT DISTINCT ?relation
WHERE {
    ns:%s ?relation ?x .
    FILTER (?x != ns:%s)
}
```

```
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT DISTINCT ?relation
WHERE {
    ?x ?relation ns:%s .
    FILTER (?x != ns:%s)
}
```

K.2 Entity Search

```
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT ?tailEntity
WHERE {
    ns:%s ns:%s ?tailEntity .
}
```

```
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT ?tailEntity
WHERE {
    ?tailEntity ns:%s ns:%s .
}
```