Ontology-Guided Reverse Thinking Makes Large Language Models Stronger on Knowledge Graph Question Answering

Runxuan Liu[1], **Bei Luo**[2], **Jiaqi Li**[3,4*], **Baoxin Wang**[1,3], **Ming Liu**[1,5†], **Dayong Wu**[3], **Shijin Wang**[3], **Bing Qin**[1,5]

[1]Harbin Institute of Technology, Harbin, China
[2]Beijing University of Posts and Telecommunications, Beijing, China
[3]Joint Laboratory of HIT and iFLYTEK, Beijing, China
[4]University of Science and Technology of China, Hefei, China
[5]Pengcheng Laboratory, Shenzhen, China
{rxliu,mliu,qinb}@ir.hit.edu.cn, luobei@bupt.edu.cn

Abstract

Large language models (LLMs) have shown remarkable capabilities in natural language processing. However, in knowledge graph question answering tasks (KGQA), existing methods rely on entity vector matching, but the purpose of the question is abstract and difficult to match with specific entities. As a result, it is difficult to efficiently establish reasoning paths to the purpose, which leads to information loss and redundancy. To address this issue, inspired by human reverse thinking, we propose Ontology-Guided Reverse Thinking (ORT), a novel framework that constructs reasoning paths from purposes back to conditions. ORT operates in three key phases: (1) using LLM to extract purpose labels and condition labels, (2) constructing label reasoning paths based on the KG ontology, and (3) using the label reasoning paths to guide knowledge retrieval. Experiments on the WebQSP and CWQ datasets show that ORT achieves state-of-the-art performance and significantly enhances the capability of LLMs for KGQA.

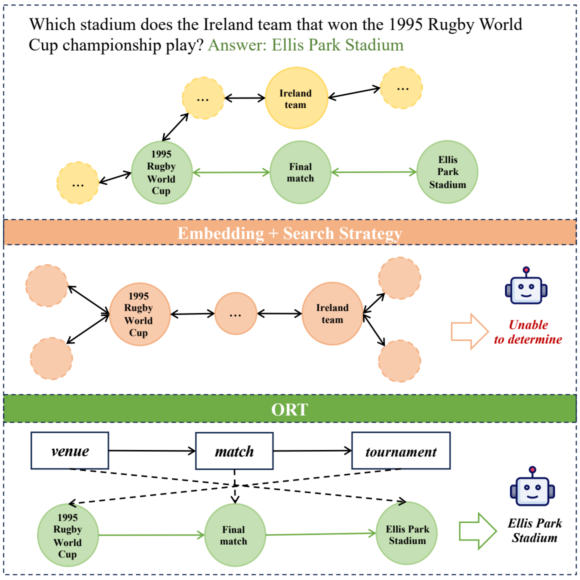


Figure 1: Example of previous methods and ORT. Traditional methods are limited to entity-centric reasoning through vector matching and path collection. In contrast, ORT enables ontology-aware reasoning by identifying conceptual intents, constructing reverse-label reasoning paths, and guiding targeted traversal to the correct answers.

1 Introduction

LLMs have made significant achievements in natural language processing, excelling in tasks such as semantic understanding (Raiaan et al., 2024), text generation (Shen et al., 2024), machine translation (Hu et al., 2024a), dialogue systems (Zhang et al., 2019), sentiment analysis (Li et al., 2025), and text summarization (Basyal and Sanghvi, 2023). LLMs have also been applied in various scenarios, such as the medical field (Wu et al., 2024b) and scientific research support (Wu et al., 2024a).

The rapid development of LLMs has sparked interest in combining LLMs with knowledge graphs to improve KGQA performance (Hu et al., 2024b). Existing approaches typically adopt two paradigms. The first is *fine-tuning methods*, such as LPKG (Wang et al., 2024) and RoG (Luo et al., 2024). However, creating high-quality training data is resource-intensive (Cao et al., 2023). Additionally, knowledge graphs are highly structured data, and when faced with questions that have not been fine-tuned, the quality is difficult to guarantee (Jiang et al., 2024). The second is *embedding + search methods*, such as MindMap (Wen et al., 2024) and Think-on-Graph (Sun et al., 2023), which rely on entity embeddings and graph traversal but are unable to handle conceptual targets absent in KG entities. As shown in Figure 1, "stadium" is a concept, not an entity in the knowledge graph, so "Embedding + Search Strategy" can only find paths between "1995 Rugby World Cup"

[*] Corresponding author.
[†] Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 15269–15284
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

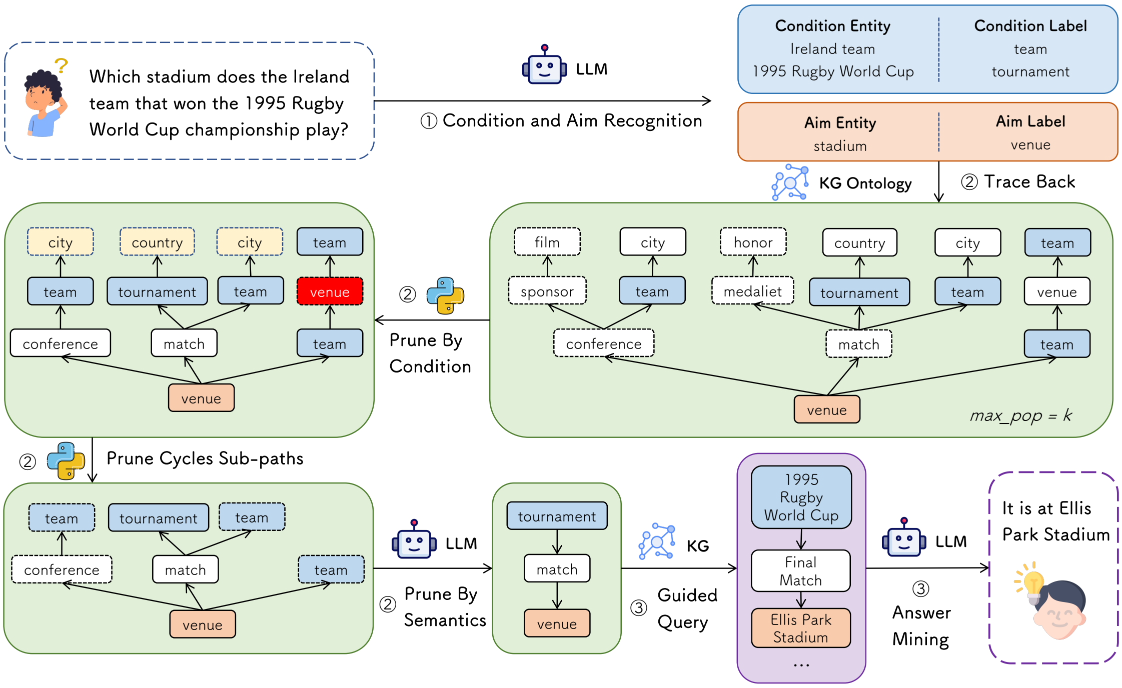


Figure 2: The overall framework of ORT. Starting from a question, the LLM is used to identify conditions and aims in the question, along with their corresponding labels. Using the aim label as the root node, the system iteratively queries related labels on the knowledge graph ontology until the backward max-hop limit is reached. Paths that do not contain condition labels, paths after the last condition label in each sequence, and loops are then pruned. The reasoning paths are used as guidance to query the knowledge graph, and the LLM summarizes the entity paths to derive the final answer.

and “Ireland Team” and their neighbors, but cannot reach “Ellis Park Stadium”.

In this paper, we propose a novel method named Ontology-guided Reverse Thinking (ORT). Our approach begins by extracting not only known entities from the question but also its underlying aims using LLMs, where these aims are represented as entity labels. Building upon these labels, we establish a reasoning framework that combines both ontological structures and reverse thinking principles. Specifically, we construct a reasoning tree that originates from the identified aims and progresses toward the known conditions, effectively creating label reasoning paths with the knowledge graph ontology. This reverse-oriented approach incorporates path pruning, eliminating unnecessary branches during the reasoning process. The refined reasoning paths then guide targeted knowledge queries in the knowledge graph, followed by using LLMs to aggregate knowledge and generate answers. This integrated methodology enables precise knowledge retrieval while minimizing interference from irrelevant information, ultimately enhancing the accuracy of the LLM’s responses.

The experimental results demonstrate that our method significantly improves the answer coverage and quality of LLM KGQA. Compared to direct responses from LLMs, our method achieves a Hit@1 improvement of at least 25.43% and an F1 score improvement of at least 25.82%. As a plug-and-play approach, it greatly enhances the efficiency of LLM KGQA.

In summary, our main contributions include: 1) We first introduce a human-like problem-solving approach for KGQA: Ontology-Guided Reverse Thinking. 2) As a plug-and-play method, we enable LLMs to efficiently understand the structure of the knowledge graph. 3) Experimental results demonstrate a significant improvement in the LLM’s KGQA ability, achieving state-of-the-art among the models studied.

2 Methodology

As shown in Figure 2, the entire algorithm is divided into three steps:

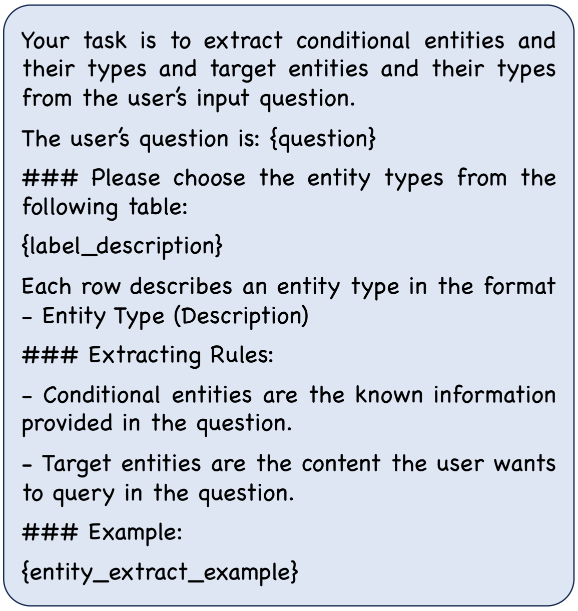
1. **Condition and Aim Recognition**: Prompt the LLM to understand the known conditions and the solving aims of the question.

2. **Ontology-Guided Reverse Thinking Rea-**

soning: Use the Reverse Thinking Reasoning method to construct label reasoning paths on the knowledge graph ontology.

3. **Guided Answer Mining**: Use the label reasoning paths to guide querying and prompt the LLM to generate the final answer.

2.1 Aim and Condition Recognition



Your task is to extract conditional entities and their types and target entities and their types from the user's input question.

The user's question is: {question}

Please choose the entity types from the following table:

{label_description}

Each row describes an entity type in the format - Entity Type (Description)

Extracting Rules:

- Conditional entities are the known information provided in the question.

- Target entities are the content the user wants to query in the question.

Example:

{entity_extract_example}

Figure 3: Prompt template for condition and aim recognition.

This step extracts the condition entities $\mathcal{C}_E = \{c_1, c_2, \ldots, c_n\}$, labels of condition entities $\mathcal{C}_L = \{cl_1, cl_2, \ldots, cl_n\}$, aim entities $\mathcal{A}_E = \{a_1, a_2, \ldots, a_n\}$, and labels of aim entities $\mathcal{A}_L = \{al_1, al_2, \ldots, al_n\}$ from the question by prompting LLM.

Condition is defined as the known key information in the question, while aim is defined as the content the user wants to query through the question. The aim entity refers to the entity in the user question that conveys the intended purpose. In fact, the aim entity is not used in the subsequent processing steps. It serves as an intermediate step for obtaining its corresponding labels. The aim entity labels represent a set of related labels of the aim entity in the knowledge graph, play a crucial role in establishing a mapping from the user question to the knowledge graph ontology. This label identification process effectively addresses the limitations of relying solely on vector-based matching.

We provide the LLM with a Label List of the knowledge graph, prompting the LLM to first extract $\mathcal{C}_E$ and $\mathcal{A}_E$, and then assign labels to the respective entities. The main content of the prompt template is shown in Figure 3, and the complete content of the prompt can be found in Appendix D.

2.2 Ontology-Guided Reverse Thinking Reasoning

Knowledge graph reasoning differs from document reasoning in that its data is structured, making the effective use of structural information particularly important (Thambi and Reghuraj, 2022). We propose for the first time the use of a knowledge graph ontology (KG ontology) to construct label reasoning paths, thereby guiding KG queries to enhance the reasoning ability of LLM with knowledge graphs.

The way we construct paths on KG ontology is Reverse Thinking Reasoning. The constructed path is named as label reasoning path $\mathcal{R}_P$, or abstract reasoning path.

2.2.1 Step I. Construct the Neighbor Label Dictionary

The ontology of the knowledge graph consists of several relation-defined triples. For each label l_i in a triple, we collect all other labels $l_k, l_{k+1}, \ldots$ that appear in the same relation-defined triple.

To express this, we introduce a function $\mathcal{N}(l_i)$, which denotes the set of labels l_k that appear in the same triple as l_i:

$$\mathcal{N}(l_i) = \{l_k \mid (l_i, \text{relationship}, l_k) \in \mathcal{G}\} \qquad (1)$$

where $\mathcal{G}$ represents the set of all triples in the knowledge graph. Then, we construct a neighbor label dictionary, denoted as $\mathcal{D}$, where l_i is the key, and $\mathcal{N}(l_i)$ is the value associated with it:

$$\mathcal{D} = \{l_i : \mathcal{N}(l_i)\} \qquad (2)$$

For example, given the following relationships in the knowledge graph ontology: $l_1 \rightarrow l_2$, $l_1 \rightarrow l_3$, and $l_3 \rightarrow l_1$, the neighbor label dictionary $\mathcal{D}$ will be:

$$\mathcal{D} = \begin{cases} l_1 & : [l_2, l_3], \\ l_2 & : [l_1], \\ l_3 & : [l_1] \end{cases}$$

2.2.2 Step II. Construct the Reverse Reasoning Tree

First, since the length of $\mathcal{A}_L$ may be greater than 1, we create a virtual root node and add all aim labels $\mathcal{A}_L = \{al_1, al_2, \ldots, al_m\}$ as its child nodes.

Then, we recursively traverse all child nodes, querying the neighbor label dictionary $\mathcal{D}$ to add all neighboring labels $\mathcal{N}(l_i)$ as child nodes of each current node.

This process continues recursively until the maximum recursion depth max_pop is reached. The maximum recursion depth is determined based on the number of hops of the question.

The reverse reasoning tree, denoted as $\mathcal{T}$, is built as a recursive structure where the nodes represent labels from the knowledge graph, and the edges represent the relationships between them. Due to the limited space in the image, the relationships between the entity labels are not explicitly shown.

2.2.3 Step III. Prune By Conditions

Starting from the root node, we perform a depth-first search (DFS) and record the current path.

When a leaf node is reached, we check if any node in the path matches the condition labels $\mathcal{C}_L$. The pruning is performed as follows:

- If the path contains no condition label nodes, the entire path is removed.
- If the path contains condition label nodes, only the last condition node and its preceding nodes are retained, while the subsequent nodes are deleted.

The pruning algorithm is recursively applied to each child node, using a copy of the path to avoid contaminating the original path. The output is the tree after pruning by conditions, denoted as $\mathcal{T}_{\text{Condition}}$.

2.2.4 Step IV. Prune Cycle Sub-paths

Due to the possibility of bidirectional relationships between two labels, cycles may exist in the reasoning paths.

A depth-first search (DFS) is performed on $\mathcal{T}_{\text{Condition}}$, adding the current node's name to the visited set visited. The pruning algorithm is recursively called for each child node of the current node. If the current node's name already exists in the visited set visited, the edge between the current node and its parent is removed, effectively eliminating the cycle. During backtracking, the current node's name is removed from the visited set so that other paths can access it. The output is the tree after pruning cycles, denoted as $\mathcal{T}_{\text{Cycle}}$.

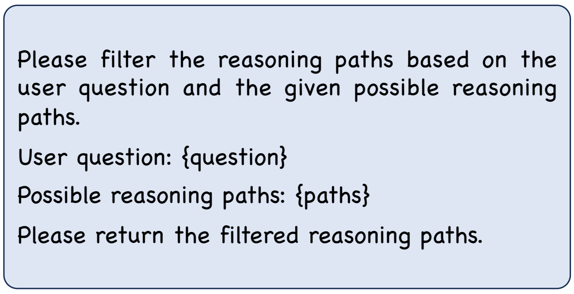
Algorithm 1: Prune Paths by Conditions

```
Input: $\mathcal{R}_P \leftarrow$
        all label reasoning paths by DFS,
        $\mathcal{C}_L \leftarrow$ condition labels
Output: $\mathcal{T}_{\text{Condition}}$
Function PrunePathsByConditions($\mathcal{R}_P$, $\mathcal{C}_L$):
    $\mathcal{T}_{\text{Condition}} \leftarrow \emptyset$;
    foreach $path \in \mathcal{R}_P$ do
        $condition\_indices \leftarrow$
          $[i \mid node_i \in path$ and $node_i \in \mathcal{C}_L]$;
        if $condition\_indices \neq \emptyset$ then
            $last\_condition\_index \leftarrow$
              last element of $condition\_indices$;
            $\mathcal{T}_{\text{Condition}}$.append($path[: last\_condition\_index + 1]$);
    return $\mathcal{T}_{Condition}$;
```

2.2.5 Step V. Prune By Semantics

As shown in Figure 2, after pruning by conditions and cycles, interference paths such as "team → conference → venue" may still exist. To remove these irrelevant paths, semantic information is used for pruning.

A depth-first search (DFS) is performed on all paths of $\mathcal{T}_{\text{Cycle}}$, and the paths are reversed to forward paths. These paths, together with the problem, are input into LLM. The model is prompted using a template to output the paths that are beneficial for answering the question. The main content of the prompt template is shown in Figure 4, and the complete content can be found in Appendix D.



Please filter the reasoning paths based on the user question and the given possible reasoning paths.

User question: {question}

Possible reasoning paths: {paths}

Please return the filtered reasoning paths.

Figure 4: Prompt template for prune by semantics with LLM.

2.3 Guided Answer Mining

Through Ontology-Guided Reverse Thinking Reasoning, abstract reasoning paths for solving the problem are obtained. They are then used to guide the forward knowledge graph query process to collect entity reasoning paths.

A tree structure is used to store the results of each query step. The process is driven by traversing the abstract path, which consists of a sequence of labels. For each reasoning path, the first node is a condition node, and all entities that satisfy the label of this node are added as child nodes to the tree. Then, for each of these child nodes, the next label in the abstract path is used to query the neighboring entities of the current entity. Only those neighbors whose label matches the next label in the abstract path are retained and added as children of the current child node. This process continues iteratively, following the order of labels in the abstract path.

If there are many neighboring entities satisfying the next label, and the number exceeds the limit, top_k neighboring entities are randomly selected and added to the tree. This process is recursively applied until the reasoning path is fully traversed.

After the forward entity tree is built, a depth-first search (DFS) is performed to collect all entity paths, which are then input into LLM along with the problem to generate the final answer. The complete content of the prompt can be found in Appendix D.

3 Experiments

3.1 Experimental Setup

Benchmarks We conducted experiments on two widely used KGQA datasets: WebQuestionSP (WebQSP) (tau Yih et al., 2016) and ComplexWebQuestions (CWQ) (Talmor and Berant, 2018). Both datasets are constructed by extracting data from the Freebase knowledge graph. In our experiments, we follow RoG (Luo et al., 2024) to construct knowledge graphs for WebQSP and CWQ. More details can be found in Appendix A.

Evaluation Metrics Following previous work (Luo et al., 2024; Zhang et al., 2022; Li et al., 2024; Tan et al., 2024), we use Hit@1 and F1 as evaluation metrics. We also provide detailed results for accuracy, precision, and recall in Appendix C. Hit@1 refers to selecting the top-1 prediction and checking whether the true answer is included. If yes, the score is 1; otherwise, it is 0. This measures the answer coverage. Since the prediction may contain invalid content, the F1 score is also used as an evaluation metric to balance precision and recall.

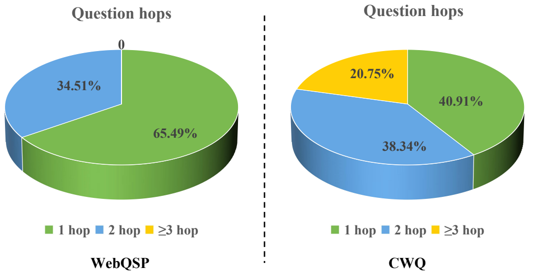


Figure 5: Statistics of the question hops in WebQSP and CWQ.

Baselines We compared a total of five categories of baselines, including:

- **Embedding-based**: KV-Mem (Miller et al., 2016) and NSM (He et al., 2021);
- **Retrieval-based**: GraphNet (Sun et al., 2018) and SR (Zhang et al., 2022);
- **LLM**: GPT-4, DeepSeek-v3 (DeepSeek-AI et al., 2024), and Qwen-max;
- **Fine-tuned LLM knowledge graph reasoning methods (KGR FT)**: KD-CoT (Wang et al., 2023) and RoG (Luo et al., 2024);
- **Non-fine-tuned LLM knowledge graph reasoning methods (KGR w/o FT)**: MindMap (Wen et al., 2024) and KG-Retriever.

Details of these baselines can be found in Appendix A.

3.2 ORT Achieves SOTA

As shown in Table 1, ORT achieves state-of-the-art performance on both WebQSP and CWQ. The base LLM for all LLM+KGs(non-Fine-tuned) methods including our method shown in Table 1 is DeepSeek-v3.

Compared to small-scale models based on embedding or retrieval, on WebQSP, Hit@1 improves by 20% to 42.7%, and F1 improves by 7.7% to 37.3%; on CWQ, Hit@1 improves by 22.7% to 54.5%, and F1 improves by 15.5% to 46.9%.

ORT also outperforms partially *KGR FT* methods such as KD-CoT and RoG. This demonstrates that our method not only enhances question-answering performance but also reduces costs, improves model scalability, and enhances adaptability to different knowledge graphs.

Table 1: The result of our method and other baseline methods on the WebQSP dataset and the CWQ dataset.

Type	Method	WebQSP		CWQ	
		Hit@1	F1	Hit@1	F1
Embedding	KV-Mem (Miller et al., 2016)	46.7	34.5	18.4	15.7
	NSM (He et al., 2021)	68.7	62.8	47.6	42.4
Retrieval	GraftNet (Sun et al., 2018)	66.4	60.4	36.8	32.7
	SR+NSM (Zhang et al., 2022)	68.9	64.1	50.2	47.1
	SR+NSM+E2E (Zhang et al., 2022)	69.5	64.1	49.3	46.3
LLMs	GPT-4o	61.8	43.6	38.2	32.9
	Qwen-max	59.0	40.0	36.4	29.5
	DeepSeek-v3 (DeepSeek-AI et al., 2024)	64.0	43.9	41.1	33.8
LLM+KGs(Fine-tuned)	KD-CoT (Wang et al., 2023)	68.6	52.5	55.7	-
	RoG (Luo et al., 2024)	85.7	70.8	62.6	56.2
LLM+KGs(non-Fine-tuned)	KG Retriever	63.0	42.9	46.7	40.2
	MindMap (Wen et al., 2024)	64.9	47.1	48.8	43.3
	ORT	**89.4**	**71.8**	**72.9**	**62.6**

Besides, compared to pure LLM and *KGR w/o FT* methods, ORT also achieves significant improvements, which will be discussed later.

Furthermore, as shown in Figure 5, WebQSP primarily focuses on single-hop questions, with 65.49% requiring only one hop and no questions exceeding three hops, whereas CWQ contains more complex multi-hop questions, with 20.75% requiring three or more hops. The inferior performance of current methods on CWQ compared to WebQSP further underscores the limitations of existing approaches in addressing multi-hop reasoning tasks.

3.3 Soars LLM's KGQA Ability

As shown in Figure 6, we conducted comparative experiments on three LLMs: GPT-4o, DeepSeek-v3, and Qwen-Max. Compared to direct answers, LLMs using ORT led to more than a 25% improvement in Hit@1 and F1 scores across both datasets.

On WebQSP, Hit@1 of the three LLM respectively improved by 25.7%, 25.3%, and 29.14%, and F1 score increased by 28.23%, 27.96%, and 31.69%. On CWQ, Hit@1 improved by 27.23%, 31.79%, and 31.45%, and F1 score increased by 25.82%, 28.83%, and 28.30%. Details can be found in Table 2.

This not only demonstrates that ORT effectively enhances the performance of LLMs on KGQA tasks but also highlights that ORT is not limited to a specific LLM. It serves as a universal enhancement strategy and can be directly used for improving LLM performance. It has the potential to become an effective, convenient, and important tool in such a domain.

3.4 ORT Outperforms Peers

ORT achieves better performance compared to other methods of the same type. Using the same base LLM (GPT-4, DeepSeek-v3, and Qwen-max), ORT was compared with MindMap and KG Retriever, as shown in Table 3. On WebQSP, ORT achieved an average improvement of 26.56% in Hit@1 and 26.27% in F1 over MindMap across the three base models. On CWQ, ORT outperformed MindMap with an average improvement of 20.18% in Hit@1 and 16.86% in F1.

3.5 Ablation Study

Through ablation study, we aim to demonstrate the effectiveness and necessity of Reverse Think-

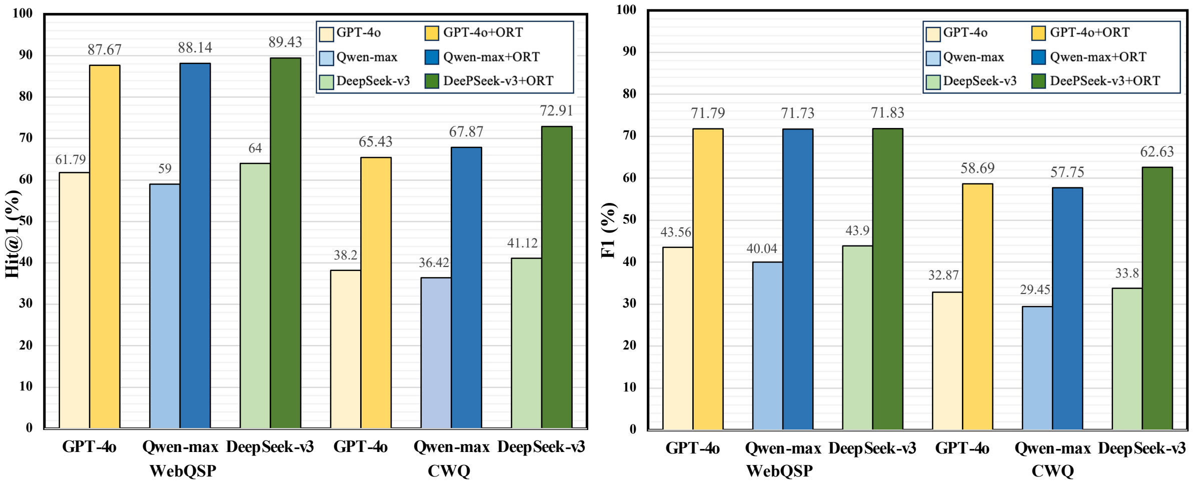


Figure 6: Comparison of LLM vs. LLM+Our Method on WebQSP and CWQ Datasets. The left side is Hit@1 (%), and the right side is F1 (%).

Table 2: Comparison of LLM vs. LLM+Our Method on WebQSP and CWQ Datasets

Method	WebQSP		CWQ	
	Hit@1	F1	Hit@1	F1
GPT-4o	61.79	43.56	38.20	32.87
GPT-4o + ORT	**87.67** (↑25.7)	**71.79** (↑28.23)	**65.43** (↑27.23)	**58.69** (↑25.82)
QWen-max	59.00	40.04	36.42	29.45
QWen-max + ORT	**88.14** (↑29.14)	**71.73** (↑31.69)	**67.87** (↑31.45)	**57.75** (↑28.3)
DeepSeek-v3	64.0	43.9	41.12	33.80
DeepSeek-v3 + ORT	**89.43** (↑25.43)	**71.83** (↑27.93)	**72.91** (↑31.79)	**62.63** (↑28.83)

Table 3: The results of non-Fine-Tuned LLM KG Reasoning methods on WebQSP and CWQ

Method	WebQSP		CWQ	
	Hit@1	F1	Hit@1	F1
ORT + GPT-4o	**87.67**	**71.79**	**65.43**	**58.69**
MindMap + GPT-4o	61.17	46.09	51.33	44.84
KG Retriever + GPT-4o	60.15	42.44	46.67	41.14
ORT + DeepSeek-v3	**89.43**	**71.83**	**72.91**	**62.63**
MindMap + DeepSeek-v3	64.92	47.14	48.83	43.30
KG Retriever + DeepSeek-v3	63.01	42.87	47.67	40.20
ORT + QWen-max	**88.14**	**71.73**	**67.87**	**57.75**
MindMap + QWen-max	59.46	43.31	45.50	40.35
KG Retriever + QWen-max	57.16	39.91	45.00	38.99

ing Reasoning, Knowledge Graph Structure-Based Reasoning, and Rule-Guided Reasoning. We designed it in three parts:

1. **w/o LLM Filter**: In this part, we removed using LLMs for pruning based on the semantics of questions and paths.
2. **Trace Forward**: We additionally designed a forward reasoning algorithm, which collects reasoning paths starting from the conditions and iterating towards the goals.
3. **w/o Rules**: In this part, label reasoning paths are not constructed, and instead, LLM directly generates answers.

The experimental results are shown in Table 4, including Hit@1, F1, Precision, and Recall. Precision measures the proportion of correct predictions among all predicted results, while Recall measures the proportion of correct predictions identified from all ground truth instances.

w/o LLM Filter Analysis As seen in Table 4, on WebQSP, Hit@1 and Precision decreased, but Recall and F1 improved. To ensure the reliability

Table 4: Ablation Experiment Results

Method	WebQSP				CWQ			
	Hit@1	Precision	Recall	F1	Hit@1	Precision	Recall	F1
ORT	**89.43**	**80.92**	74.51	71.83	**72.91**	**65.57**	**66.03**	**62.63**
w/o LLM Filter	86.58	75.10	**78.54**	**73.01**	62.58	57.45	54.76	53.24
Trace Forward	77.82	71.92	61.91	58.73	60.73	51.50	54.18	49.30
w/o Rules	64.00	56.91	46.55	43.87	41.12	36.42	36.40	33.80

of predictions and reduce hallucinations, we still choose to retain the use of LLM to filter abstract paths.

Trace Forward Analysis To contrast with Trace Back, we designed the Trace Forward method, which performs forward reasoning starting from the conditions. The basic pipeline is to extract the conditions from the question, and then iteratively perform a breadth-first search from several conditions on the KG ontology to construct a reasoning tree. A depth-first search from the conditions outputs all paths, which, along with the question, are given to LLM to filter abstract paths that semantically match. Then, abstract paths are used to retrieve entity paths from the knowledge graph, which are passed to LLM to generate the final answer.

The potential drawback of this method is that it may collect irrelevant abstract paths, and overly depends on LLM to generate reasoning paths. As seen in Table 4, on WebQSP, Trace Forward's Hit@1 decreased by 11.61%, and F1 decreased by 13.10%. One CWQ, Trace Forward's Hit@1 decreased by 12.18%, and F1 decreased by 13.33%. However, this method still performed better than MindMap, which highlights the necessity of utilizing the KG ontology and using abstract paths to guide knowledge retrieval.

w/o Rules Analysis Finally, to demonstrate the necessity of Rule-Guided Reasoning, we conducted experiments without rule guidance, i.e., directly using LLM to generate answers without generating abstract paths. The experimental results show that this method's performance significantly decreased.

4 Related Work

Small-scale models for KGQA Small-scale methods for knowledge graph question answering (KGQA) can be divided into two categories: embedding-based and retrieval-based methods. Embedding-based methods, such as KV-Mem (Miller et al., 2016) and NSM (He et al., 2021), represent entities and relations in a low-dimensional vector space, performing well on simple, single-hop queries. However, they struggle with complex, multi-hop queries due to difficulty in capturing intricate path information. To address this, retrieval-based models like GraphNet (Sun et al., 2018) and SR (Zhang et al., 2022) construct subgraphs or paths for reasoning, showing improvements in multi-hop tasks by better leveraging structural relationships. Yet, both methods are limited by incomplete utilization of the full structural information in the knowledge graph.

Fine-tuning LLMs for KGQA In recent years, the rapid development of large language models (LLMs) has sparked interest in combining LLMs with knowledge graphs to improve KGQA performance. Models like RoG (Luo et al., 2024), KD-CoT (Wang et al., 2023), UniKGQA (Jiang et al., 2022), and DeCAF (Yu et al., 2022) have demonstrated impressive results by fine-tuning LLMs to generate reasoning paths and produce answers. These models excel at tackling complex KGQA tasks, where multi-hop reasoning is required. However, fine-tuning LLMs often demands vast computational resources and labeled datasets, making it challenging to scale these methods for practical, real-world applications.

Non-fine-tuning LLMs for KGQA Some recent approaches focus on methods that utilize LLMs for KGQA without requiring additional training. MindMap Wen et al. (2024) is one such method that extracts entities from the query and performs a breadth-first search in the knowledge graph to generate reasoning paths. Additionally, Chen et al. (2024) proposes a model that feeds all the relations in the knowledge graph to the LLM to help generate relational paths. Although these methods can avoid the high computational costs associated with fine-tuning, they often suffer from a lack of deep understanding of the underlying structure of the knowledge graph, which can lead to the generation of lower-quality reasoning paths.

5 Conclusion

In this paper, we simulate the cognitive paradigm that humans use to solve complex problems and propose the Ontology-Guided Reverse thinking method for knowledge graph question answering. We use LLMs to understand the intent of the question and generate the corresponding labels. By leveraging the knowledge graph ontology, we use Reverse-Thinking Reasoning to form label reasoning paths, followed by guided knowledge graph queries and answer aggregation. Experimental results show that our method significantly improves the accuracy and answer coverage.

Limitations

For this work, we want to address two areas for improvement in the future. First, when querying the knowledge graph along reasoning paths, there may be a large number of entities satisfying the label constraints, which could lead to irrelevant results. Second, when generating the final answer, inputting all entity paths into the LLMs may introduce irrelevant paths, potentially lowering the accuracy of the answer.

Acknowledgement

The research in this article is supported by the New Generation of Artificial Intelligence National Science and Technology Major Project (2023ZD0121503), the National Science Foundation of China (U22B2059, 62276083).

References

Lochan Basyal and Mihir Sanghvi. 2023. Text summarization using large language models: A comparative study of mpt-7b-instruct, falcon-7b-instruct, and openai chat-gpt models. *ArXiv*, abs/2310.10449.

Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction mining: Instruction data selection for tuning large language models.

Zhongwu Chen, Long Bai, Zixuan Li, Zhen Huang, Xiaolong Jin, and Yong Dou. 2024. A new pipeline for knowledge graph reasoning enhanced by large language models without fine-tuning. In *Conference on Empirical Methods in Natural Language Processing*.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, and Bing-Li Wang. 2024. Deepseek-v3 technical report. Technical report, DeepSeek-AI.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.

Yuchen Hu, Chen Chen, Chao-Han Yang, Ruizhe Li, Dong Zhang, Zhehuai Chen, and EngSiong Chng. 2024a. GenTranslate: Large language models are generative multilingual speech and machine translators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 74–90, Bangkok, Thailand. Association for Computational Linguistics.

Yuntong Hu, Zhihan Lei, Zhengwu Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024b. Grag: Graph retrieval-augmented generation. *ArXiv*, abs/2405.16506.

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

Pengcheng Jiang, Lang Cao, Cao Xiao, Parminder Bhatia, Jimeng Sun, and Jiawei Han. 2024. Kg-fit: Knowledge graph fine-tuning upon open-world knowledge. *ArXiv*, abs/2405.16412.

Mufei Li, Siqi Miao, and Pan Li. 2024. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *ArXiv*, abs/2410.20724.

Yunxin Li, Zhenyu Liu, Zitao Li, Xuanyu Zhang, Zhenran Xu, Xinyu Chen, Haoyuan Shi, Shenyuan Jiang, Xintong Wang, Jifang Wang, Shouzheng Huang, Xinping Zhao, Borui Jiang, Lanqing Hong, Longyue Wang, Zhuotao Tian, Baoxing Huai, Wenhan Luo, Weihua Luo, Zheng Zhang, Baotian Hu, and Min Zhang. 2025. Perception, reason, think, and plan: A survey on large multimodal reasoning models. *arXiv preprint arXiv:2505.04921*.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. *International Conference on Learning Representations*, abs/2310.01061.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *ArXiv*, abs/1606.03126.

Mohaimenul Azam Khan Raiaan, Md. Saddam Hossain Mukta, Kaniz Fatema, Nur Mohammad Fahad, Sadman Sakib, Most. Marufatul Jannat Mim, Jubaer Ahmad, Mohammed Eunus Ali, and Sami Azam. 2024. A review on large language models: Architectures, applications, taxonomies, open issues and challenges. *IEEE Access*, 12:26839–26874.

Jocelyn Shen, Joel Mire, Hae Won Park, Cynthia Breazeal, and Maarten Sap. 2024. HEART-felt narratives: Tracing empathy and narrative style in personal stories with LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1026–1046, Miami, Florida, USA. Association for Computational Linguistics.

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

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Sai Wang, Chen Lin, Yeyun Gong, Heung yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *ArXiv*, abs/2307.07697.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *ArXiv*, abs/1803.06643.

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. 2024. Paths-over-graph: Knowledge graph empowered large language model reasoning. *ArXiv*, abs/2410.14211.

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 *Annual Meeting of the Association for Computational Linguistics*.

Sincy V. Thambi and P. C. Reghuraj. 2022. Towards improving the performance of question answering system using knowledge graph - a survey. *2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, pages 672–679.

Junjie Wang, Mingyang Chen, Binbin Hu, Dan Yang, Ziqi Liu, Yue Shen, Peng Wei, Zhiqiang Zhang, Jinjie Gu, Jun Zhou, Jeff Z. Pan, Wen Zhang, and Huajun Chen. 2024. Learning to plan for retrieval-augmented large language models from knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7813–7835, Miami, Florida, USA. Association for Computational Linguistics.

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

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2024. MindMap: Knowledge graph prompting sparks graph of thoughts in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10370–10388, Bangkok, Thailand. Association for Computational Linguistics.

Dayong Wu, Jiaqi Li, Baoxin Wang, Honghong Zhao, Siyuan Xue, Yanjie Yang, Zhijun Chang, Rui Zhang, Li Qian, Bo Wang, Shijin Wang, Zhixiong Zhang, and Guoping Hu. 2024a. Sparkra: A retrieval-augmented knowledge service system based on spark large language model. *ArXiv*, abs/2408.06574.

Junde Wu, Jiayuan Zhu, and Yunli Qi. 2024b. Medical graph rag: Towards safe medical large language model via graph retrieval-augmented generation. *ArXiv*, abs/2408.04187.

Donghan Yu, Shenmin Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, J. Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. *ArXiv*, abs/2210.00063.

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 *Annual Meeting of the Association for Computational Linguistics*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B. Dolan. 2019. Dialogpt : Large-scale generative pre-training for conversational response generation. In *Annual Meeting of the Association for Computational Linguistics*.

A EXPERIMENT DETAILS

A.1 Datasets

To evaluate the performance of ORT on knowledge graph question and answer tasks, we conducted experiments on two multi-hop datasets (CWQ (Talmor and Berant, 2018) and WebQSP (Devlin et al., 2019)). The questions in both datasets cover various domains, including people, places, events, etc. Due to the complexity of the questions, traditional question answering systems and search-based engines often struggle to provide valuable knowledge. FreeBase (Bollacker et al., 2008) serves as the background knowledge graph for both datasets, containing approximately 88 million entities, 20,000 relationships, and 126 million triples.

Similar to the datasets used in ROG, we extracted 3,531 question-answer pairs from the CWQ dataset as the test set, which includes 2,294,264 triples and 4,726 relationships. We also extracted 1,628 question-answer pairs from the WebQSP dataset as the test set, which includes 2,277,228 triples and 5,051 relationships. For details, see Table 5.

A.2 Metrics

Accuracy is the ratio of the number of correct predictions to the total number of predictions. The formula is as follows:

$$\text{Accuracy} = \frac{\sum_{i=1}^{N} \mathbb{I}(\hat{y}_i \in A_{\text{gold},i})}{N} \tag{3}$$

Hit@1 is whether the most probable prediction among the model's multiple outputs contains the ground truth. If yes, the Hit@1 score is 1; otherwise, the score is 0. Because our method has only one output, there is no need to select the prediction with the highest probability. For example, consider the question "What religion does India follow?" The correct answer is "Hinduism," and the model's predicted answers are "Christianity, Hinduism, Islam." In this case, since "Hinduism" appears in the model's predicted answers and it is the correct answer, the Hit@1 score is 1. The formula is as follows:

$$Hit@1 = \mathbb{I}(\exists \hat{y}_i \in A_{\text{gold}}) \tag{4}$$

Precision is the ratio of the number of correct predictions to the total number of predictions. The formula is as follows:

$$Precision = \frac{\sum_{i=1}^{N} \mathbb{I}(\hat{y}_i \in A_{\text{gold},i})}{N_{\text{pred}}} \tag{5}$$

Recall measures how many of the standard answers the model can correctly predict. The calculation method is the same as that of accuracy.

F1 score is the harmonic mean of precision and recall. The formula is as follows:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6}$$

A.3 Baselines

Baselines are grouped into 12 baseline methods into 5 categories: 1) **Embedding-based methods**, 2) **Retrieval-augmented methods**, 3) **LLM**, 4) **LLM+KGs (Fine-tuned)**, and 5) **LLM+KGs (non-Fine-tuned)**. The detailed information for each baseline is as follows:

Embedding-based methods

- KV-MEM (Miller et al., 2016) employs a key-value memory network to store triples and performs multi-hop reasoning by iterating operations over memory.

- NSM(He et al., 2021) uses a sequential model to mimic the multi-hop reasoning process.

Retrieval-augmented methods

- GraftNet (Sun et al., 2018) retrieves relevant subgraphs from knowledge graphs with entity linking.

- SR+NSM (Zhang et al., 2022) introduces a relation-path retrieval mechanism to fetch subgraphs for multi-hop reasoning.

- SR+NSM+E2E (Zhang et al., 2022) further adopts an end-to-end training strategy to jointly train the retrieval and reasoning modules of SR+NSM.

LLM methods

- GPT-4 is a large language model developed by OpenAI, renowned for its excellent performance across a wide range of natural language processing tasks.

- DeepSeek-v3 (DeepSeek-AI et al., 2024) is an advanced model designed for deep reasoning and retrieval-augmented tasks, focusing on domain-specific knowledge extraction.

- Qwen-max is a large model optimized for multilingual and multi-task learning, known for its strong capabilities in both generative and analytical tasks.

Table 5: The statistics of the used datasets.

Datasets	Complex WebQuestions	WebQuestionSP
Domain	English General Q&A	English General Q&A
KG dataset	FreeBase	FreeBase
Question	3531	1628
Node	684846	781490
Triple	2294264	2277228
Relationship	4726	5051

LLM+KGs (Fine-tuned) methods

- KD-COT (Wang et al., 2023) retrieves relevant knowledge from KGs to formulate faithful reasoning plans for LLMs.
- ROG (Luo et al., 2024) combines knowledge graphs (KGs) and large language models (LLMs) to achieve reliable and interpretable reasoning through a planning-retrieval-reasoning framework.

LLM+KGs (non-Fine-tuned) methods

- KG Retriever aims to find the shortest path between each pair of question entities, and then retrieves the final prompt from the KG to guide the LLM in answering the question. The key difference between MindMap and KG Retriever is that they do not use diverse multiple pieces of evidence in the LLM, nor do they ORT the evidence sources.
- MindMap (Wen et al., 2024) integrates knowledge graphs (KGs) and large language models (LLMs) by using KGs to provide explicit knowledge and reasoning paths, enhancing the LLM's reasoning ability and transparency while revealing the thought process of the LLM.

B Implementation Details

B.1 Label List Construction

We utilize the datasets provided by RoG to conduct our experiments. The relations in the triples of the knowledge graph contain label information. For example, in the triple [Jamaica, meteorology.cyclone_affected_area.cyclones, Tropical Storm Keith], the entity "Jamaica" is assigned the label *cyclone_affected_area*, while "Tropical Storm Keith" is labeled as *cyclones*.

B.2 Ontology Construction

We traverse the dataset triples and extract entity label information in a normalized format, constructing abstract triples of the form [label, relation, label]. These abstract triples collectively form the knowledge graph ontology.

B.3 Hyperparameter Settings

The hyperparameters used in our experiments are as follows:

- **Maximum reasoning hops:** `MAX_POP = 5`. The maximum hop count for the WebQSP and CWQ datasets is 4, so the maximum reasoning hops are set to 5 to ensure sufficient reasoning depth.
- **Maximum number of neighbors:** `TOP_K = 10`. Here, neighbors refer to entities in the *Guided Answer Mining* phase that satisfy the `next_label` constraint.

C ADDITIONAL RESULTS

We have added Accuracy, Precision, and Recall to further observe the Model Improvement Comparison between WebQuestionSP and ComplexWebQuestions. You can find the details in Table 6.

D PROMPTS

We demonstrate all the prompt templates used, including "Aims and Conditions Recognition", "Prune by Semantics" and "Generate Final Answer With LLMs", as shown in Figure 7, Figure 8, and Figure 9.

E CASES

We will present two cases in Table 7 and Table 8 to illustrate the process of our method.

Extract Aims and Conditions Template

I am developing a knowledge graph enhanced question answering system.
Your task is to extract **conditional entities and their types** and **destination entities and their types** from user input questions.

Please select the type of entity from the following table:
Each line describes an entity type, in the format of - entity type (description information)
{label_description}

Rules:
-The conditional entity is the known information provided in the problem;
-The target entity is the content that the user wants to query in the problem;
-If there is no suitable entity, please use 'none' to indicate.

Output format:
-Separate the conditional entity and the destination entity with a period;
-The format of each entity is * * "Entity Name, Entity Type" * *;
-If there are multiple conditional entities or destination entities, use * * ";" * * (semicolons) to separate them;
-If there is only one of the conditions and purposes, for example:
-Output when there is only a conditional entity and no destination entity: ce1,cl1; ce2,cl2.none,none
-Output when there is only a destination entity and no conditional entity: none,none.ae1,al1; ae2,al2
-Only output the final answer, without including unnecessary explanations, clarifications, or text.

Example:
Example1:
Input: Lou Seal is the mascot for the team that last won the World Series when?
Output: Lou Seal,mascot. championship, championship
......

The user's question is: {question}
Please generate an answer that conforms to the above format:

Figure 7: The prompt template for "Aim and Condition Recognition"

Prune By Semantics Template

Please filter the reasoning paths based on the user question and the given possible reasoning paths.

User question: {question}
Possible reasoning paths: {paths}

Explanation:
- I have used a LLM to extract known conditions from the user question.
- Starting from these known conditions, I performed a depth-first search in the domain knowledge graph to extract all reasoning paths that start with the labels of these conditions.
- Each path begins with a condition entity, and the path connects multiple entity labels.

Filter criteria:
-Try to filter out paths that are helpful for the answer as much as possible. If the user asks' What could be the problem? ', then the pathways for diseases, medical examinations, and medication should be preserved. A separate pathway to the disease should also be kept.
-Ensure that the output paths are not duplicated.

Please return the filtered reasoning paths.

Figure 8: The prompt template for "Prune by Semantics"

Generate Final Answer Template

The user has input the following question: {question}

I can provide you with some reference content, where each set of content consists of two parts: conditions and objectives.
- Conditions: The known information from the question.
- Objectives: The goals that the question seeks to address.

Below is the conditions and objectives: {last_node_str}

I can also provide you the complete reasoning paths which maybe useful for you. I wish you cloud utilize your reasoning ability to answer users' question
- Each path is described by nodes and edges in the following format: [Entity Type] Entity Name -> (Relation) [Entity Type] Entity Name -> ...
- Each node includes [Entity Type] Entity Name.
- Each edge is represented by an arrow ->, with the edge information enclosed in parentheses, e.g., (Relation).
- Starting from the root node, the path is described step by step, including nodes and their relationships, until reaching the leaf node.

The abstract paths are as follows: {rules_string}
Below is the reasoning paths: {reasoning_path_str}

The output can refer to the following format requirements: {ReferenceTemplate}

Please follow the reference content to answer the question, applying logical reasoning as needed to generate the final answer.

Figure 9: The prompt template for "Guided Answer Mining"

Table 6: Detailed Experiment Results of Model Improvement Comparison between WebQSP and CWQ

Method	WebQSP					CWQ				
	Accuracy	Hit@1	Precision	Recall	F1	Accuracy	Hit@1	Precision	Recall	F1
GPT-4o	43.29	61.79	61.07	43.29	43.56	33.61	38.20	36.56	33.61	32.87
GPT-4o + ORT	71.86	87.67	88.00	71.86	71.79	58.99	65.43	63.21	58.99	58.69
GPT-4o + MindMap	47.81	61.17	50.33	47.81	46.09	45.52	51.33	49.50	45.52	44.84
GPT-4o + KG Retriever	44.72	60.15	46.64	44.72	42.44	40.82	46.67	45.77	40.82	41.14
QWen-max	41.39	59.00	55.65	41.39	40.04	31.66	36.42	32.29	31.66	29.45
QWen-max + ORT	74.30	88.14	81.00	74.30	71.73	61.61	67.87	58.97	61.61	57.75
QWen-max + MindMap	46.09	59.46	46.57	46.11	43.31	40.59	45.50	44.04	40.59	40.35
QWen-max + KG Retriever	42.23	57.16	43.93	42.25	39.91	40.03	45.00	42.31	40.03	38.99
DeepSeek-v3	46.55	64.00	56.91	46.55	43.87	36.40	41.12	36.42	36.40	33.80
DeepSeek-v3 + ORT	74.51	89.43	80.92	74.51	71.83	66.03	72.91	65.57	66.03	62.63
DeepSeek-v3 + MindMap	50.68	64.92	50.10	50.68	47.14	44.07	48.83	46.79	44.07	43.30
DeepSeek-v3 + KG Retriever	47.88	63.01	46.18	47.88	42.87	42.95	47.67	41.87	42.95	40.20

Table 7: Two cases for better understanding of our method: Case 1.

Question:
Lou Seal is the mascot for the team that last won the World Series when?

Aims:
[[”championship”, ”championship”]]

Conditions:
[[”Lou Seal”, ”mascot”]]

Rule_Paths:
```
mascot -> game -> season -> championship
mascot -> team -> season -> championship
mascot -> team -> championship
mascot -> school -> team -> championship
mascot -> brand -> team -> championship
mascot -> team -> league -> championship
mascot -> team -> relationship -> championship
mascot -> brand -> relationship -> championship
mascot -> game -> event -> championship
mascot -> team -> event -> championship
```

Selected_Rule_Paths:
```
mascot -> team -> championship
mascot -> team -> event -> championship
mascot -> team -> season -> championship
```

Reasoning_Paths:
reasoning path 1: [mascot] Lou Seal -> team [team] San Francisco Giants -> champion [championship] 2010 World Series
reasoning path 2: [mascot] Lou Seal -> team [team] San Francisco Giants -> championship [championship] 2014 World Series
reasoning path 3: [mascot] Lou Seal -> team [team] San Francisco Giants -> champion [championship] 2012 World Series
...
reasoning path 82: [mascot] Lou Seal -> team_mascot [team] San Francisco Giants -> league [season] m.0crt4b6
reasoning path 83: [mascot] Lou Seal -> team_mascot [team] San Francisco Giants -> team [season] National League West -> championship [championship] National League Division Series

Final_Answer:
2014 World Series

Table 8: Two cases for better understanding of our method: Case 2.

Question:
What is the predominant religion where the leader is Ovadia Yosef?

Aims:

```
[["religion", "religion"]]
```

Conditions:

```
[["Ovadia Yosef", "person"]]
```

Rule_Paths:

```
person -> religion
person -> party -> celebrity -> religion
person -> language -> region -> religion
person -> title -> membership -> religion
person -> group -> membership -> religion
person -> leadership -> organization -> religion
person -> child -> organization -> religion
person -> location -> organization -> religion
person -> parent -> organization -> religion
person -> title -> leader -> religion
person -> leadership -> leader -> religion
person -> location -> choice -> religion
```

Selected_Rule_Paths:

```
person -> leadership -> leader -> religion
person -> title -> leader -> religion
person -> leadership -> organization -> religion
```

Reasoning_Paths:

reasoning path 1: [person] Ovadia Yosef -> leader [leadership] m.048bcbz -> leader [leader] Ovadia Yosef -> religion [religion] Judaism

reasoning path 2: [person] Ovadia Yosef -> leader [leadership] m.048bcbz -> leader [leader] Ovadia Yosef -> religion [religion] Haredi Judaism

reasoning path 3: [person] Ovadia Yosef -> leader [leadership] m.048bcbz -> religious_leadership [leader] Ovadia Yosef -> religion [religion] Judaism

reasoning path 4: [person] Ovadia Yosef -> leader [leadership] m.048bcbz -> religious_leadership [leader] Ovadia Yosef -> religion [religion] Haredi Judaism

...

reasoning path 20: [person] Ovadia Yosef -> religious_leadership [leadership] m.048bcbz -> religious_leadership [organization] Ovadia Yosef -> religion [religion] Haredi Judaism

reasoning path 21: [person] Ovadia Yosef -> religious_leadership [leadership] m.048bcbz -> organization [organization] Chief Rabbinate of Israel

Final_Answer:
Judaism