LLM-Based Discriminative Reasoning for Knowledge Graph Question Answering

Anonymous ACL submission

Abstract

Large language models (LLMs) based on generative pre-trained Transformer have achieved remarkable performance on knowledge graph question-answering (KGQA) tasks. However, LLMs often produce ungrounded subgraph planning or reasoning results in KGQA due to the hallucinatory behavior brought by the generative paradigm. To tackle this issue, we propose READS to reformulate the KGQA process into discriminative subtasks, which simplifies the search space for each subtasks. Based on the subtasks, we design a new corresponding discriminative inference strategy to conduct the reasoning for KGQA, thereby alleviating hallucination and ungrounded reasoning issues in LLMs. Experimental results show that the proposed approach outperforms multiple strong comparison methods, along with achieving state-of-the-art performance on widely used benchmarks WebQSP and CWQ. [1]

1 Introduction

Large language models (LLMs) have shown remarkable reasoning capabilities in KGQA task (Yu et al., 2022; Huang and Chang, 2023; Wang et al., 2023b), especially the feasibility to prompt the LLMs to generate searching and reasoning results through the LLMs' built-in knowledge. Typically, based on the given question, LLMs can be prompted to provide a plan for the question-related subgraph through one-time generation. After retrieving the subgraph, LLMs can directly generate the answers along with the reasoning steps using the subgraph as context. Utilizing internal knowledge or reasoning ability distilled from stronger models like GPT-4, the generative KGQA model can effectively conduct knowledge graph reasoning, along with achieving state-of-the-art performance on the KGQA tasks (Mondorf and Plank, 2024; LUO et al., 2024; Sun et al., 2024a).

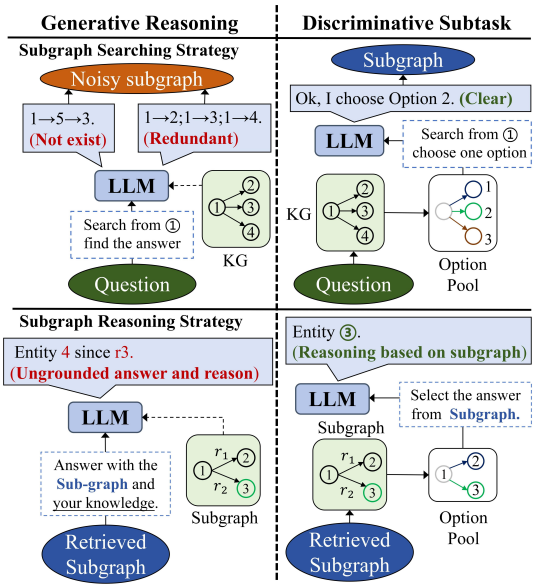


Figure 1: The generation-based methods tend to generate unsupported or redundant subgraphs and reasoning results (left), while the proposed method address the issue by establishing proper searching space for each of the KGQA subtasks (right).

Despite their success, the generative reasoning methods often produce ungrounded planning or reasoning results due to the hallucinatory behavior (Zhang et al., 2023; Sun et al., 2024b; Pan et al., 2024), which is opposite to the deterministic characteristic of knowledge reasoning process (Garcez et al., 2015; Xiong et al., 2024). As shown in Figure 1, when searching question-related subgraphs, generation-based methods come with not existed path "*1->5->3*" due to hallucinatory planning , or retrieve redundant paths at one step "*1->2; 1->3; 1->4*" as a compensate to the generation uncertainty (upper left). When conducting answer inference on the retrieved subgraph, the generation-based methods may generate unreasonable step "*since r3*" as inference chain or even entity "*4*" that

[1] Our code and data will be released upon acceptance.

do not exist in the subgraphs as answers (bottom left). The hallucinatory behavior of the generative LLMs hinders the advancement of KGQA.

To address the issue, we propose LLM-Based **Rea**soning With **D**iscriminative **S**ubtasks (READS) to strengthen the LLM-based knowledge reasoning process. READS decomposes the KGQA process into three discriminative subtasks: graph searching, graph pruning, and answer inference. The decomposition aims to explicitly simulate the capabilities of searching for question-related knowledge, identifying semantic constraints, and inferring the answer position on the subgraph, respectively. Meanwhile, READS simplifies search space from the knowledge graph without toolboxes, along with designed discriminative inference strategy to conduct the reasoning of KBQA effectively. In summary, our main contributions are as follows:

- We introduce READS, an novel reasoning framework that explicitly models KGQA reasoning skills by deconstructing the KGQA process into three discriminative subtasks.
- An effective corresponding discriminative inference strategy is designed to conduct the reasoning of KGQA for READS, thereby significantly alleviating hallucination and ungrounded reasoning issues.
- Experimental results demonstrate that READS achieved state-of-the-art performance on two widely used benchmarks.

2 Related Works

Generative Approaches. The challenge of KGQA task lies in how to conduct precise reasoning on the knowledge graphs (Miller et al., 2016; Yasunaga et al., 2021; Zhu et al., 2024), early works tried to teach models to construct database queries for knowledge graphs, allowing them to directly retrieve answers from the graph (Gu and Su, 2022; Ye et al., 2022). With the advent of LLM's long-horizon planning and reasoning capability (Zhong et al., 2024; Wang et al., 2024), the focus of KGQA research shifts toward leveraging the reasoning capabilities of a single LLM for knowledge inference (Jiang et al., 2022). One straightforward way is to directly schedule the question-related subgraph using the LLM's knowledge (Hong et al., 2023; Wang et al., 2023a). Typical approach like RoG employs chain-like subgraph planning and distills GPT-4's Chain-of-Thought reasoning capability to achieve reliable reasoning processes over knowledge graphs, achieving state-of-the-art performance (LUO et al., 2024). Despite their success, one concern is that those methods often provides incorrect and ungrounded reasoning results due to the hallucinatory generation process.

Interactive and Discriminative Approaches. An alternative approach is to design effective tools and generation strategies to retrieve environment information from the knowledge graph to enhance the step-by-step reasoning process, as seen in approaches such as ToG, KGAgent and GoG (Sun et al., 2024a; Jiang et al., 2024; Xu et al., 2024). However, these interactive generation approaches cannot avoid the influence of hallucinations. Even when provided with environmental information or recalled subgraphs, the model may still arrive at incorrect reasoning results. As claimed in PANGU, using discriminative strategy can effectively mitigates the hallucination problem (Gu et al., 2023). Despite PANGU's success, integrating tools such as search and answer retrieval within the same search space may also lead the LLM to make erroneous decisions.

In this paper, we propose to reformulate KGQA into three subtasks to explicitly model the KGQA skills and design discriminative strategies to effectively enhance LLM's reasoning capability on the knowledge graph.

3 READS Framework

In this section, we propose a novel framework that reformulates the KGQA task into three discriminative subtasks, including question-related subgraph searching, question-related subgraph pruning, and answer inference.

Formulation of KGQA task. Given a question Q and the knowledge graph entities E contained in the question, the KGQA task asks the model to recall golden answers A_{gold} as much as possible. The model has to retrieve question-related subgraphs from the knowledge graph and infer the right answer based on the subgraphs.

The knowledge graph KG used in this paper is Freebase[2], which consists of knowledge triplets represented as $t = (s, p, o)$ including the subject entity s, the object entity o, and the predicate p that connects these two entities.

[2]The two benchmarks used in this work are constructed using Freebase (Bollacker et al., 2008).

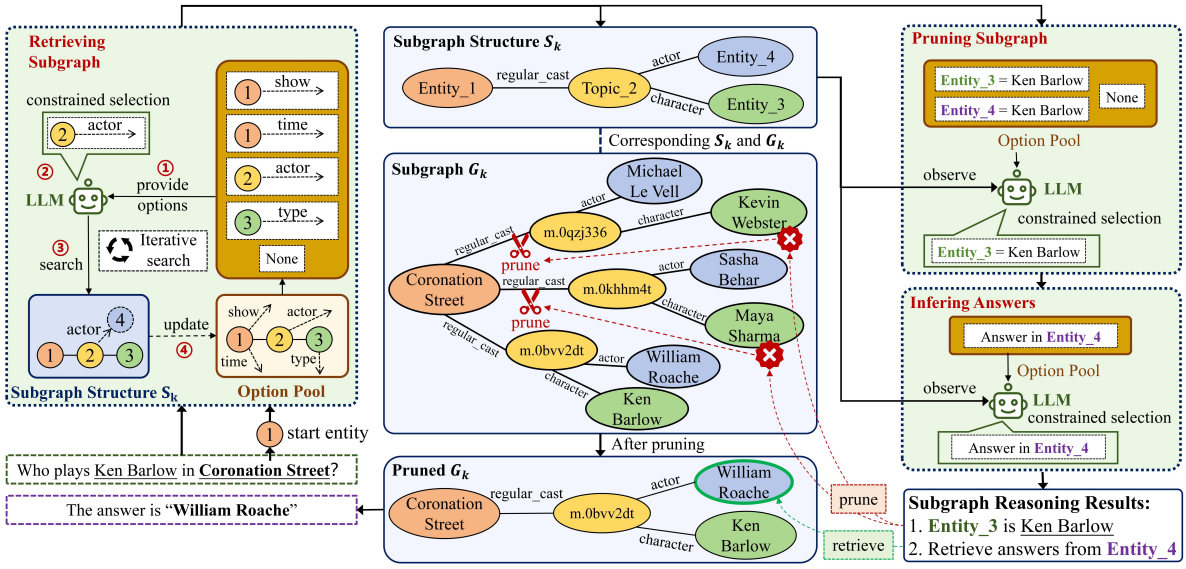


Figure 2: The proposed READS for KGQA. Start from the question (bottom left) with a given starting entity "Coronation Street", READS sequentially conducts subgraph retrieval, subgraph pruning, and answer inference. Then READS automatically uses the reasoning results to prune G_k and then retrieve the answers from it. In this figure, the node's color in subgraph G_k (middle center) represents its position in subgraph structure S_k (top center).

3.1 Question-related subgraph Searching

Given the knowledge graph, the question-related subgraph searching subtask aims to retrieve the question-related subgraph G_k and thereby summarizes an abstract structure S_k for each input question. Specifically, G_k is the subgraph comprising only the necessary knowledge to correctly answer the question Q, which can be represented by a set of triplets:

$$G_k = \{t_i | t_i = (s_i, p_i, o_i)\}. \quad (1)$$

Following UniKGQA (Jiang et al., 2022), we use "semantic nodes" to represent a group of entities sharing same structural position in the knowledge graph. The summarized abstract structure S_k groups all the entities into "semantic nodes" based on their position in G_k:

$$S_k = \{t_i | t_i = (s_{abs}, p_j, o_{abs})\}, \quad s_{abs}, o_{abs} \in Group(G_k). \quad (2)$$

For example, if G_k includes two triples, $(a, friend\_of, b)$ and $(a, friend\_of, c)$, its abstract structure S_k is $\{entity_1, friend\_of, entity_2\}$. Here both entities b and c are grouped into the abstract node $entity\_2$ since they connect to the same entity a with the same relation $friend\_of$. Note that S_k only groups the entities and keeps the name of the relations.

3.2 Question-related Subgraph Pruning

Based on the abstract structure S_k, the question-related subgraph pruning task aims to map all the question-related constraint entities C to nodes in S_k. C denotes the intersection of question mentioned entities $E_{question}$ and all entities in Freebase $E_{freebase}$:

$$C = E_{question} \cap E_{freebase}. \quad (3)$$

Let Node(S_k) represents the set of nodes in S_k, the mapping results between entities in C and nodes in S_k is represented as:

$$\{(C_i, N_i) | C_i \in C, N_i \in Node(S_k)\}. \quad (4)$$

For example, when answering the question: *"What is the name of the team who won the Super Bowl in 2011?"*, there are two constraint entities C_1: "Super Bowl" and C_2: "2011". Now that the retrieved subgraph G_k is always a tree rooted from the starting entity, any branches contain information against the the information in C_1 should be pruned from its root. In order to better focus on the LLM-based discriminative reasoning process, we assume that the entities mentioned in the questions have already been linked to Freebase entities through rule-based recognition methods.

3.3 Answer Inference

Given the question-related subgraph structure S_k, the answer inference subtask aims to locate the

position of the answer A_{pos} which corresponds to the position of A_{gold} in S_k:

$$\begin{aligned} A_{pos} &= \text{Grouped}(A_{gold}), \\ A_{gold} &\in \text{Node}(G_k), \\ A_{pos} &\in \text{Node}(S_k). \end{aligned} \tag{5}$$

Once the position of the answer A_{pos} is selected, the corresponding group of entities in G_k will be regarded as the final answers.

4 READS Discriminative Reasoning

After we propose the framework of READS, we are able to design efficient reasoning strategy to facilitate graph retrieval, graph pruning and answer inference. Based on the subtasks, we are able to explicitly model the KGQA process as illustrated in Figure 2. We design discriminative strategies to achieve the subtasks and construct training data to augment the LLM-based reasoning process.

4.1 Searching Strategy

Compared to previous methods that rely on agent toolboxes, READS opts for a discriminative searching approach which only observes and updates the subgraph structure S_k.

In each iteration, based on the retrieved subgraph structure S_k, the LLM selects one option from the option pool (as shown in Figure 2). Each option includes a starting node s_{next} in S_k and a neighboring relation p_{next}, forming the next triple $t_{next} = (s_{next}, p_{next}, o_{next})$. A new node o_{next} is added to S_k along with its corresponding entities in G_k retrieved from the knowledge graph. READS maintains an option pool that includes all feasible triples for search. We can formulate each step of the discriminative searching strategy in READS as:

$$t_{next} = argmax(\mathbb{P}(t|S_k, Q), t \in pool), \tag{6}$$

where $\mathbb{P}$ represents the probability distribution over options provided by the LLM using a constrained beam search algorithm based on output logits. An additional option, 'None', is always available to terminate the search process.

Retention of Node Information. READS further enriches the information contained within the subgraph structure S_k by labeling the semantic nodes with entity types. Entities in Freebase can be classified into one of the following types: entity, topic, date, and num (details are provided in Appendix A). READS will recognize the type of retrieved entities, and add new nodes in S_k.

4.2 Pruning Strategy

After obtaining the question-related subgraph G_k along with its structure S_k, READS maps all constraints mentioned in the question onto S_k to perform subgraph pruning. A triplet C_n = (c_{pos}, c_{opt}, c_{tar}) has to be chosen from the option pool, where c_{tar} is the constraint entity mentioned in the question, c_{pos} is the target position for applying the constraint, and c_{opt} is the operator to define the type of logical resolution used for applying the constraint. READS restrict the operator c_{opt} to one of the seven types $\{=, <, \leq, >, \geq, \text{min}, \text{max}\}$ and combine each constraint with all possible operators and positions to form the option pool.

Based on the question Q and S_k, READS asks the LLM to iteratively select the constraints until the LLM selects 'None' or there are no options left:

$$C_n = argmax(\mathbb{P}(C_n|C_1..., C_{n-1}, S_k, Q)). \tag{7}$$

The pruning process is conducted at the level of subtrees rooted from the starting node in G_k (as shown in Figure 2), retaining only the subtrees that meet the constraints.

4.3 Answering Strategy

When answering questions with a large number of answers, previous generative methods often fail to capture all the correct answers, even if the reasoning steps are successfully generated. To address this problem, READS focuses on locating the positions of answers within the subgraph structure S_k to simultaneously retrieve all possible answers. Based on S_k, READS determines the answer position A_{pos} using:

$$A_{pos} = argmax(\mathbb{P}(n|S_k, C, Q), n \in S_k), \tag{8}$$

where $\mathbb{P}$ is also given by the LLM based on S_k,C, and Q. Positions for applying constraints can not be chosen again. All entities in G_k corresponding to the position A_{pos} will be listed as the answer.

5 Experiments

5.1 Datasets and Settings

Data preprocessing. Based on the proposed subtasks, we construct training data based on the original training sets. We get 121,023 subgraph searching samples and 46,885 subgraph pruning and answer inference samples. For more details of our data preprocessing and training data construction method, please refer to Appendix B.

Method	WebQSP			CWQ		
	Hits@1	Recall	F1	Hits@1	Recall	F1
Llama2-7b zero-shot (Touvron et al., 2023)*	0.403	-	0.293	0.297	-	0.272
Llama3-8b zero-shot (Dubey et al., 2024)*	0.303	-	0.257	0.305	-	0.278
Qwen2.5-7b zero-shot (Yang et al., 2024)*	0.284	-	0.237	0.259	-	0.241
GPT-4-turbo zero-shot (Achiam et al., 2023)*	0.632	-	-	0.483	-	-
Llama2-7b SPARQL Generation*	0.747	-	-	0.656	-	-
KV-Mem (Miller et al., 2016)	0.467	-	0.345	0.184	-	0.157
GraftNet (Sun et al., 2018)	0.664	-	0.604	0.368	-	0.327
QGG (Lan and Jiang, 2020)	0.730	-	0.738	0.369	-	0.374
NSM (He et al., 2021)	0.687	-	0.628	0.476	-	0.424
SR+NSM+E2E (Zhang et al., 2022)	0.695	-	0.641	0.493	-	0.463
DECAF (DPR+FiD-3B) (Yu et al., 2022)	0.821	-	0.788	-	-	-
UniKGQA (Jiang et al., 2022)	0.772	-	0.722	0.512	-	0.490
PANGU (Gu et al., 2023)	0.796	-	-	0.622	-	-
KD-CoT (Wang et al., 2023a)	0.686	-	0.525	0.557	-	-
ToG w/GPT-4 (Sun et al., 2024a)	0.826	-	-	0.676	-	-
KG-Agent (Jiang et al., 2024)	0.833	-	0.810	0.722	-	0.692
RoG (Top-3 relation path) (LUO et al., 2024)*	0.795	0.756	0.701	0.567	0.573	0.547
READS (Ours)	**0.840**	**0.860**	**0.845**	**0.802**	**0.837**	**0.820**

Table 1: The results of our method compared with previous approaches on WebQSP and CWQ. Asterisk (*) denotes the results we reproduced. Note that the Hits@1 result reported in the original RoG paper (WebQSP 0.857, CWQ 0.626) is not calculated in the right way, see the author's response here.

Benchmarks. To evaluate the knowledge graph question-answering capability of the proposed method, we choose two widely used benchmarks, WebQSP (Yih et al., 2016) and CWQ (Talmor and Berant, 2018). These two benchmarks are constructed based on Freebase knowledge graph.

Metrics. We choose commonly used metrics Hits@1 and F1 for the evaluation process following previous works (LUO et al., 2024; Sun et al., 2024a). For detailed definition and implementation of the metrics, please refer to Appendix C.

Baselines We use previous reproducible SOTA generation-based KGQA method RoG as our baseline. RoG make full use of LLM planning and chain-of-thought reasoning capability to achieve remarkable KGQA performance (LUO et al., 2024). We also listed typical methods like ToG and KGAgent with interactive reasoning strategy (Sun et al., 2024a; Jiang et al., 2024), PANGU with single-task discriminative strategy (Gu et al., 2023). The zero-shot performance of widely used LLMs is listed for comparison. We also finetuned llama2-7b to directly generate SPARQL queries for each of the question, and then execute those queries on Freebase to get the answer.

Base Model. We choose Llama2-7b as the base model of READS following RoG. For implementation with GPT-4, see Section 5.7.

5.2 Main results

The performance of READS on WebQSP and CWQ is presented in Table 1. According to the results, our porposed READS framework achieves state-of-the-art performance on these two benchmarks, with improvements in both Hits@1 and F1, indicating an enhanced capability of the LLM to handle KGQA tasks. Besides, the READS method abandons the use of internal model knowledge, yet still achieves better KGQA performance, which sufficiently demonstrates that the proposed framework can effectively enhance the knowledge reasoning capabilities of LLMs (refer to Appendix D). We also test READS on more challenging dataset GrailQA (Gu et al., 2021), the results are shown in Appendix E.

5.3 Searching Capability Analysis

To validate READS's enhancement on the capability of LLM to retrieve question-related subgraphs, we design two metrics, relation recall and minimum graph edit distance, to measure the difference between the retrieved subgraph G_k and the golden subgraph G_{gold} extract from the SPARQL query given by the benchmarks.

Relation recall measures the proportion of golden relations edges that are successfully predicted, which reflects the method's sensitivity to retrieve the most relevant relations towards the

Leaf Number	2					3			4			5			Avg.
Total Hop	**1**	**2**	**3**	**4**	**5**	**3**	**4**	**5**	**4**	**5**	**6**	**5**	**6**	**7**	
Relation Recall R_{rel}															
RoG	0.853	0.644	0.381	0.280	0.254	0.429	0.266	0.270	0.286	0.179	0.169	0.186	0.266	0.283	0.339
READS	**0.887**	**0.887**	**0.903**	**0.897**	**0.972**	**0.859**	**0.853**	**0.899**	**0.748**	**0.656**	**0.826**	**0.867**	**0.837**	**0.863**	**0.853**
Minimum Graph Edit Distance $D(G_k, G_{gold})$															
RoG	0.479	2.494	3.929	5.462	7.727	3.071	3.746	5.394	1.760	4.780	5.681	5.441	8.100	10.438	4.893
READS	**0.097**	**0.209**	**0.315**	**0.625**	**0.181**	**0.338**	**1.069**	**1.490**	**1.521**	**3.658**	**3.000**	**1.235**	**1.550**	**1.578**	**1.204**

Table 2: We use relation recall and minimum graph edit distance as the metrics to measure the quality of retrieved subgraphs with different type of structures.

given question from the knowledge graph:

$$R_{rel} = \frac{\text{count}(\{R|R \in G_k\} \cap \{R|R \in G_{gold}\})}{\text{count}(\{R|R \in G_{gold}\})}. \quad (9)$$

Minimum edit distance $D(G_1, G_2)$ is defined as the total number of operations required to transform one graph G_1 into another graph G_2 by sequentially adjusting its edges one by one:

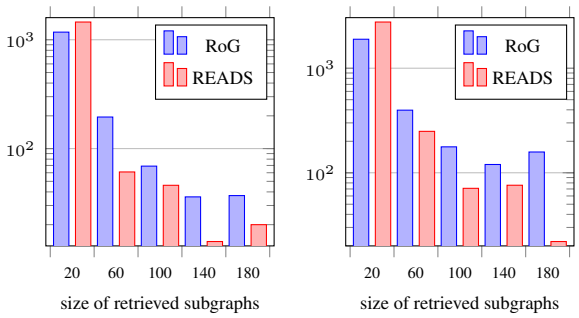
$$D(G_1, G_2) = \min_n(\text{Edit}^n(G_1) == G_2). \quad (10)$$

The lower the distance $D(G_k, G_{gold})$ is, the smaller the structural difference between the predicted graph structure and the correct reasoning subgraph is. We combine the WebQSP and CWQ datasets and classify the test set based on the structure of the given golden subgraph with two features: number of leaf nodes and the total number of relations. The detailed statistic result can be found in Appendix G. We compare READS with finetuned generative method RoG to evaluate the method's performance to retrieve different types of subgraphs, the results are shown in Table 2.

Across all types of subgraph structures, it is evident that our method consistently achieves higher relation recall and lower average edit distance, which demonstrates significant enhancement of the LLM's capability to search for question-related subgraphs with our proposed searching strategy.

5.4 Pruning-Answering Capability Analysis

Following the analysis of searching capability, we move on to evaluate the pruning and answering capability of our proposed strategies. We calculate the average size of the retrieved subgraphs. As shown in Figure 3, there is a significant reduction in the average size of the retrieved subgraphs, indicating that the READS method effectively improves the efficiency of subgraph retrieval by recalling fewer but higher-quality subgraph triples.



(a) WebQSP (b) CWQ

Figure 3: The number of cases with the size (number of triplets) of retrieved subgraphs.

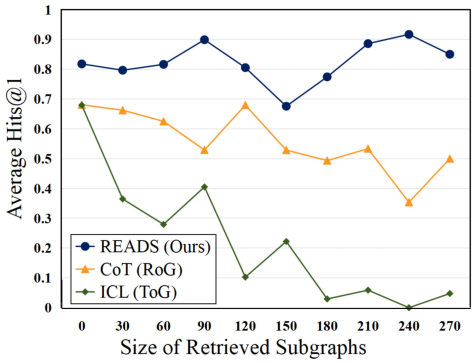


Figure 4: The trend of average Hits@1 as the size (number of triplets) of retrieved subgraph increases.

We analyze the impact of the size of the retrieved subgraph (i.e., the number of triples included) on the overall performance of the strategy (the result is shown in Figure 4). In addition to using RoG with finetuned chain-of-thought reasoning, we implement the in-context reasoning strategy proposed by ToG with the subgraphs retrieved by READS. To ensure fairness, we use Llama2_7b as the base model for all experiments. As shown in Figure 4, as the number of recalled subgraph triples increases, the performance of generative reasoning methods declines, whereas the strategy adopted

by READS remains more stable. Redundant subgraph significantly increase the context length, thereby affecting the performance of the generative reasoning process. This observation suggests that generation-based reasoning strategies are more sensitive to the size of the subgraphs compared to the strategy employed by READS.

5.5 Subtask Ablation Study

To validate the effectiveness of the reformulation approach adopted in READS, we ablate the strategies in READS one at a time and observe the changes in performance. The implementations are: 1) answer: Generate the answer based on the subgraph G_k rather than determining position on S_k; 2) pruning: Skip the pruning process and rely on answer generation process to filter answers; 3) searching: Directly generate subgraph paths based on the question using the strategy in RoG. 4) entity type: Erase the entity type on S_k, an extra ablation implementation to evaluate the effectiveness of entity information retention for LLM reasoning.

Model	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
READS	0.840	0.845	0.802	0.820
- answer	0.761	0.744	0.684	0.679
- pruning	0.737	0.764	0.548	0.632
- searching	0.739	0.803	0.444	0.581
- entity type	0.764	0.776	0.741	0.770

Table 3: Ablation study of the strategies in READS.

The results are shown in Table 3. Firstly, all three tasks experienced a performance decline when employing strategies similar to previous work, demonstrating the effectiveness of the task framework proposed by READS. Secondly, the subgraph search task showed the greatest performance difference before and after ablation, indicating that the model's subgraph search capability is the most critical within the current framework. Lastly, entity type information has been proven to effectively assist large models in conducting more precise reasoning processes.

5.6 Error Type Analysis

To analyze the effect of adopting READS on ungrounded reasoning with hallucinatory behavior, we collected and examined the frequency of error cases in READS. Since we decompose the KGQA process into three subtasks executed sequentially, we can categorize all errors into the following three types: 1) E1 stands for abscense of answer in the retrieved subgraph, corresponding to searching subtask; 2) E2 stands for lack of filering of the answer set, corresponding to pruning subtask; 3) E3 stands for mistakenly chosen the position of answer, corresponding to answering subtask (refer to Appendix F for more details). We analyzed the frequency of these different error types, and the results are shown in Figure 5.

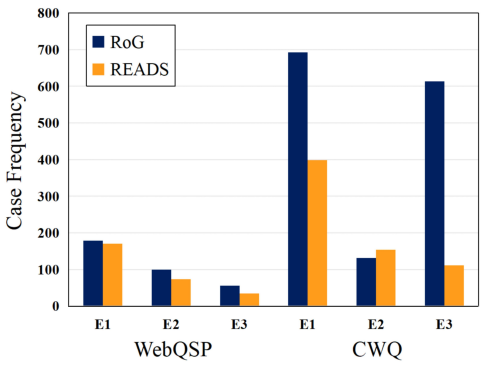


Figure 5: Case frequency of different types of errors, E1 corresponds to searching subtask; E2 corresponds to pruning subtask; E3 corresponds to answering subtask.

According to the result, compared to the generation-based method, READS significantly reduces the frequency of E3 errors on both benchmarks and also reduces the overall frequency of E2 errors, which proves that READS alleviates ungrounded reasoning behaviors in most cases by applying discriminative reasoning strategies. The dropped case frequency of E1 is consistent with our claim that the READS enhances subgraph searching capability of the LLM.

5.7 Hallucination vs. Internal Knowledge

As mentioned earlier, to mitigate hallucinations, READS abandons the capability of LLMs to generate answers directly, focusing instead on discriminative subtasks. To further analyze the balance between hallucination and LLM internal knowledge, we conduct experiments with a strong model GPT-4 and our base model llama2-7b using different answering strategies.

We employed three strategies: 1) Zero-shot: directly listing answers based on the question; 2) READS: using our proposed framework; 3) Augmented: generating answers based on subgraphs extracted by READS. Since we could not fine-tune or constrain the generation process of GPT-4, we presented it with a pool of options and asked it to make a selection.

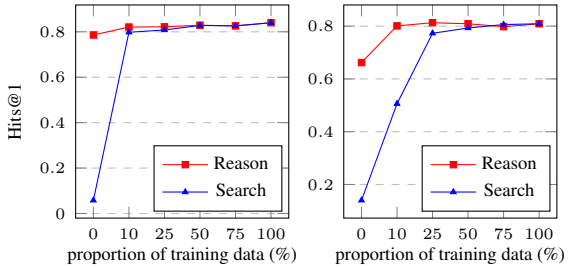
Strategy	WebQSP		CWQ	
	GPT-4	Llama2	GPT-4	Llama2
Zero-shot	**0.632**	0.403	**0.483**	0.297
READS	0.544	**0.840**	0.346	**0.802**
Augmented	**0.856**	0.791	**0.792**	0.632

Table 4: The Hits@1 under different strategies.

As shown in Table 4, compared to the READS process, GPT-4 achieves better results than Llama2 in generating answers based on subgraphs. We believe this illustrates the differences in strategic adaptability between scaled models and 7b-size models. For the commonly used 7b-size models, applying a constrained generation framework may better enhance their ability to inference answers.

5.8 Subtask Data Efficiency

We examine the training efficiency of the READS subtasks. We combine subgraph pruning and answer inference in the figure as reasoning component and fine-tune two separate models from scratch using Llama2-7b. When evaluating the performance of one model, we use the other model in its fully fine-tuned form.



(a) WebQSP (b) CWQ

Figure 6: The Hits@1 performance using different proportion of finetuning data.

As shown in Figure 6, both models require only about 25% of the training data to reach near the best performances. In terms of data requirements among subgraph retrieval, subgraph pruning and answer inference, subgraph retrieval demands more data and poses greater challenges to the LLM.

5.9 Further Analysis

Model Universality of READS. To analyze the model Universality of READS, we test the performance of the READS method based on different backbone models, and the results are shown in Table 5. The results indicate that changing the base model has no significant impact on the method's performance, highlighting the universality of the READS approach.

Base Model	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
Vicuna-7b	0.809	0.828	0.778	0.794
Llama-7b	0.830	0.842	0.799	0.823
Llama2-7b	0.840	0.845	0.802	0.820
Llama3-8b	0.827	0.845	0.812	0.831
Qwen2.5-7b	0.825	0.840	0.809	0.821

Table 5: Model universality of READS.

Reasoning Cost. The interactive analysis between LLMs and knowledge graphs can be quite time-consuming, particularly when large subgraphs introduce long contexts that further hinder reasoning efficiency. However, through the implementation of a highly efficient reasoning strategy, READS has significantly reduced both the average number of model calls per question and the number of tokens per request. As demonstrated in Table 6, READS has halved the cost and achieved a similar average number of model calls as RoG, which plans a subgraph through a single generation.

Method	WebQSP			CWQ		
	input	output	calls	input	output	calls
RoG	343.3	47.4	4.0	490.1	42.9	4.0
ToG	-	-	11.2	-	-	14.3
READS	178.9	10.8	3.9	206.6	12.4	5.7

Table 6: The average model calls per question and average number of input/output tokens per request.

Case Study. We present cases of solving KGQA problems using the READS method in Appendix I. READS provides effective explicit intermediate reasoning information, which adds to the readability of the overall KGQA process.

6 Conclusion

In this paper, we propose a novel LLM-based reasoning framework READS to reformulate KGQA process, aiming to alleviate the hallucination issues in existing generative methods and enhance the LLM's reasoning capability. Experimental results proves our claim that by decomposing KGQA and adopting designed discriminative strategies, we can enhances the capability of LLMs to retrieve question-related subgraphs and mitigate ungrounded reasoning results caused by hallucinations in the generation process.

Limitations

Though our proposed READS framework has shown competitive KGQA performance and is proven to enhance the LLM's reasoning capability, we identify several limitations that requires further improvement. In the future, we will focus on the following directions to extend the current work:

1) Entity linking: Existing methods assume that the entity linking process is done before the KGQA process (LUO et al., 2024; Sun et al., 2024a); In this work we follow the previous works to assume that the entity linking has already been completed. This is a common issue faced by the KGQA methods, we will explore how to eliminate this assumption to achieve reliable KGQA process.

2) Demand on labeled data: Although our method effectively enhances the knowledge reasoning capabilities of large models and demonstrates competitive performance across multiple datasets, we assume the existence of a gold query. Given the strong zero-shot KGQA capability and reasoning capability of GPT-4, works that does not rely on a gold query either requires GPT-4 to annotate the reasoning process (such as RoG) or combines the knowledge memory of strong models to improve overall performance (such as ToG, GoG, etc.) (LUO et al., 2024; Sun et al., 2024a; Xu et al., 2024). In the future works, we will explore the possibility of using model-generated pseudo-labels or constructing self-summarized memories to deal with this issue.

References

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

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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Artur d'Avila Garcez, Tarek R Besold, Luc De Raedt, Peter Földiak, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luis C Lamb, Risto Miikkulainen, and Daniel L Silver. 2015. Neural-symbolic learning and reasoning: contributions and challenges. In *2015 AAAI Spring Symposium Series*.

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4928–4949.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

Yu Gu and Yu Su. 2022. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731, Gyeongju, Republic of Korea. International Committee on 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. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21. ACM.

Ruixin Hong, Hongming Zhang, Hong Zhao, Dong Yu, and Changshui Zhang. 2023. Faithful question answering with monte-carlo planning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3944–3965.

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.

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

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

LINHAO LUO, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In

The Twelfth International Conference on Learning Representations.

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

Philipp Mondorf and Barbara Plank. 2024. Beyond accuracy: Evaluating the reasoning behavior of large language models–a survey. *arXiv preprint arXiv:2404.01869*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge & Data Engineering*, (01):1–20.

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

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

Kai Sun, Yifan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024b. Head-to-tail: How knowledgeable are large language models (llms)? aka will llms replace knowledge graphs? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 311–325.

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.

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

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

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023b. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634.

Haoyi Xiong, Zhiyuan Wang, Xuhong Li, Jiang Bian, Zeke Xie, Shahid Mumtaz, Anwer Al-Dulaimi, and Laura E Barnes. 2024. Converging paradigms: The synergy of symbolic and connectionist ai in llm-empowered autonomous agents. *arXiv preprint arXiv:2407.08516*.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Kang Liu, and Jun Zhao. 2024. Generate-on-graph: Treat llm as both agent and kg in incomplete knowledge graph question answering. *arXiv preprint arXiv:2404.14741*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

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

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

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and

logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

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, Dublin, Ireland. Association for Computational Linguistics.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

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

Type	Definition	Example
entity	Real entities include person\school\events and so on	Micheal
topic	Topic id entities which is used to connect entities with the same topic, its id has no actual meanings	m.01428y
num	Numbers	240.15
date	Dates	2015\08\10

Table 7: Entity types with its definition and example

A Semantic Entity Types

Here we demonstrate different semantic entity types in Table 7.

B Data Preprocessing

Training Data for Subgraph Searching. We make use of the SPARQL data available in existing benchmarks to form the training data. In WebQSP and CWQ, each question is associated with a SPARQL query. The direct execution of this query yields the answer to the open question. We obtain the correct subgraph structure required to solve each problem by decomposing the SPARQL statements. Unlike ROG (LUO et al., 2024), in finetuning process READS always presents the model with the correct knowledge subgraph structure rather than the shortest path starts from the question entity and ends at the answer entities.

Training Data for Subgraph pruning and answer inference. To finetune the LLMs to be capable of constraint determination and answer inference, we also construct constraint/answer locating samples from the SPARQL queries in WebQSP and CWQ. The input is a complete subgraph structure with all feasible options of constraints or answer positions, the golden output is the correct position of the constraint and the answer.

Freebase preprocessing. Due to the huge volume of established Freebase knowledge graph, directly interacting with Freebase through SPARQL is inefficient and may result in unnecessary syntax errors. Following UniKGQA (Jiang et al., 2022), we extract subgraphs from Freebase using breadth-first search for each question, which are then used for the subgraph searching process. Additionally, we expand these subgraphs using the SPARQL

queries provided in the benchmarks to ensure the presence of constraint branches.

SPARQL queries contain the subgraph information necessary to complete a comprehensive graph query. These queries are composed of graph structure triples and filtering conditions. In WebQSP, CWQ, and most KGQA datasets built on Freebase, each question corresponds to a specific SPARQL query. Therefore, the paths included in SPARQL effectively represent the correct subgraph structure required to answer the current question.

In previous works, subgraphs obtained using shortest path search methods typically formed chain-like structures. Compared to the information contained in SPARQL, these structures: 1) might not be logically coherent search paths, and 2) could miss some branches on certain nodes along the path. To enable our method to proceed smoothly, we extracted additional subgraph structures with all possible branches related to the question from Freebase based on the structural information inherent in the SPARQL queries. These were added to the original dataset (for a reference to the original dataset, see RoG). The specific implementation can be found in the corresponding functions in the open-source code, and will not be elaborated here.

C Metrics

Here we outline the metrics calculation formulas and their corresponding meanings that were not detailed in the main text.

Hits@1. Hits@1 calculates the proportion of questions for which the first answer given by the model is correct. Given A_{pre} is the predicted list of answers, and A_{gold} is the list of golden answers, $A_{pre}[0]$ as the very first answer the model predict, then we have:

$$Hits@1 = \frac{count(A_{pre}[0] \in A_{gold})}{count(questions)}. \quad (11)$$

For example, if the correct answer is "apple" and the model answers "pear, apple, banana," then Hits@1 for this question is 0. It is important to note that this metric can sometimes be miscalculated as follows:

$$Hits@1 = \frac{count(A_{pre} \cap A_{gold} \neq \emptyset)}{count(questions)}. \quad (12)$$

With this incorrect calculation, the Hits@1 would be higher. For the above example, the Hits@1 for this question would be 1.

Leaf Node Number	2	3	4	5
with threshold	0.847	0.741	0.625	0.604
w/o threshold	0.845	0.750	0.636	0.738

Table 8: The average Hits@1 performance on questions with different subgraph structures. Manually add minimum branch threshold during tree search process. The performance drops as we manually add the threshold.

Method	GrailQA Dev			
	i.i.d	compositional	zero-shot	overall
PANGU	0.844	0.746	0.716	0.754
READS	0.921	0.759	0.626	0.718

Table 9: The Hits@1 performance on GrailQA.

F1. We adopt the same calculation method as previous work, using the Macro-F1 scoring method. First, we calculate the precision and recall for each test sample. Then, we average them based on the number of samples to obtain the overall recall and precision. Finally, we use the harmonic mean of the overall recall and precision to calculate the overall F1 score.

D Ungrounded Reasoning Behavior

The previous generation-based method can sometimes provide the correct answer even when the subgraph does not contain the correct answer, whereas READS does not exhibit this behavior (see case C2 in Table 11).

E Result on GrailQA

We test our proposed READ on more challenging benchmark GrailQA's development set (Gu et al., 2021), the results are shown in Table 9. Compared to the previous single task discriminative method PANGU (Gu et al., 2023), although not achieving overall SOTA performance, READS enhanced the KGQA reliability on both i.i.d and compositional questions, which proves the effectiveness of the reformulation strategy used in READS.

F Types of Errors

We categorize the answering process into five scenarios, with three of these ultimately resulting in incorrect answers. We present the overall definitions in Table 10 and the frequency statistics for the five scenarios in Table 10. Here we further explain the definition of the three error cases.

Does retrieved subgraph contains correct answer?	Is the very first answer predicted correct?	Case type
Yes	Yes	C1
	No, but the correct answer exist in the predicted list	E2
	No, and there is no correct answer in the predicted list	E3
No	Yes	C2
	No	E1

Table 10: Error Case type definitions.

Case Type	CWQ		WebQSP	
RoG	**Total**	**Seperate**	**Total**	**Seperate**
C1	2390	1645	1363	1208
E2		132		99
E3		613		56
C2	1057	364	257	78
E1		693		179
READS				
C1	3049	2784	1449	1342
E2		154		73
E3		111		34
C2	398	0	171	0
E1		398		171

Table 11: Frequency count of different cases.

Leaf Count	Edge count in the subgraph								Total	Percentage
	1	2	3	4	5	6	7	8		
2	921	1453	1267	400	22	0	0	0	4063	0.802
3	0	0	278	217	208	5	0	0	708	0.139
4	0	0	0	71	41	44	2	3	161	0.031
5	0	0	0	0	34	40	57	0	131	0.025

Table 12: Statistics of questions with different knowledge subgraph structure. This is the statistic result combining WebQSP and CWQ's test set.

Tree Search Stage Prompt Template

Below is a wikipedia question,
you can retrieve a graph to help you answer the question.
The retrieved graph information is given as information triple like (Entity1, Relation, Entity2),
or only the name of the start entity.
Decide which entity and corresponding relation to retrieve next,
response in form of 'entity+relation'.
Response 'None' if the retrieved graph is
informative enough to answer the question.
Question:
<The Question>
Retrieved graph: <The retrieved subgraph structure S_k, in forms of triples>
Next retrieve:

Tree Pruning Stage Prompt Template

Locate Constraints:
Below is a question with a support graph presented as
triples (entity A, relation, entity B).
The entity name in the support graph is 'type_id'.
'Type' denotes the entity type, which includes four types:
ordinary entity (entity), topic entity (topic),
number (num), and date (date).
'Id' is an incremental identifier used to distinguish entities.
Please match all the constrains with one of the entity in the support graph.
Support graph:
<The retrieved subgraph structure S_k, in forms of triples>
Question:
<The Question>
Constraints:
<List of constraint entities>
Determine result:

Locate Answer:
Below is a question with a support graph presented as
triples (entity A, relation, entity B).
The entity name in the support graph is 'type_id'.
'Type' denotes the entity type, which includes four types:
ordinary entity (entity), topic entity (topic),
number (num), and date (date).
'Id' is an incremental identifier used to distinguish entities.
Please select the answer from the support graph by choosing the right entity.
Support graph:
<The retrieved subgraph structure S_k, in forms of triples>
Question:
<The Question>
Answer entity:

Table 13: Prompt Template use in READS

E1 (failed subgraph searching) is directly related to the graph search ability of the model. If the answer is not included in the retrieved subgraph, the model can not actually obtain the answer through inference and pruning.

We attribute E2 to lack of subgraph pruning as it indicates the presence of incorrect answer entities at the selected answer position. We detect E2 as the cases when the first answer is wrong all the correct answers are listed after the The lack of pruning may be caused by: 1) Omission of branches in the structure, which means the LLM fails to retrieve necessary entities; 2) Failure on matching constraints with the correct position. To avoid such errors, the model should have stronger searching and constraint locating capabilities.

We attribute E3 to wrong answer location since the answer list contains no golden answer. Although generation-based methods generates the answer rather than selecting the position, the inability to infer the answer from the graph containing the correct answer is considered as a similar location error. To avoid such errors, the model should have stronger subgraph reasoning and answer positioning capabilities.

G Statistics of Subgraph Structure

After categorizing questions based on the structure of their corresponding knowledge subgraphs, we count the number of questions in each class(see Figure 12), and find that there is a relative scarcity of graph-structured data with single or multiple branches.

Many questions with leaf count 2 is free from constraints, while these issues make up the vast majority(80.2%) of the test set. This proportional relationship also appears in the training set, which means the model will see more simple graph structures during training process. This may lead the model to prematurely halt the search by favoring structures with fewer branches. However, introducing minimum branching threshold to force the LLM to search more branches before it terminates the search stage may obstacle normal tree search behavior (see Table 8). This remains a topic worth to be discussed in the future.

H Prompt Templates

We demonstrate all the prompt templates used in READS in Table 13, including the template for tree searching, locating constraints and the answer.

I Case Study

We present two clear process examples of conducting KGQA tasks using READS in Table 14.

Case 1

Question:
what does jamaican people speak?
Tree search stage output S_k:
('Jamaica', 'location.country.languages_spoken', 'entity_1')
Real subgraph G_k retrieve from Freebase:
('Jamaica', 'location.country.languages_spoken', 'Jamaican English')
('Jamaica', 'location.country.languages_spoken', 'Jamaican Creole English Language')
Tree pruning stage output:
No constrain, Answer is "entity_1"
READS Output:
'Jamaican English', 'Jamaican Creole English Language'

Case 2

Question:
who plays the voice of kitt in knight rider?
Tree search stage output S_k:
("Knight Rider", "tv.tv_program.regular_cast", "topic_1")
("topic_1", "tv.regular_tv_appearance.character", "entity_2")
("topic_1", "tv.regular_tv_appearance.actor", "entity_3")
Real subgraph G_k retrieve from Freebase:
("Knight Rider", "tv.tv_program.regular_cast", "m.03lj4m5")
("m.03lj4m5", "tv.regular_tv_appearance.character", "KITT")
("m.03lj4m5", "tv.regular_tv_appearance.actor", "William Daniels")
...
("m.02h9cb0", "tv.regular_tv_appearance.actor", "David Hasselhoff")
("m.03lj4m0", "tv.regular_tv_appearance.actor", "Edward Mulhare")
("m.03lj4lk", "tv.regular_tv_appearance.actor", "Patricia McPherson")
Tree pruning stage output:
Constraint: "entity_2= KITT", Answer is "entity_3"
READS Output:
'William Daniels'

Case 3

Question:
where did kevin love go to college?
Tree search stage output S_k:
("Kevin Love", "people.person.education", "topic_1")
("topic_1", "education.education.institution", "entity_2")
("entity_2", "common.topic.notable_types", "entity_3")
Real subgraph G_k retrieve from Freebase:
("Kevin Love", "people.person.education", "m.04ftwdc")
("Kevin Love", "people.person.education", "m.045z4gx")
("m.04ftwdc", "education.education.institution", "University of California, Los Angeles")
("m.045z4gx", "education.education.institution", "Lake Oswego High School")
("University of California, Los Angeles", "common.topic.notable_types", "College/University")
("Lake Oswego High School", "common.topic.notable_types", "School")
Tree pruning stage output:
Constraint: "entity_3 = College/University", Answer is "entity_2"
READS Output:
'University of California, Los Angeles'

Table 14: Case Study of READS