Self-Reflective Planning with Knowledge Graphs: Enhancing LLM Reasoning Reliability for Question Answering

Anonymous EMNLP submission

Abstract

Recently, large language models (LLMs) have demonstrated remarkable capabilities in natural language processing tasks, yet they remain prone to hallucinations when reasoning with insufficient internal knowledge. While integrating LLMs with knowledge graphs (KGs) provides access to structured, verifiable information, existing approaches often generate incomplete or factually inconsistent reasoning paths. To this end, we propose Self-Reflective Planning (SRP), a framework that synergizes LLMs with KGs through iterative, reference-guided reasoning. Specifically, given a question and topic entities, SRP first searches for references to guide planning and reflection. In the planning process, it checks initial relations and generates a reasoning path. After retrieving knowledge from KGs through a reasoning path, it implements iterative reflection by judging the retrieval result and editing the reasoning path until the answer is correctly retrieved. Extensive experiments on three public datasets demonstrate that SRP surpasses various strong baselines and further underscore its reliable reasoning ability.

1 Introduction

Recent breakthroughs in large language models (LLMs) have significantly advanced natural language processing (Brown et al., 2020; Ouyang et al., 2022; Bubeck et al., 2023). Nonetheless, LLMs largely depend on knowledge gained through pretraining, and consequently, they tend to struggle with hallucination when solving questions that extend beyond their scope of knowledge.

To tackle these problems, researchers attempt to integrate LLMs with knowledge graphs (KGs), capitalizing on the explicit structure and semantic knowledge to reinforce factual grounding (Hogan et al., 2021). Prior research has explored two categories of approaches: (1) fine-tuning methods that equip models with the ability to utilize information in KGs (Shi et al., 2021; Zhang et al., 2022);

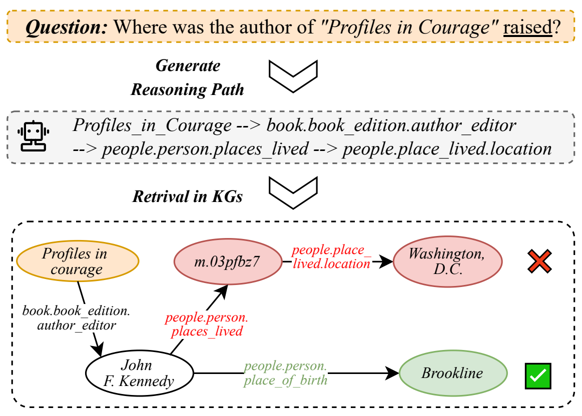

Question: Where was the author of "Profiles in Courage" raised?
Generate Reasoning Path
Profiles_in_Courage --> book.book_edition.author_editor --> people.person.places_lived --> people.place_lived.location
Retrival in KGs
Profiles in courage
m.03pfbz7
people.place_lived.location
Washington, D.C.
book.book_edition.author_editor
people.person.places_lived
John F. Kennedy
people.person.place_of_birth
Brookline

Figure 1: An illustration of the question answering over knowledge graphs.

and (2) guiding LLMs with prompts to make plans and decisions in order to search for the knowledge in KGs for answering questions (Li et al., 2023b; Cheng et al., 2024).

Despite their effectiveness, existing approaches still face notable challenges. On the one hand, fine-tuning-based models require high-quality, large-scale data for fine-tuning. Such requirements not only limit their adaptability to new domains but also demand substantial computational resources. On the other hand, prompt-based approaches primarily focus on planning effective retrieval, while falling short in reflecting the retrieved information. For example, in Figure 1, given the question "*Where was the author of 'Profiles in Courage' raised*", the model generates a reasoning path to search for relevant knowledge in the KG. The result obtained from the reasoning path is "*Washington, D.C.*", which is incorrect as the relation "*people.place_lived.location*" in the reasoning path points to where the person lived, but the question asks about where the person "was raised". While the model retrieves a relevant result, it lacks reflection on whether the reasoning path truly aligns with the intent of the question. As a consequence, it overlooks the correct relation "*people.person .place_of_birth*" and correct entity "*Brookline*".

To address these shortcomings, we propose Self-Reflective Planning (SRP), a KG-enhanced LLM framework with reliable planning and reflection ability. Specifically, SRP first designs a Reference Searching module, which searches for relevant references as evidence. Subsequently, a Path Planning module is designed to check initial relations linked to topic entities and generate the reasoning path. After that, we develop a Knowledge Retrieval module to instantiate the reasoning path with the KG. More importantly, to achieve reliable reflection, a Reflection and Reasoning module is employed to iteratively judge retrieval results and edit the reasoning path until the answer is retrieved.

In summary, the main contributions of our work could be summarized as follows.

- We explore methods for reliable planning and reflection with LLMs to generate accurate reasoning path, offering a new perspective for building more reliable question answering algorithms.
- We propose a Self-Reflective Planning (SRP) framework, which can generate the effective reasoning path and refine reasoning errors with reliable reflection.
- Extensive evaluations on WebQSP, CWQ, and GrailQA datasets demonstrate SRP's superior performance compared to competitive baselines, underscoring the reliability and accuracy of SRP. The code is available on https://anonymous.4open.science/r/SRP-E06C.

2 Related Work

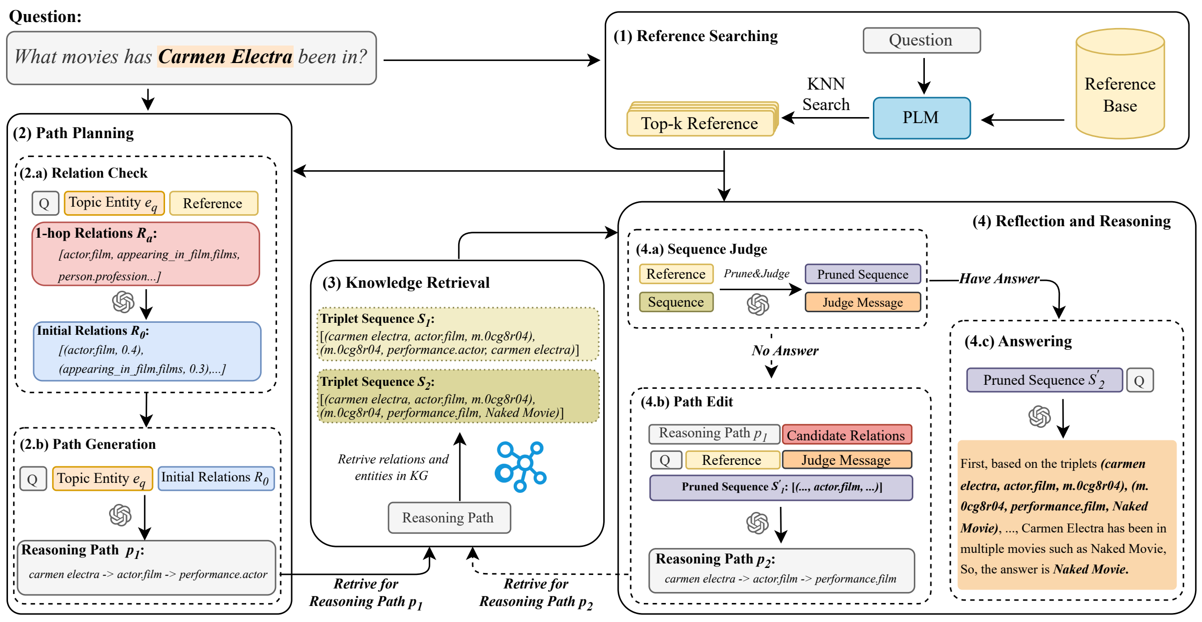
LLM Reasoning. To promote reasoning ability of LLMs, numerous researchers have directed LLMs to incorporate their thought processes into their outputs rather than merely delivering direct answers (Wei et al., 2022; Hao et al., 2024; Kojima et al., 2022). Initially, Chain of Thought (CoT) (Wei et al., 2022) framework was developed to present several examples of intermediate reasoning steps in natural language as prompts. Subsequently, various adaptations of CoT reasoning, such as Self-Consistency (Wang et al., 2022), Tree-of-Thought (Yao et al., 2023), Graph-of-Thought (Besta et al., 2024), were introduced to motivate the reasoning ability of LLMs. To facilitate LLMs in achieving a cognitive process that parallels human thinking, many studies (Madaan et al., 2023; Wang et al., 2024; Guan et al., 2024; Shinn et al., 2023) have devised self-correction mechanisms using feedback to amend faulty reasoning and ensure precision. Our work is also inspired by self-correction and proposes a self-reflection mechanism to better plan and correct the reasoning path for searching effective knowledge in the KG.

KG-enhanced LLM. LLMs are pre-trained on extensive datasets, but they still encounter issues such as outdated information, hallucinations, and unclear decision-making processes (Zhang et al., 2024a, 2023, 2024b). A promising solution to these problems is utilizing KGs to provide explicit and modifiable knowledge to LLMs. Earlier research incorporated KGs during the pre-training (Wang et al., 2021) or fine-tuning (Luo et al., 2023) phases, but these methods demand substantial computational resources and perform poorly when dealing with problem that were not encountered during the training or fine-tuning period. Consequently, some approaches (Baek et al., 2023; Yang et al., 2024; Cheng et al., 2024) first extracted information from KGs and then directly supplied explicit knowledge to LLMs. These methods eliminate the requirement for pre-training or fine-tuning the model and achieve satisfactory performance. However, this method still falls short in accurately extracting question-relevant knowledge from the KG, which reduces the reliability of model. To solve this problem, our work can improve reliability of reasoning path through reference searching, checking the first relation and self-reflection, which is effective for retrieve relevant knowledge in KGs.

3 Problem Definition

Combining KGs with LLMs to answer questions belongs to knowledge graph question answering (KGQA) task, which is about resolving natural language question with KGs. KG is a set of triples, i.e., $\{(e, r, e')|e, e' \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ are the sets of entities and relations. Given a question q, a set of topic entities $\mathcal{T}_q$ extracted within q, and a KG $\mathcal{G}$, the objective of KGQA is to infer valid answers $\mathcal{A}_q$. To achieve this, LLM needs to plan reliable reasoning paths, retrieve the triplet sequences based on reasoning paths, and answer questions with triplet sequence. The reasoning path and triplet sequence are defined as below.

Reasoning Path. Reasoning Path is sequences starting from entity and followed by one or more



Question:
What movies has Carmen Electra been in?
(1) Reference Searching
Question
KNN Search
PLM
Reference Base
Top-k Reference
(2) Path Planning
(2.a) Relation Check
Q
Topic Entity e_q
Reference
1-hop Relations R_a:
[actor.film, appearing_in_film.films, person.profession...]
Initial Relations R_0:
[(actor.film, 0.4), (appearing_in_film.films, 0.3),...]
(2.b) Path Generation
Q
Topic Entity e_q
Initial Relations R_0
Reasoning Path p_1:
carmen electra -> actor.film -> performance.actor
(3) Knowledge Retrieval
Triplet Sequence S_1:
[(carmen electra, actor.film, m.0cg8r04), (m.0cg8r04, performance.actor, carmen electra)]
Triplet Sequence S_2:
[(carmen electra, actor.film, m.0cg8r04), (m.0cg8r04, performance.film, Naked Movie)]
Retrive relations and entities in KG
Reasoning Path
Retrive for Reasoning Path p_1
Retrive for Reasoning Path p_2
(4) Reflection and Reasoning
(4.a) Sequence Judge
Reference
Sequence
Prune&Judge
Pruned Sequence
Judge Message
Have Answer
No Answer
(4.b) Path Edit
Reasoning Path p_1
Candidate Relations
Q
Reference
Judge Message
Pruned Sequence S'_1: [(..., actor.film, ...)]
Reasoning Path p_2:
carmen electra -> actor.film -> performance.film
(4.c) Answering
Pruned Sequence S'_2
Q
First, based on the triplets (carmen electra, actor.film, m.0cg8r04), (m.0cg8r04, performance.film, Naked Movie), ..., Carmen Electra has been in multiple movies such as Naked Movie, So, the answer is Naked Movie.

Figure 2: The framework of our SRP method. It consists of four main parts: (1) Reference Searching, (2) Path Planning, (3) Knowledge Retrieval and (4) Reflection and Reasoning.

relations, e.g., $p = e_0 \rightarrow \hat{r}_0 \rightarrow \hat{r}_1 \rightarrow \cdots \rightarrow \hat{r}_l$, where each $\hat{r}_i$ is a predicted relation generated by LLM and l is the length of reasoning path.

Triplet Sequence. Triplet Sequence is a sequence of triplets retrieved from KGs. Given reasoning path p, the triplet sequence is $S = \{(e_d,\ r_d,\ e_{d+1})\}_{d=0}^{D}$, where $e_d, e_{d+1} \in \mathcal{E}$ and $r_d \in \mathcal{R}$. Noted that each r_d is corresponding to a predicted relation $\hat{r}_d$ in p.

4 Methodology

4.1 Overview

As shown in Figure 2, our SRP includes four modules: 1) Reference Searching module that searches for reference, 2) Path Planning module that generates reasoning path, 3) Knowledge Retrieval module that retrieves triplet sequence and 4) Reflection and Reasoning module that performs judgement, editing and answering.

Given a question, module 1) first searches for references to guide module 2) and 4). Then, module 2) generates initial reasoning path. Module 3) bridges Module 2) and Module 4), achieving iterative retrieval and refinement of reasoning paths. The prompts of SRP are illustrated in Appendix C.

4.2 Reference Searching

To guide the reasoning process, module retrieves relevant cases from reference base, which contains massive referable cases about solving question with KGs. A case includes a question, reasoning paths that can be utilized to retrieve answers in KGs and the correct answers of the question, which is denoted as reference. We extract references from training set. To construct the reference base, each question in training set is encoded into dense vectors with pretrained language model and then clustered (e.g., via K-Means). We randomly select the same number of references from each cluster and store them in the reference base.

During process of planning and self-reflection, the model encodes the input question and compares it against the embeddings of questions from reference base using similarity-based search (e.g., K-Nearest Neighbors) to retrieve the Top-k most similar corresponding references. These references are integrated into downstream modules to enrich the reasoning process with relevant historical contexts, as illustrated in Figure 2, thereby motivating reasoning ability of LLM to enhance accuracy and reliability of planning and self-reflection.

4.3 Path Planning

The Path Planning module is designed to generate reliable initial reasoning path, which includes two parts: relation check and path generation.

Relation Check. Before path generation, it is necessary to check each 1-hop relation of topic entity e_q, which means relation directly connected to

e_q. This is because the LLM lacks context knowledge of e_q in KGs. Checking the 1-hop relations connected to e_q before generating the reasoning path p can help LLMs predict reliable p and align p with the actual structure of the KG, thereby enhancing the reliability of reasoning path. It identifies the most relevant 1-hop relations connecting e_q to other entities within the KG. The 1-hop relations of e_q are denoted as $R_a = \{r^a_0, r^a_1, r^a_2 \ldots\}$. Guided by the information from the references obtained in reference searching, the LLM assesses each relation in R_a based on its relevance to the given question q. The LLM scores the R_a, and the top-K relations with the highest scores are selected as the initial relations $R_0 = \{r^0_0, r^0_1, \ldots, r^0_k\}$ to participate in the path generation.

For demonstration, consider the example in Figure 2, the LLM checks 1-hop relations of e_q. The LLM scores 1-hop relations of *carmen electra*, such as *actor.film*, *appearing_in_film.film*, *person.profession* and so on. The relevance scores assigned by the LLM are used to select relations including *actor.film* and *appearing_in_film.film* as the top-K candidate relations, with indicative scores of 0.4 and 0.3, respectively. These selected R_0 serve as candidates for the first relation starting from e_q in reasoning path, thereby reducing uncertainty in the subsequent path generation.

Path Generation. The path generation process generates the complete reasoning path p starting from the topic entity e_q with LLM, leveraging the initial relations R_0 obtained through relation check. Continuing the example of Figure 2, the LLM predicts p starting from the topic entity *carmen electra*. Taking inputs that include the question q, R_0 (e.g., *actor.film*), and e_q, the LLM generates the reasoning path "*carmen electra* $\rightarrow$ *actor.film* $\rightarrow$ *performance.actor*", denoted as the reasoning path p_1. This predicted reasoning path serves for further retrieval in KGs and might be refined one or more times to improve reliability of the reasoning path.

4.4 Knowledge Retrieval

The Knowledge Retrieval module aims to instantiate the reasoning path $p = e_0 \rightarrow \hat{r}_0 \rightarrow \hat{r}_1 \rightarrow \cdots \rightarrow \hat{r}_l$ by searching the corresponding triplet sequence $S = \{(e_d,\ r_d,\ e_{d+1})\}_{d=0}^{D}$, where D is the length of s, and $D \leq l$. Each r_d in S has a corresponding predicted relations $\hat{r}_d$ in p.

Following method in Readi (Cheng et al., 2024), we conduct retrieval by comparing semantic similarity. The retrieval is starting from e_0, which is also the topic entity e_q of the question. To retrieve relation corresponding to $\hat{r}_0$, we first utilize BM25 and Contriever (Izacard et al., 2022) to obtain relations semantically similar to $\hat{r}_0$, denoted as $R^s_0 = \{r^s_{0,\,0},\ r^s_{0,\,1},\ \ldots\}$. Then, we examine the 1-hop relations connected to e_0. If there is one or more relations among e_0 that exists in R^s_0, then the corresponding relation r_0 and entity e_1 in KGs will be retrieved. The retrieval process will be iterated until each predicted relation in p finds corresponding relation in KGs or a predicted relation in p fails to find corresponding relation in KGs.

For the instance in Figure 2, the reasoning path p_1 from Path Planning is successfully retrieved, whose corresponding triplet sequence is *[(carmen electra, actor.film, m.0cg8r04), (m.0cg8r04, performance.actor, carmen electra)]*, denoted as triplet sequence S_1. The edited path generated from path edit (reasoning path p_2) is also retrieved as *[(carmen electra, actor.film, m.0cg8r04), (m.0cg8r04, performance.film, Naked Movie)]*, denoted as triplet sequence S_2.

4.5 Reflection and Reasoning

The Reflection and Reasoning module enables LLM to evaluate retrieved triplets, then choose to refine the reasoning path for next retrieval in KGs or answer question with retrieved information based on evaluation result. It consists of three parts: sequence judge, path edit and answering.

Sequence Judge. The sequence judge aims to determine whether the current reasoning path p needs to be edited, and retain the subsequence related to the question in the triplet sequence S to eliminate irrelevant triplets. This process is designed to provide reliable information for answering questions and path editing.

Under the guidance of references, LLM assesses whether the answer of question or relevant information is in triplets of S and outputs judgement result and its thinking process. If the judgement result is "have answer", p will not be edited and self-reflection will be stopped, then SRP will answer the question. If the judgement result is "no answer", p will be edited in path edit part for next retrieval. LLM will also generate pruned sequence $S' = \{(e'_d,\ r'_d,\ e'_{d+1})\}_{d=0}^{D'}$, where $D' \leq D$. It is the sub-sequence of S which relevant to answering question. By evaluating S', we can identify that there might be predicted relations in p that need to be corrected. These relations may not be retrievable in the KG or may not be helpful for answering the question. The evaluation result of S' and thinking process of LLM are concatenated as judge message for path edit process.

For the example in Figure 2, the triplet sequence S_1 is judged to have no answer, and the LLM considers *(m.0cg8r04, performance.actor, carmen electra)* is meaningless for answering question, so LLM generated pruned sequence S'_1 *[(carmen electra, actor.film, m.0cg8r04)]* and corresponding judge message. For triplet sequence S_2, LLM judges that answer is in the sequence, so the self-reflection is stopped and the model will answer question with pruned sequence S'_2 which is same as triplet sequence S_2.

Path Edit. Path edit refines the reasoning path p to better align with the question q and reasoning goals, triggered when the triplet sequence S lacks sufficient information. Path edit requires the 1-hop relations linked to the tail entity of the pruned sequence S' as the candidate relations for the edit process, which is obtained from retrieval module. As shown in Figure 2, with information from references, judge message and S', LLM edits the reasoning path p_1 with candidate relations. It modifies the relation *performance.actor* to *performance.film* and generates edited reasoning path "*carmen electra → actor.film → performance.film*", denoted as reasoning path p_2.

Answering. If the judgement result is "have answer", LLM will reason for answering question based on pruned sequence S' from sequence judge part. Using Chain-of-Thought (CoT) reasoning prompts, the LLM generates the final answer of question based on S' . This step involves analyzing the triplets in S' to deduce the answer. For the instance in Figure 2, based on triplets *(carmen electra, actor.film, m.0cg8r04)*, *(m.0cg8r04, performance.film, Naked Movie)*, LLM can give conclusion that "*the answer is Naked Movie*".

5 Experiments

5.1 Experiment Setup

Datasets and Evaluation Metrics. To evaluate the reasoning ability of SRP, we conduct experiments on three multi-hop KGQA datasets called WebQuestionsSP (WebQSP) (Yih et al., 2016), ComplexWebQuestions (CWQ) (Talmor and Berant, 2018) and GrailQA (Gu et al., 2021). More statistics are illustrated in Table 1. Besides, all of the three datasets utilize Freebase (Bollacker et al., 2008) as the background KG. To ensure computational efficiency in processing the large-scale GrailQA dataset, our experimental framework adopts the same test samples established in ToG (Sun et al., 2023). You can refer to Appendix A for more details.

Datasets	#Train	#Test	#Max hop
WebQSP	3,098	1,628	2
CWQ	27,639	3,531	4
GrailQA	44,337	1,000	4

Table 1: Statistics of three datasets.

Consistent with established methodologies in previous work (Cheng et al., 2024; Li et al., 2023a; Jiang et al., 2023), we employ exact match accuracy (Hits@1) as the primary evaluation metric for model performance assessment.

Implementation Details. In Reference Searching and Knowledge Retrieval module, we adopt the all-MiniLM-L6-v2 based on sentence-transformers (Reimers and Gurevych, 2019) as the representation model. We extract 100 questions and corresponding references from each dataset to construct reference base. The number of retrieved reference for each test question is set to be k = 4.

We adopt gpt-3.5-turbo (OpenAI, 2022) and gpt-4.1-mini (OpenAI, 2025) as the LLMs to conduct experiments, denoted as SRP-GPT3.5 and SRP-GPT4.1-mini. Temperature is 0.3 for all modules. Following previous research (Cheng et al., 2024), we utilize a Pyserini as a hybrid searcher with BM25 and Contriver (Izacard et al., 2022). For each relation in reasoning path, we retrieve top-5 similar relations on Freebase. We deploy Virtuoso server to search for knowledge from Freebase.

Benchmark Methods. In order to verify the performance of our SRP model, we compare SRP with the state-of-the-art KGQA methods: 1) Fine-Tuned model methods, including TransferNet (Shi et al., 2021), TIARA (Shu et al., 2022), SR+NSM+E2E (Zhang et al., 2022) and Flexkbqa (Li et al., 2024); 2) LLM-only methods, including GPT3.5 (OpenAI, 2022) and GPT4.1-mini (OpenAI, 2025); 3) Prompting LLM methods, including KB-BINDER (Li et al., 2023b), StructGPT (Jiang et al., 2023), ToG (Sun et al., 2023) and Readi (Cheng et al.,

Method	**WebQSP**	**CWQ**	**GrailQA**			
			overall	**I.I.D.**	**Compositional**	**Zero-shot**
Fine-tuned Method						
TransferNet*	71.4	48.6	-	-	-	-
TIARA*	75.2	-	73.0	**87.8**	**69.2**	68.0
SR+NSM+E2E*	69.5	49.3	-	-	-	-
Flexkbqa*	60.6	-	62.8	71.3	59.1	60.6
LLM-only Method						
GPT3.5	63.8	45.7	25.8	20.0	18.7	30.8
GPT-4.1-mini	64.2	52.4	36.5	34.2	26.3	41.1
Prompting LLM Methods						
KB-BINDER*	74.4	-	50.6	-	-	-
StructGPT*	72.6	54.3	-	-	-	-
ToG-GPT3.5*	76.2	57.1	68.7	70.1	56.1	72.7
Readi-GPT3.5	74.5	56.7	67.0	67.9	53.5	71.4
Readi-GPT4.1-mini	80.9	60.2	71.7	67.5	60.1	77.6
SRP-GPT3.5 (ours)	78.6	58.7	71.2	68.3	58.6	76.9
SRP-GPT4.1-mini (ours)	**83.6**	**69.0**	**78.8**	75.8	62.6	**85.8**

Table 2: Performance comparison with different baselines on three KGQA dataets. **Bold font** represents the optimal result, while the underline font represents the sub-optimal result. The results labeled with * are cited from their original publications.

2024). It is worth noting that, we utilize GPT3.5 and GPT4.1-mini to motivate Readi in experiment, denoted as Readi-GPT3.5 and Readi-GPT4.1-mini. Results of ToG (ToG-GPT3.5) are cited from (Chen et al., 2024). More details can be seen from Appendix B.

5.2 Experiment Result

As shown in Table 2, SRP consistently demonstrates state-of-the-art (SOTA) performance on both the WebQSP and CWQ benchmarks, while achieving the second-highest overall performance on GrailQA, demonstrating its capacity to effectively leverage an external KG for robust retrieval and multi-hop reasoning. First, by integrating a systematic retrieval module with self-reflection reasoning prompts, SRP not only detects relevant facts more accurately but also navigates complex reasoning pathways in a manner that outperforms prompting-based LLM baselines such as Readi. This superiority becomes evident on both GPT-3.5 and GPT-4.1-mini backbones, highlighting SRP's adaptability to different large language model architectures and underscoring the versatility of our retrieval-centric design. Second, although SRP does not incorporate any pre-training or fine-tuning steps, it still surpasses notable fine-tuned models, including TIARA, SR+NSM+E2E, and Flexkbqa; this observation underscores the power of a well-designed prompting approach to capture nuanced relationships within a knowledge graph, thus compensating for the absence of additional parameter updates. Third, when compared against LLM-only methods—where no external knowledge is used—SRP reveals significant gains in accuracy, indicating that the KG provides crucial information and helps avoid the hallucinations or logical gaps that can arise when relying purely on learned language representations.

Method	**WebQSP**	**CWQ**	**GraliQA**
SRP	**78.5**	**58.7**	**71.2**
w/o relation check	76.9	56.7	62.9
w/o self-reflection	76.9	55.5	68.7
w/o reference	75.7	57.6	69.8
w/ random reference	77.0	58.2	70.9

Table 3: Ablation study

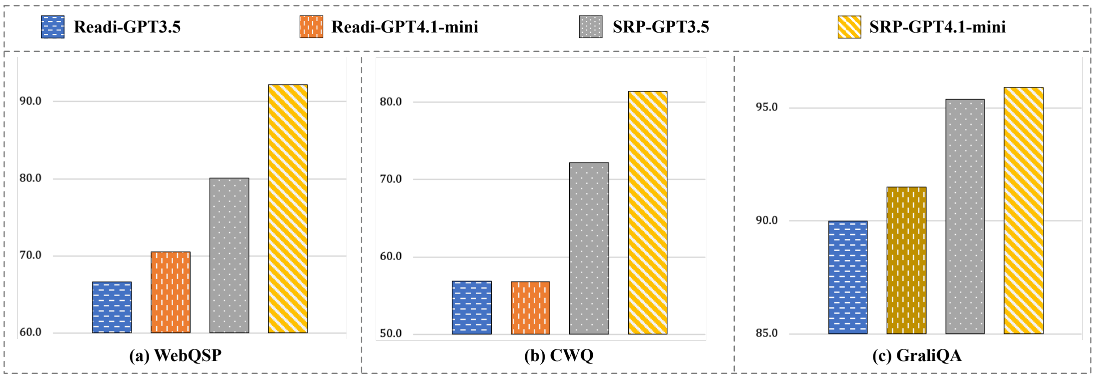


Figure 3: Reliable answering rate of Readi and SRP on theree datasets.

Methods	WebQSP	CWQ	GraliQA
Readi-GPT3.5	55.7	44.2	68.3
Readi-GPT4.1-mini	64.3	41.6	74.2
SRP-GPT3.5	69.9	60.5	80.3
SRP-GPT4.1-mini	85.3	70.3	85.0

Table 4: Searching success rate of Readi and SRP on theree datasets.

5.3 Ablation Study

Table 3 presents the ablation study findings for our proposed SRP model across three benchmark datasets. We conduct ablation studies from the following three perspectives.

First, omitting the relation check part in Path Planning (- *relation check*) significantly undermines performance of SRP, which underscores the necessity of validating relation correctness to generate reliable reasoning paths.

Second, removing the sequence judge and path edit part in Reflection and Reasoning module (-*self-reflection*) causes a notable drop on CWQ, revealing effects of reflection mechanisms for judging retrieved results and refining reasoning path.

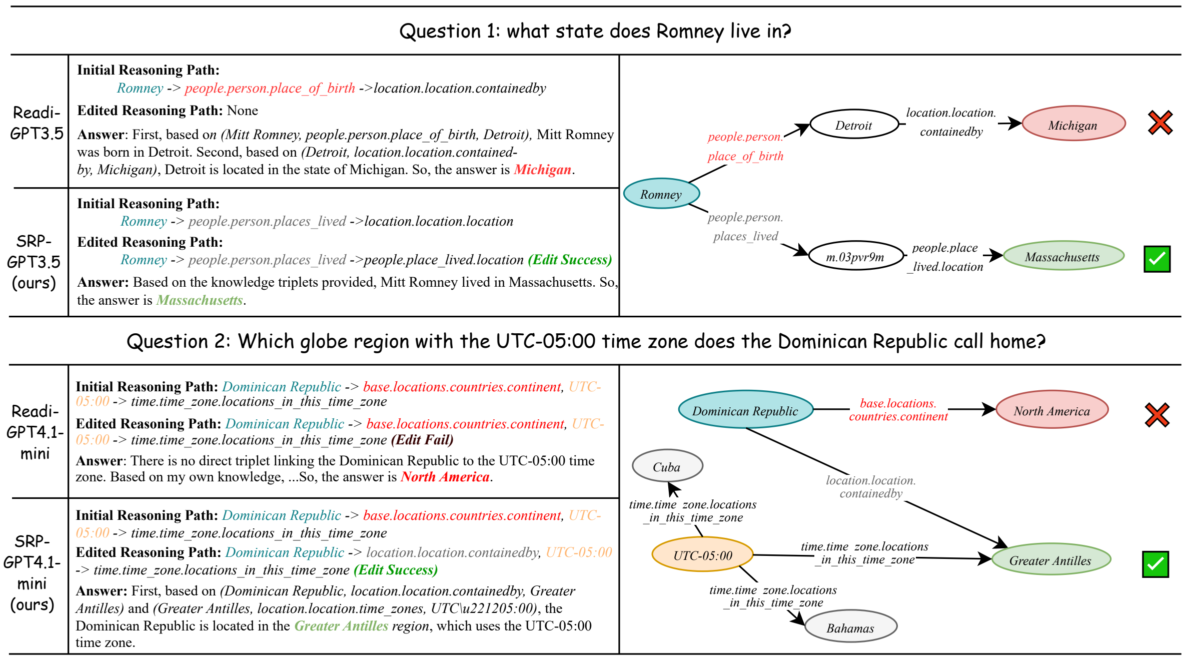
Third, eliminating the Reference Searching module (- *reference*) yields a pronounced performance drop, highlighting the importance of reference; for instance, scores on WebQSP decline from 78.5% to 75.7%, demonstrating the benefits of leveraging external information for guidance. Furthermore, replacing the searched references with random-sampled references (*random reference*) also reduces the result from 71.2% to 70.9% on GrailQA and from 78.5% to 77.0% on WebQSP, implying that carefully searched references are superior to arbitrary references.

5.4 Reliability Study

To further substantiate the reliability of our proposed SRP framework, we evaluate both the searching success rate and the reliable answering rate using the WebQSP, CWQ, and GrailQA datasets. The searching success rate denotes how efficiently each approach retrieves answer of question from the KG, while the reliable answering rate reflects defined as the proportion of correct answers supported by factual triples from the KG, thereby capturing the overall reliability of each system.

Table 4 presents the searching success rate for of Readi-GPT3.5, Readi-GPT4.1-mini, SRP-GPT3.5, and SRP-GPT4.1-mini. The data illustrate that SRP's performance of retrieval valuable knowledge from KGs substantially exceeds Readi on three datasets. Even when Readi uses GPT4.1-mini, Readi's searching success rate is still lower than SRP-GPT3.5 on WebQSP, CWQ, and GrailQA, which demonstrates that the reasoning paths generated by SRP through reliable planning and reflection are more effective in retrieving the knowledge needed to answer the question.

As illustrated in Figure 3, we analyze each model's reliable answering rate. Overall, SRP consistently outperforms Readi across three datasets. For instance, on WebQSP, SRP-GPT4.1-mini achieves a reliable answering rate of 92.2%, significantly higher than the 70.5% reached by Readi-GPT4.1-mini. The gap remains evident on CWQ and persists in GrailQA, where SRP maintains over 95% reliability. These findings indicate that SRP's correct answers are more frequently grounded in triples from KGs, reflecting strong factual support and high explanatory credibility.



Question 1: what state does Romney live in?

Readi-GPT3.5	**Initial Reasoning Path:** *Romney* -> *people.person.place_of_birth* ->*location.location.containedby* **Edited Reasoning Path:** None **Answer**: First, based on *(Mitt Romney, people.person.place_of_birth, Detroit),* Mitt Romney was born in Detroit. Second, based on *(Detroit, location.location.containedby, Michigan)*, Detroit is located in the state of Michigan. So, the answer is ***Michigan***.
SRP-GPT3.5 (ours)	**Initial Reasoning Path:** *Romney* -> *people.person.places_lived* ->*location.location.location* **Edited Reasoning Path:** *Romney* -> *people.person.places_lived* ->*people.place_lived.location* ***(Edit Success)*** **Answer:** Based on the knowledge triplets provided, Mitt Romney lived in Massachusetts. So, the answer is ***Massachusetts***.

Question 2: Which globe region with the UTC-05:00 time zone does the Dominican Republic call home?

Readi-GPT4.1-mini	**Initial Reasoning Path:** *Dominican Republic* -> *base.locations.countries.continent, UTC-05:00* -> *time.time_zone.locations_in_this_time_zone* **Edited Reasoning Path:** *Dominican Republic* -> *base.locations.countries.continent, UTC-05:00* -> *time.time_zone.locations_in_this_time_zone* ***(Edit Fail)*** **Answer**: There is no direct triplet linking the Dominican Republic to the UTC-05:00 time zone. Based on my own knowledge, ...So, the answer is ***North America***.
SRP-GPT4.1-mini (ours)	**Initial Reasoning Path:** *Dominican Republic* -> *base.locations.countries.continent, UTC-05:00* -> *time.time_zone.locations_in_this_time_zone* **Edited Reasoning Path:** *Dominican Republic* -> *location.location.containedby, UTC-05:00* -> *time.time_zone.locations_in_this_time_zone* ***(Edit Success)*** **Answer:** First, based on *(Dominican Republic, location.location.containedby, Greater Antilles)* and *(Greater Antilles, location.location.time_zones, UTC\u221205:00)*, the Dominican Republic is located in the ***Greater Antilles*** *region*, which uses the UTC-05:00 time zone.

Romney
people.person.place_of_birth
Detroit
location.location.containedby
Michigan
people.person.places_lived
m.03pvr9m
people.place_lived.location
Massachusetts
Dominican Republic
base.locations.countries.continent
North America
location.location.containedby
Greater Antilles
Cuba
time.time_zone.locations_in_this_time_zone
UTC-05:00
time.time_zone.locations_in_this_time_zone
time.time_zone.locations_in_this_time_zone
Bahamas

Figure 4: Case study of Readi and SRP. Question 1 comes from WebQSP, and question 2 comes from CWQ.

5.5 Case Study

Figure 4 illustrates two typical example, where Question 1 comes from WebQSP and Question 2 comes from CWQ. To further highlight the reliability of the SRP in generating and correcting reasoning paths, we conducted a case study on the WebQSP and CWQ datasets. As shown in Figure 4, we present the initial reasoning paths, edited reasoning paths, and the answers inferred based on retrieval results of SRP and Readi. The retrieval results are presented in the form of node graphs. GPT3.5 and GPT4.1-mini are utilized in question 1 and in question 2 respectively.

In Question 1, with reliable planning, SRP selects the correct initial relation (*people.person .place_of_birth*), thereby establishing a solid foundation for generating correct reasoning path. This deliberate approach ultimately guides SRP to identify "Michigan" as the most plausible answer. Conversely, Readi initiates its reasoning path with an inappropriate relation (*people.person.places_lived*), which leads the reasoning path of Readi to deviate from the correct answer.

In Question 2, although SRP's initial reasoning path is not flawless, by reflection on retrieved information, SRP swiftly edits the path with correct relation "*location.location.containedby*". In contrast, Readi fails to realise that relation "*base.locations.countries.continent*" was inconsistent with the problem due to lack of reflection on triplets sequence. The capacity for iterative refinement illustrates SRP's resilience and adaptability, underscoring the importance of reflection in knowledge-based QA.

These cases demonstrate the reliability of the reasoning paths generated by SRP through the reliable planning and reflection. Through these reasoning paths, SRP can obtain more fact knowledge relevant to answering questions from the KG, thereby enhancing the reliability of reasoning.

6 Conclusion

In this paper, we propose Self-Reflective Planning (SRP) framework to address the limitations of LLMs in generating reliable reasoning path due to implicit knowledge reliance. To strengthen the reliability of its reasoning, SRP searches references for guidance, enforcing systematic relation-checking to ensure initial reasoning path alignment with structures of KGs, and repeatedly refining these reasoning path through self-reflection mechanism that preserves factual consistency. Extensive experiments on three challenging QA benchmarks demonstrate SRP's superior performance, achieving state-of-the-art results. We hope our work will motivate future studies about KG-enhanced LLM question answering.

Limitations

While SRP shows promising results demonstrates that SRP can retrieve valuable knowledge with reliable reasoning path, two limitations still exists in SRP which should be resolved in future work.

First, the iterative reflection process increases computational costs compared to single-pass methods. This is because the reflection mechanism iteratively utilizes LLM to judge and edit for refining reasoning path, increasing the number of LLM calls. We hope to explore reliable and more efficient mechanisms to reduce computational costs in future research.

Second, performance of Reference Searching partially depends on the quality of the reference, which could be impacted by domain shifts. Because we use data from training set, this limitation doesn't influence the performance of SRP. Future work will explore dynamic reference adaptation and lightweight verification mechanisms to enhance the robustness of Reference Searching.

References

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

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

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

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

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

Sébastien Bubeck, Varun Chadrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4.

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. *arXiv preprint arXiv:2410.23875*.

Sitao Cheng, Ziyuan Zhuang, Yong Xu, Fangkai Yang, Chaoyun Zhang, Xiaoting Qin, Xiang Huang, Ling Chen, Qingwei Lin, Dongmei Zhang, et al. 2024. Call me when necessary: Llms can efficiently and faithfully reason over structured environments. *arXiv preprint arXiv:2403.08593*.

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.

Jian Guan, Wei Wu, Peng Xu, Hongning Wang, Minlie Huang, et al. 2024. Amor: A recipe for building adaptable modular knowledge agents through process feedback. *Advances in Neural Information Processing Systems*, 37:126118–126148.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

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

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023a. Few-shot in-context learning for knowledge base question answering. *arXiv preprint arXiv:2305.01750*.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023b. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980, Toronto, Canada. Association for Computational Linguistics.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 18608–18616.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2025. Introducing gpt-4.1 in the api.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. TransferNet: An effective and transparent framework for multi-hop question answering over relation graph. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4149–4158, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: Multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8108–8121, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

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, New Orleans, Louisiana. Association for Computational Linguistics.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

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

Yifei Wang, Yuyang Wu, Zeming Wei, Stefanie Jegelka, and Yisen Wang. 2024. A theoretical understanding of self-correction through in-context alignment. *arXiv preprint arXiv:2405.18634*.

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

Linyao Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. 2024. Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3091–3110.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Wen-tau Yih, Matthew Richardson, Chris 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, Berlin, Germany. Association for Computational Linguistics.

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.

Weichao Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2024a. Pretraining data detection for large language models: A divergence-based calibration method. *arXiv preprint arXiv:2409.14781*.

Yi-Fan Zhang, Weichen Yu, Qingsong Wen, Xue Wang, Zhang Zhang, Liang Wang, Rong Jin, and Tieniu Tan. 2024b. Debiasing multimodal large language models. *arXiv preprint arXiv:2403.05262*.

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*.

A Datasets

We adopt three widely adopted multi-hop knowledge graph question answering (KGQA) datasets.

- WebQuestionsSP (WebQSP) (Yih et al., 2016) is a popular KGQA dataset derived from the WebQuestions dataset (Berant et al., 2013), designed for complex question answering over Freebase (Bollacker et al., 2008). It contains questions annotated with SPARQL queries, enabling research on semantic parsing and knowledge base question answering.

- Complex WebQuestions(CWQ) (Talmor and Berant, 2018) extends the original WebQSP dataset by introducing compositional questions that require multi-hop reasoning. It consists of natural language questions that are more complex and involve multiple relations or constraints.

- GrailQA (Gu et al., 2021) is a large-scale KGQA dataset built on Freebase (Bollacker et al., 2008). It evaluate models' generalization abilities across three levels: I.I.D., compositional, and zero-shot.

B Baselines

We comepare SRP with baselines grouping into 3 categories: 1) Fine-tuned model method; 2) LLM only methods; 3) Prompting LLM methods. The details of each baseline are describe as follows.

Fine-tuned model methods:

- TransferNet (Shi et al., 2021) utilizes a transparent framework to transfer knowledge across domains by aligning feature distributions for improved domain adaptation.

- TIARA (Shu et al., 2022) is a framework that leverages BERT for schema item retrieval and T5 for plan generation, employing constrained decoding to ensure grammatical correctness.

- SR+NSM+E2E (Zhang et al., 2022) trains an encoder to retrieve relevant relations and constructs reasoning paths based on the retrieved relations.

- Flexkbqa (Li et al., 2024) is a flexible KGQA framework that uses LLMs to adapt to different knowledge graphs and query languages with minimal annotated data.

LLM only methods:

- GPT3.5 (OpenAI, 2022) is an advanced AI language model developed by OpenAI, capable of understanding and generating human-like text across a wide range of tasks.

- GPT4.1-mini (OpenAI, 2025) is a streamlined version of the GPT-4.1 model, optimized for efficient natural language processing with reduced computational demands while maintaining core performance capabilities.

Prompting LLM methods:

- KB-BINDER (Li et al., 2023b) is designed to address the heterogeneity of items across different KGs, facilitating few-shot in-context learning for KGQA tasks.

- StructGPT (Jiang et al., 2023) utilizes an interface for KG data to enable finite knowledge access and filtering, leveraging a LLM to repeatedly infer answers or subsequent planning.

- ToG (Sun et al., 2023) is a training-free framework that incorporates LLMs with KGs through iterative beam search, enhancing deep reasoning capabilities, ensuring knowledge traceability and correctability, and enabling flexible, cost-effective deployment across diverse models and datasets.

- Readi (Cheng et al., 2024) is a novel framework that enables LLMs to efficiently and faithfully reason over structured environments by initially generating a reasoning path, instantiating it on the environment, and invoking targeted editing only when necessary.

Table 5 presents performance of ToG-GPT3.5, ToG-GPT4, SRP-GPT3.5 and SRP-GPT4.1-mini on three datasets, where ToG-GPT4 means ToG motivated by GPT-4. Results of ToG-GPT3.5 and ToG-GPT4 are cited from (Chen et al., 2024). Because of the lack of experiment data on ToG motivated by GPT4.1-mini and the difference of parameters size between GPT4 and GPT4.1-mini, we doesn't present results of ToG-GPT4 and compare ToG-GPT4 with SRP-GPT4.1-mini in table 2. As shown in table 5, SRP performs better than ToG when motivated by GPT-3.5. Although the performance of SRP-GPT4.1-mini is slightly inferior to that of ToG-GPT4. in GrailQA, SRP-GPT4.1-mini is better than ToG-GPT4 in WebQSP and CWQ. More importantly, the performance gap on GrailQA can be reasonably attributed to the fact that GPT-4.1-mini has significantly fewer parameters than GPT-4, which may limit its capacity in complex multi-hop reasoning tasks. Despite this, SRP-GPT4.1-mini still demonstrates strong generalization and competitive performance, indicating the robustness of our SRP framework even under lighter backbone models.

C Prompts

We provide prompts of SRP in this section. Prompts of relation check, path generation, sequence judge, path edit and answering are in Table 6, Table 7, Table 8, Table 9 and Table 10. Noted that the demonstration of prompts of relation check is from (Sun et al., 2023), and prompts of answering is following (Cheng et al., 2024).

For number of few-shot demonstration, we utilize 3 shots in WebQSP and 4 shots in CWQ and GrailQA for path generation. In path edit, we utilized 5 shots in WebQSP and CWQ and 4 shots in GrailQA. For relation check, sequence judge and answering, we utilizes 1 shots, 2 shots and 5 shots in each dataset respectively. When test Readi in GrailQA, we utilized 6 shots prompts for reasoning path generation and 4 shots demonstration for reasoning path edit.

Method	WebQSP	CWQ	GrailQA			
			overall	I.I.D.	Compositional	Zero-shot
ToG-GPT3.5	76.2	57.1	68.7	70.1	56.1	72.7
SRP-GPT3.5	**78.6**	**58.7**	**71.2**	**68.3**	**58.6**	**76.9**
ToG-GPT4	82.6	67.6	**81.4**	**79.4**	**67.3**	**86.5**
SRP-GPT4.1-mini	**83.6**	**69.7**	78.8	75.8	62.6	85.8

Table 5: Performance of ToG-GPT3.5, ToG-GPT4, SRP-GPT3.5 and SRP-GPT4.1-mini on three datasets.

Instruction
Please retrieve 3 relations (separated by semicolon) that contribute to the question and rate their contribution on a scale from 0 to 1 (the sum of the scores of 3 relations is 1). Note: (1) please refer to the 4 examples of relation paths to give your score, if some example are similar to the question and the first relation of its relation path appears in candidate relation, give this relation a good rate; (2) please output relation and score in the format of ('relation', score).

Reference
Here are 4 examples of questions and associated relation paths which connect to correct answer of question:
Question: {reference question}
Relation Path: {reference reasoning path}
......

Demonstration Example
Question: Name the president of the country whose main spoken language was Brahui in 1980?
Topic Entity: Brahui Language
Candidate Relations: language.human_language.main_country; language.human_language.language _family; language.human_language.iso_639_3_code; base.rosetta.languoid.parent; language.human _language.writing_system; base.rosetta.languoid.languoid_class; language.human_language .countries_spoken_in; kg.object_profile.prominent_type; base.rosetta.languoid.document; base .ontologies.ontology_instance.equivalent_instances; base.rosetta.languoid.local_name; language .human_language.region Answer:
1. ('language.human_language.main_country', 0.4): This relation is highly relevant as it directly relates to the country whose president is being asked for, and the main country where Brahui language is spoken in 1980.
2. ('language.human_language.countries_spoken_in', 0.3): This relation is also relevant as it provides information on the countries where Brahui language is spoken, which could help narrow down the search for the president.
3. ('base.rosetta.languoid.parent', 0.2): This relation is less relevant but still provides some context on the language family to which Brahui belongs, which could be useful in understanding the linguistic and cultural background of the country in question.

Table 6: Prompts of relation check.

Instruction

You are tasked with generating relation paths to help searching for answers in Freebase based on given question. I will provide you with:
1. A question.
2. One or more topic entity that is central to the question.
3. A set of valuable relations associated with the topic entity.

Your goal is to generate relation paths that start with the topic entity and follow a sequence of relations to help answer the question.

Demonstration Example

Question: who played princess leia in star wars movies?
Topic Entity: princess leia
Valuable Relations: {“princess leia”: [’film.film_character.portrayed_in_films’, ’tv.tv_character .appeared_in_tv_program’, ’film.film_character.movie’, ’movie.movie_character.movie’]}
Thought: Firstly, the path should cover the movies portrying princess leia. Secondly, the path should cover the actors in that movie.
Path: {
```
  “princess leia”:[
    “princess leia -> film.film_character.portrayed_in_films -> film.performance.actor”,
    “princess leia -> movie.movie_character.movie -> film.actor.actor”
  ]
}
```

Table 7: Prompts of path generation.

Instruction

Here are some triplet sequences [(h_0, r_0, t_0), ..., (h_n, r_n, t_n)] that may contain information helpful for solving the problem. Please analyze the following triplet sequences and retain the subsequences within each triplet sequence that are useful for answering the question, while removing the subsequences that are not helpful. Please first output your Thinking Process, then output the retained parts of each triplet sequence. If you believe the answer to the question appears at the end of the triplet sequence (i.e., the answer is the tail entity t_n of the last triplet), directly return this sequence. If you think that the entire triplet sequence, except for the head entity h_0 of the first triplet, is unrelated to the question, return an empty list [].

Note: (1) The retained part of the triplet sequence should be a continuous subsequence, and the removed part should also be a continuous subsequence; you cannot return non-continuous triples from the original sequence. (2) If it is possible to retain, the retained part should include at least the first triplet of the sequence. (3) The format of the output triplet sequence should be the same as the input triplet sequence. (4) If you believe the answer to the question appears in the triplet sequences, please give "<HAVE_ANSWER>" in the end of your Thinking Process. If you do not believe the answer to the question appears in the triplet sequences, please give "<NO_ANSWER>" in the end of your Thinking Process.

Reference

Here are 4 examples of some questions, associated relation and answer of question.
Question: {reference question}
Relation Path: {reference reasoning path}
Answer: {reference answer}
......

Demonstration Example

Question: where is aviano air force base located?
Triplet sequences:
1. [("Aviano Air Base", "location.location.containedby", "Italy")]
2. [("Aviano Air Base", "aviation.airport.serves", "Aviano")]

Thinking Process: First, based on the triplet ("Aviano Air Base", "location.location.containedby", "Italy"), I can answer the question. So, I think these triplet sequences have enough information to answer the question. <HAVE_ANSWER>
Retained sequences:
1. [("Aviano Air Base", "location.location.containedby", "Italy")]
2. []

Table 8: Prompts of sequence judge.

Instruction
Task: Given an Inital Path and some feedback information of a Question, please correct the Inital Path.
Note:
(1)When you receive Error Message, please edit the path based on Instantiate Paths. For example, if the Error Message is “relation XXX not instantiated”, you should modify this relation with candidate relation; if the Error Message is “<cvt></cvt> in the end”, you should add a candidate relation to a Instantiate Path which you think is relevant to question; if the Error Message is “Current Information is not enough”, please analysis Instantiate Paths and Candidate Relations, then generate a new path which is more relevant to question; (2) please refer to the 4 examples of relation paths to correct the Inital Path; (3) Avoid generating Final Path that are the same as the Initial Path.

Reference
Question: {reference question}
Relation Path: {reference reasoning path}
......

Demonstration Example
Question: What major religion in the UK has a place of worship named St. Mary’s Cathedral, Batticaloa?
Initial Path: United Kingdom -> location.location.religions -> place.religion.major_religions
»» Error Message
1. <cvt></cvt> in the end.
2. relation “place.religion.major_religions” not instantiated.
»» Instantiation Context
Instantiate Paths: United Kingdom -> location.location.contains -> Heaton railway station
United Kingdom -> location.statistical_region.religions -> <cvt></cvt>
United Kingdom -> location.location.contains -> Bakersfield, Nottingham
United Kingdom -> location.location.contains -> Knockloughrim
United Kingdom -> location.location.contains -> Oakenshaw
Candidate Relations: {’United Kingdom -> location.statistical_region.religions’: [’location.religion _percentage.date’, ’location.religion_percentage.percentage’, ’location.religion_percentage .religion’], ’United Kingdom -> location.location.contains’: [’location.location.containedby’, ’location.location.geolocation’, ’type.object.type’]}
»» Corrected Path
Goal: The Initial Path starts from United Kingdom, which should cover the major religion in United Kingdom.
Thought: In Instantiate Paths, I find that United Kingdom has some religions, described by a cvt node. In candidates, I find “location.religion_percentage.religion” most relevant to major religions.
Final Path: United Kingdom -> location.statistical_region.religions -> location.religion _percentage.religion

Table 9: Prompts of path edit.

Instruction
Given a question and the associated retrieved knowledge graph triplets (entity, relation, entity), you are asked to answer the question with these triplets. When you answer the question, please first give your answer with your own knowledge, then give your answer with knowledge from retrieved knowledge graph triplets. If the given knowledge triples is not enough or missing, you can use your own knowledge. Use { } to enclose the answer! Please think step by step.

Demostration Example Q: Find the person who said “Taste cannot be controlled by law”, where did this person die? Knowledge Triplets: (Taste cannot be controlled by law., media_common.quotation.author, Thomas Jefferson) A: First, based on (Taste cannot be controlled by law., media_common.quotation.author, Thomas Jefferson), the person who said “Taste cannot be controlled by law” is Thomas Jefferson. Second, no Triplet provided can answer where Thomas Jefferson’s dead, however, based on my owned knowledge, Thomas Jefferson died in Charlottesville. So, the answer is Charlottesville .

Table 10: Prompts of answering.