Enhancing Large Language Models with Pseudo- and Multisource-Knowledge Graphs for Open-ended Question Answering

Anonymous ACL submission

Abstract

Mitigating the hallucinations of Large Language Models (LLMs) and enhancing them is a crucial task. Although some existing methods employ model self-enhancement techniques, they fall short of effectively addressing unknown factual hallucinations. Using Knowledge Graph (KG) enhancement approaches fails to address the generalization across different KG sources and the enhancement of open-ended answer questions simultaneously. To tackle these limitations, there is a framework that combines **Pseudo-Graph Generation** and **Atomic Knowledge Verification** proposed. The enhancement of LLM using KG in an open-ended question-answering setting is implemented by leveraging the Pseudo-Graph Generation. Atomic Knowledge Verification utilizes atomic-level knowledge querying and verification to achieve generalizability under different KG sources. Compared to the baseline, this approach yields a minimum improvement of 11.5 in the ROUGE-L score for open-ended questions. For precise questions, we observe a minimum accuracy improvement of 7.5. Moreover, there is also demonstration that this framework exhibits generalizability across different KG sources. In summary, our results pave the way for enhancing LLMs by incorporating Pseudo- and Multisource-KGs, particularly in the context of open-ended questions.

1 Introduction

Large language models (LLMs) (Brown et al., 2020; Ouyang et al., 2022; OpenAI, 2023; Touvron et al., 2023; Chowdhery et al., 2022) have achieved remarkable results in the field of question answering tasks. They obtain the capability to handle various questions through pre-training on a large scale of data with a massive amount of parameters. However, LLMs still face issues of hallucination and lack of specific domain knowledge when dealing with complex problems (Huang et al., 2023; Ye et al., 2023).

To mitigate the hallucination of models and thus improve the accuracy of model responses, various methods have been proposed.

The first is to use the model's own capabilities to address the model's hallucination about uncertain knowledge. Chain-of-Thought (CoT) prompting (Wei et al., 2022) method, by having the model generate intermediate processes in its responses, has improved the accuracy of the model's answers. The Self-Consistency (SC) (Wang et al., 2023b) method enhances the robustness of CoT by considering a synthesis of multiple models' thought processes. However, these methods cannot fundamentally solve the problem of hallucinations in LLMs because of the errors or missing knowledge in the training data of LLMs (Ye et al., 2023). Therefore, we need to introduce external knowledge to enhance the LLMs, thereby achieving mitigation of hallucinations.

The second approach is to use knowledge graphs (KGs) to enhance LLMs. Knowledge graphs, like Wikidata, Freebase and YAGO, are highly valued in LLMs tasks due to their structured knowledge, high accuracy, and timely updates of information (Pan et al., 2024). Therefore , how to extract knowledge from knowledge graphs to enhance large models is a important researched field. A straightforward approach is to prompt (Chang and Fosler-Lussier, 2023) or fine tune (SQL-PALM) (Sun et al., 2023) LLMs to generate Structured Query Language (SQL). However, the schema of different knowledge graphs may vary, limiting the generalization ability of this method. To address the generalization issue across different knowledge graphs, one approach is to encode the knowledge graph semantically and enhance it through retrieval-based methods (Lewis et al., 2020). However, for questions where the relationship is not explicitly stated or for open-ended answer questions, the effectiveness of semantic retrieval may be limited. For example, it would be difficult to find the en-

methods	abilities					
	No training	No linking	Knowledge enhanced	Multi graph	Robustness	Open-ended QA
CoT	✔	✔	✘	✘	✘	✔
RAG	✔	✔	✔	✘	✔	✘
SQL-PALM	✘	✔	✔	✘	✘	✘
ToG	✔	✘	✔	✔	✘	✘
KGR	✔	✘	✔	✘	✔	✘
Ours	✔	✔	✔	✔	✔	✔

Table 1: Abilities of some representative methodologies. 1) No training means that the method requires no training; 2) No linking indicates that this method does not require entity linking within KGs; 3) Knowledge enhanced indicates that this method uses external knowledge to enhance LLMs; 4) Multi graph means the method exhibits good generalization across various KG sources; 5) Robustness refers to the property that current errors have minimal impact on subsequent steps; 6) Open-ended question indicates that this method can enhance LLMs in questions with open-ended answers.

tity "Leonardo da Vinci" in Wikidata solely based on the question "Who is the most famous painter in the world". For other methods like KGR (Guan et al., 2023) and ToG (Anonymous, 2024), although they have achieved good results, there are some limitations. ToG leaks the QID of entities in the KG during the reasoning process, and KGR does not explicitly indicate entity linking. These limitations can affect the generalization ability of these models in practical applications. In summary, the previous KG-enhanced LLM methods cannot simultaneously address the following two problems: 1) utilizing KG to enhance open-ended question answering, and 2) generalization across different knowledge graphs.

To address the aforementioned issues, **Pseudo-Graph Generation** and **Atomic Knowledge Verification** are propose. In Pseudo-Graph Generation, the framework first use LLMs to generate pseudo-triples relevant to the question. From this, it can use LLM to clarify the knowledge needed in the open-answer questions. This approach can handle open-ended question answering because the hallucination property of LLMs could be leveraged. Even if LLMs experience hallucinations during the generation process, they can still effectively construct the framework of the required knowledge, allowing us to perform queries in open-ended answers. For Atomic Knowledge Verification, we perform semantic querying on the semantically encoded KG based on these pseudo-triples. This approach exhibits generalization across different KGs because both the querying and verification processes are based on atomic-level knowledge, independent of the KG schema. All in all, LLMs are utilized to verify the pseudo-triples based on the queried triples from different KG sources, resulting in the desired answer. Therefore, the method can alleviate factual hallucinations by enhancing LLMs with external knowledge from different KG sources.

Our contributions are listed below:

- The Pseudo-Graph Generation leverages the LLMs to generate pseudo-triples relevant to the question, allowing the framework to utilize KG as knowledge augmentation for LLMs in open-ended question answering.
- Atomic Knowledge Verification uses atomic-level knowledge. Therefore, it ensures that the framework has good generalization across different KG sources.
- We introduce an open-ended question-answering dataset in a KG-enhanced setting named Natural Questions. Our experimental results show that our method not only performs excellently on this dataset but also demonstrates strong performance on existing datasets such as QALD-10(Perevalov et al., 2022) and SimpleQuestions(Bordes et al., 2015a).

2 Related Work

Table 1 demonstrates a comparison of the capabilities of the representative methods.

2.1 Self Enhanced LLMs

Directly fine-tuning LLMs to achieve performance improvements is difficult, due to the huge computational resources consumed. Wei et al., 2022 has shown that the Chain-of-Thought (CoT) prompt method can stimulate reasoning in LLMs. They

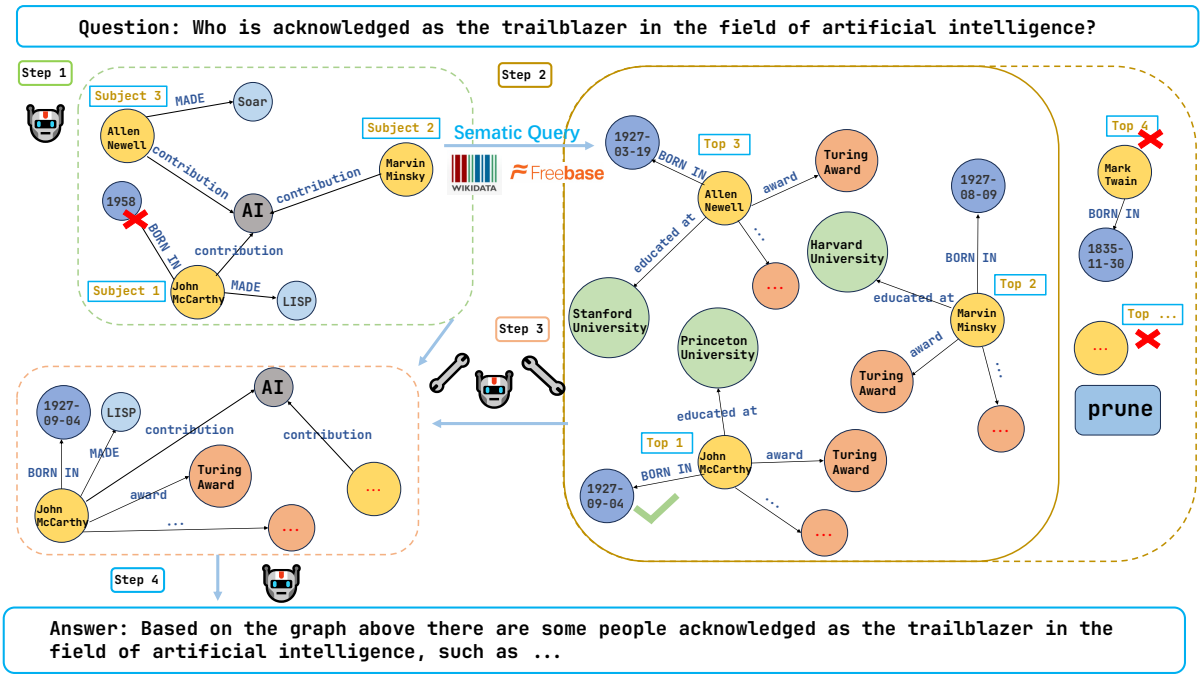

Figure 1: The over view of our method. In step 1, we prompt LLM to generate pseudo-graph G_p related to the question. For step 2, the triples generated are used to query sematic KG and get the ground graph G_g. In step 3, LLM verifies the G_p. Finally, LLM provide the answer based on the fixed G_p.

enhance LLMs by generating reasoning processes during the answer generation of answer. The introduction of this method has sparked a series of follow-up works. Zero-shot-CoT (Kojima et al., 2022) use "Let's think step by step" eliciting effective CoT reasoning. Auto-CoT (Zhang et al., 2023) automates the construction of high-quality CoTs. The Self-Consistency (Wang et al., 2023b)(SC) method considers a synthesis of multiple models' thought processes. Additionally, other methods have incorporated knowledge into CoT, such as Knowledge-driven CoT (Wang et al., 2023a) and KAM-CoT (Mondal et al., 2024)

2.2 KG Enhanced LLMs

Simply enhancing LLMs through prompts is far from sufficient. For instance, with new questions like "What kind of chips does the Apple Vision Pro use?" which involve new knowledge not covered by LLMs, enhancement through simple prompts is inadequate. Because of the structured knowledge, high accuracy, and timely updates of information (Pan et al., 2024), Enhancing LLMs with KGs is a practical method.

A straightforward approach is to prompt (Chang and Fosler-Lussier, 2023) or fine tune (SQL-PALM) (Sun et al., 2023) LLMs to generate Structured Query Language (SQL). However, for the prompt method, generating SQL without providing entity IDs to LLMs is difficult. For example, when we ask ChatGPT[1]: *Please tell me what is the QID of Yellow River in wikidata?*, it returns the output as Q1826. But Yellow River's real QID is Q2066882. For the fine-tuning approach, firstly, it requires a significant amount of computational resources. Secondly, the schema of different knowledge graphs may vary, limiting the generalization ability of this method. Embedding representations, like Trans-E (Bordes et al., 2013), of KGs is a good method to address the issue of differing schemas among various KGs. However, for multi-hop relationships and large-scale KGs, the embedding approach makes it difficult to memorize knowledge (Li et al., 2023). ToG (Anonymous, 2024) and KGR (Guan et al., 2023) are methods for augmenting models by introducing knowledge from KGs in the model inference process. ToG utilizes the model to search for relevant entities and relationships within KGs to solve complex problems. KGR identifies relevant entities existing within KGs from the answers of LLMs, thereby achieving corrections to the answers. However, these methods all exhibit ambiguity in the entity-linking step. ToG directly leaks the entity's ID, while KGR does not explicitly indicate the entity linking. This signifi-

[1]https://chat.openai.com/

cantly weakens the generalizability of these methods in practical applications. And, to the best of our knowledge, methods of enhancing LLMs based on KGs not yet have been applied to questions with open-ended answers.

3 Methodology

In this section, we would like to describe the process of the approach. The general flow of the method can be seen in Figure 1. First, we define a triple of the form $G = \{O, R, T\}$, where O denotes the set of subjects, R is the set of relations, and S is the set of objects.

3.1 Generation of Pseudo-graph

For processing of generating the pseudo-graph, initially, it is aimed to utilize LLM to directly generate fact-related triples. However, when the LLM has not been fine-tuned, enabling it to understand and generate fact-related triples is a relatively challenging task. Since LLMs are trained on large-scale natural corpora (Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023; Chowdhery et al., 2022). They are more inclined to use continuous language for responses, rather than answering with discrete triples. Relying on directly generating triples may lead to the model producing triples that do not conform to rules, for example:

$$< YangtzeRiver >< flows \quad Hubei >$$

Considering that LLMs have decent code capabilities (Yetiştiren et al., 2023), programming languages are adopted as an intermediary bridge between natural language and triples.

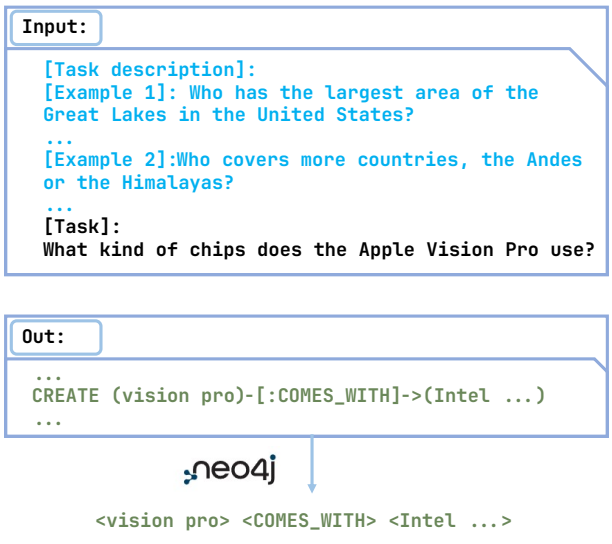

Figure 2: Generation of pseudo-graph

Specifically, the LLM is instructed with two examples in Cypher language, asking it to generate Cypher queries that could solve the problem. Then, we run the Cypher queries on Neo4j[2] and decode them into the form of triples. This ensures that the model outputs in a manner it is familiar with. Additionally, in the subsequent semantic querying process, it resembles the structure of triples in semantics. Finally, there is the pseudo-graph $G_p = \{O_p, R_p, T_p\}$ get from LLM.

3.2 Atomic Knowledge Verification

3.2.1 Sematic Querying

First, for the semantic knowledge graph, we extracted a subset of subgraphs $G_{base} = \{O_{base}, R_{base}, T_{base}\}$ from Wikidata or Freebase based on the questions. Then we use Sentence-BERT (Reimers and Gurevych, 2019) to encode the triples after parsing them into semantic form.

Following that, the cosine similarity between each triple in the G_h and that in the G_{base} is calculated. We select for each triple in G_p the top 10 in G_{base} as the result of the query to form $G_t = \{O_t, R_t, T_t\}$. Next, we can extract relevant entities and relationships. However, due to the large number of triples in G_p, the resulting relationships and entities obtained from the query can be quite extensive, sometimes even exceeding the maximum token limit of LLM.

So a pruning method to address this problem should be proposed. For the ToG (Anonymous, 2024) method, they use LLM scoring to prune relationships. However, for pruning method, if we do the same, that won't make efficient use of the knowledge graph G_p generated by LLM. Moreover, relying heavily on LLM can lead to accumulated errors. So here is proposed a two-step pruning method that eliminates the need for LLM judgment. Firstly, we utilize the size k of S_p in G_p to identify the top k entities from S_t with the highest number of triples as candidates. This step helps us to eliminate some less popular entities with the same name. For example, in Wikidata, there are 7 entities labeled as "YaoMing", but the basketball player Yao Ming is the most popular one. So for the question "Where was Yao Ming born?", it is more inclined to select the basketball player Yao Ming as the answer entity. Next, cosine similarity are utilize during semantic querying to further prune and rank entities. For each subject s belonging to S_t, it can

[2]https://neo4j.com/

calculate the average semantic score (the cosine similarity when querying with G_p) of all triples with s as the subject as the entity confidence score. At the same time, entities with scores below 0.7 are filtered out. Finally, we get the ground graph G_g.

It is worth noting that our method is significantly different from previous approaches. ToG (Anonymous, 2024) leaks the entity's ID in the KG during model decision-making. The KGR (Guan et al., 2023) method does not clearly specify how to perform entity linking. These methods have limitations in practical applications.

3.2.2 Pseudo-Graph Verification

When verifying G_h with LLM, the triples of entity s in S_g with higher scores are place closer to G_h. This approach is more advantageous for LLM to establish better attention between the G_h and G_g. Then, there are two simple examples used to enable LLM to perform self-verification of the knowledge graph. Afterwards, we obtain the final knowledge graph G_f.

Park et al., 2023 utilizes LLM to verify the KG generated by LLM. However, their method focuses on verifying the consistency of the model in solving subproblems in multi-hop questions. Moreover, they enhance LLM using retrieval-based methods using the subproblems on the text corpus, which establishes a weak coupling relationship between the knowledge graph, the question, and the model.

3.3 Answer Generation

In this step, we use two examples to teach LLM how to answer questions based on the knowledge graph. Then, the model is instructed to generate answers to questions using the examples, the question itself, and G_f.

4 Experimens

4.1 Models

In our experiments, we used GPT-3.5 and GPT-4 (OpenAI, 2023) as large language models to generate graphs, perform verification, and produce the final answers. Sentence-BERT (Plenz et al., 2023) is chosen as the encoder for the semantic KG and as the query module to query knowledge related to the triples generated by the LLM.

4.2 Datasets

To verify that the method has validity in natural question and answer, it is tested on three different types of datasets. They include single-hop questions, multi-hop questions, and open-answer quizzes: SimpleQuestions (Bordes et al., 2015a), QALD-10 (Perevalov et al., 2022), Nature Questions.

- SimpleQuestions (Bordes et al., 2015a) employs a manually annotated method to generate corresponding questions based on facts in the knowledge base, using Freebase as the source of answers.
- QALD-10 (Perevalov et al., 2022) is a multilingual, multi-hop question answering dataset that uses Wikidata as its answer knowledge base and includes multiple languages. In our experiments, English is choose for the question-and-answer tasks.
- Nature Questions is a dataset we compiled, featuring questions people commonly ask in daily life that include open-ended answers, multiple-answer responses, and queries about new knowledge. We manually constructed 50 questions for this dataset, writing three answers for each question, expecting the answer will be comprehensive enough.

Detail Due to the closure of the Freebase API, we used a subset of FB2M (Bordes et al., 2015b) as our knowledge base on Freebase. Furthermore, because the number of questions in SimpleQuestions (Bordes et al., 2015b) is too large (100k), 1000 questions are randomly selected for GPT-3.5. Due to the price factor of the GPT-4, there are 150 data chosen for the SimpleQuestion test. For both QALD-10 and Nature Questions, we use the full dataset for testing and constructing the corresponding semantic KG based on the questions.

Evaluation Metrics Regarding evaluation metrics, for both SimpleQuestions (Bordes et al., 2015a) and QALD-10 (Perevalov et al., 2022), we adopted the Hit@1 metric as the measure of question answering accuracy. For the Nature Questions dataset, ROUGE-L-f1 (Lin, 2004) is used to evaluate the accuracy and comprehensiveness of LLM's answers.

4.3 Baselines

In order to judge the validity of the Pseudo-Graph Generation and Atomic Knowledge Verification method, the following baselines for comparison are chosen:

- **IO** (Brown et al., 2020) We use the standard input-output (IO) prompt as for conducting

Method		Hit@1		ROUGE L
		SimpleQuestions	QALD-10	Nature Questions
GPT-3.5	IO	20.2	38.7	20.5
	CoT	22.0	40.5	23.2
	SC	21.2	41.1	-
	QSM	27.5	34.2	23.8
	Ours	**34.3**	**48.6**	**37.5**
GPT-4	IO	29.9	44.7	20.9
	CoT	32.2	48.9	27.7
	SC	36.0	48.9	-
	QSM	31.3	46.2	27.0
	Ours	**40.0**	**56.5**	**39.2**

Table 2: The main results of our experiments. **Bold** indicates the best-performing method on the dataset for the model. SimpleQuestions and QALD use Hit@1 for evaulation. And Nature Questions evals with ROUGE-L.

direct input-output testing of the model, with 6 in-context examples.

- **Chain of Thougnt** (CoT) (Wei et al., 2022) It encourages the model to generate the thinking process during the output generation to enhance the model, with 6 in-context examples.
- **Self-Consistency** (SC) (Wang et al., 2023b) We use a sampling temperature of 0.7 and perform three sampling iterations, using voting to process the results in our experiments.
- **Question Sematic Matching** (QSM) Directly matching the question with the semantic knowledge graph for retrieval.

4.4 Main Results

Our main results can be seen in Table 2. It demonstrates the effectiveness of the framework in open-ended questions and in traditional precise questions

4.4.1 Comparison With Other Methods

Firstly, we can observe from Table 2 that the the Pseudo-Graph Generation and Atomic Knowledge Verification method method achieves better results than the baselines across different LLMs and datasets. That QSM performs worst on QALD-10. Especially on QALD with GPT-3.5 can also be observed. It performs 4.5 points lower compared to the IO method. This to some extent indicates the challenge of directly matching the semantics of the question to obtain triples in multi-hop questions. The continuous nature of question expression contrasts with the discontinuous nature of semantic triples, leading to a certain gap between the two.

Advantage in deterministic question answering: For deterministic question answering, Pseudo-Graph Generation and Atomic Knowledge Verification methodmethod achieves the highest accuracy on the QALD-10 dataset. On the QALD-10 dataset, the method even outperforms the ToG method, which achieves a result of 54.7 with GPT-4(Anonymous, 2024). Furthermore, this method with all models achieves a higher accuracy than the fine-tuned SOTA(Borroto et al., 2022), whose accuracy is 45.4. The improvement is also evident on the SimpleQuestions. This method achieves improvements of 10.9(GPT-3.5) and 5.3(GPT-4), respectively, compared to the second baseline on the two models. We also found that, with the enhancement of the approach, the factual hallucination of GPT-3.5 can be effectively addressed. It even outperforms various GPT-4 baselines on SimpleQuestion dataset.

Better performance in nature questions answering: On the Nature Questions answering dataset, the method also achieves significant improvements in terms of the ROUGE-L evaluation metric. With the Pseudo-Graph Generation and Atomic Knowledge Verification method method, for open problems, GPT-3.5 performs even better than GPT-4 under CoT prompt enhancement. And GPT-4 could improve about at least 11.5 in ROUGE-L. This indicates that this method can be effectively applied to real-world problems as well. Taking into account the improvement from QALD-10 (multi-hop questions) and Nature Questions (open-ended questions), there is a conclusion that the method is effective for complex questions.

In conclusion, the above results indicate that good performance in both precise question-answering and open-ended question-answering tasks has been achieved by LLMs with this method.

4.4.2 Generalization Across Different KG Sources

Additionally, we also proof the generalization capability of our method across different knowledge graph sources by evaluating it on the same set of questions but with varying KG sources. GPT-3.5 is selected as the model for the test. SimpleQuestions and Nature Questions are selected to validate the generalization capability of the method. We compared the improvement of model compared to the CoT on different datasets and different knowledge graphs to validate its performance. From the

Method	SimpleQuestion	Nature Questions
CoT	22.0	23.2
Our/ Freebase	38.2	26.7
Gain	+16.2	+3.5
Our/ Wikidata	28.1	37.5
Gain	+6.1	+14.3

Table 3: Performance on SimpleQuestions and Nature Questions with different KG sources. The SimpleQuestions is based on the Freebase as KG sources.

Table 3, it is evident that the method has shown improvement compared to the CoT method on the same set of questions across different KG sources. It is worth mentioning that SimpleQuestions is based on Freebase as the KG source. However, certain single-hop relations in Freebase might require multi-hop in Wikidata. This factor was not considered during the construction of the knowledge graph, which led to a smaller improvement in performance.

In conclusion, the results above have demonstrated to some extent the generalization capability of the method across different KG sources for the same set of questions.

4.5 Ablation Study

In this section, we aim to verify whether components of the method are operational. A most important question is: Are the verification steps useful? It is possible that when we use the model to generate a pseudo-graph, we activate the internal knowledge of the model, thereby enhancing the model's performance. Following this, pseudo-graph verification steps may lightly reduce the accuracy of the pseudo-graph. Thereby this makes the results superficially better than other baselines. Therefore, GPT-3.5 and GPT-4 are selected for testing on QALD-10 and Nature Questions. During the experiment, we compared the accuracy of directly providing the model with the pseudo-graph to determine whether the verification steps could perform well.

Method	QALD-10	Nature Questions
CoT	40.5	23.2
Pseudo-Graph	44.4	24.3
Gain from CoT	+3.9	+1.1
Verification	48.6	37.5
Gain from Pseudo-Graph	+4.2	+13.2

Table 4: GPT3.5's performance on QALD-10 and Nature Questions with different KG.

Method	QALD-10	Nature Questions
CoT	48.9	27.7
Pseudo-Graph	53.9	24.4
Gain from CoT	+5.0	-3.3
Verification	56.5	39.2
Gain from Pseudo-graph	+2.6	+14.8

Table 5: GPT4's performance on QALD-10 and Nature Questions with different KG.

Pseudo-Graphs Generation stimulate model's knowledge: From Table 4 , we can see that for GPT-3.5, using the model to generate pseudo-graphs, as opposed to the traditional CoT, can to some extent activate the model's factual capabilities. Regarding Table 5, for the QALD-10 dataset, the pseudo-graph also enhanced the traditional CoT with better performance.

Atomic Knowledge Verification increases precision and breadth of knowledge: Additionally, from Table 4, verification steps did not lead to a decline in the model's performance; instead, they resulted in an improvement in accuracy from pseudo-graph. In Table 5, for Nature Questions, the pseudo-graph actually led to a certain decline in the model's performance. This could be because, when the model generate the pseudo-graph, it becomes more inclined to output knowledge that it is certain of. This leads to the pseudo-graph not being comprehensive in terms of facts. Moreover, the performance of GPT-4 when using CoT has been quite good. So, there is a slight decline in performance in this case. However, this also further

demonstrates that the Atomic Knowledge Verification steps indeed perform effectively.

In summary, from the results mentioned above, it is not difficult to find that the verification steps can work effectively. Moreover, this results can to some extent explain the function of the two steps. Utilizing the model to generate pseudo-graphs can more effectively stimulate the model's self-factual retrieval capability compared to the traditional CoT method. Additionally, the Atomic Knowledge Verification steps can not only correct erroneous facts in precise questioning but also significantly enhance the factual accuracy and comprehensiveness of the model's answers in open-ended questions.

4.6 Error Analysis

It is aimed to identify some of the reasons for errors by analyzing the four steps. The details can be seen in Appendix A.2.

4.6.1 Generation of Pseudo-Graph

In the approach to generating pseudo-graphs using LLM, we first utilize LLM to generate Cypher statements and then parse them. Therefore, there might be instances where errors occur during the LLM's generation of Cypher statements. However, since Cypher is a relatively simple and formally structured language, the model seldom makes mistakes in writing the code in actual practice. GPT-3.5 exhibited a 0.6% error rate on QALD-10 and SimpleQuestion datasets. For other cases, the model did not make mistakes in generating Cypher statements.

We discovered that the primary reason for errors in generating Cypher by the model is: during generation, it mistakenly believes that a query needs to be made in the KG and thus uses the "MATCH". This could be related to the fact that the model primarily focuses on querying the knowledge graph during the training process.

4.6.2 Sematic Querying

During the process of semantic querying, there were instances where entities were not matched. This could be due to the issue of threshold settings in the semantic querying process. Additionally, during the pruning process based on the pseudo-graph, there might also be cases where the answer entities were inadvertently removed.

4.6.3 Pseudo-graph Verification

Errors may also occur during the process of fact verification using LLM. For example, some inaccuracies in the model during the verification process could lead to errors in the final graph. The proportion of new errors caused by verification could be calculated, compared to directly using the pseudo-graph, for two models on the QALD-10 dataset, as a percentage of the total errors. For GPT-3.5 and GPT-4, the proportions are 15.2% and 13.8%, respectively. This is to a certain extent within our acceptable range, and it also reflects to some degree that GPT-4's verification performance is superior.

We discovered that the main errors in the model's verification process were caused by directly appending the base graph after the pseudo-graph and not making modifications to the pseudo-graph.

4.6.4 Answer Generation

We found that in terms of answering, the model largely follows the graph for responses.

5 Conclusion & Future Work

A framework that combines Pseudo-Graph Generation and Atomic Knowledge Verification is proposed to enhance LLMs for open-ended questions. Pseudo-Graph Generation utilizes the property of LLM hallucination-even under erroneous conditions provide us with a framework for knowledge points. Atomic Knowledge Verification contains atomic-level fact semantic querying and verification to solve the generalization issue for the same question across different KG sources. We implemented the use of KG for LLM augmentation in open-answer questions scenarios. Experimental results show that the framework not only achieves good results in traditional precise questioning, but also gets an equally good boost in the natural question-answering. Our approach points out a feasible direction for the enhancement of LLM using KG in practical applications.

In future work, we would like to utilize better semantic encoding models to enhance semantic querying. Whether there are better pruning strategies to improve the quality of the knowledge acquired is also a direction to ponder. There is also a reflection to build an additional Pseudo-Graph Verification module to better enhance the knowledge of the LLM. Finally, we want to engineer our framework and build means of enhancing LLM that can be practically applied.

6 Limitations

However, the method still has certain limitations. For example, during Semantic Querying, it is possible that the answer entity may be omit during pruning, resulting in errors. Additionally, during Pseudo-Graph Verification, due to using LLM to verify itself, there may be a bias towards the LLM's pseudo-graph, leading to unsuccessful verification.

References

Anonymous. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015a. Large-scale simple question answering with memory networks. *ArXiv*, abs/1506.02075.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015b. Large-scale simple question answering with memory networks.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Manuel Borroto, F. Ricca, Bernardo Cuteri, and Vito Barbara. 2022. Sparql-qa enters the qald challenge. In *NLIWoD@ESWC*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Shuaichen Chang and Eric Fosler-Lussier. 2023. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings. *arXiv preprint arXiv:2305.11853*, abs/2305.11853.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311.

Xinyan Guan, Yanjiang Liu, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. 2023. Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Rui Li, Xu Chen, Chaozhuo Li, Yanming Shen, Jianan Zhao, Yujing Wang, Weihao Han, Hao Sun, Weiwei Deng, Qi Zhang, and Xing Xie. 2023. To copy rather than memorize: A vertical learning paradigm for knowledge graph completion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6335–6347, Toronto, Canada. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Debjyoti Mondal, Suraj Modi, Subhadarshi Panda, Rituraj Singh, and Godawari Sudhakar Rao. 2024. Kam-cot: Knowledge augmented multimodal chain-of-thoughts reasoning. *ArXiv*, abs/2401.12863.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

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

Jinyoung Park, Ameen Patel, Omar Zia Khan, Hyunwoo J. Kim, and Joo-Kyung Kim. 2023. Graph-guided reasoning for multi-hop question answering in large language models.

Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. 2022. Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, pages 229–234.

Moritz Plenz, Juri Opitz, Philipp Heinisch, Philipp Cimiano, and Anette Frank. 2023. Similarity-weighted construction of contextualized commonsense knowledge graphs for knowledge-intense argumentation tasks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6130–6158, Toronto, Canada. Association for Computational Linguistics.

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.

Ruoxi Sun, Sercan O Arik, Hootan Nakhost, Hanjun Dai, Rajarishi Sinha, Pengcheng Yin, and Tomas Pfister. 2023. Sql-palm: Improved large language modeladaptation for text-to-sql. *arXiv preprint arXiv:2306.00739*, abs/2306.00739.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/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. *ArXiv*, abs/2308.13259.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. 2023. Cognitive mirage: A review of hallucinations in large language models.

Burak Yetiştiren, Işık Özsoy, Miray Ayerdem, and Eray Tüzün. 2023. Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

A Appendix

A.1 Our prompts

Our prompts can be seen in the Figure 3, 4, 5.

A.2 Error cases

Error cases can be seen in the Figure 6, 7, 8.

```
[Task description]:
You should the answer {Question} in the following steps:
<step 1> Find out what {Knowledge Planing} do you need to solve the {Question}
<step 2> Strictly fill the {Knowledge Planing} to construct the {Knowledge Graph} as as complete as possible  with (Cypher)

[Example 1]:
{Question}: Who has the largest area of the Great Lakes in the United States?

<step 1> {Knowledge Planning}:
To answer the question of who has the largest area of the Great Lakes in the United States,
we need to gather information about the Great Lakes, their individual areas, and the states they are located in.
Here are the key points we should include in the knowledge graph:

- Introduction:
  - Provide a brief overview of the Great Lakes and their significance.
- Basic Information:
  - List the names of the Great Lakes along with their respective areas.
  - Provide information about the states that border each Great Lake.
- Largest Area:
  - Identify the specific state that has the largest area of a particular Great Lake.

<step 2> {Knowledge Graph}:
Here's a knowledge graph that represents the information we gathered:
```
// Create Great Lakes nodes
CREATE (superior :Lake {name: 'Lake Superior', area: 82000})
CREATE (michigan :Lake {name: 'Lake Michigan', area: 58000})
CREATE (huron :Lake {name: 'Lake Huron', area: 23000})
CREATE (ontario :Lake {name: 'Lake Ontario', area: 19000})
CREATE (erie :Lake {name: 'Lake Erie', area: 9600})
...
```

[Example 2]:
{Question}: Who covers more countries, the Andes or the Himalayas?

<step 1> {Knowledge Planning}:
To effectively answer the question "Who covers more countries, the Andes or the Himalayas?"
I need to gather information about the Andes and the Himalayas, as well as the countries they span. Here are the key points I want to include in the knowledge graph:

- Introduction:
  - Briefly describe the Andes and the Himalayas.
- Geography:
  - List the countries that the Andes cover.
  - List the countries that the Himalayas cover.

<step 2> {Knowledge Graph}:
```
// Create Andes node
CREATE (andes:MountainRange {name: "Andes"})

// Create Himalayas node
CREATE (himalayas:MountainRange {name: "Himalayas"})

// Create countries and connect them to the Andes
CREATE (andes)-[:COVERS]->(ecuador:Country {name: "Ecuador"})
CREATE (andes)-[:COVERS]->(colombia:Country {name: "Colombia"})
CREATE (andes)-[:COVERS]->(venezuela:Country {name: "Venezuela"})
CREATE (andes)-[:COVERS]->(bolivia:Country {name: "Bolivia"})
CREATE (andes)-[:COVERS]->(peru:Country {name: "Peru"})
CREATE (andes)-[:COVERS]->(chile:Country {name: "Chile"})
CREATE (andes)-[:COVERS]->(argentina:Country {name: "Argentina"})

// Create countries and connect them to the Himalayas
CREATE (himalayas)-[:COVERS]->(india:Country {name: "India"})
CREATE (himalayas)-[:COVERS]->(china:Country {name: "China"})
CREATE (himalayas)-[:COVERS]->(nepal:Country {name: "Nepal"})
CREATE (himalayas)-[:COVERS]->(bhutan:Country {name: "Bhutan"})
CREATE (himalayas)-[:COVERS]->(myanmar:Country {name: "Myanmar"})
CREATE (himalayas)-[:COVERS]->(pakistan:Country {name: "Pakistan"})
CREATE (himalayas)-[:COVERS]->(afghanistan:Country {name: "Afghanistan"})

// Additional information
CREATE (andes)-[:KNOWN_FOR]->(mountainClimbing:Concept {name: "Mountain Climbing"})
CREATE (himalayas)-[:KNOWN_FOR]->(mountainClimbing)

// Information about height
CREATE (andes)-[:HAS_PROPERTY]->(height:Property {name: "High Altitude"})
CREATE (himalayas)-[:HAS_PROPERTY]->(height)
```

[Task]:
{Question}:
```

Figure 3: Prompt for pseudo-graph. We partially omit the section involving generated code due to the large number of lines it occupies.

```
[Task description]:
Please based the "gold graph" below deleting redundant content from "graph to fix" and adding missing content from "graph tofix" to help me solve the [problem], follow the format in [Example]:
[Example]:
[problem]:  "Who has the largest area of the Great Lakes in the United States? "
"gold graph":
[entity_0]:
<Lake Superior> <area> <82350>
<Lake Superior> <connects with> <Keweenaw Waterway>

[entity_1]:
<Lake Michigan> <area> <57750>
<Lake Michigan> <part of> <Lake Michigan-Huron>

[entity_2]:
<Lake Huron> <area> <59600>

[entity_3]:
<Lake Ontario> <area> <18529>
<Lake Ontario> <inflows> <Niagara River>
<Lake Ontario> <inflows> <Millhaven Creek>
<Lake Ontario> <inflows> <Black River>

[entity_4]:
<Lake Erie> <area> <25700>
<Lake Erie> <length> <388>

[entity_5]:
<Dongting Lake> <area> <4350>

"graph to fix":
<Lake Superior> <AREA> <82000>
<Lake Michigan> <AREA> <58000>
<Lake Huron> <AREA> <23000>
<Dongting Lake> <AREA> <259430>

"Fixed graph":
<Lake Superior> <area> <82350>
<Lake Michigan> <area> <57750>
<Lake Huron> <area> <59600>
<Lake Ontario> <area> <18529>
<Lake Erie> <area> <25700>

[Example]:
[problem]:  "What is the population of China?"
"gold graph":
[entity_0]:
<China> <population> <1375198619>
<China> <population> <1442965000>
<China> <population> <1443497378>

[entity_1]:
<United States of America> <population> <332278200>
<United States of America> <population> <331449281>


"graph to fix":
<China> <Number of population> <1463725000>

"Fixed graph":
<China> <population> <1443497378>

[Task]:
Please based the [Example] above deleting redundant content from "graph to fix" and adding missing content from "graph to fix with the "gold graph" to help me solve the [problem].
Please note that you only need to extract the information necessary for me to solve the [problem], thank you!
If "graph to fix" has triples that are not in the "gold graph", just delete them!
If "graph to fix" has triples that are conflict with the "gold graph", replace them with the ones in the "gold graph".!
If there are time-varying triples such as "<China> <population> <1443497378>", the "gold graph": will already have these triplesin chronological order, so you can just pick the last one that
comes up.
[problem]:
```

Figure 4: Prompt for pseudo-graph Verification

```
[Task description]:
Please use the [problem] below to answer the [question], follow the format in [Example], you need to mark your answer with "{}":
[Example]:
[problem]: "What is the population of China?"
[graph]:
<China> <population> <1442965000>
<China> <population> <1443497378>
<United States of America> <population> <331449281>
[answer]: Based on the [graph] above, the population of China is {1443497378}.

[Example]:
[problem]: "Who has the largest area of the Great Lakes in the United States? "
[graph]:
<Lake Superior> <area> <82350>
<Lake Michigan> <area> <57750>
<Lake Huron> <area> <59600>
<Lake Ontario> <area> <18529>
<Lake Erie> <area> <25700>
[answer]: Based on the [graph] above, the largest of the Great Lakes is {Lake Superior} which area is 82,350.

[Task]:
If there are time-varying triples such as "<China> <population> <1443497378>", the [graph] will already have these triples in chonological order, so you can just pick the last one that comes up.
You need to mark the answer entity to the question with "{}".
If [graph] has no triples, answer with your own knowledge.
[problem]:
```

Figure 5: Prompt for answer generation

```
[ground graph]:
[entity_0]:
<Stevie Wonder> <description> <American musician (born 1950)>
<Stevie Wonder> <sex or gender> <male>
<Stevie Wonder> <occupation> <singer-songwriter>
<Stevie Wonder> <occupation> <record producer>
<Stevie Wonder> <occupation> <singer>
<Stevie Wonder> <occupation> <pianist>
<Stevie Wonder> <occupation> <vocalist>
<Stevie Wonder> <occupation> <music arranger>
<Stevie Wonder> <occupation> <drummer>
<Stevie Wonder> <discography> <Stevie Wonder discography>
<Stevie Wonder> <Commons category> <Stevie Wonder>
<Stevie Wonder> <instance of> <human>
<Stevie Wonder> <image> <Stevie Wonder 1973.JPG>
<Stevie Wonder> <genre> <rhythm and blues>
<Stevie Wonder> <genre> <soul music>
<Stevie Wonder> <genre> <funk>
<Stevie Wonder> <genre> <jazz>
<Stevie Wonder> <genre> <blues>
<Stevie Wonder> <genre> <pop music>
<Stevie Wonder> <ISNI> <0000000108675094>
<Stevie Wonder> <record label> <Motown>
<Stevie Wonder> <social media followers> <+1180000>

[entity_1]:
<Kati Kovács> <description> <Hungarian recording artist; actress and singer>
<Kati Kovács> <instance of> <human>
<Kati Kovács> <occupation> <singer>
<Kati Kovács> <occupation> <actor>
<Kati Kovács> <occupation> <lyricist>
<Kati Kovács> <occupation> <recording artist>
<Kati Kovács> <image> <KovacsKatiKossuth-díj.jpg>
<Kati Kovács> <Commons category> <Kati Kovács>
<Kati Kovács> <ISNI> <0000000369547648>
<Kati Kovács> <genre> <jazz>
<Kati Kovács> <genre> <pop rock>
<Kati Kovács> <genre> <folk rock>
<Kati Kovács> <record label> <Hungaroton>
<Kati Kovács> <sex or gender> <female>
<Kati Kovács> <discography> <Kati Kovács discography>

[graph]:
<Madam Satan> <HAS_PROPERTY> <Music Genre>
<Music Genre> <VALUE> <Jazz>
<Madam Satan> <HAS_PROPERTY> <Type>
<Type> <VALUE> <Nightclub>
<Madam Satan> <HAS_PROPERTY> <Capacity>
<Capacity> <VALUE> <500>
<Madam Satan> <LOCATED_IN> <Philadelphia>
```

Verification was not possible due to incorrect pruning of the correct entity.

Figure 6: Wrong pruning of the right entity.

```
[ground graph]:

[graph]:
<Francis Galton> <PUBLISHED> <Natural Inheritance>
<Francis Galton> <PUBLISHED> <Inquiries into Human Faculty and Its Development>
<Francis Galton> <PUBLISHED> <Hereditary Genius>
<Francis Galton> <INFLUENCED> <Anthropometry>
<Francis Galton> <INFLUENCED> <Psychometrics>
<Francis Galton> <INFLUENCED> <Statistics>
<Francis Galton> <INFLUENCED> <Eugenics>
<Karl Pearson> <WORKED_IN> <Statistics>
<Charles Davenport> <WORKED_IN> <Eugenics>
<Alfred Binet> <WORKED_IN> <Psychometrics>
<Harry Hollingworth> <WORKED_IN> <Psychometrics>
```

Prune too much

Figure 7: The threshold is too high, making all entities pruned.

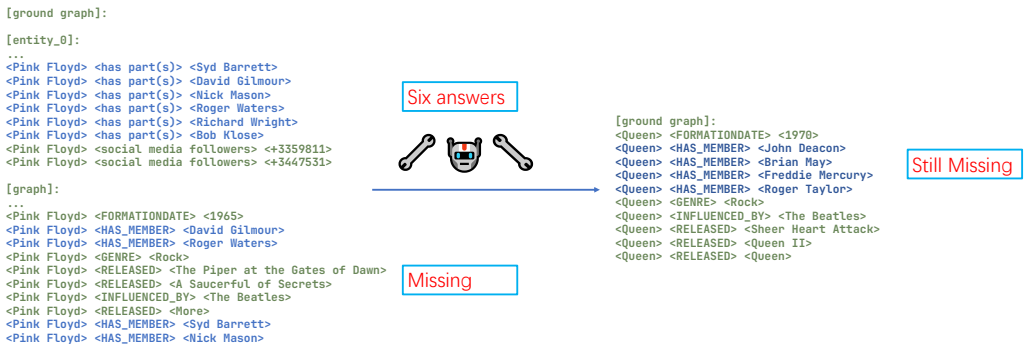

Figure 8: LLM error verification.