EffiQA: Efficient Question-Answering with Strategic Multi-Model Collaboration on Knowledge Graphs

Zixuan Dong[1], **Baoyun Peng**[2,*], **Yufei Wang**[1], **Jia Fu**[1], **Xin Zhou**[1], **Yongxue Shan**[1], **Weiguo Chen**[1], **Kangchen Zhu**[1], **Xiaodong Wang**[1,*]

[1]National University of Defense Technology, [2]Academy of Military Sciences

Correspondence: pengbaoyun13@alumni.nudt.edu.cn

Abstract

While large language models (LLMs) have shown remarkable capabilities in natural language processing, they struggle with complex, multi-step reasoning tasks involving knowledge graphs (KGs). Existing approaches that integrate LLMs and KGs either underutilize the reasoning abilities of LLM or suffer from prohibitive computational costs due to tight coupling. To address these limitations, we propose a novel collaborative framework named EffiQA that can strike a balance between performance and efficiency via an iterative paradigm. EffiQA consists of three stages: global planning, efficient KG exploration, and self-reflection. Specifically, EffiQA leverages the commonsense capability of LLMs to explore potential reasoning pathways through global planning. Then, it offloads semantic pruning to a small plug-in model for efficient KG exploration. Finally, the exploration results are fed to LLMs for self-reflection to further improve global planning and efficient KG exploration. Empirical evidence on multiple KBQA benchmarks shows EffiQA's effectiveness, achieving an optimal balance between reasoning accuracy and computational costs. We hope the proposed new framework will serve as a step forward in enabling efficient, knowledge-intensive querying through the integration of LLMs and KGs, fostering future research on knowledge-based question answering.

1 Introduction

Large language models (LLMs) (Radford et al., 2019; Ouyang et al., 2022; Touvron et al., 2023) have shown impressive capabilities across various natural language processing tasks, generating coherent and context-sensitive responses that demonstrate deep linguistic insights (Wei et al., 2022). However, they struggle with complex, multi-step reasoning tasks, including complex arithmetic

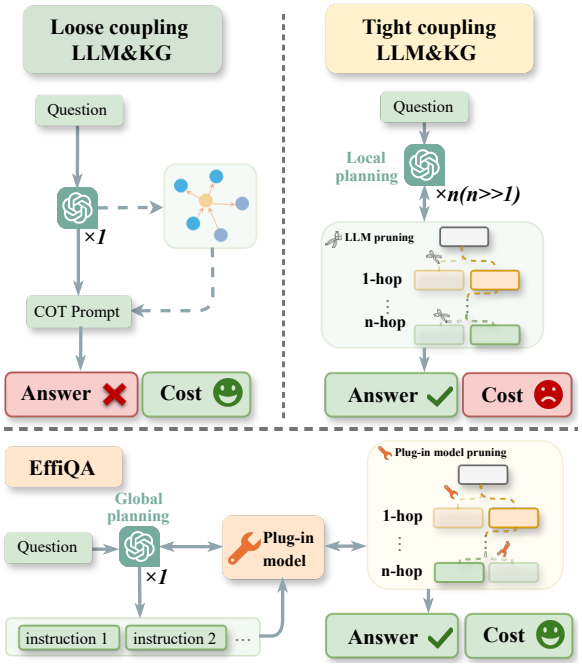

Figure 1: LLM-based KBQA includes three paradigms: loose-coupling (LLM-only with prompts), tight-coupling (LLM exploring KG iteratively), and moderate-coupling (LLM for planning, plug-in model for KG exploration).

(Wang et al., 2022), commonsense (Zhao et al., 2024), symbolic reasoning (Pan et al., 2023b), and multi-hop question answering (Yasunaga et al., 2021; Pan et al., 2024a).

To enhance the reasoning abilities of LLMs, techniques like chain-of-thought (CoT) prompting (Wei et al., 2022) have been developed, enabling step-by-step rationale generation before arriving at the final answer. Despite improved performance on various reasoning tasks (Yao et al., 2024; Besta et al., 2024; Turpin et al., 2024), CoT prompting sometimes fails to generate sufficient sub-questions to gather all necessary information, leading to issues like hallucinations (Lyu et al., 2023; Lin et al., 2021), opaque reasoning (Suzgun et al., 2022), and reliance on outdated data (Borgeaud et al., 2022;

*Corresponding Author

Proceedings of the 31st International Conference on Computational Linguistics, pages 7180–7194
January 19–24, 2025.

Izacard et al., 2023).

To address these challenges, researchers have explored integrating external knowledge sources, such as knowledge graphs (KGs) (Sun et al., 2021; Yasunaga et al., 2021; Zhang et al., 2022a; Li et al., 2023b; Pan et al., 2024a), into the reasoning process. These methods typically involve retrieving information from KGs, augmenting the prompt, and feeding it into LLMs. However, these loose-coupling paradigms often rely on simple data retrieval, failing to harness the full reasoning potential of LLMs. Consequently, their success hinges on the completeness and quality of the KGs.

Recent approaches have explored tighter integration between LLMs and KGs, such as Think-on-Graph (ToG) (Sun et al., 2023), Reasoning-on-Graph (Luo et al., 2023), and Chain-of-Knowledge (Li et al., 2023b). These methods use LLMs to iteratively explore entities and relations in KGs, achieving better performance but at the cost of excessive computational resources and the risk of inefficiency due to potential path explosion, which can lead to suboptimal reasoning in dynamic queries.

Striking a balance between the coupling degree of LLMs and KGs, such that it can fully utilize their respective capabilities to enhance KBQA performance while maintaining efficiency, remains a challenge. To address this challenge, we propose a novel collaborative framework EffiQA for LLM-based KBQA, which leverages the commonsense capability of LLMs for global planning while offloading semantic pruning to a small plug-in model.

Specifically, EffiQA consists of the following stages:

Global Planning. In this stage, LLM is employed to decompose the question into several semantically coherent trajectories and generate exploration instructions for exploring potential reasoning pathways and extending search space beyond the knowledge graph's structural limits.

Efficient KG exploration. In this stage, a plug-in model is employed for semantic pruning based on global planning to remove irrelevant nodes and paths during the KG search process.

Self-reflection. In this stage, LLM will proceed to self-reflect on the exploration results to refine the global planning, leading to improved planning and exploration in the subsequent iteration.

To strike a balance between performance and efficiency, EffiQA introduces an enhanced querying strategy that tightly couples the LLM's instructions with constrained semantic pruning of the KG. This allows EffiQA to selectively expand the most promising graph regions based on semantics and type of entities, substantially reducing the search space while maintaining relevance. A fine-grained semantic matching process further focuses the pruning on conceptually relevant relationships. By having the LLM provide high-level guidance while offloading computationally expensive KG traversal to a specialized model, EffiQA achieves a balanced integration.

Different from previous methods, this collaborative approach can not only enhance the reasoning performance by combining the strengths of LLMs and KGs but also improve operational efficiency. Figure 1 shows the difference between the proposed method and previous methods. By striking a balanced integration between LLMs and KGs, EffiQA redefines the standards for efficient, knowledge-intensive querying in KBQA tasks.

2 Related Work

2.1 Integration of External Knowledge Sources

Recent advancements in LLMs have focused on enhancing reasoning capabilities by integrating external information (Guo et al., 2023; Yang et al., 2024). Notable examples include BlenderBot3 (Shuster et al., 2022) and Atlas(Izacard et al., 2023), which achieves marked enhancements in its performance on the Knowledge Intensive Language Tasks (KILT) benchmark (Petroni et al., 2020). These developments illustrate a growing trend towards the dynamic incorporation of external data sources into LLMs, which serves to enrich their foundational knowledge base and diminish the frequency of inaccuracies in generated content (Baek et al., 2023b; Pan et al., 2024b).

2.2 Knowledge-Enhanced Reasoning in LLMs

LLMs enhanced with structured external knowledge have shown immense potential for accurately understanding user intentions (Jiang et al., 2023). Nevertheless, these models often struggle with complex reasoning tasks, such as multi-hop Knowledge Base Question Answering (KBQA), primarily due to their limited capability in decomposing multi-step problems into essential intermediate steps needed to derive an answer (Guan et al., 2024). In response to this challenge, the Chain

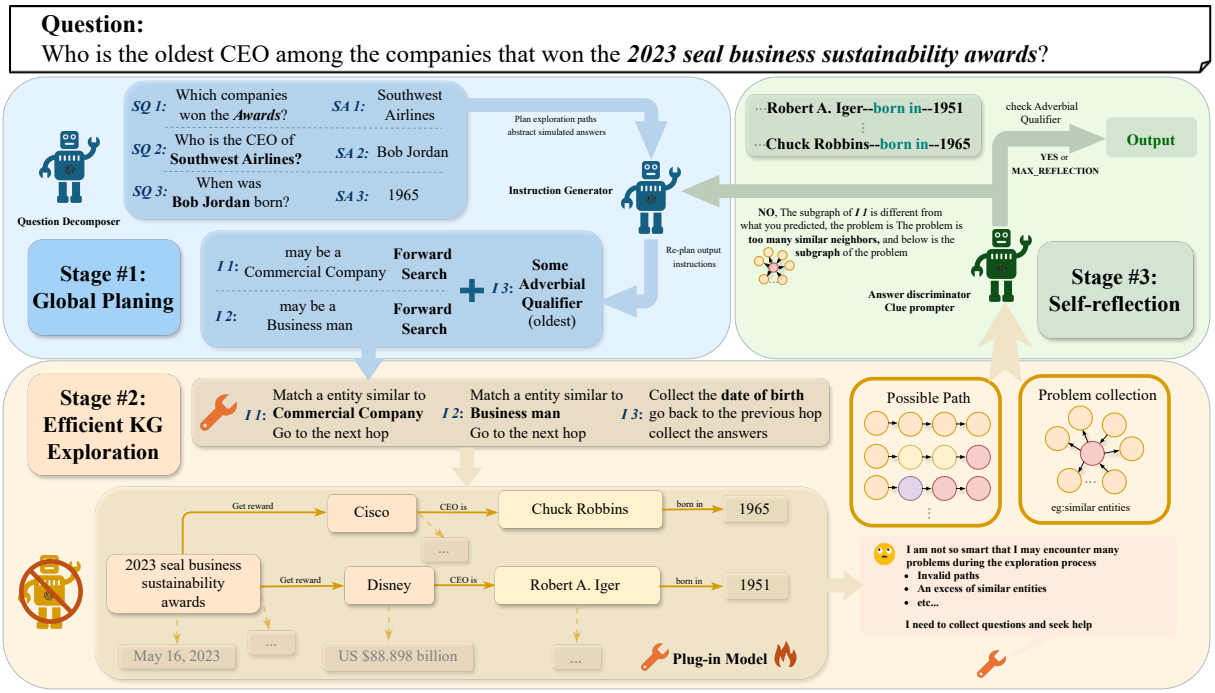

Figure 2: The EffiQA workflow consists of three stages. First, the LLM decomposes the problem and generates instructions that include simulated answers and actions based on the problem's logic. Next, EffiQA employs a plug-in model to execute these instructions, perform efficient knowledge graph exploration, and identify potential issues. Finally, the LLM reviews the identified problems, iteratively replans, and produces answers once sufficient information is available.

of Thought (CoT) method was introduced (Wei et al., 2022), further developed into variations such as Auto-CoT and Zero-Shot-CoT (Zhang et al., 2022b; Kojima et al., 2022). This method exemplifies a structured prompting technique that significantly enhances the efficiency of LLMs in navigating complex reasoning tasks. Concurrently, innovative frameworks like Chain-of-Knowledge (CoK) (Li et al., 2023b) and Think-on-Graph (ToG) (Sun et al., 2023) have been developed, integrating knowledge retrieval directly into the reasoning process, thereby enriching the depth and improving the factual accuracy of the generated responses.

2.3 Advanced Frameworks for Structured Knowledge Utilization

The evolution of knowledge integration strategies has significantly advanced the development of frameworks that facilitate more profound interactions between LLMs and knowledge graphs. For instance, a work (Pan et al., 2023a) has introduced a method in which LLMs employ a greedy search algorithm to navigate KGs, enabling more nuanced interactions with data. In a related development, the Clue-Guided Path Exploration (CGPE) (Tao et al., 2024) framework, which effectively combines a knowledge base with an LLM, using clues from queries to guide systematic exploration, which in turn helps reduce computational loads. Concurrently, the Verification and Editing (VE) framework has been developed by (Zhao et al., 2023). This framework seeks to refine the reasoning outputs of LLMs, which consequently increases both the fidelity and reliability of the model's responses.

3 Method

The EffiQA framework consists of three main components that are executed iteratively: global planning, efficient KG exploration, and self-reflection. In the global planning stage, LLM is leveraged to decompose the input question into semantically coherent trajectories and generate exploration instructions, which helps explore potential reasoning pathways and extend the search space beyond the structural limits of the KG. Then, a small plug-in model is employed for efficient KG exploration, performing breadth-first search and semantic pruning on KG with the help of exploration instructions. In the self-reflection stage, LLM will reflect on the exploration results to revamp global planning and KG exploration for further improvement. Figure

2 shows the overall framework of the proposed method.

3.1 Global Planning

In the global planning phase, the LLM leverages its powerful reasoning abilities to globally plan the exploration pathway. At this stage, LLM will give a set of instructions based on the given question and initial entity to guide the plug-in model to explore the graph.

Query Decomposition Initially, the LLM identifies the primary subject entity e_0 from the query and deconstructed into M sub-questions $Q = \{q_1, q_2, \ldots, q_M\}$ with adverbial qualifiers $D = \{d_1, d_2, \ldots, d_N\}$, where $(N \leq M)$. This process is formulated as:$Q, D \leftarrow \text{Decompose}(query, e_0)$.

Instruction Generation and Optimization After decomposition, the LLM generates simulated answers $A_i \leftarrow \text{Simulate}(e_i, q_i, d_i)$ and constructs corresponding instructions instructions $\leftarrow \text{S}(e_i, q_i, d_i)$ to guide the plug-in model in efficiently navigating the knowledge graph. Specifically, "forward search" instructions are created for sub-questions, and "adverbial qualifier" instructions are designed to handle qualifiers; these are collectively referred to as action instructions. Additionally, we have developed other action instructions tailored for the plug-in model and generate the appropriate matching instructions. For example, Figure 3 illustrates the retrieval of a sports event type.

3.2 Efficient KG exploration

The simulated answers and actions generated by global planning can effectively guide the plug-in model based on an intuitive prior premise: when LLM answers a question, the content of the answer may be inaccurate, but will generate simulated answers of the same type or with similar semantics as the answer. Using the simulated answers, we implement a constrained search with semantic pruning, selectively expanding promising entities to optimize efficiency.

Initialization and Systematic Exploration Graph initialization define the knowledge graph $G = (E, R)$ and start BFS from e_0 and for each entity $e_{current}$ in BFS, evaluate relations r, select a representative tail entity, and perform semantic matching with simulated answers using entity descriptions and triples.

Algorithm 1 EffiQA Framework

Input: Query q, Knowledge Graph G
$e_0 \leftarrow$ ExtractEntity(q)
while not converged **do**
  $Q, D \leftarrow$ DecomposeQuery(q, e_0)
  $A \leftarrow$ SimulateAnswers(Q, D)
  Instr $\leftarrow$ GenerateInstr(e_0, Q, D)
  for each instr in Instr **do**
    $Paths, Problems \leftarrow$KGExplore($G$, e_0, instr, A)
  end for
  $Result, Clues \leftarrow$Aggregate(Paths, Problems)
  if Result satisfactory **then**
    Output: Final Answer
    break
  else
    RevisePlan($Clues$)
  end if
end while

Semantic Matching and Graph Traversal EffiQA uses fine-grained entity typing (FGET) based on the Ontonotes dataset (Dan et al., 2014) for semantic matching, ensuring exploration focuses on pertinent relations despite potential inaccuracies For example, as shown in Figure 3, even if inaccurate instruction content is given due to large model time constraints, plug-in model execution instructions still allow the search for premises that are conceptually consistent despite being inaccurate in the simulation Continuing below, we demonstrate the robustness of the semantic matching process in managing time-sensitive inaccuracies by focusing on categorical relevance rather than precise data accuracy.

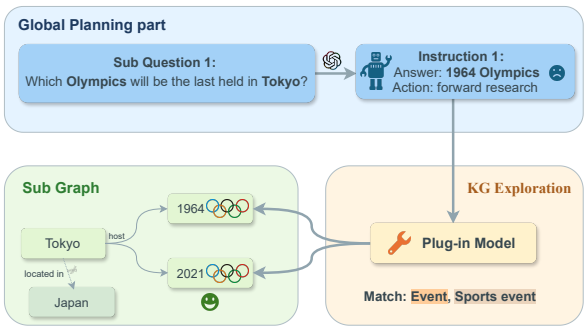

Figure 3: Despite the challenges associated with the temporal constraints of the training corpus (up to March 2021), the plug-in model effectively executes accurate match pruning. The detailed entity typing depicted in the figure serves solely for illustrative purposes

If a representative entity a'_{rep} matches a_i, extend exploration to all entities linked by r_i, controlling branching via a threshold. Unmatched relations r are pruned. Each entity is also checked against adverbial qualifiers d_i. Upon reaching the reasoning depth, the search ends and qualifying paths are collected. Problems encountered (e.g., unmatched paths, excessive branching) are collected with their subgraphs for further processing.

3.3 Self-reflection

Detailed Answer Aggregation In this phase, the LLM synthesizes results from efficient KG exploration to ensure answers are contextually relevant and precisely address the query. Each reasoning path is verified against its adverbial qualifiers d_i, and terminal entities are checked against the final constraints q_{final}. When multiple initial entities e_0 are present, validated paths are intersected to identify consistently relevant entities, enhancing answer robustness.

Strategic Replanning and Iteration Problematic paths are flagged for re-evaluation and replanning. During this process, the LLM reflects on issues identified in the efficient KG exploration's subgraphs and the generated clue prompts. It then performs a global replanning that integrates the problematic subgraphs and clue prompts, ensuring comprehensive coverage and addressing reasoning gaps. This iterative approach systematically resolves inaccuracies and inefficiencies, leading to more robust final answers. The final selection is based on intersected entities e'_{final}, ensuring the answer is derived from validated paths and fully meets the initial query requirements.

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets and Metrics

To rigorously evaluate the performance of the EffiQA framework, we selected five benchmark datasets: Complex Web Questions (CWQ) (Talmor and Berant, 2018), WebQuestionSP (WebQSP) (Yih et al., 2016), GrailQA (Gu et al., 2021), QALD10-en (Perevalov et al., 2022), and Simple Questions (Bordes et al., 2015). These datasets were chosen for their varying complexity and the different types of reasoning challenges they present, ranging from simple fact retrieval to sophisticated multi-hop questions. We assessed the framework on two primary metrics: cost efficiency, which measures the resource consumption per query, and answer accuracy (Hits@1), defined by the precision of responses compared to the gold standards.

At the same time, since the recall rate is crucial during path exploration, incorrect pruning will cause errors and lead to wrong answers. To retain the correct path while avoiding incorrect pruning, we define pruning recall as the fraction of correct paths retained during exploration. When calculating efficiency, we use the average number of floating-point operations (FLOPs) per query as an indicator of computational overhead. Cost efficiency is defined as the ratio of pruning recall rate to FLOPs, which quantifies the effectiveness of pruning relative to its computational cost, as shown Equation 1 below:

$$\textit{Cost Efficiency} = \frac{\textit{Pruning Recall Rate}}{\textit{FLOPs}} \quad (1)$$

4.1.2 Baselines

EffiQA is benchmarked against key established methods, including methods of reasoning without external knowledge, such as standard prompts (IO prompts) (Brown et al., 2020) and CoT prompts (Zhang et al., 2022b), as well as some state-of-the-art methods using external knowledge reasoning. These comparisons cover hint-based methods, fine-tuning methods, and different KBQA architectures. IO prompts and CoT prompts are two methods that do not require external knowledge, whereas the tight-coupling KBQA method is the most advanced approach for KBQA. These comparisons help illustrate the necessity of external knowledge in answering complex queries and demonstrate the cost-effectiveness of EffiQA compared to existing state-of-the-art methods. Through these comparisons, the flexibility and cost-effectiveness of EffiQA across various datasets can be proven.

4.1.3 Experiment Details

Comprehensive experiments on EffiQA were performed using two distinct LLMs: GPT-3.5-turbo, GPT-4[1] and four models with different parameter sizes for comparison. These models were selected to assess the framework's scalability and performance across varying computational capabilities. GPT-4 was utilized to explore the limits of performance in more complex scenarios. To ensure

[1]Both GPT-3.5-turbo and GPT-4 can be accessed at https://openai.com/

Method	CWQ	WebQSP	GrailQA	QALD10-en	Simple Questions
Without external knowledge					
IO prompt w/GPT-3.5-turbo$^{\alpha}$	37.6	63.3	29.4	42.0	20.0
CoT w/GPT-3.5-turbo$^{\alpha}$	38.8	62.2	28.1	42.9	20.3
SC w/GPT-3.5-turbo$^{\alpha}$	45.4	61.1	29.6	45.3	18.9
With external knowledge					
Prior FT SOTA	70.4$^{\beta}$	82.1$^{\gamma}$	75.4$^{\delta}$	45.4$^{\epsilon}$	85.8$^{\zeta}$
Prior Prompting SOTA	-	74.4$^{\eta}$	53.2$^{\eta}$	-	-
Prior tight-coupling SOTA$^{\alpha}$	72.5	82.6	81.4	54.7	66.7
EffiQA (ours) w/GPT-3.5-turbo	52.1	65.2	63.3	46.2	65.7
EffiQA (ours) w/GPT-4	**69.5**	**82.9**	**78.4**	**51.4**	**76.5**

Table 1: Comparison between EffiQA and related methods on KBQA tasks. Method α (Sun et al., 2023) has a significantly higher cost compared to EffiQA. β (Das et al., 2021), γ (Yu et al., 2022), δ (Gu et al., 2022), ϵ (Borroto et al., 2022), ζ (Baek et al., 2023a), η (Li et al., 2023a).

consistent and reproducible results, the temperature setting for all interactions with these models was fixed at 0, eliminating randomness in responses.

The plug-in model used for node semantic matching and path pruning utilizes RoBERTa (Liu et al., 2019), fine-tuned on the modified OntoNotes v5 dataset[2], and a entity typing training set of over 10,000 entities generated using GPT-4 for fine-tuning. The dataset is generated using the classification standards from the Context-Dependent Fine-Grained Entity Type Tagging method (Dan et al., 2014) as the basis for entity typing. Since only the path pruning process involves entity typing, to ensure accurate recognition of the named entity recognition (NER) part, we use special markers to mark the entities that need to be classified and then fine-tune the model to ensure that the specified entities can be correctly identified.

4.2 Main Results

4.2.1 Comparison to Other Methods

We compare EffiQA with some frameworks without external knowledge. As a baseline for large model capabilities, that is, the knowledge contained in the large models themselves, as can be seen from Table 1, the methods without external knowledge generally perform poorly, reflecting the LLMs dependence on knowledge graphs for answering knowledge-based questions. EffiQA leverages both large and smaller specialized language models integrated with external knowledge graphs, surpassing all methods that do not use external knowledge. At the same time, EffiQA provides a distinctive advantage over traditional fine-tuning approaches with its plug-and-play capability that requires no dataset-specific training.

Comparison with methods using external knowledge shows that even without any fine-tuning, EffiQA still outperforms existing fine-tuning methods. Particularly noteworthy is its performance on single-hop datasets. In comparison to ToG, which tightly couples with LLMs and KGs, EffiQA demonstrates competitive strength. Notably, EffiQA excels in single-hop datasets like SimpleQuestions, where it achieves 65.7% accuracy with GPT-3.5-turbo and 76.5% with GPT-4, underscoring its effective global planning for enhancing accuracy without sacrificing recall—a common shortfall in larger models that heavily prune data. The framework's strategy for addressing simple questions enhances accuracy without compromising the recall rate, a common issue in large models that employ aggressive pruning techniques, where EffiQA's global planning proves highly effective.

EffiQA also demonstrates competitive results on multi-hop datasets such as ComplexWebQuestions (CWQ) and WebQuestionsSP (WebQSP), scoring 69.5% and 82.9% respectively when using GPT-4. These results demonstrate that EffiQA's strategy of leveraging external knowledge significantly improves LLMs' deep reasoning capabilities, effectively managing complex queries that often pose challenges to tightly-coupled approaches that integrate LLMs and knowledge graphs, as they typ-

[2]OntoNotes v5 can be downloaded at https://huggingface.co/datasets/conll2012_ontonotesv5

ically consume at least twice the resources of EffiQA while achieving similar performance.

4.2.2 Performance with Different Backbone Models

In exploring the integration of EffiQA with different LLM backbones, we conducted ablation studies using a range of different scale models in table 2, since EffiQA relies heavily on LLM with powerful reasoning capabilities to give accurate instructions, we chose Llama3.1-8B (Vavekanand and Sam, 2024), GPT-3.5-turbo, Deepseek-V2, and GPT-4 as our planning and aggregation iteration modules. These studies aim to evaluate how the underlying LLM affects the overall accuracy of the system on multiple datasets such as CWQ and WebQSP.

Method	CWQ	WebQSP
Fine-tuned Baseline		
NSM $^{\alpha}$	53.9	74.3
DeCAF $^{\beta}$	70.4	82.1
Prompting Baseline		
KD-CoT $^{\gamma}$	50.5	73.7
LLMs		
COT (Llama3.1-8B)	32.8	56.6
EffiQA (Llama3.1-8B)	37.4	58.3
Gain	**+4.6**	**+1.7**
COT (DeepSeek-V2)	41.2	57.8
EffiQA (DeepSeek-V2)	61.7	67.4
Gain	**+20.5**	**+9.6**
COT (GPT-3.5-turbo)$^{\delta}$	38.8	62.2
EffiQA (GPT-3.5-turbo)	52.1	65.2
Gain	**+13.3**	**+3.0**
COT (GPT-4)$^{\delta}$	46.0	67.3
EffiQA (GPT-4)	69.5	82.9
Gain	**+23.5**	**+15.6**

Table 2: Performance comparison of methods on CWQ and WebQSP datasets. EffiQA consistently improves performance. α (He et al., 2021), β (Yu et al., 2022), γ (Wang et al., 2023), δ (Sun et al., 2023).

The results show EffiQA's performance improves with model capacity and complexity. Notably, Deepseek-V2 activates fewer parameters than GPT-3.5-turbo yet outperforms it, demonstrating the MOE architecture's effectiveness. GPT-4 excels in complex multi-hop queries due to strong inference capabilities. EffiQA is highly sensitive to the LLM's reasoning abilities, such as global planning and self-reflection, which enhance accuracy by improving instructions and PLM pruning, reducing repeated reasoning. When integrated with Llama3.1-8B, EffiQA gains performance, but improvements are limited by the smaller scale, highlighting its reliance on robust planning typically found in larger models.

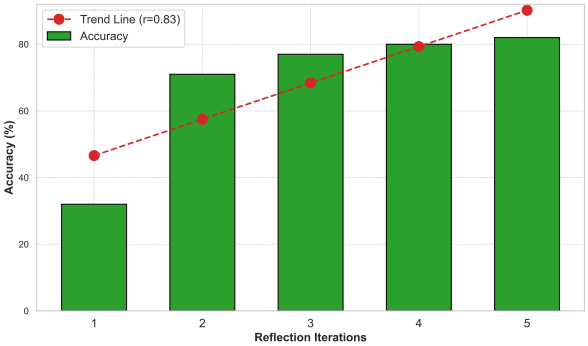

Figure 4: Accuracy by Reflection Iterations on WebQSP Dataset (MAX_REFLECTION=5)

4.3 Ablation study

4.3.1 Effect of Reflection Iterations on Accuracy

To investigate the impact of reflection iterations on model accuracy, we conducted experiments on the WebQSP dataset with a maximum of 5 reflection iterations. As shown in Figure 4, the accuracy of the model improves consistently with an increasing number of reflection iterations. The trend line in the graph indicates a strong positive correlation (r=0.84) between the number of reflections and the accuracy rate, demonstrating that iterative reflections contribute to better performance in entity classification tasks.

The progressive improvement suggests that incorporating multiple reflection iterations enables the model to refine its understanding and make more accurate predictions, possibly by reinforcing correct classifications and learning from mistakes. This reinforces the utility of our approach, especially in scenarios where high accuracy is paramount. By employing up to 5 reflection iterations, the model achieves significant performance gains, providing a balance between computational cost and accuracy enhancement.

4.3.2 Model scale, Efficiency and Performance

In order to prove that when PLM obtains LLM instructions, the recall rate of semantic matching

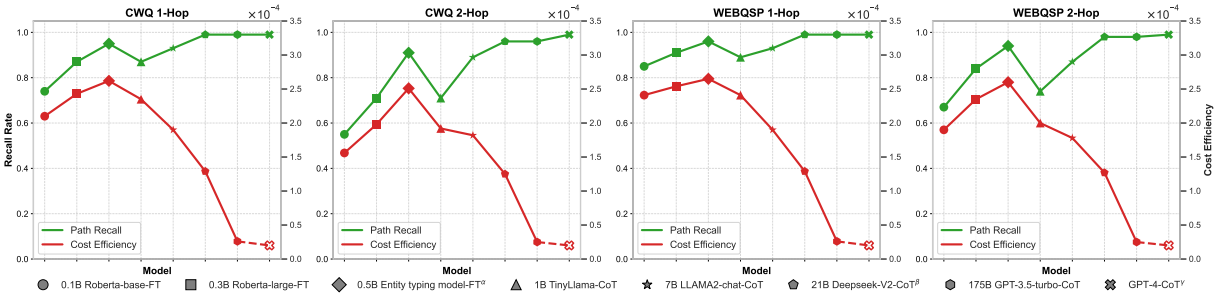

Figure 5: Path Recall Rate and Cost Efficiency of various model sizes on CWQ and WebQSP datasets. The medium-sized entity typing model achieves an optimal balance between recall and cost. α: RoBERTa models fine-tuned with increased parameters. β: Activation parameters calculated using the MoE model. γ: GPT-4 parameter estimates based on scale, as exact numbers are undisclosed.

and path pruning can achieve equivalent effects to LLM direct pruning, but its cost-effectiveness has obvious advantages compared with large models, we compare Models of different sizes perform in path pruning and semantic matching tasks through entity typing. The following figure 5 shows the change curve of recall rate and cost-effectiveness of path pruning by models of different sizes under different hop numbers in the CWQ and WebQSP data sets.

Among which models above 1B are the current mainstream decoder-only architecture, using the generative classification method. For these models, the CoT method is used and 2-shot is used for inference, while the models below 1B are dedicated entity typing models. Inference is performed after fine-tuning. The fine-tuning process uses 2*NVIDIA 4090 24G, the learning rate is set to 5e-6, and the original data set is used plus the generated training set for fine-tuning for 20 epochs.

It is worth noting that the recall rate of path pruning is not equal to the accuracy of the model's direct classification of entities, because the plug-in model always tends to classify semantically similar entities into one category. If the plug-in model is execute fine-grained entities typing and gives an incorrect result, the correct path may still be recalled successfully due to exploring the semantic similarity of entities, so the recall rate of path pruning is usually higher than the recall rate of entity typing.

It can be seen from the results that some dedicated fine-tuning plug-in models have achieved performance comparable to existing large models in path matching tasks based on entity classification, even better than LLM with a small number of parameters. At the same time, their cost-effectiveness is significantly higher than that of LLMs.

4.3.3 Computational Cost

Cost analysis highlights EffiQA's ability to significantly reduce the number of queries per input question for large models, thereby reducing computational expense. This efficiency not only enhances the applicability of the framework in resource-limited environments but also emphasizes its commercial potential for scalable real-world applications. In the cost consumption experiment on the KBQA data set, we set the exploration depth and width of TOG to 3 and stopped it when the LLM requests for a problem exceeded 30 times.

At the same time, we set the number of Self-reflections of EffiQA on single-hop problems. The threshold is set to 5, and the multi-hop value is set to 10. Tests are conducted on the CWQ and WebQSP data sets. We divide the cost consumption results of the above data sets into single and multi-hop calculations respectively. From Table 3, we can see that whether it is 1-hop or multi-hop inference on KG, the number of inferences and request cost of this solution are at least halved compared to the existing SOTA model TOG.

Method	1-Hop	Multi-Hop
TOG w/GPT-3.5-turbo	16.7	25.6
TOG w/GPT-4	14.8	21.4
EffiQA w/GPT-3.5-turbo	4.7	7.3
EffiQA w/GPT-4	3.2	6.5

Table 3: Average number of calls to LLM per question

5 Conclusion

In this work, we proposed EffiQA, a new integration paradigm of LLMs and KGs for multi-step reasoning. Through an iterative paradigm of global

LLM planning, efficient KG exploration, and self-reflection, EffiQA balances leveraging LLM capabilities with maintaining computational efficiency. The global planning outlines promising trajectories and generates instructions to guide semantic pruning during efficient KG traversal, reducing search spaces. Exploration results then refine the global plan iteratively. Extensive experiments demonstrate EffiQA's ability to optimally balance accuracy and costs.

Limitations

EffiQA exhibits sensitivity to the capabilities of large models, relying on their reasoning abilities for optimal performance. This dependence means that any limitations in the large model's ability can directly affect EffiQA's outcomes. Furthermore, the plug-in model encounters performance bottlenecks when scaling to larger or more complex knowledge graphs. As the knowledge graph grows, the computational effort required for effective exploration and semantic pruning increases, potentially slowing down processing and limiting the system's efficiency in extensive datasets. These factors constrain EffiQA's scalability and adaptability in more demanding scenarios.

References

Jinheon Baek, Alham Fikri Aji, Jens Lehmann, and Sung Ju Hwang. 2023a. Direct fact retrieval from knowledge graphs without entity linking. *arXiv preprint arXiv:2305.12416*.

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

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*, volume 38, pages 17682–17690.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Manuel Borroto, Francesco Ricca, Bernardo Cuteri, and Vito Barbara. 2022. Sparql-qa enters the qald challenge. In *Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference, Hersonissos, Greece*, volume 3196, pages 25–31.

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.

Gillick Dan, Lazic Nevena, Ganchev Kuzman, Kirchner Jesse, and Huynh David. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay-Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. *arXiv preprint arXiv:2104.08762*.

Yu Gu, Xiang Deng, and Yu Su. 2022. Don't generate, discriminate: A proposal for grounding language models to real-world environments. *arXiv preprint arXiv:2212.09736*.

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.

Xinyan Guan, Yanjiang Liu, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. 2024. Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18126–18134.

Tiezheng Guo, Qingwen Yang, Chen Wang, Yanyi Liu, Pan Li, Jiawei Tang, Dapeng Li, and Yingyou Wen. 2023. Knowledgenavigator: Leveraging large language models for enhanced reasoning over knowledge graph. *arXiv preprint arXiv:2312.15880*.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 553–561.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251.

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*.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Lidong Bing, Shafiq Joty, and Soujanya Poria. 2023b. Chain of knowledge: A framework for grounding large language models with structured knowledge bases. *arXiv preprint arXiv:2305.13269*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

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

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*.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*.

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.

Jeff Z Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, et al. 2023a. Large language models and knowledge graphs: Opportunities and challenges. *arXiv preprint arXiv:2308.06374*.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023b. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.

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

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

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. IEEE.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2020. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, et al. 2022. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv:2208.03188*.

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

Yueqing Sun, Qi Shi, Le Qi, and Yu Zhang. 2021. Jointlk: Joint reasoning with language models and knowledge graphs for commonsense question answering. *arXiv preprint arXiv:2112.02732*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Dehao Tao, Feng Huang, Yongfeng Huang, and Minghu Jiang. 2024. Clue-guided path exploration: An efficient knowledge base question-answering framework with low computational resource consumption. *arXiv preprint arXiv:2401.13444*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2024. Language models don't always say what they think: unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36.

Raja Vavekanand and Kira Sam. 2024. Llama 3.1: An in-depth analysis of the next-generation large language model.

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

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*.

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*, pages 1–20.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

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

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

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022a. Greaselm: Graph reasoning enhanced language models for question answering. *arXiv preprint arXiv:2201.08860*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022b. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. *arXiv preprint arXiv:2305.03268*.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2024. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36.

A Alternative Pruning Strategy: Clustering-Based Approach

In addition to the classification-based semantic pruning employed in EffiQA, we investigated an alternative pruning strategy utilizing clustering as a tool model. This approach aimed to group semantically similar entities within the knowledge graph to streamline the exploration process. Specifically, we leveraged a BERT-based model to generate embeddings for each entity and applied K-Means clustering to partition these embeddings into distinct clusters. The hypothesis was that clustering could effectively reduce the search space by allowing the system to focus on the most relevant clusters based on the query context.

However, empirical evaluations on the WebQSP dataset revealed significant limitations of the clustering-based approach. The clustering model achieved an accuracy of approximately 20%, which is substantially lower than the classification-based method integrated into EffiQA. This low accuracy can be attributed to several factors:

- **Model Precision:** The BERT-based clustering model struggled to accurately group entities with nuanced semantic relationships, leading to high levels of intra-cluster heterogeneity.
- **Noise Sensitivity:** Clustering algorithms are inherently sensitive to noise and outliers, which are prevalent in large-scale knowledge graphs. This sensitivity resulted in the formation of clusters that contained a significant number of irrelevant entities.
- **Recall Rate:** The recall rate for the clustering-based approach was notably low, meaning that many relevant entities were inadvertently pruned during the clustering process. This shortfall undermined the effectiveness of the pruning strategy, as essential information required to answer complex queries was often excluded.

Furthermore, the presence of substantial noise and the inability to reliably distinguish between closely related entities significantly hindered the clustering model's performance. As a result, the clustering-based pruning strategy did not achieve the desired balance between efficiency and accuracy, making it an unsuitable alternative to the classification-based approach within the EffiQA framework.

B Case Study: Query Processing Analysis

B.1 Successful Query Processing

Query Example: "Where did the 'Country Nation World Tour' concert artist go to college?"

In this case, LLM processes the query successfully by following a structured reasoning flow. The system first identifies the primary entity, the artist associated with the 'Country Nation World Tour,' and classifies it as the center entity for exploration. Based on the instruction, the LLM plans a search focusing on educational attributes linked to this artist within the knowledge graph. By traversing relevant relations and sampling tail entities, the model efficiently retrieves the correct college information. Finally, the system validates the retrieved results through cross-checking to ensure accuracy and consistency. This structured reasoning and validation pipeline demonstrates the system's ability to handle complex queries effectively.

B.2 Failure in Query Processing

Query Example: "Where is Whistler Mountain located in the Eastern Time Zone?"

In this case, LLM generates an incorrect hypothetical answer, incorrectly interpreting "where" to refer to a geopolitical entity (GPE) rather than a location (LOC). This erroneous hypothesis misguides the plug-in model during the classification phase, causing it to assign an incorrect category to the instruction entity. Consequently, the search process focuses on irrelevant GPE-related relations and tail entities within the knowledge graph, producing invalid candidate paths and failing to retrieve meaningful results.

This failure highlights a critical limitation of the system: its reliance on the LLM's ability to generate semantically accurate initial hypotheses. Errors at this stage propagate through the classification, relation exploration, and candidate generation phases, ultimately leading to an incorrect final answer.

B.3 Discussion

These cases demonstrate the dual challenges of semantic understanding and instruction-driven reasoning in structured query processing. In the first example, the system effectively aligns hypothesis generation, entity classification, and relation exploration with the instruction, enabling accurate information retrieval. However, the second example reveals a significant vulnerability: the system's

reliance on the LLM's initial hypothesis. An incorrect assumption, such as misclassifying 'Whistler Mountain,' propagates errors throughout the reasoning pipeline, leading to invalid candidate paths and incorrect answers.

To address this issue, future research should focus on developing mechanisms to validate LLM-generated hypotheses before classification and search. Integrating consistency checks between the LLM and plug-in models during early reasoning stages could reduce the propagation of errors. Furthermore, incorporating a feedback loop to iteratively refine hypotheses and classifications may improve robustness, ensuring that errors in the initial stage do not compromise the entire reasoning process.

C KG Exploration Algorithm

In this section, we present the detailed procedure of the Knowledge Graph (KG) exploration mechanism employed in our model, which involves iteratively sampling and classifying entities, while utilizing a self-reflection module for error correction. The key steps of the KG exploration process are outlined below in an algorithmic format.

The knowledge graph exploration process is detailed in Algorithm 2. To ensure clarity, we encapsulate complex operations into modular functions with descriptive names and clear input-output specifications.

The Algorithm 2 takes a knowledge graph $\mathcal{KG}$, an initial instruction I, and a center entity e_c as input, and outputs a set of candidate paths $\mathcal{P}$. The algorithm initializes a queue $\mathcal{Q}$ with the center entity and the initial instruction. Exploration proceeds iteratively, dequeuing the next entity-instruction pair for processing until the queue is empty.

At each step, the current entity e_c is classified based on the instruction I. For each relation r connected to e_c, the algorithm samples tail entities T_r, classifies them against the instruction I, and checks if the classifications match. If they match, the relation and its tail entities are added to the candidate paths $\mathcal{P}$.

Feedback mechanisms handle cases where the number of candidate paths exceeds a predefined threshold or when no candidates match the instruction's classification. In the former case, the algorithm triggers a self-reflection process to analyze the current subgraph and prune unnecessary paths. In the latter case, the algorithm reflects on the current subgraph to refine the exploration process.

Algorithm 2 Instruction-Driven KG Exploration

Input: Knowledge Graph $\mathcal{KG} = \{E, R, T\}$, Initial instruction I, Center entity e_c
Output: Candidate paths $\mathcal{P}$
$\mathcal{P} \leftarrow \emptyset$
$\mathcal{Q} \leftarrow \{(e_c, I)\}$
while $\mathcal{Q}$ is not empty **do**
  $(e_c, I) \leftarrow \textsc{Dequeue}(\mathcal{Q})$
  $C_I \leftarrow \textsc{Classify}(e_c, I, \mathcal{KG})$
  for each relation $r \in R(e_c)$ **do**
    $T_r \leftarrow \textsc{Sample}(\mathcal{KG}, e_c, r)$
    $C_r \leftarrow \textsc{Classify}(T_r, I, \mathcal{KG})$
    if $\textsc{Match}(C_r, C_I)$ **then**
      $\mathcal{P} \leftarrow \mathcal{P} \cup \textsc{AddPath}(e_c, r, T_r)$
    end if
  end for
  if $\textsc{TooMany}(\mathcal{P})$ **then**
    $\textsc{Reflect}(\mathcal{KG}, e_c, \mathcal{P})$
  else if $\textsc{NoMatch}(C_r, C_I)$ **then**
    $\textsc{Reflect}(\mathcal{KG}, e_c)$
  else
    $I_{\text{next}} \leftarrow \textsc{NextInstruction}(I)$
    if I_{next} exists **then**
      $\textsc{Enqueue}(\mathcal{Q}, (e_c, I_{\text{next}}))$
    end if
  end if
end while
Return: Candidate paths $\mathcal{P}$

If no issues arise, the algorithm retrieves the next instruction and enqueues the corresponding entity-instruction pair for further exploration. The process terminates when the queue is empty, returning the final set of candidate paths $\mathcal{P}$.

D Training Details

In order to enhance the accuracy and robustness of the classification process, we adopt a comprehensive strategy for representing entities in the knowledge graph. Each entity $e \in E$ is represented as a concatenated textual description derived from the neighboring triples in the graph. Specifically, we construct the representation by combining the subject, relation, and object from all associated triples, ensuring that the contextual information of the entity is fully captured. This representation serves as the input to the classification model. For the instruction entity, a textual representation is first generated by LLM based on the instruction context.

The resulting instruction-specific sentence is then concatenated with the corresponding triples to form the full context for entity classification.

The classification model is trained using a combination of original training data and additional samples generated through augmentation. To expand the training dataset, we utilize Freebase to sample new examples for entity classification tasks. The additional samples are designed to cover a broader range of entity types and scenarios, particularly those underrepresented in the original data. This augmented dataset ensures that the model learns from a more diverse and comprehensive set of examples, improving its generalization capabilities. Importantly, these newly sampled examples do not replace the original training data but complement it, thereby retaining the strengths of the initial dataset while addressing its limitations.

To address the challenge of distinguishing between closely related entity categories, we introduce subcategories for classes that are prone to confusion. These subcategories are designed based on the guidelines proposed in previous work (Dan et al., 2014), providing finer-grained distinctions that guide the model during training. For instance, if certain event entities (EVENT) exhibit overlapping characteristics, such as those involving sports and accidents, subcategories like "sports event" and "accident" are manually defined based on their contextual features derived from surrounding triples. This manual refinement addresses ambiguity and enhances the model's ability to differentiate between these subcategories. Relevant data is then incorporated into the training process to further improve the model's classification performance for these newly introduced subcategories.

The training process is structured to focus solely on minimizing the classification error between annotated entities and their corresponding ground-truth labels, ensuring that the model does not rely on predictions made during inference. This separation ensures a robust training process that is independent of potential biases introduced in the inference phase. Regularization techniques and learning rate schedules are employed to prevent overfitting, while the diversity of the training data helps improve the model's ability to generalize across various entity categories.

E Prompts

E.1 Decomposed Sub-questions

As shown in figure 6, this functionality involves decomposing a complex question into a series of logically connected sub-questions, following the reasoning path in the knowledge graph. The answers to these sub-questions collectively form the final answer to the original question. And

========== Prompt Input ==========

Given a complex question, decompose it into a series of sub-questions that follow the logical structure of the knowledge graph and the question, corresponding to the reasoning path in the knowledge graph.

The final answer of the decomposed sub-questions should be the answer to the original question.

For each step:
Write the sub-question corresponding to this step.
Use logical sequencing where the answer to one step becomes the subject for the next.

Example 1~n
<Example>

Question:
<Question>

========== LLM Output ==========

Initial Entity: <Initial Entity>

Step 1: Sub-question: <Sub-question>

Step 2: Sub-question: <Sub-question>

Figure 6: The prompt template for Decomposed Sub-questions

E.2 Instruction Generation

In figure 7, this functionality answers the decomposed sub-questions and provides specific instructions based on semantic information and the structure of the knowledge graph. This guides the reasoning process at each step.

E.3 Re-planning

As shown in figure 8, this functionality involves re-planning reasoning steps when a [problem] interrupts the current reasoning path in the knowledge graph. The subgraph representing the problem is given by [triple 1, triple 2, ..., triple n], and the goal is to restructure the reasoning steps to resolve the issue.

========== **Prompt Input** ==========

Using the decomposed sub-questions, answer each sub-question and provide specific instructions based on semantic information and the knowledge graph structure. Output the complete hypothetical answers and instruction actions.

In this process, instruction each step as either:
[Adverbial Qualifier]: Applies a specific condition or context to narrow the scope of an answer.
[Forward Search]: Identifies information by querying the knowledge graph directly.

For each step:
Instruction its type: [Adverbial Qualifier] or [Forward Search].
Write the sub-question.
Provide a hypothetical answer (based on typical responses).
Write the knowledge graph instruction or reasoning action to resolve the step.

#Example 1~n
<Example>

Sub-questions:
<Decomposed Sub-questions from Prompt 1>

========== **LLM Output** ==========

Step 1:
Sub-question: <Sub-question>
Hypothetical Answer: <Hypothetical Answer>
Knowledge Graph Instruction: <Instruction>

Step 2:
Sub-question: <Sub-question>
Hypothetical Answer: <Hypothetical Answer>
Knowledge Graph Instruction: <Instruction>

Figure 7: The prompt template for Instruction Generation

========== **Prompt Input** ==========

<Part of Decomposed Sub-questions prompts>
In the process of exploring a Knowledge Graph, you encounter a <problem> that hinders the current reasoning path. This problem is represented in the subgraph, which consists of the following triples:
<triple 1, triple 2 ... triple n>
Given this issue, you need to re-plan the reasoning steps and provide a new path to effectively address the problem.

Your original plan was:
<Original reasoning steps>

The problematic step is:
<Problematic step in the reasoning process>

For each step in the re-planning process, you need to:
- Identify the source of the issue and how it affects the reasoning flow.
- Reconstruct the sub-questions logically based on the problem and the subgraph information.
- Adjust the reasoning path to ensure that the answer to each sub-question logically leads to the next, ultimately resolving the original problem.

Ensure that the final answer to the re-planned sub-questions provides a solution to the <problem>.

#Example 1~n
<Example>

========== **LLM Output** ==========

Initial Entity: <Initial Entity>

Step 1: Sub-question: <Re-planned Sub-question>

Step 2: Sub-question: <Re-planned Sub-question>

Figure 8: The prompt template for Instruction Generation