ODA: Observation-Driven Agent for integrating LLMs and Knowledge Graphs

Anonymous ACL submission

Abstract

The integration of Large Language Models (LLMs) and knowledge graphs (KGs) has achieved remarkable success in various natural language processing tasks. However, existing methodologies that integrate LLMs and KGs often navigate the task-solving process solely based on the LLM's analysis of the question, overlooking the rich cognitive potential inherent in the vast knowledge encapsulated in KGs. To address this, we introduce Observation-Driven Agent (ODA), a novel AI agent framework tailored for tasks involving KGs. ODA incorporates KG reasoning abilities via global observation that enhances reasoning capabilities through a cyclical paradigm of observation, action, and reflection. Confronting the exponential explosion of knowledge during observation, we innovatively design a recursive observation mechanism. Subsequently, we integrate the observed knowledge into the action and reflection modules. Through extensive experiments, ODA demonstrates state-of-the-art performance on several datasets, notably achieving accuracy improvements of 12.87% and 8.9%.

1 Introduction

Large language models (LLMs) (Touvron et al., 2023; Scao et al., 2022; Muennighoff et al., 2022; Brown et al., 2020) have exhibited extraordinary capabilities across a variety of natural language processing tasks. Despite their impressive accomplishments, LLMs often struggle to provide accurate responses to queries that necessitate specialized expertise beyond their pre-training content. In response to this limitation, a natural and promising approach involves the integration of external knowledge sources, such as knowledge graphs (KGs), to augment LLM reasoning abilities. KGs provide structured, explicit, and explainable knowledge representations, offering a synergistic method to overcome the intrinsic constraints of LLMs. The fusion

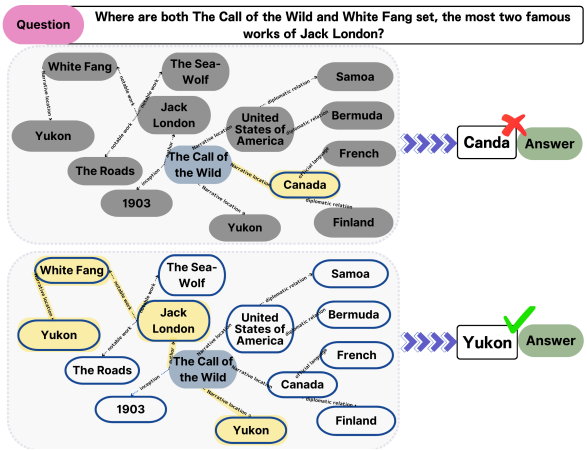

Figure 1: An example of LLM integrating KG. Observed entities are in white while non-observed entities are in gray. Entities selected by the agent for answering the question are highlighted in yellow.

of LLMs with KGs has garnered significant interest in recent research (Pan et al., 2024), underlying a vast array of applications (Zhang et al., 2023; Do et al., 2024; Sun et al., 2023b).

Existing methodologies for solving tasks that integrate KGs with LLMs can be categorized into two groups. The first one involves retrieving relevant triples from KGs in response to specific questions (Wang et al., 2023b; Jiang et al., 2023). The second part adopts an explore-exploit strategy, directing the knowledge utilization process within the graph according to the question (Sun et al., 2023b; Guo et al., 2023). However, both categories navigate the task-solving process by merely relying on the LLM's analysis of the question, overlooking the rich cognitive potential inherent in the abundant knowledge encapsulated in KGs. KGs, which store a wealth of informative and symbolic facts, should deeply participate in the reasoning process together with LLM rather than being merely treated as a static repository of knowledge (Pan et al., 2024). As the example in the upper panel of Figure 1, LLM analyzes the question and navigates towards *Narra-*

tive location relation of entity *The Call of The Wild*. However, this entity has many neighboring entities with that relation, leading LLM to incorrectly infer *Canada* as the answer. In contrast, the bottom panel demonstrates how KG provides key patterns that reveal both *The Call of The Wild* and *White Fang* share the location *Yukon*. If LLM could observe this information beforehand, it would precisely guide its reasoning process towards the correct answer (as shown in the bottom panel). Therefore, LLM should adopt an overall observation to incorporate the extensive knowledge and intricate patterns embedded within the KG. Achieving this objective presents two primary challenges: firstly, a global observation of the KG can result in an exponential growth in the number of triples. As shown in the upper panel of Figure 1, fully processing all 3-hop connections for *The Call of the Wild* is impractical. Secondly, the integration of such comprehensive observation into the existing reasoning paradigms of LLMs presents another challenge. How to combine the observation with the reasoning process of LLM matters for solving the tasks.

Motivated by this, we introduce a novel framework, the Observation-Driven Agent (ODA), aimed at sufficiently integrating the capabilities of both LLM and KG. ODA serves as an AI Agent specifically designed for KG-centric tasks. ODA engages in a cyclical paradigm of observation, action, and reflection. Within ODA, we design a novel observation module to efficiently draw reasoning patterns of KG. Our observation module avoids the problem of exponential growth of triples via recursive progress. This approach ensures ODA integrating abilities of KG and LLM while mitigating the challenges associated with excessive data in KG, improving the efficiency and accuracy. Following the observation phase, ODA takes action by amalgamating insights derived from LLM inferences with the observed KG patterns. ODA can perform actions of three distinct types: Neighbor Exploration, Path Discovery, and Answering. Subsequently, ODA reflects on its internal state, considering both the outcomes of its actions and the prior observations. This iterative process continues until ODA accomplishes the task at hand.

We conduct extensive experiments to testify to the effectiveness of ODA on four datasets: `QALD10-en`, `T-REx`, `Zero-Shot RE` and `Creak`. Notably, our approach achieved state-of-the-art (SOTA) performance compared to competitive baselines. Specifically, on `QALD10-en` and `T-REx` datasets, we observed remarkable accuracy improvements of 12.87% and 8.9% over competitive baselines. We conclude the contributions as follows:

- We propose ODA, an AI Agent tailored for KG-centric tasks. ODA conducts observation to incorporate the reasoning ability of KG.
- We further design action and reflection modules to integrate observation into LLM reasoning to exploit the capabilities of both sides.
- We conduct experiments on four datasets and achieve SOTA performances.

2 Methods

In this work, we aim to solve tasks associated with KG. Let q represent a user question. The task T can be defined as generating an answer Y given a question q, task-relevant entities $E = \{e_0, e_1, ..., e_k\}$, and a KG denoted as G. Formally, the task T can be expressed as:

$$T : (q, E), G \rightarrow Y$$

Employing an iterative approach, ODA tackles the challenges inherent in KG-centric tasks. In contrast to existing methods that couple LLMs and KGs and rely solely on analyzing the LLM's query, ODA integrates observed knowledge from the KG into the entire reasoning process, resulting in more informed decisions. To achieve this objective, our ODA system, illustrated in Figure 2, primarily comprises three key modules for task resolution:

- **Observation**: This module efficiently observes and processes relevant knowledge from the KG environment, generating an observation subgraph (denoted as O_i). By leveraging insights and patterns gleaned from the KG, this subgraph is incorporated into a reasoning LLM. This synergistic integration equips ODA with enhanced capabilities from both the LLM and KG, allowing it to tackle tasks more effectively.
- **Action**: Drawing upon both the global observation subgraph O_i and ODA memory (denoted as M), the action module,represented by a_i, strategically selects the most suitable action to execute on the KG, ensuring the accurate answering of the question.

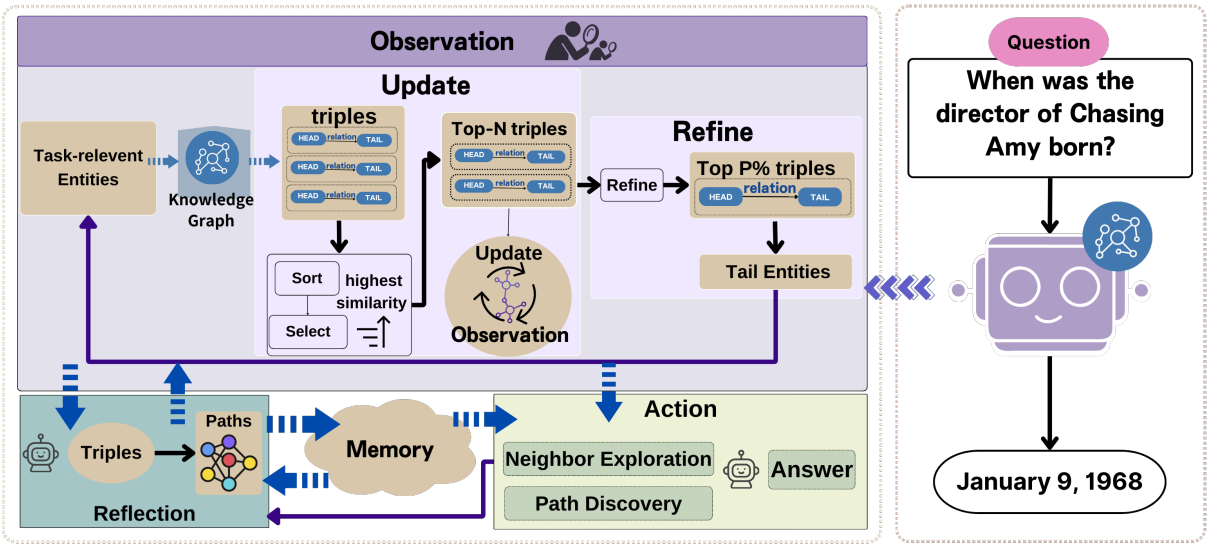

Figure 2: Diagram of ODA.

- **Reflection**: Utilizing the global observation subgraph O_i, the reflection module provides feedback by reflecting on the knowledge obtained from the action step. The reflected knowledge is then stored in memory M for the next iteration, facilitating continuous reasoning.

Through this iterative process, ODA dynamically updates both its observation subgraph O_i and memory M, enabling it to perform precise question-answering. Each module is discussed in detail in the following sections.

The observation module is specifically designed to inspect global KG knowledge and navigate the reasoning process with the KG environments. It can effectively guide the action module towards better decision-making. At each iteration i, it leverages task-relevant entities E_i and a question q to generate an observation subgraph O_i. This process can be formulated as:

$$O_i = \text{Observation}([E_i, q])$$

Initially, the task-relevant entities are populated with the entities embedded within the question q.

For KG-centric tasks, the global observation incurs the problem of an explosive number of nodes. To address the scalability challenge during observation subgraph updates, we propose a D-turn observe strategy, where D represents the maximum hop depth. Each turn has two steps: update and refine. The update step focuses on expanding the subgraph, while the refining step ensures its appropriate size without loss of information.

For each entity $e \in E_i$, the observation module initializes the observation entities as a set $E_o^d = \{e\}$ at hop depth d, where d represents the current search depth within the knowledge graph.The two-step update and refine process iterates for each entity e until D is reached. The specific details are described below:

Algorithm 1 Observation

Require: Question q, limit D, N, and P

```
Initialize task-relevant entities E_i with the entities in q
for e ∈ E_i do
    Set d = 0
    Initialize observation entities E_o^d = {e}
    while d < D do
        for entity ∈ E_o^d do
            Extract the neighboring triples
            for (r, t) ∈ triples do
                Cosine similarity(q, r + t)
            end for
            Sort(triples) using similarity scores
        end for
        Append top N triples to O_i
        Extract top P% triples from the top N triples
        Update E_o^d with t in top P% triples
        Increment d : d = d + 1
    end while
end for
```

2.1 Observation

Update:

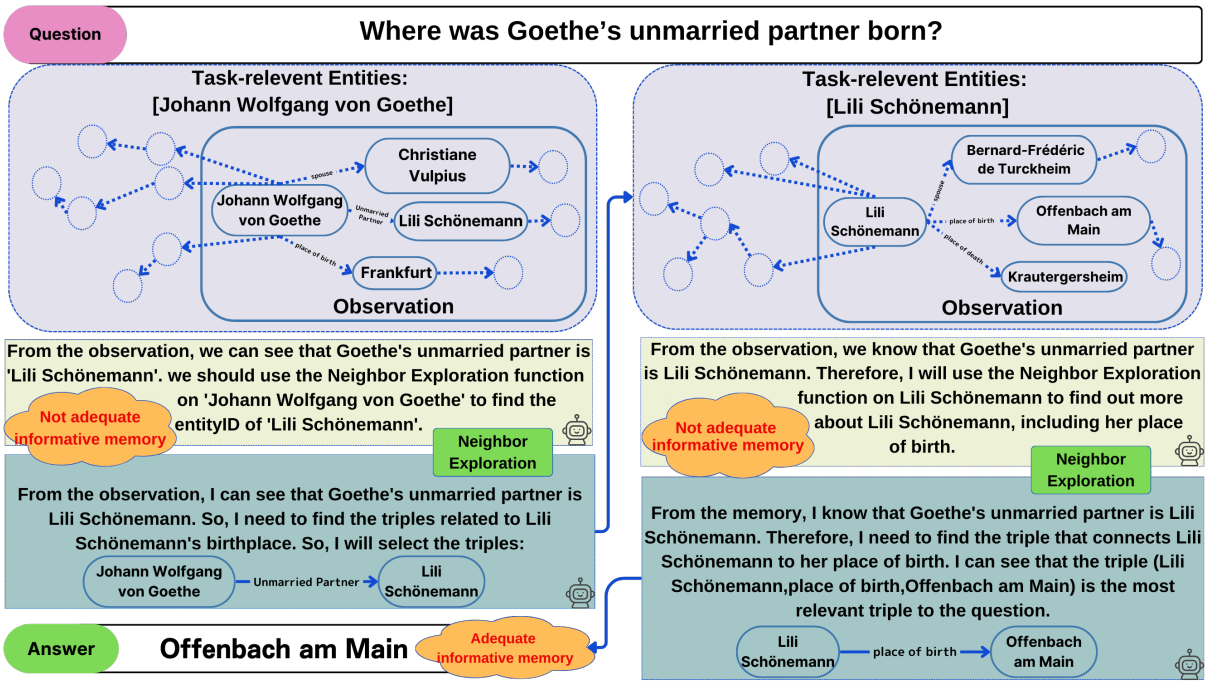

Figure 3: An example workflow of ODA. In this case, ODA initiates the obervation with entity *Johann Wolfgang von Goethe*. During the first iteration on the left side, the Neighbor Exploration of *Johann Wolfgang von Goethe* is selected, and the reflected triple *(Johann Wolfgang von Goethe, unmarried Partner, Lili Schöneman)* is stored in memory. Subsequently, The observation of *Lili Schöneman* then guides ODA to choose Neighbor Exploration action, and leads to the retention of the triple *(Lili Schöneman, place of birth, Offenbach am Main)* in memory, as shown on the right side. Once sufficient knowledge has been accumulated, ODA triggers the Answer action, correctly identifying *Offenbach am Main* as the answer.

- For each entity in E_o^d, neighboring triples are extracted from KG. Each triple takes the form $[e, r, t]$, where r signifies the relation, and t denotes the tail entity.

- The similarity score between the question and the combination of relation r and tail t in each triple, is measured using cosine similarity.[1]:

$$\text{Cosine Similarity}(\mathbf{v}_q, \mathbf{v}_{r+t}) = \frac{\mathbf{v}_q \cdot \mathbf{v}_{r+t}}{\|\mathbf{v}_q\| \|\mathbf{v}_{r+t}\|}$$

- The neighboring triples are sorted in descending order based on their similarity scores.

- Append the Top-N triples to the observation subgraph O_i.

Refine:

- The Top-N triples are further refined by retaining only the top $P\%$ with the highest similarity scores.

- The tail entities from the refined top $P\%$ triples are identified as the starting observation entities E_o^d for the next iteration.

Then we obtain the observation subgraph O_i for i^{th} turn. It is then utilized for the next modules.

2.2 Action

Within an AI Agent framework, memory plays a crucial role as a long-term storage component (Shinn et al., 2023; Yao et al., 2023). In our ODA , memory M is involved in guiding both the action and reflection modules. In its interaction with the action module, memory assists in selecting the most appropriate action, thereby facilitating successful task completion.

Harnessing the power of an LLM, the action module crafts strategic prompts to generate optimal actions. Based on its memory M, observation subgraph O_i, and historical actions $a_{<i}$, the ODA selects the most accurate action a_i.

$$a_i = \text{Action}([O_i, a_{<i}, M])$$

We propose three core actions designed to empower ODA :

[1]We use the GPT text-embedding-ada-002 model from OpenAI for encoding.

- **Neighbor Exploration**: This action explores the KG neighborhood of the entity in task-relevant entities E_i, retrieving all neighboring triples. This helps build context and understand interconnectedness within the KG for ODA .

- **Path Discovery**: Given two entities in task-relevant entities E_i, this action searches for all possible paths connecting them. Each path consists of interconnected triples, allowing the ODA to explore various connections and potentially uncover hidden relationships.

- **Answer**: This action responds to the question only if the required information is present in memory M.

Upon selecting an answer action, ODA halts the iterative loop of observation, action, and reflection. Leveraging the reliable knowledge within memory M, it can then directly formulate the answer to the question. Alternatively, if a Neighbor Exploration or Path Discovery action is selected, ODA strategically extracts relevant knowledge from the KG as a set of triples. These extracted triples are then fed into the subsequent reflection step for further processing.The prompt used here can be found in Table 7.

2.3 Reflection

The reflection module plays a crucial role in evaluating the triples generated from the action step and subsequently updating ODA memory M. Designed specifically for KG tasks, memory M adopts a format comprising a network of paths that align with the innate structure of KGs, aimed at optimizing efficiency and relevance. Each path within M holds a set of triples. By strategically integrating the observed subgraph O_i and existing memory M, the reflection module provides invaluable feedback, guiding future decision-making. This process can be formalized as:

$$M = \text{Reflection}([O_i, a_i, M])$$

Given that memory M is structured as a network of paths, the reflection module navigates these paths to identify the first suitable one for integrating the new triple. This suitability stems from the alignment of the last triple's tail t within the chosen path with the new triple's entity e. If a matching path exists, the triple is appended. Otherwise, a new path is created within the memory subgraph.The specific prompt text used for the reflection module is provided in Table 8.

The observation, action, and reflection modules collaborate in an iterative process until either the Answer action is triggered or the maximum iteration limit is reached. Figure 3 depicts an example of how observation, action, and reflection work together.

3 Experiments

Dataset	Test	Entity	Type	License
QALD10-en	333	396	Multi-hop	MIT License
T-REx	5000	4943	Slot-Filling	MIT License
Zero-Shot RE	3724	3657	Slot Filling	MIT License
Creak	1371	516	Fact Checking	MIT License

Table 1: Dataset statistics. **Entity** stands for the entity derived from the question within the datasets.

3.1 Dataset

To evaluate the performance of our ODA , we conduct experiments on four KBQA datasets: QALD10-en (Perevalov et al., 2022),the fact-checking dataset Creak (Onoe et al., 2021), T-REx (Elsahar et al., 2018), and Zero-Shot RE (Petroni et al., 2021). We evaluate ODA on the test set for each dataset listed in Table 1. The Hits@1(Sun et al., 2019) accuracy with exact match is utilized as our evaluation metric.

3.2 Setup

We utilized the GPT-4 (OpenAI, 2023) model as the ODA via the OpenAI API. Throughout our experiments, we consistently configured the temperature value of GPT-4 to 0.4 and set the maximum token length to 500.

For the observation step, we set P_t to 10 and N_t to 50. Furthermore, the ODA loop was capped at a maximum of 8 iterations. Lastly, the maximum hop depth D is set to 3.

To establish Wikidata KG database and retrieve information from it, we employed the *simple-wikidata-db*[2] Python library. This library provides various scripts for downloading the Wikidata dump, organizing it into staging files, and executing distributed queries on the data within these staged files. Specifically, we deployed the Wikidata dump across five AWS EC2 instances, each consisting of a 768GB machine with 48 cores.

[2]https://github.com/neelguha/simple-wikidata-db

Considering that our ODA relies heavily on continuous interaction with the KG, we discovered that the real-time extraction of required Wikidata knowledge on AWS achieved an average completion time of 50 seconds per question-answer pair within QALD10-en dataset. However, as the KGQA dataset expanded, the cost of using the Wikidata database on AWS became prohibitively expensive. Consequently, to address the computational expenses involved, we devised a solution by generating an offline subgraph for each KGQA dataset. This offline subgraph captures all the triples within a 3-hop radius of the entities in each dataset, including the properties of both the entities and the relations involved. Notably, generating such a subgraph for the T-REx dataset, with its 4943 entities (as listed in Table 1), takes approximately 54 minutes and 42.834 seconds in practice.

3.3 Baseline Models

To comprehensively evaluate ODA effectiveness, we conduct a rigorous benchmark against several SOTA models across diverse categories. The comparison encompasses various models, starting with prompt-based approaches that do not utilize external knowledge. These include direct answering with GPT-3.5 and GPT-4, as well as the Self-Consistency (Wang et al., 2023c) and CoT (Sun et al., 2023b). On the other hand, Kownledge-combined models are considered, which incorporate fine-tuned techniques such as SPARQL-QA (Santana et al., 2022), RACo (Yu et al., 2022), RAG (Petroni et al., 2021) and Re2G (Glass et al., 2022). Additionally, there is ToG (Sun et al., 2023a) model, which integrates LLM with KG to bolster question-answering proficiency.

3.4 Main Result

Our ODA method outperforms existing methods, as shown in Table 2. On average, our method achieves an accuracy gain of up to 19.58% compared to direct answering with GPT-4, 19.28% compared to fine-tuned models, and 7.09% compared to TOG. These results demonstrate the efficiency and effectiveness of our method in comparison to other state-of-the-art methods.

Furthermore, our ODA significantly outperforms the prompt-based methods across various datasets, particularly showing an improvement of 65.50% and 23.77% on Zero-Shot REx and QALD10-en , respectively. These results underscore the importance of leveraging external knowledge graphs for reasoning and completing the question-answering task.

Compared to the fine-tuned method, our ODA method demonstrates superior performance. Specifically, our method achieves a performance gain of 21.27% for the QALD10-en dataset, 6.99% for the Creak dataset, and 50.56% for the Zero-Shot RE dataset. Notably, this interaction between the LLM and KG, as our method employs, proves more effective than data-driven fine-tuned techniques, despite requiring no explicit training.

Our ODA method exhibits significant performance gains over the TOG method across most datasets, with improvements of 12.87% (QALD10-en), 8.9% (T-REx), and 7% (Zero-Shot RE), despite both methods leveraging large language models and knowledge graphs. This performance disparity highlights the critical role of our observation module and the effectiveness of deeply incorporating reasoning from KG. Specifically, our method demonstrates significantly stronger performance on the QALD10-en dataset, known for its multi-hop and complex reasoning requirements. This achievement underscores our ODA ability to effectively exploit the rich knowledge and patterns within KG, seamlessly combining the reasoning strengths of both LLM and KG to tackle complex questions successfully.

4 Discussion

To better understand the key factors influencing our ODA, we conducted extensive analysis experiments. To conserve computational resources, we kept the previously mentioned datasets (QALD10-en, Creak, T-REx, and Zero-Shot RE) but randomly sampled 400 examples each for Creak, T-REx, and Zero-Shot RE.

4.1 Effect of Observation

To assess the efficacy of the observation module, we conducted comprehensive experiments with the model without observation. During the action step, ODA selects the action only based on the knowledge within memory. Subsequently, the reflection step reflects on the triples outputted by the action and updates memory without the guide from observation.

A statistical comparison was performed to evaluate the performance of the ODA with and without observation across all datasets (see Table 3). The results show that the ODA with observation out-

Method	QALD10-en	Creak	T-REx	Zero-Shot RE	Average
Without external knowledge					
Direct answering(GPT3.5)	44.74	90.00	37.78	37.14	52.42
Direct answering(GPT4)	57.10	94.52	57.72	55.50	66.21
Self-Consistency(GPT3.5)(Wang et al., 2023c)	45.30	90.80	41.80	45.40	55.83
COT(GPT3.5)(Sun et al., 2023a)	42.90	90.10	32.00	28.80	48.45
Leveraging external knowledge					
Fine-tuned SOTA	45.40(1)	88.20(2)	87.70(3)	44.74(4)	66.51
TOG-R(GPT4)(Sun et al., 2023a)	54.70	95.40	75.50	86.90	78.13
TOG(GPT4)(Sun et al., 2023a)	53.80	**95.60**	77.10	88.30	78.70
ODA (GPT-4)	**66.67**	95.19	**86.00**	**95.30**	**85.79**

Table 2: Performance Comparison of different methods. The fine-tuned SOTA includes:(1)(Santana et al., 2022),(2)(Yu et al., 2022),(3)(Glass et al., 2022),(4)(Petroni et al., 2021)

performs the ODA without observation, with an average improvement of 3.14%. Specifically, for QALD10-en dataset, the ODA with observation outperforms the ODA without observation by 5.41%. Since QALD10-en involves multi-hop reasoning, the improved performance of the ODA with observation indicates that the observation module enhances the reasoning ability of the agent, enabling more accurate action selection and reflection.

We can further illustrate the benefits of the observation module with a practical case. In this scenario (see Table 5), question is *Where are both The Call of the Wild and White Fang set, the most two famous works of Jack London?*. Without observation, ODA generated the memory, such as *(The Call of the Wild, narrative location, Canada)*, ultimately produced the wrong answer of *Canada*. However, with the observation module, the ODA correctly reasons the memory,such as *(The Call of the Wild, narrative location, Yukon), (White Fang, narrative location, Yukon)*. As a result, the ODA with observation provides the correct answer, *Yukon*. This case exemplifies how the observation module improves the accuracy of action selection and reflection, consequently enhancing the reasoning ability of ODA .

By incorporating observation information, ODA reasoning power undergoes a dramatic leap, therefore generate an accurate answers. This boost stems from the synergistic interplay between the observation module, harnessing the KG's inherent reasoning capabilities, and LLM, which further amplifies those strengths.

4.2 Effect of Observation on Reflection

In this section, we discuss the impact of observation on reflection module. Three non-observation reflection methods were designed to verify whether observation can enhance the effectiveness of reflection. The similarity-based involves reflecting on the triples from action steps by calculating similarity. In this approach, triples are first sorted based on the similarity score between the $r + t$ and the question. The top-k triples are then selected and stored in memory for the next iteration. The random-based method randomly picks k triples from the action's output and stores them in memory. Finally, the generated-fact method creates k natural language question-related facts for storage. All methods use $k = 15$.

Table 3 showcases our ODA dominance over all three non-observation methods. It achieved an average accuracy increase of 2.48% compared to the similarity-based method, 6.00% compared to the random-based method, and 3.66% compared to the generated-fact method.

In specific cases (see Table 6), when answering the question *What is the capital of the prefecture Tokyo?*, the generated-fact method resulted in problematic facts, such as *Tokyo is the capital of Tokyo,Tokyo is the capital of Japan*. These were essentially hallucinations created by the LLM based on the given question. They misled the agent resulting in incorrect answers. In contrast, the reflection of our ODA leveraging observation yielded factual knowledge, *(Tokyo, capital, Shinjuku)*, consequently enabling the ODA to answer the question correctly.

The findings of Table 3 reveal that observation enables reflection module to generate more ac-

Method	QALD10-en	Creak	T-REx	Zero-Shot RE	Average
Without Observation	61.26	95.50	82.00	91.75	82.63
Similarity-based Reflection	61.26	95.20	83.20	93.50	83.29
Random-based Reflection	58.56	89.00	79.50	92.00	79.77
Generated-fact Reflection	63.66	91.00	80.00	93.75	82.10
ODA	**66.67**	**96.00**	**85.40**	**95.00**	**85.77**

Table 3: Ablation Comparison

curate memories, which translates to improved question-answering accuracy for ODA. This result underscores the value of both leveraging KG reasoning capabilities and fostering deep collaboration between KG and LLMs.

4.3 Performance across Different Backbone Models

We evaluated the effectiveness of ODA across various backbones by analyzing its impact on T-REx and QALD10-en datasets. Our ODA approach significantly outperformed the direct answering methods using GPT-3.5 and GPT-4, as evidenced by the Table 4. Notably, ODA yielded a remarkable 30.4% improvement with the direct answering using GPT-3.5 model on QALD10-en dataset. This experiment suggests the generalizability of ODA across different LLMs.

Method	T-REx	QALD10-en
Direct answering(GPT3.5)	37.60	44.74
ODA (GPT3.5)	**68.00**	**49.71**
Direct answering(GPT4)	57.44	57.10
ODA (GPT4)	**86.00**	**66.67**

Table 4: Performance comparison using different backbone models

5 Related Works

KG-enhanced LLM Knowledge Graph-enhanced Language Models utilize two primary methodologies when tackling tasks that require integration with KGs. The first involves the extraction of relevant triples from KGs in response to posed questions. Wang et al. (2023b) prompt LLMs to generate explicit knowledge evidence structured as triples. Similarly, Jiang et al. (2023) develop specialized interfaces for gathering pertinent evidence from structured data, thus enabling LLMs to focus on reasoning tasks based on this information. Baek et al. (2023) retrieve facts related to the input question by assessing semantic similarities between the question and associated facts, subsequently prepending these facts to the input. Meanwhile, Li et al. (2023) iteratively refine reasoning rationales by adapting knowledge from the KG. Wang et al. (2023a) dissect complex questions using predefined templates, retrieve entities from the KG, and generate answers accordingly.

The second approach employs an explore-exploit strategy that guides the knowledge utilization process within the graph. Sun et al. (2023b) perform an iterative beam search on the KG to identify the most promising reasoning pathways and report the outcomes. Guo et al. (2023) selectively accumulate supporting information from the KG through an iterative process that incorporates insights from the LLM to address the question.

AI Agent In the domain of AI agents, Yao et al. (2022) utilize LLMs to interleave the generation of reasoning traces with task-specific actions. Wu et al. (2023) design an agent framework that is adaptable, conversational, and capable of operating in various modes that leverage combinations of LLMs, human input, and auxiliary tools. Chen et al. (2023) focus on creating expert agents capable of solving complex tasks.

6 Conclusion

In this work, we design ODA framework for KG-centric tasks. In ODA, we introduce KG observation mechanism to combine the reasoning abilities of KG with LLM. We first propose the observation method to mitigate the problem of explosive number of triples in KG when tackling complex tasks. Then we fuse the observation into the action and reflection modules to further enhance the overall performance. We conduct extensive experiments, and the results clearly illustrate the effectiveness of our framework, highlighting its capability to enhance performance across four KBQA datasets, particularly in handling complicated questions.

Limitation

Given the diverse nature of KG-related tasks spanning multiple domains and requiring a broad range of capabilities, the implementation of a multi-agent system is promising to significantly enhance task performance. We leave the integration of our observation mechanism with multi-agent sysmtem to future work.

References

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

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.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. 2023. Autoagents: A framework for automatic agent generation. *arXiv preprint arXiv:2309.17288*.

Quyet V Do, Tianqing Fang, Shizhe Diao, Zhaowei Wang, and Yangqiu Song. 2024. Constraintchecker: A plugin for large language models to reason on commonsense knowledge bases. *arXiv preprint arXiv:2401.14003*.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2G: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715, Seattle, United States. Association for Computational Linguistics.

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*.

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

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2023. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *arXiv preprint arXiv:2305.13269*.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

Yasumasa Onoe, Michael J. Q. Zhang, Eunsol Choi, and Greg Durrett. 2021. Creak: A dataset for commonsense reasoning over entity knowledge.

R OpenAI. 2023. Gpt-4 technical report. *arXiv*, pages 2303–08774.

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*.

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.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Manuel Alejandro Borroto Santana, Bernardo Cuteri, Francesco Ricca, and Vito Barbara. 2022. SPARQLQA 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 (ESWC 2022)*, volume 3196 of *CEUR Workshop Proceedings*, pages 25–31, Hersonissos, Greece. CEUR-WS.org.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

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

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

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space.

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*.

Chaojie Wang, Yishi Xu, Zhong Peng, Chenxi Zhang, Bo Chen, Xinrun Wang, Lei Feng, and Bo An. 2023a. keqing: knowledge-based question answering is a nature chain-of-thought mentor of llm. *arXiv preprint arXiv:2401.00426*.

Jianing Wang, Qiushi Sun, Nuo Chen, Xiang Li, and Ming Gao. 2023b. Boosting language models reasoning with chain-of-knowledge prompting. *arXiv preprint arXiv:2306.06427*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023c. Self-consistency improves chain of thought reasoning in language models.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2023. Retroformer: Retrospective large language agents with policy gradient optimization.

Wenhao Yu, Chenguang Zhu, Zhihan Zhang, Shuohang Wang, Zhuosheng Zhang, Yuwei Fang, and Meng Jiang. 2022. Retrieval augmentation for commonsense reasoning: A unified approach. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4364–4377, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yichi Zhang, Zhuo Chen, Wen Zhang, and Huajun Chen. 2023. Making large language models perform better in knowledge graph completion. *arXiv preprint arXiv:2310.06671*.

A Case Study

To demonstrate ODA effectiveness, we highlight several representative cases from the `QALD10-en` dataset. For improved readability, we replaced entity IDs in Outputs with their corresponding names. We selected two examples to illustrate how ODA utilizes observation to correctly address the question in Table 5 and Table 6.

Question	*Where are both The Call of the Wild and White Fang set, the most two famous works of Jack London?*
Action	**Thought**: The question asks about the setting of two works, The Call of the Wild and White Fang, both by Jack London. The observation provides some information about the narrative locations of these works, but to confirm and provide a specific answer, I will use the GetNeighbor function on The Call of the Wild. **Action**: GetNeighbor **Entity_id**: *The Call of the Wild*
Reflection	**Thought**: First, I will focus on the triples related to the locations of the two works, The Call of the Wild and White Fang, as the question is asking about the settings of these two novels. I will also consider the triples related to the author, Jack London, as the question mentions him. I will select the triples that are most directly related to the settings of the two novels. **Triples**: *(The Call of the Wild, Narrative location, Yukon), (The Call of the Wild, Narrative location, Canada), (The Call of the Wild, author, Jack London)*
Action	**Thought**:The question asks for the setting of both The Call of the Wild and White Fang, two notable works of Jack London. From the observation, we can see that The Call of the Wild has narrative locations in the United States of America, Alaska, Canada, and Yukon. However, we don't have the narrative location for White Fang. Therefore, I will use the GetNeighbor function on the entityID of White Fang to find its narrative location. **Action**: GetNeighbor **Entity_id**: *White Fang*
Reflection	**Thought**:First, I will look for triples that directly answer the question. The question is asking for the setting of two works by Jack London, namely, The Call of the Wild and White Fang. Therefore, I will look for triples that contain these two works and a relation that indicates a setting or location. From the candidate triples, I can see that there are two triples that indicate a narrative location for White Fang: (White Fang, Narrative location, Yukon), (White Fang, Narrative location, Canada). **Triples**: *(White Fang, Narrative location, Yukon), (White Fang, Narrative location, Canada)*
Action	**Thought**:The question is asking for the locations where both The Call of the Wild and White Fang are set. From the reference memory, I can see that The Call of the Wild is set in the United States of America, Alaska, Canada, and Yukon. White Fang is set in Canada and Yukon. The common locations for both books are Canada and Yukon. **Answer**: [*Canada*,*Yukon*]

Table 5: Case 1

Question	*The question is What is the capital of the prefecture Tokyo ?*
Action	**Thought**: The question is asking for the capital of the prefecture Tokyo. The candidate entity ID 'Q1490' corresponds to Tokyo. I can see from the observation that there is a triple (Tokyo, capital, Shinjuku) which might answer the question. However, to confirm this, I will execute a GetNeighbor action on 'Q1490' to get all the triples where Tokyo is the head. **Action**: GetNeighbor **Entity_id**: *Tokyo*
Reflection	**Thought**: The question is asking for the capital of Tokyo. From the observation, we can see that Tokyo is the capital of Japan and it is a prefecture of Japan. The capital of Tokyo is Shinjuku. Therefore, we should select the triples that contain this information. **Triples**: *(Tokyo, instance of, prefecture of Japan), (Tokyo, capital , Shinjuku), (Tokyo, capital, Japan)*
Action	**Thought**:The question is asking for the capital of the prefecture Tokyo. From the reference memory, it is stated that the capital of Tokyo is Shinjuku. Therefore, the answer to the question is Shinjuku. **Answer**: *Shinjuku*

Table 6: Case 2

B Prompt

This section presents the prompts for the action and reflection modules in tables 7 and 8.

Action	Prompt
Neighbor Exploration & Path Discovery	GetNeighbor(entityID: str) -> List[Tuple[str, str, str]]: Description: Returns triplets containing the given entityID as the head and its corresponding entityID as the tail. GetPath(entityID1: str, entityID2: str) -> List[List[Tuple[str, str, str]]]: Description: Returns all triplets linking the two given entityIDs. Example Usage:GetPath("Q30", "Q25231") returns all triplets connecting 'Q30' and 'Q25231'.
Answer	You are a agent that answer questions based on the reference memory and your knowledge. Here are the reference memory:[Memory]. You can use it to help you answer the quesiton. Here is the question you are asked to answer the question:[Question]. Ensure that your answer contains one answer or a list of answer, and each answer should be only one or several words,a phrase, a number,true or false, or a date, no other information or descripation in answer.

Table 7: Action Prompt Description

Field	Prompt
Reflection	You are an agent that provides answers based on a KG. You queried some triples [triples] from last action step and their corresponding labels:[entities labels] from the KB based on the question: [Question]. Now you are asked to select related triples, so you can answer the question in the future by using them. Here are the observation: [Obervation] for guiding you to select the right triples from the candidate triples. Also, here is the memory: [Memory]. You can use it to help you select the right triples from the candidate triples.

Table 8: Reflection Prompt Description