KG-Agent: An Efficient Autonomous Agent Framework for Complex Reasoning over Knowledge Graph

Jinhao Jiang[1], Kun Zhou[2], Wayne Xin Zhao[1]*, Yang Song[3]*, Chen Zhu[4], Hengshu Zhu[4], Ji-Rong Wen[1]

[1]Gaoling School of Artificial Intelligence, Renmin University of China.
[2]School of Information, Renmin University of China.
[3]Nanbeige Lab, BOSS Zhipin. [4]Career Science Lab, BOSS Zhipin.
jiangjinhao@ruc.edu.cn, batmanfly@gmail.com

Abstract

In this paper, we aim to improve the reasoning ability of large language models (LLMs) over knowledge graphs (KGs) to answer complex questions. Inspired by existing methods that design the interaction strategy between LLMs and KG, we propose an autonomous LLM-based agent framework, called **KG-Agent**, which enables a small LLM to actively make decisions until finishing the reasoning process over KGs. In KG-Agent, we integrate the LLM, multifunctional toolbox, KG-based executor, and knowledge memory, and develop an iteration mechanism that autonomously selects the tool and then updates the memory for reasoning over KG. To guarantee the effectiveness, we leverage program language to formulate the multi-hop reasoning process over the KG and synthesize a code-based instruction dataset to fine-tune the base LLM. Extensive experiments demonstrate that only using 10K samples for tuning LLaMA2-7B can outperform competitive methods using larger LLMs or more data, on both in-domain and out-domain datasets. Our code and data will be publicly released.

1 Introduction

Despite the remarkable performance on various NLP tasks (Brown et al., 2020; Zhao et al., 2023), large language models (LLMs) still have limited capacities in solving complex tasks (Hu et al., 2023b) solely based on their parametric knowledge, *e.g.,* multi-hop and knowledge-intensive reasoning (Lan et al., 2023). Knowledge graph (KG), which stores massive knowledge triples in a graph-structured format, has been broadly used to complement LLMs with external knowledge (Pan et al., 2023; Gu et al., 2021; Fang et al., 2024).

Due to the large volume and structured format of KG data, it is not easy for LLMs to effectively utilize the information from KG. Recent work mainly adopts *retrieval-augmented* (Ye et al., 2022) or *synergy-augmented* (Jiang et al., 2023b) methods to enhance LLMs with KG data. The former approach retrieves and serializes the task-related triples as part of the prompt for LLMs, while the latter approach designs an information interaction mechanism between KG and LLMs to iteratively find the solution to the question. In particular, synergy-augmented methods can benefit from the structured search on KG (*e.g.,* SPARQL) and the language understanding capacity of LLMs, achieving comparable or even better performance compared with previous state-of-the-art methods (Gu et al., 2023).

However, there are still two major limitations on existing synergy-augmented methods. First, the information interaction mechanism between LLM and KG is often pre-defined (*e.g.,* following a human-crafted multi-round plan), which cannot flexibly adapt to various complex tasks (Luo et al., 2023; Jiang et al., 2023b). For instance, it would become ineffective to handle the unintended requirements in the reasoning process, *e.g.,* varied difficulties or constraints. Second, these methods (Wang et al., 2023a) mostly rely on stronger closed-source LLM APIs (*e.g.,* ChatGPT and GPT-4) to solve complex tasks. However, the distilled plans or procedures, also limited to special task settings or capacity levels, may not be best suited for instructing these weaker models.

To address these issues, in this paper, we propose the **KG-Agent**, an autonomous LLM-based agent framework for complex reasoning tasks over KG. The motivations are twofold: (1) designing autonomous reasoning approaches that can actively make decisions during reasoning, without human assistance; (2) enabling relatively small models (*e.g.,* 7B LLM) to effectively perform complex reasoning, without reliance on close-sourced LLM APIs. To achieve this, our approach makes three major technical contributions. First, we extend the LLM's capacity to manipulate structured data by curating a multifunctional toolbox, enabling LLM

* Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9505–9523
July 27 - August 1, 2025

to perform discrete or advanced operations (*e.g.,* filtering, counting, and retrieval) on KG data and intermediate results. Second, we leverage existing KG reasoning datasets for synthesizing code-based instruction data to fine-tune the LLM, where we first generate the program according to the reasoning chain on KG and then synthesize the instruction data. Third, we propose an autonomous iteration mechanism based on tool selection and memory updation that integrates the tuned LLM, multifunctional toolbox, KG-based executor, and knowledge memory, for autonomously reasoning over KG.

Our extensive evaluation results on both in-domain and out-of-domain tasks (*i.e.,* KG-based question answering (KGQA) and open domain question answering (ODQA) affirms the effectiveness of our KG-Agent. We consolidate our contributions and results as follows:

• **Autonomous and General KG Agent.** To the best of our knowledge, KG-Agent is the first method to develop an autonomous agent using a relatively small LLM (7B).

• **Efficient Training and Inference.** KG-Agent is trained on only 10K data (*e.g.,* 22.6% of GrailQA), and inference faster (*e.g.,* nearly 3× inference speed compared to StructGPT).

• **Strong Performance.** KG-Agent performs the best across competitive methods on both in-domain and out-of-domain datasets, achieving a 7.5% relative improvement in F1 on CWQ compared to ReasoningLM, and an 8.5% relative improvement in accuracy on TQ-Wiki compared to BART-Large.

2 Preliminary

Knowledge Graph (KG). A knowledge graph typically consists of a large number of fact triples, expressed as $\mathcal{G} = \{\langle e, r, e' \rangle | e, e' \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the entity set and relation set, respectively. A triple $\langle e, r, e' \rangle$ describes a factual knowledge that a relation r exists between the head entity e and tail entity e'. Each entity e is assigned a unique entity ID (or string value), and belongs to one entity type t such as *Country* and *Person*. Furthermore, we introduce *neighboring relations* to denote both the incoming and outgoing relations for a set of entities $\{e\}$, denoted as $\mathcal{R}_{\{e\}} = \{r | \langle e, r, e' \rangle \in \mathcal{G}\} \cup \{r | \langle e', r, e \rangle \in \mathcal{G}\}$.

Problem Formulation. In this work, we assume that a KG is available and contains the answer entities for the given natural language question. Our objective is to develop a LLM-based agent that can autonomously infer the answer to the question based on the given KG. As it has been shown that domain-specific interface is helpful for LLMs to manipulate the structured data (Jiang et al., 2023b), we further assume that a toolbox can be provided to facilitate the access to the information of KG. Formally, given a natural language question q, and a toolbox $\mathcal{T}$ and a KG $\mathcal{G}$, we aim to develop a capable agent to deduce the final answers $A_q = \{e\}$ for the question q by leveraging the tools in $\mathcal{T}$ and the knowledge information in $\mathcal{G}$.

3 Approach

In this part, we present the proposed **KG-Agent** for autonomously solving complex reasoning tasks over KG. The core of our KG-Agent framework is a well-instructed LLM, which can autonomously make decisions when reasoning over KG. We first extend the LLM's capacities by designing a toolbox with supporting tools to manipulate the KG data or intermediate results (Section 3.1). To enhance the step-by-step reasoning capacity, we leverage existing knowlege graph question answering (KGQA) datasets to synthesize KG reasoning programs and convert them into formatted instruction tuning data (Section 3.2). Finally, we design an effective agent framework based on the knowledge memory to support autonomous reasoning over KG (Section 3.3). Next, we give the technical details of KG-Agent.

3.1 Toolbox for Knowledge Graph

Since LLMs struggle to accurately manipulate the structured data (Jiang et al., 2023b), we construct a supporting toolbox for easing the utilization of the KG information. According to existing work (Gu et al., 2021; Cao et al., 2022), reasoning over KG (*e.g.,* Freebase or Wikidata) typically requires three fundamental operations, namely extracting information from KG, filtering irrelevant information based on the semantics of the question, and operating on the extracted information. Therefore, we design three types of tools for LLMs reasoning over KG, *i.e.,* extraction, semantic, and logic tools.

• **Extraction tools** aim to facilitate the access to information from KG. Considering the basic data types in KG, we design five tools to support the access to the relations (*get_relation*), the head/tail entities (*get_head_entity*/*get_tail_entity*), and entities with specific type or constraint (*get_entity_by_type*/ *get_entity_by_constraint*), *w.r.t.* some entity set or

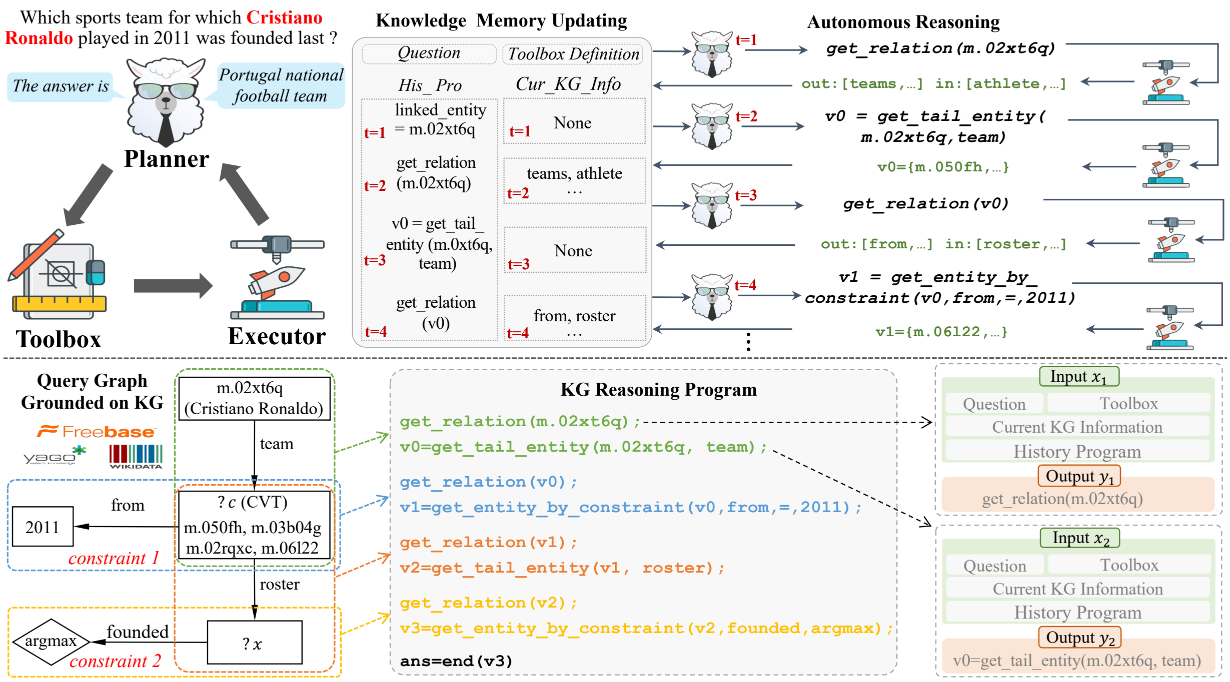

Figure 1: The overview of our proposed KG-Agent. The top half is the workflow of our agent, and the bottom half is an example of instruction fine-tuning data synthesis and the prompt template for the input-output pairs. For brevity, we simplify the relation surface form.

other input information (*e.g.,* relation or type).

• **Logic tools** aim to support basic manipulation operations on the extracted KG information, including entity counting (*count*), entity set intersection (*intersect*) and union (*union*), condition verification (*judge*), and ending the reasoning process with the current entity set as the final answer(s) (*end*).

• **Semantic tools** are developed by utilizing pre-trained models to implement specific functions, including relation retrieval (*retrieve_relation*) and entity disambiguation (*disambiguate_entity*). These tools extend the basic operations on KGs and can support advanced functionalities for KG reasoning.

We summarize the detailed definition and usage of the tools in Table 9 at the Appendix C. Note that since the format and usage for each tool have been defined in a unified way, the toolbox for KG can be flexibly extended according to the real needs.

3.2 KG-Agent Instruction Tuning

To enable the autonomous reasoning process, we construct a high-quality instruction dataset for fine-tuning a small LLM (*i.e.,* LLaMA2-7B). For this purpose, we first leverage existing KG based question answering (KGQA) datasets to generate the KG reasoning program, and then decompose it into multiple steps. Finally, each step is formulated as the instruction data with input and output.

3.2.1 KG Reasoning Program Generation

Instead of distilling from close-sourced LLMs (*e.g.,* GPT-4), we propose to leverage existing KGQA datasets to synthesize the KG reasoning program. These KGQA datasets contain the annotated SQL queries that can be executed to directly extract the answer entities for each question. In particular, the SQL query generally includes the relation chain, conditions, or constraints, which are beneficial for reasoning program synthesis. Concretely, we first ground the SQL query on the KG to obtain a query graph, then extract the reasoning chain and constraint conditions from the query graph, and finally decompose the chain into multiple code snippets as the reasoning program.

Reasoning Chain Extraction. Since the whole KG is extremely large and contains irrelevant data, the first step is to acquire a small KG subgraph related to the question, referred to as *query graph*. Following previous work (Yin et al., 2020), we obtain the query graph from the KG via rule match. As shown in Figure 1 (b), the query graph has a tree-like structure that can be directly mapped to a logical form (Yin et al., 2020), and it can clearly depict the execution flow of the SQL query to obtain the answer. Second, starting from the mentioned entity in the question (*i.e., Cristiano Ronaldo*), we adopt breadth-first search (BFS) to visit all the nodes on the query graph. This strategy would finally produce a reasoning chain (*e.g., teams→roster_team*) linking the start entity to the answer entity, and the relevant constraint conditions (*e.g., roster_from* = "2011") or numerical operation (*e.g., founded* must be last) can be naturally involved in this process.

Reasoning Program Generation. After extracting the reasoning chain, we next convert it into multiple interrelated triples, where each triple generally corresponds to an intermediate reasoning step. Finally, we reformulate the triples into several function calls with the code format, which represents the tool invocation and can be executed to obtain the corresponding triples based on the KG. Given a triple $\langle e, r, e' \rangle$, we craft a rule-based method to synthesize the function calls that represent the information flow from e to e'. Specifically, we start from the *get_relation(e)* function call to obtain the current candidate relations $\{r\}$ associated with e on the KG. Then, we select one relation r and pass it to other required function calls (*e.g., get_tail_entity* or *get_entity_by_constraint*), and finally obtain new entities. Following the order of the reasoning chain, we generate all the function calls to compose the final KG reasoning program for producing the instruction dataset. We show one example in Figure 1 (b) to intuitively illustrate the conversion process from the annotated SQL query to our required KG reasoning program.

3.2.2 KG Reasoning Instruction Synthesis

After obtaining the reasoning program on KG, we further utilize it for synthesizing instruction data for supervised fine-tuning (SFT). As discussed in Section 3.2.1, our instruction data is highly based on the reasoning program, which is aligned with the intermediate reasoning steps for KGQA.

Input-Output Pair Construction. The synthetic KG reasoning program consists of multiple function calls in a sequence. For each function call, we aim to construct an input-output pair as the instruction. Specifically, the input contains the question, toolbox definition, current KG information (*i.e.,* the next candidate relations of the current entity set), and history reasoning program before the current step; and the output is the function call at the current step. Next, after executing the function call at the current reasoning step, the history reasoning program and current KG information in the input will be accordingly updated, and the output

Method	**Work Flow**	**Base Model**	**Tool**	**Memory**	**Multi Task**
Pangu	pd	T5-3B	✗	✗	✗
StructGPT	pd	ChatGPT	✓	✗	✗
RoG	pd	LLaMA-7B	✗	✗	✗
ChatDB	auto	ChatGPT	✗	✓	✗
KB-BINDER	pd	CodeX	✗	✗	✗
KG-Agent	auto	LLaMA2-7B	✓	✓	✓

Table 1: Comparison of different methods. ***Work Flow*** describes that the interaction way between the LLM and KG is pre-defined ("pd") or autonomous ("auto"). ***Multi Task*** means whether to support generalization across different KGs via multi-task learning.

will be updated as the function call at the next step. By iterating the above process, for each sample in the KGQA datasets, we can obtain multiple input-output pairs derived from the corresponding reasoning program, which depict the complete reasoning trajectory on the KG. To help LLMs better understand, we further utilize a unified prompt, as shown in Figure 1 (c), to format each input-output pair and obtain the final instruction tuning data.

Agent Instruction Tuning. Based on the above formatted instruction tuning data, we perform supervised fine-tuning on a small LLM (*i.e.,* LLaMA-7B), which is much smaller than the backbone models in previous work (Jiang et al., 2023b). Formally, for each sample, we formulate all input-output pairs of the complete trajectory in the format of $\{\langle x_1, y_1\rangle, ..., \langle x_t, y_t\rangle, ..., \langle x_n, y_n\rangle\}$, where $\langle x_t, y_t\rangle$ represent the input and ground-truth response in the t-th step and n represents the total steps. For simplicity, we denote each input and output as x and y below. During the instruction tuning process, we feed the input x and output y into the decoder-only LLM and minimize the cross-entropy loss on the ground-truth response y as:

$$\mathcal{L} = -\sum_{k=1}^{m} \log \Pr(y_k | x, y_{<k}), \qquad (1)$$

where m denotes the number of tokens in y, y_k and $y_{<k}$ are the k-th and previous tokens in the output.

3.3 Autonomous Reasoning over KG

After instruction tuning, we further design an effective agent framework that enables KG-Agent to autonomously perform multi-step reasoning over KG for answer finding. The overall illustration of KG-Agent is shown in Figure 1 (a). It mainly contains four components, *i.e.,* the core instruction-tuned

LLM (Section 3.2), referred to as the *LLM-based planner*, the multifunctional *toolbox* (Section 3.1), the *KG-based executor* for executing the tool invocation, and the *knowledge memory* to record the context and currently useful information in the whole process. Next, we introduce how KG-Agent performs autonomous reasoning over KG.

Knolwedge Memory Initialization. The knowledge memory preserves the currently useful information to support the LLM-based planner for making decisions. It mainly contains four parts of information, *i.e.,* natural language question, toolbox definition, current KG information, and history reasoning program. The former two parts are initialized with the given question and toolbox definition, which remain unchanged during the reasoning process. The later two parts are initialized as an empty list, which will be constantly updated at each step after LLM generating the function call and executor invoking the corresponding tool.

Planner for Tool Selection. Based on the current knowledge memory, the LLM-based planner selects a tool to interact with KG at each step. Specifically, all the parts in the current knowledge memory will be formatted with corresponding prompt template to compose the input (used in Agent Instruction Tuning in Section 3.2.2), and then the LLM will generate one function call by selecting a tool and its arguments from the input. Generally, the planner needs to invoke tools from the pre-defined toolbox to address four types of task requirements, *i.e.,* linking the mentioned entity to KG (*e.g.,* "*get_candidate_entity*" and "*disambiguate_entity*"), accessing the KG information (*e.g., "get_relation"* and *"get_head_entity"*), processing the intermediate results (*e.g., "count"* and *"intersect"*), or returning the final answer to end the reasoning process (*e.g., "end"*).

Executor for Memory Updating. After the planner generates the function call, the KG-based executor will execute it using a program compiler. It can cache or operate the intermediate variables, and extract new entities or relations from the KG. After execution, the knowledge memory will be accordingly updated. First, the current function call will be added into the history reasoning program. Second, if the invoked tool is to obtain the new information from the KG (*e.g.,* "*get_relation*"), the executor will add it into the KG information for updating the knowledge memory.

Iterative Autonomous KG-Agent. The KG-Agent framework autonomously iterates the above tool selection and memory updation process to perform step-by-step reasoning, where the knowledge memory is used to maintain the accessed information from KG. In this way, the multi-turn decision-making process of the agent is like walking on the KG along relations. Once reaching the answer entities, the agent will automatically stop the iterative process. Note that the whole process is agnostic to the task types (*e.g.,* question answering) and some specific KGs. Therefore, our approach is a general framework that can be applied to a variety of complex tasks that require reasoning over any KGs.

3.4 Comparison to Previous Work

We give a comparison in Table 1. Existing methods of reasoning over KG can be categorized into two classes based on their workflow. The first line of research, such as KB-BINDER (Li et al., 2023), Pangu (Gu et al., 2023), StructGPT (Jiang et al., 2023b), and RoG (Luo et al., 2023), crafted a pre-defined interaction way between LLM and KG, which cannot flexibly adapt to various complex tasks. Another line of research, such as ChatBD (Hu et al., 2023a), conducted autonomous reasoning with chain-of-thought and memory augmented. However, it relies on the strong closed-source LLM APIs (*e.g.,* ChatGPT) and cannot use tools to implement some specialized operations (*e.g.,* count). Our KG-Agent is the first autonomous agent framework to support the complex interaction between LLM and KG with tool and memory augmented. Furthermore, we implement this autonomous agent by instruction tuning a smaller 7B open-source LLM compared to the backbone LLM in KB-BINDER, StructGPT, and ChatDB. At the same time, the agent instruction tuning data is constructed from various KGs (*e.g.,* Wikidata and Freebase), which helps our KG-Agent to learn the general autonomous decision making capabilities over various KGs.

4 Experiment

4.1 Experimental Setup

We select four commonly-used KGQA datasets as in-domain datasets, *i.e., WebQSP*, *CWQ*, and *GrailQA*, which are based on Freebase, and *KQA Pro*, which is based on Wikidata. And we select three ODQA datasets as out-of-domain datasets, *i.e., WQ*, *NQ*, and *TQ*. Further, we consider three

Model	WebQSP		CWQ		GrailQA (F1)			
	Hits@1	F1	Hits@1	F1	Overall	I.I.D.	Compositional	Zero-shot
GraftNet	66.4	60.4	36.8	32.7	-	-	-	-
NSM	68.7	62.8	47.6	42.4	-	-	-	-
SubgraphRetrieval	69.5	64.1	49.3	46.3	-	-	-	-
UniKGQA	75.1	70.2	50.7	48.0	-	-	-	-
ReasoningLM	78.5	71.0	69.0	64.9	-	-	-	-
RNG-KBQA	-	75.6	-	-	76.8	89.0	68.9	74.7
Uni-Parser	-	75.8	-	-	76.5	88.3	71.4	73.4
ArcaneQA	-	75.6	-	-	76.9	89.2	73.9	72.8
PanGu w/ T5-3B	-	79.6	-	-	83.4	-	-	-
TIARA	75.2	78.9	-	-	81.9	91.2	74.8	80.7
FC-KBQA	-	76.9	-	56.4	83.8	91.5	77.3	83.1
ROG	**85.7**	70.8	62.6	56.2	-	-	-	-
ChatGPT	67.4	59.3	47.5	43.2	25.3	19.6	17.0	31.2
Davinci-003	70.8	63.9	51.4	47.6	30.1	23.5	22.0	36.4
GPT-4	73.2	62.3	55.6	49.9	31.7	25.0	20.6	39.2
StructGPT	72.6	63.7	54.3	49.6	54.6	70.4	44.3	50.5
Ours	83.3	**81.0**	**72.2**	**69.8**	**86.1**	**92.0**	**80.0**	**86.3**

Table 2: The results on the test set of WebQSP and CWQ, and dev set of GrailQA, which are based on Freebase KG. We copy part of the results from Jiang et al. (2023b); Gu et al. (2023); Luo et al. (2023) and evaluate ChatGPT,Davinci-003, GPT-4, and StructGPT with OpenAI API. Bold font denotes the best performance.

types of baseline methods, *i.e., subgraph-based reasoning*, *LM-based seq2seq generation*, and *LLM-based methods* for comparison on in-domain datasets, and *Fine-tune based* and *LLM-based* methods for out-of-domain datasets. We show the details of the above datasets, baselines, evaluation protocol, and implementation in Appendix B.

4.2 Main Results

Results on In-domain Datasets. Table 2 and Table 3 show the results on in-domain datasets based on Freebase and Wikidata, respectively. First, LM-based seq2seq generation methods can achieve better F1 score compared to the subgraph-based reasoning methods on the WebQSP and KQA Pro. It indicates that the SPARQL query generated by the LM can obtain a more complete answer set, and the structured query can better support some complex operations (*e.g.,* maximum, count) than the traditional subgraph-based reasoning methods. Second, although LLMs are powerful, directly using Davinci-003, ChatGPT, and even GPT-4 still has a large performance gap compared with the best fine-tuned methods in WebQSP, GrailQA, and KQA Pro, indicating the difficulty of answering complex questions solely by LLMs.

Finally, our KG-Agent is substantially better than all other competitive baselines in all datasets after instructing tuning on the mixed data. With the mutual augmentation between different datasets, our approach achieves 1.7%, 7.5%, and 2.7% improvements of F1 on WebQSP, CWQ, and Grailqa, respectively. Benefiting from the autonomous reasoning mechnism, our approach can perform reasoning on the two KGs and obtain consistent improvement on all datasets.

Results on Out-of-domain Datasets. After instruction tuning, we directly evaluate the zero-shot performance of our KG-Agent on the out-of-domain datasets. As shown in Table 4, although fine-tuned with full data, the small pre-trained language models (*e.g.,* T5 and BART) can not effectively answer these factual questions. Owing to the large-scale parameters, Davinci-003 and ChatGPT performs well on NQ and TQ, which are constructed based on Wikipedia, the corpus that they may have been pre-trained on. However, they perform not well on WQ, which is constructed based on Freebase KG. In contrast, our KG-Agent only needs to learn how to interact with KG instead of memorizing the specific knowledge. Thus, it can utilize the external KG in zero-shot setting, and achieve consistent improvement compared to fine-tuned pre-trained language models.

4.3 Further Analysis

Transfer to Domain-specific KG. To evaluate the transferability of our approach on other KGs, we test our KG-Agent on the MetaQA dataset which is based on a movie domain KG. Follow-

Model	Overall	Multi-hop	Qualifier	Comparison	Logical	Count	Verify	Zero-shot
KVMemNet	16.61	16.50	18.47	1.17	14.99	27.31	54.70	0.06
EmbedKGQA	28.36	26.41	25.20	11.93	23.95	32.88	61.05	0.06
RGCN	35.07	34.00	27.61	30.03	35.85	41.91	65.88	0.00
RNN SPARQL	41.98	36.01	19.04	66.98	37.74	50.26	58.84	26.08
BART SPARQL	89.68	88.49	83.09	96.12	88.67	85.78	92.33	87.88
ChatGPT	24.96	24.22	26.37	39.15	25.51	10.76	54.70	15.67
Davinci-003	31.02	29.58	31.58	49.8	29.62	16.70	65.54	21.83
GPT-4	37.43	34.82	37.15	55.75	36.81	15.27	72.93	27.28
Ours	**92.15**	**91.03**	**87.90**	**96.32**	**91.28**	**88.21**	**92.86**	**91.40**

Table 3: The accuracy on the test set of KQA Pro, which is based on Wikidata KG. The results of Davinci-002,GPT-4, and ChatGPT are evaluated by us and the results of other baselines are copied from Cao et al. (2022).

Models	NQ-Wiki	TQ-Wiki	WQ-Freebase
T5-Base	30.94	27.63	24.06
T5-Large	31.21	29.40	24.70
BART-Base	29.47	25.43	21.95
BART-Large	32.60	33.05	26.33
Davinci-003	51.94	88.57	23.81
ChatGPT	**57.49**	**88.68**	23.23
Ours	33.00	35.89	**28.90**

Table 4: The results on the subsets of the dev sets from the out-of-domain ODQA datasets.

Models	MQA-1hop	MQA-2hop	MQA-3hop
GraftNet	82.5	-	-
EmbedKGQA	92.0	40.7	34.6
NSM	94.8	97.0	91.0
TransferNet	96.5	97.5	90.1
ChatGPT	61.9	31.0	43.2
StructGPT	94.2	93.9	80.2
Ours	**97.1**	**98.0**	**92.1**

Table 5: The results on the three subsets of MetaQA. We copy the results of baselines from Jiang et al. (2023b).

ing existing work (He et al., 2021; Jiang et al., 2023b), we show the one-shot results on the test set in Table 5. ChatGPT performs not well when directly answering these domain-specific questions, where the performance drops 45% absolutely on the MQA-3hop subset compared to the supervised fine-tuned TransferNet model. After equipping the LLM with the KG, StructGPT can greatlt outperform ChatGPT with about 37% improvement. In contrast, our KG-Agent can obtain consistent performance improvement compared to the competitive supervised fine-tuning baselines on all subsets. It indicates that the agent indeed learns the general ability about reasoning on KG, which can be efficiently transferred to other KGs.

Proportion	WebQSP	CWQ	GrailQA	Average
1:10:5	80.0	69.8	86.1	**78.6**
<u>2</u>:10:5	81.2	68.7	83.3	77.8
1:<u>20</u>:5	78.9.	73.6	78.8	77.1
1:10:<u>10</u>	80.8	66.9	84.3	77.3

Table 6: The F1 scores on three in-domain datasets after instruction tuning under different sampling proportions. We highlight the changed proportion with an underline.

Effect of Instruction Amount. We explore how the amount of instructions affects the performance of KG-Agent and show the results in Figure 2. With a constant sampling proportion, we scale the total amount from 2k to 64k in an exponential way and evaluate the F1 and Hist@1 scores on WebQSP and CWQ datasets. As we can see, the performance increases with more instruction tuning data, and eventually reaches a stable state, which indicates the importance of data amount. At the same time, with the data amount increasing from 16k to 64k, the KG-Agent doesn't obtain a remarkable performance improvement. We think this is relevant to the variety of our instruction tuning data, which is illustrated in existing work (Chung et al., 2022; Aribandi et al., 2022). Therefore, we will construct more various samples in the future, and could further boost the performance.

Effect of Tuning Data Proportion. Our experiment finds that only sampling 10K samples from existing datasets is enough for backbone LLM to learn the autonomous decision making capability. Here, we conduct a further ablation study to explore the impact of sampling proportion on the agent's performance when keeping the total amount of instruction tuning data constant. Specifically, we evaluate the agent performance of WebQSP, CWQ, and GrailQA when doubling the proportion of one

Base LLM	WebQSP	CWQ	Average
ROG w ChatGPT	70.8	56.2	63.5
Ours w LLaMA2-7B	81.0	69.8	75.4
w Phi2-3B	76.9	65.5	71.2
w Mistral-7B	80.3	68.5	74.4
w CodeLLaMA-7B	82.0	69.9	76.0
w LLaMA3-7B	83.5	72.1	77.8

Table 7: The F1 results on the WebQSP and CWQ with different base LLMs.

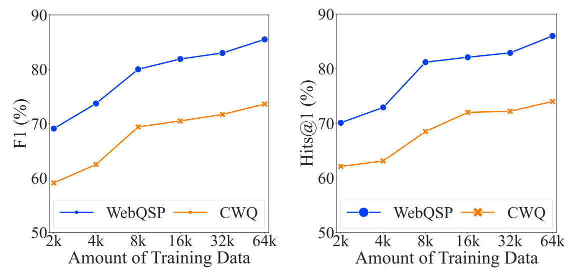

Figure 2: The F1 (Left) and Hits@1 (Right) scores of KG-Agent on the test set of WebQSP and CWQ with a various amount of instruction tuning data.

dataset while maintaining the other two dataset proportions. We show the results in Table 6. We can see that as the sampling proportion of a certain dataset increases, the agent performance on it consistently improves. However, for the average performance on all three datasets, all variants are lower than our selected proportion, indicating that the proportion we chose is suitable for the LLM to balance and master more comprehensive and general abilities.

Effect of Different Base LLMs. We further investigate the generalizability of our approach by utilizing other different mainstream open-source LLMs as the base model for KG-Agent, such as Phi2-3B, Mistral-7B, CodeLLaMA-7B, and LLaMA3-7B. The results are presented in Table 7, which indicates that both of these LLMs can achieve superior performance compared to closed-source LLMs. Models with the same parameter scale exhibit similarly strong performance. Additionally, larger LLMs tend to perform better when comparing these 7B LLMs with Phi2-3B model. These results demonstrate that our method is adaptable to various LLMs.

Scalability and Efficiency. In the above, our KG-Agent can achieve superior performance compared to closed-source LLMs relying on the only 7B open-source LLM. We further investigate whether the performance of KG-Agent aligns with the parameter size of base LLMs. Additionally, we examine the inference latency to compare KG-Agent's efficiency with other state-of-the-art approaches. We show the detailed results in Appendix D. In general, the results demonstrate that our method can obtain better results by increasing model parameters. Besides, our method demonstrates a time advantage over API-based LLMs and is comparable in speed to existing methods.

Case Study. We present an example to show the details of the workflow along with the input and output of our KG-Agent, as shown in Appendix D.1.

5 Related Work

Recent research has LLMs for reasoning over KGs, primarily through retrieval-augmented and synergy-augmented methods. Retrieval-augmented approaches retrieve triples from KGs but often lose structured information and introduce redundancy. In contrast, synergy-augmented methods enable multiple interactions between LLMs and KGs, allowing for more flexible reasoning, though they still follow fixed protocols. Additionally, LLM-based agents like ReAct and AutoGPT have emerged for autonomous task-solving, relying heavily on powerful closed-source LLMs. The proposed KG-Agent distinguishes itself as the first framework for complex KG reasoning using a smaller 7B LLM, facilitating autonomous decision-making without human intervention. We give a more detailed description in Appendix A.

6 Conclusion

In this work, we proposed an autonomous agent framework to synergize LLMs and KGs to perform complex reasoning over KG, namely KG-Agent. We first curated a toolbox for KG, consisting of three types of tools to support the typical operations when reasoning on KG. Then, we developed an autonomous iteration mechanism based on tool selection-then-memory updation that integrates the LLM, multifunctional toolbox, KG-based executor, and knowledge memory, for reasoning over KG. Finally, with only 10K synthesized code-based tuning samples, our autonomous agent with 7B LLaMA2 model, which mostly outperforms strong baselines with full-data tuning or larger LLMs.

Limitations

Although KG-Agent demonstrates remarkable performance across various complex factual question answering tasks, there are some limitations of our method. First, we only use the LLaMA2-7B as the backbone LLM, which has a strong capability after instruction tuning. Hence, more experiments are required to evaluate other LLMs with comparable parameter sizes, such as Mistral-7B (Jiang et al., 2023a) or CodeLLaMA-7b (Rozière et al., 2023). Second, we focus on reasoning over the KG to answer the factual questions. We should consider extending our framework to deal with more types of knowledge sources, *e.g.,* databases or tables. Third, we only evaluate factual question answering tasks based on KG. Future work should include wider evaluation scenarios to evaluate the universality of our method, *e.g.,* data-to-text and formal-language-to-text (Xie et al., 2022). Finally, we have tried our best to tune the LLM only to answer the questions based on the KG information, and avoid generating discriminatory and risky responses for user questions. However, we should add more rule-based methods to post-process the predictions and filter the illegal responses.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China under Grant No. 92470205 and 62222215, Beijing Municipal Science and Technology Project under Grant No. Z231100010323009, and Beijing Natural Science Foundation under Grant No. L233008.

References

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Prakash Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. 2022. Ext5: Towards extreme multi-task scaling for transfer learning. In *ICLR*. OpenReview.net.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 6101–6119. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.*

Tianqing Fang, Zeming Chen, Yangqiu Song, and Antoine Bosselut. 2024. Complex reasoning over logical queries on commonsense knowledge graphs. *arXiv preprint arXiv:2403.07398.*

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4928–4949. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond I.I.D.: three levels of generalization for question answering on knowledge bases. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 3477–3488. ACM / IW3C2.

Yu Gu and Yu Su. 2022. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 1718–1731. International Committee on Computational Linguistics.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 553–561. ACM.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023a. Chatdb: Augmenting llms with databases as their symbolic memory. *CoRR*, abs/2306.03901.

Xuming Hu, Junzhe Chen, Xiaochuan Li, Yufei Guo, Lijie Wen, Philip S. Yu, and Zhijiang Guo. 2023b. Do large language models know about facts? *CoRR*, abs/2310.05177.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *CoRR*, abs/2310.06825.

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

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yaliang Li, and Ji-Rong Wen. 2023c. Reasoninglm: Enabling structural subgraph reasoning in pre-trained language models for question answering over knowledge graph. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3721–3735. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023d. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Complex knowledge base question answering: A survey. *IEEE Trans. Knowl. Data Eng.*, 35(11):11196–11215.

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

Yudong Liu, Xu Zhang, Shilin He, Hongyu Zhang, Liqun Li, Yu Kang, Yong Xu, Minghua Ma, Qingwei Lin, Yingnong Dang, Saravan Rajmohan, and Dongmei Zhang. 2022. Uniparser: A unified log parser for heterogeneous log data. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 1893–1901. ACM.

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

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1400–1409.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap. *CoRR*, abs/2306.08302.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3505–3506. ACM.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural*

Language Processing, EMNLP 2020, Online, November 16-20, 2020, pages 5418–5426. Association for Computational Linguistics.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950.

Apoorv Saxena, Aditay Tripathi, and Partha P. Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4498–4507.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

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

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje F. Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 8108–8121. Association for Computational Linguistics.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. Progprompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 11523–11530. IEEE.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4231–4242.

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. *CoRR*, abs/2307.07697.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 641–651. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2023a. A survey on large language model based autonomous agents. volume abs/2308.11432.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2023b. A survey on large language model based autonomous agents. *CoRR*, abs/2308.11432.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 602–631.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023. MM-REACT: prompting chatgpt for multimodal reasoning and action. *CoRR*, abs/2303.11381.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 6032–6043. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8413–8426.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 5773–5784. Association for Computational Linguistics.

Lingxi Zhang, Jing Zhang, Yanling Wang, Shulin Cao, Xinmei Huang, Cuiping Li, Hong Chen, and Juanzi Li. 2023. FC-KBQA: A fine-to-coarse composition framework for knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1002–1017. Association for Computational Linguistics.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6069–6076.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2023. Memorybank: Enhancing large language models with long-term memory. *CoRR*, abs/2305.10250.

A Related Work

LLM-based KG Reasoning. Benefitting from the powerful zero-shot and few-shot capability, recent studies have leveraged LLMs to perform reasoning over KG. Recent work can be roughly divided into *retrieval-augmented* (Shu et al., 2022) and *synergy-augmented* (Gu et al., 2023) two types. The retrieval-augmented method is to retrieve and serialize the triples from the KG, and then feed it to the LLM to help generate the final results (*e.g.,* answers or SPARQL query) (Ye et al., 2022). Such a way loses the structured information in the original KG and may retrieve redundant knowledge, limiting LLMs' understanding. To relieve these problems, the synergy-augmented methods design an information interaction mechanism between LLMs and KGs to enable LLMs to query KGs multiple times to answer the question (Jiang et al., 2023b). Specifically, they either first generate the full plan (Li et al., 2023) and then ground it on KG, or make a plan step-by-step based on the KG (Luo et al., 2023). Although obtaining better performance, the information interaction mechanism in existing methods often follows a pre-defined way, which cannot flexibly adapt to various complex tasks. In contrast, our proposed KG-Agent can autonomously make decisions during reasoning over KG, without human assistance.

LLM-based Agents. Recently, LLMs have shown surprising long-horizon planning and reasoning capabilities (Shinn et al., 2023; Zhong et al., 2023), and LLM-based agents have gradually become a hot topic for autonomously solving complex interactive tasks (Wang et al., 2023b). A large number of agents focus on general-purpose task solving. As the representative projects, ReAct (Yao et al., 2023) proposes a prompting method to convert LLMs (*e.g.,* ChatGPT) as language agents, to interact with the external environment, receive the feedback, and then generate the action for next step reasoning. Then, AutoGPT[1] further empowers LLMs (*i.e.,* GPT4) with long/short-term memory management and external tools like search engines to autonomously address a user request. In addition, several other agents also focus on specific domains, such as WebGPT (Nakano et al., 2021) for the web-browsing environment, MM-REACT (Yang et al., 2023) for the multi-modal scenario, and ProgPrompt (Singh et al., 2023) for the real-life environment. However, recent works involving language agents mostly rely on stronger closed-source LLM APIs (*e.g.,* ChatGPT and GPT-4) to understand or learn to solve complex tasks. Our KG-Agent is the first autonomous agent framework to support complex reasoning over KG only relying on a relatively smaller 7B LLM.

[1] https://github.com/Significant-Gravitas/AutoGPT

B Experiment Setup

B.1 Datasets

We select four popular complex KGQA datasets as in-domain datasets, *i.e., WebQuestionsSP (WebQSP)* (Yih et al., 2016), *Complex WebQuestions 1.1 (CWQ)* (Talmor and Berant, 2018), and *GrailQA* (Gu et al., 2021), which are based on Freebase, and *KQA Pro* (Cao et al., 2022), which is based on Wikidata. And we select three representative ODQA datasets as out-domain datasets, which are *WebQuestions (WQ)* (Berant et al., 2013), *Natural Questions (NQ)* (Chen et al., 2017), and *TriviaQA (TQ)* (Joshi et al., 2017). Since we only rely on the KG to answer questions, we filter the questions in ODQA datasets that can not be linked to any entity in KG, denoted as *WQ-Freebase*, *NQ-Wiki*, and *TQ-Wiki*, respectively. Besides, we further select the *MetaQA* (Zhang et al., 2018), which is based on a domain-specific movie KG, to evaluate the generalibility of our method. The detail description of these selected datasets is as follows:

• **WebQSP** consists of 4,737 questions. The answer entities are within a maximum of 2 hops from the topic entity on the Freebase KG. We adopt the train/valid/test splits from GraftNet (Sun et al., 2018) for consistency.

• **CWQ** is constructed based on WebQSP, which is more challenging. It complicates WebQSP by extending the question entities or adding constraints to restrict the answers. The answer entities are within a maximum of 4 hops from the topic entity on the Freebase KG.

• **GrailQA** consists of 64,331 questions. Compared to WebQSP and CWQ, it focuses on a more comprehensive generalization capability evaluation from three levels (*i.e.,* i.i.d, compositional, and zero-shot).

• **KQA Pro** consists of 117,970 questions. The above three datasets are based on Freebase, and it is based on Wikidata, and require multiple reasoning capabilities including compositional reasoning, multi-hop reasoning, quantitative comparison, set operations, and etc.

• **MetaQA** comprises over 400,000 questions based on a movie domain KG, with answer entities located up to three hops away from the topic entities. Based on the number of hops, the dataset is divided into three sub-datasets: MetaQA-1hop, MetaQA-2hop, and MetaQA-3hop. Following existing work (He et al., 2021), we randomly sample just one training case for each question template from the original training set, to form a one-shot training dataset.

• **WQ** consists of 6,642 questions. The questions are mostly centered around a single named entity and are supposed to be answerable by Freebase KG. We extract 2034 questions from the original test set to compose the WQ-freebase subset.

• **NQ** consists of 323,045 questions. Each example contains a question from the Google search and the corresponding answers, which are text spans on the Wikipedia page. Following existing work (Roberts et al., 2020), we use the open version of this dataset which discards answers with more than 5 tokens. We extract 543 questions from the original test set to compose the NQ-Wiki subset.

• **TQ** consists of 110K questions. Each example contains a question authored by trivia enthusiasts, and the answers are text spans from the Web or Wikipedia. Following existing work (Roberts et al., 2020), we use its unfiltered version for evaluation. We extract 1864 questions from the original test set to compose the TQ-Wiki subset.

B.2 Evaluation Protocol

For KGQA, following existing work (Sun et al., 2018), we use Hits@1 and F1 metrics for WebQSP and CWQ datasets, F1 metric for GrailQA dataset, and Hits@1 for MetaQA. The Hits@1 evaluates the correctness of the top-ranked answer while F1 considers coverage of all the predicted answers. It's worth noting that some baselines and our approach would return all the unordered answers at the end, which is not suitable for the Hist@1 metric. For a comprehensive comparison, we randomly select one answer per question as the top-ranked answer and then calculate the average Hits@1 result by repeating this process 100 times following existing work (Shu et al., 2022). For ODQA, following existing work (Roberts et al., 2020), we report the EM metric, which evaluates whether the predicted answer is the same as the gold one after performing normalization.

B.3 Baselines for Comparison

For KGQA, we consider the following three types of baseline methods for performance comparison:

• **subgraph-based reasoning** methods which perform answer reasoning in a retrieval subgraph form KG, including GrafeNet (Sun et al., 2018), NSM (He et al., 2021), SubgraphRetrieval (Zhang et al., 2022), UniKGQA (Jiang et al., 2023d), and ReasoningLM (Jiang et al., 2023c) for datasets on Freebase, and KVMemNet (Miller et al., 2016), EmbedKGQA (Saxena et al., 2020), and RGCN (Schlichtkrull et al., 2018) for datasets on Wikidata;

• **LM-based seq2seq generation** methods which generate the final SPARQL query by fine-tuning a sequence-to-sequence language model, including RNG-KBQA (Ye et al., 2022), UniParser (Liu et al., 2022), ArcaneQA (Gu and Su, 2022), PanGu w/ T5-3B (Gu et al., 2023), TIARA (Shu et al., 2022), and FC-KBQA (Zhang et al., 2023) for datasets on Freebase, and RNN SPARQL and BART SPARQL (Cao et al., 2022) for datasets on Wikidata;

• **LLM-based** methods which utilize the powerful zero-shot or few-shot capabilities of LLMs to answer the question without fine-tuning, including ROG (Luo et al., 2023), StructGPT (Jiang et al., 2023b), gpt-3.5-turbo-instruct (Davinvi-003) [2], gpt-3.5-turbo (ChatGPT) [3], and gpt-4 (GPT-4) [4] for both in-domain datasets.

For ODQA, we focus on the closed-book setting where no documents are provided and consider the following two types of baseline methods:

• **Fine-tune based** methods which learn to predict the answers, including T5-Base, T5-Large, BART-base, and BART-Large from (Roberts et al., 2020);

• **LLM-based** methods which directly answer the questions in zero-shot setting, including gpt-3.5-turbo-instruct (Davinvi-003) and gpt-3.5-turbo (ChatGPT).

B.4 Implementation Details

For instruction tuning data construction, we randomly sample a total of 10,000 training data from in-domain datasets in a ratio of 1:5:5:10 for WebQSP, KQA Pro, GrailQA, and CWQ according to some prior empirical studies. Since we focus

[2] https://platform.openai.com/docs
[3] https://platform.openai.com/docs
[4] https://platform.openai.com/docs

on the reasoning process over KG, we suppose the entities have been given for each question following existing work (Sun et al., 2018; He et al., 2021; Jiang et al., 2023b). For instruction tuning, we use the LLaMA2-7B (Touvron et al., 2023) as our backbone LLM. We use a cosine learning rate schedule with an initial learning rate of 2e-5, a weight decay of 0.1, a batch size of 256, a maximum length of 1500, and finally fine-tune the model for 3 epochs. For the relation retrieval model and entity disambiguation model in the semantic tool, we build them following the existing work (Zhang et al., 2022; Shu et al., 2022).

We use the entire Freebase KG (which includes about 1.9 billion triples) and partial Wikidata KG (containing around 3 billion triples) to conduct the experiment. Both are typical large KGs, indicating the scalability of our framework. During the reasoning process, starting from the mentioned entities, we do not need to import all the KG information into our agent memory. Instead, the agent only needs to process the current hop of KG information, and keeps only the useful part. Concretely, the whole KG (100G) is stored in the disk, and no more than 1G of information is required to be loaded into the memory. Such a special design enables our approach to accept very large-scale KGs.

After instruction tuning, for in-domain datasets, we evaluate the performance of our KG-Agent on the test set of CWQ, WebQSP, KQA Pro, and the dev set of GrailQA. For out-domain datasets, we evaluate the zero-shot performance of our KG-Agent on the NQ-Wiki, TQ-Wiki, and WQ-Freebase. For the domain specific dataset, *i.e.,* MetaQA, we follow existing work (He et al., 2021; Jiang et al., 2023b) to extract the one-shot tuning subset from the original training set and fine-tune our KG-Agent with it. When evaluating the performance of Davinci-003, ChatGPT, and GPT4, we use the latest February version of APIs from OpenAI. And for in-domain datasets, we provide six demonstrations for each test question and parse the prediction results following existing work (Sun et al., 2023; Jiang et al., 2023b), we show the prompt with demonstration for each dataset in Table 8. For the selection of demonstrations, we randomly sample from the corresponding training set for each dataset. For out-domain datasets, since they are open-domain question answering tasks, we directly input the question to LLMs with proper prompt, as shown in Table 8, and then evaluate the

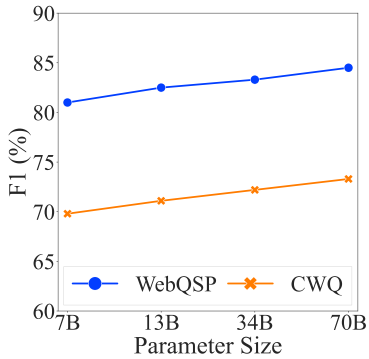

Figure 3: The F1 (Left) and Hits@1 (Right) scores of KG-Agent on the test set of WebQSP and CWQ with a various amount of instruction tuning data.

output.

We trained our model using eight A800 GPUs, each with 80 GB of memory, and a CPU with 128 cores. For testing, we utilized a single A800 GPU with 80 GB of memory and the same CPU configuration. We employ FlashAttention (Dao et al., 2022) to implement memory-efficient attention, and DeepSpeed (Rasley et al., 2020) to facilitate the training of large language models.

C Summary of Toolbox

We summarize the tool name, tool description, and the input argument and output of tools in Table 9.

D Scalability and Efficiency

Here, we further investigate whether the performance of KG-Agent aligns with the parameter size of base LLMs. We try our best to conduct a controlled experiment using a series of LLaMA2 models, including 7B, 13B, 34B, and 70B. Since the original LLaMA2-34B base model has not been released, we adopt the corresponding CodeLLaMA-34B to approximation. We evaluate the results on WebQSP and CWQ and show the results in Figure 3. Additionally, we examine the inference latency (the time taken to answer a question) to compare KG-Agent's efficiency with other state-of-the-art approaches. We select three strong baselines: RoG (LLaMA-7B+KG), StructGPT (ChatGPT+KG), and GPT-4. Although we made every effort to maintain a consistent testing environment and average the results across five inference trials, the measured time would be only used to approximate comparison, considering the potential environmental variables such as hardware differences and network latency. The results are presented in

Table 10. Our method demonstrates a time advantage over API-based LLMs and is comparable in speed to existing methods.

D.1 Case Study

As shown in Figure 1, the core process of KG-Agent is an autonomously iterative tool selection and memory update. Concretely, at each iteration, the LLM-based planner first selects a tool to interact with KG based on the current knowledge memory, which mainly contains four parts of information, i.e., natural language question, toolbox definition, current KG information, and history reasoning program. After the planner generates the function call, the KG-based executor will execute it on KG using a program compiler. After execution, the knowledge memory will be accordingly updated for the next iteration. The KG-Agent will autonomously iterates the above process. Once reaching the answer entities, the agent will automatically stop the iterative process. Here, we further present an example to show the details of the aforementioned process along with the input and output of our KG-Agent, as shown in Table 11.

Dataset	Prompt
WebQSP	Question: where is the syracuse university? Answer: [New York | Syracuse | United States of America]. Question: where is the mtv headquarters? Answer: [New York City]. Question: what are the 3 official languages of spain? Answer: [Spanish Language]. Question: what timezone is new england usa in? Answer: [Eastern Time Zone]. Question: who started southwest airlines? Answer: [Herb Kelleher | Rollin King]. Question: what was irving langmuir famous for? Answer: [Scientist]. Question: {test question} Answer:
CWQ	Question: Who is the president in the place where the government of Peru is located? Answer: [Ollanta Humala]. Question: Where did Martin Luther King attend university, that has less than 2,586 undergraduates? Answer: [Morehouse College]. Question: What movie produced by the company New Line Cinema was Taylor Lautner in? Answer: [Valentine's Day]. Question: Which year did the team that plays at Turner Field win the World Series? Answer: [1995 World Series]. Question: Which airports are in the circulation area of Il Manifesto? Answer: [Leonardo da Vinci–Fiumicino Airport | Ciampino–G. B. Pastine International Airport]. Question: What were the professions held by the publisher of "The Awakening?"? Answer: [Businessperson | Novelist | Writer | Author]. Question: {test question} Answer:
GrailQA	Question: what does the thiokol rocket do? Answer: [Launch vehicle]. Question: what is the club interest of inverness yacht club? Answer: [Sailing]. Question: who is the tour operator of kiribati? Answer: [Fly Water Adventures | Kiribati Holidays | Otintaai Tours | Molloy's Tours]. Question: 1998 marsala vergine terre arse contains what type of grapes? Answer: [Catarratto | Grillo | Ansonica]. Question: how many ice hockey coaches have coached the team that is currently coached by the eisbaren berlin? Answer: [1]. Question: court of appeal of sri lanka has what inferior court? Answer: [Supreme Court of Sri Lanka]. Question: {test question} Answer:
KQA Pro	Question: Which website officially represents Morgan Creek Productions? Answer: [http://www.morgancreek.com/]. Question: Which is shorter: The Killers, with a story set in Los Angeles, or Sherlock Holmes, produced by 20th Century Fox? Answer: [Sherlock Holmes]. Question: What is the street address for the University of San Diego? Answer: [5998 Alcala Park, San Diego, CA, 92110-2492]. Question: How is the Francis Bacon who died in New Haven related to the Yale School of Medicine? Answer: [educated at]. Question: For the film titled Aladdin, where is it published on its publication date of 2019-05-24? Answer: [United States of America]. Question: Who wrote The Postman which was published in 1985? Answer: [David Brin]. Question: {test question} Answer:
NQ-Wiki **TQ-Wiki** **WQ-Freebase**	Answer the following question with one or few words. Question: {test question}

Table 8: The prompts used for each dataset when evaluating the ChatGPT, Davinci-003, and GPT-4 models. When performing evaluation, just replace the "{test question}" with the test question.

Type	Tool	Description
Extraction Tool	get_relation	Input: entity set $\{e\}$ $\rightarrow$ Output: one-hop relations $R_{\{e\}}$ Return the incoming and outgoing relations of the given entity set $\{e\}$ on KG.
	get_head_entity	Input: entity set $\{e\}$, relation r $\rightarrow$ Output: entity set $\{e\}$ Return the head entity set of the given tail entity set $\{e\}$ along the relation r.
	get_tail_entity	Input: entity set $\{e\}$, relation r $\rightarrow$ Output: entity set $\{e\}$ Return the tail entity set of the given head entity set $\{e\}$ along the relation r.
	get_entity_by_type	Input: string type t $\rightarrow$ Output: entity set $\{e\}$ Return the entity set belonging to the given type t.
	get_entity_by_constraint	Input: entity set $\{e\}$, relation r, operator o, string value v $\rightarrow$ Output: entity set $\{e\}$ Return the new entity set whose tail entity along r satisfies the constraint condition. If v is not empty, the o should be one of {"=",">",">=","<","<="}, which means the comparison between the tail entity and string value should satisfy the operator. Else, the o should be one of {"argmax","argmin"}, which means the tail entity should be the maximum or minimum value.
	get_candidate_entity	Input: string entity mention m $\rightarrow$ Output: entity set $\{e\}$ Return the candidate linked entity set on the KG for the given entity mention m.
Logic Tool	count	Input: entity set $\{e\}$ $\rightarrow$ Output: integer Return the number of entities in the given entity set $\{e\}$.
	intersect	Input: entity set list $[\{e\}]$ $\rightarrow$ Output: entity set $\{e\}$ Return the intersection of the given list of entity sets.
	union	Input: entity set list $[\{e\}]$ $\rightarrow$ Output: entity set $\{e\}$ Return the union of the given list of entity sets.
	judge	Input: entity set $\{e\}$, relation r, operator o, string value v $\rightarrow$ Output: boolean Return a boolean value indicating whether the comparison between the tail entity of the given entity set $\{e\}$ along relation r and the given value v satisfies the operator o.
	end	Input: entity set $\{e\}$ $\rightarrow$ Output: entity set $\{e\}$ Return the entity set as the final answer and end the reasoning process.
Semantic Tool	retrieve_relation	Input: relation set $\{r\}$ $\rightarrow$ Output: relation set $\{r\}$ Retrieve relations from the given relation set $\{r\}$ that are semantically relevant to the question through neural network.
	disambiguate_entity	Input: entity set $\{e\}$ $\rightarrow$ Output: entity e Disambiguate the candidate linked entity $\{e\}$ based on the question semantics and entity information on KG (*e.g.,* one-hop relations) through neural network.

Table 9: The detailed definition and usage of all the tools.

Method	Work Flow	Base Model	Tool	Memory	Multi Task	Inference Latency
Pangu	pd	T5-3B	✗	✗	✗	0.78s
StructGPT	pd	ChatGPT	✓	✗	✗	2.63s
RoG	pd	LLaMA-7B	✗	✗	✗	0.85s
ChatDB	auto	ChatGPT	✗	✓	✗	-
KB-BINDER	pd	CodeX	✗	✗	✗	-
KG-Agent	auto	LLaMA2-7B	✓	✓	✓	0.89s

Table 10: Comparison of different methods. *Work Flow* describes that the interaction way between the LLM and KG is pre-defined ("pd") or autonomous ("auto"). *Multi Task* means whether to support generalization across different KGs via multi-task learning. *Inference Latency* refers to the rough average time, measured in seconds, required to answer a WebQSP question.

```
>>> Step 1 Input:
Generate the programs to solve the question: what is the name of justin bieber brother?
Here are some available functions:
def obtain_kg_information(entity):
    Return the in-degree and out-degree relations of the given entity from kg.
...
Here is no current KG information.
Please select a function and provide the corresponding arguments to generate the next line of solution program. The question is:
what is the name of justin bieber brother?
Solution program:
linked_entity_1 = "justin bieber"
linked_entity_2 = "male"
answer_type = none
>>> Step 1 - Output: obtain_kg_information(linked_entity_1)
>>> Step 1 - Execute: obtain the current KG information and update the memory.
```

```
>>> Step 2 Input:
Here are some available functions:
def obtain_kg_information(entity):
    Return the in-degree and out-degree relations of the given entity from kg.
...
Here is the current KG information:
For linked_entity_1 (different relations are separated by ' | '):
The out-degree relations: "people.person.gender" ... "type.object.name"
The in-degree relations: "people.sibling_relationship.sibling" ... "people.person.children"
Please select a function and provide the corresponding arguments to generate the next line of solution program. The question is:
what is the name of justin bieber brother?
Solution program:
linked_entity_1 = "justin bieber"
linked_entity_2 = "male"
answer_type = none
>>> Step 2 - Output: var_0 = get_tail_entity(linked_entity_1, "people.person.sibling_s")
>>> Step 2 - Execute: obtain the current KG information and update the memory.
```

... (We omit the intermedia steps and directly show the final step for convenience.)

```
>>> Step 7 Input:
Here are some available functions:
def obtain_kg_information(entity):
    Return the in-degree and out-degree relations of the given entity from kg.
...
Here is no current KG information.
Please select a function and provide the corresponding arguments to generate the next line of solution program. The question is:
what is the name of justin bieber brother?
Solution program:
linked_entity_1 = "justin bieber"
linked_entity_2 = "male"
answer_type = none
var_0 = get_tail_entity(linked_entity_1, "people.person.sibling_s")
var_1 = get_tail_entity(var_0, "people.sibling_relationship.sibling")
var_2 = get_head_entity(linked_entity_2, "people.person.gender")
var_3 = intersect(var_1, var_2)
>>> Step 7 - Output: ans = end(var_3)
>>> Step 7 - Execute: obtain the final answers and stop reasoning.
```

Table 11: A case study of our KG-Agent. The green text denotes the input question, the pink text denotes the toolbox definition, the blue text denotes the current KG information, the orange text denotes the task description, and the brown text denotes the history programs.