A Framework of Knowledge Graph-Enhanced Large Language Model Based on Question Decomposition and Atomic Retrieval

Yading Li[1], Dandan Song[1*], Changzhi Zhou[1], Yuhang Tian[1], Hao Wang[1], Ziyi Yang[2], Shuhao Zhang[3]

[1]School of Computer Science and Technology, Beijing Institute of Technology, China.
[2]School of Cyberspace Science and Technology, Beijing Institute of Technology, China.
[3]College of Computing and Data Science, Nanyang Technological University, Singapore.
{liyading,sdd,zhou_changzhi97,tianyuhang,wanghao,yziyi}@bit.edu.cn;
shuhao.zhang@ntu.edu.sg

Abstract

Knowledge graphs (KGs) can provide explainable reasoning for large language models (LLMs), alleviating their hallucination problem. Knowledge graph question answering (KGQA) is a typical benchmark to evaluate the methods enhancing LLMs with KG. Previous methods on KG-enhanced LLM for KGQA either enhance LLMs with KG retrieval in a single round or perform multi-hop KG reasoning in multiple rounds with LLMs. Both of them conduct retrieving and reasoning based solely on the whole original question, without any processing to the question. To tackle this limitation, we propose a framework of KG-enhanced LLM based on question decomposition and atomic retrieval, called KELDaR. We introduce question decomposition tree as the framework for LLM reasoning. This approach extracts the implicit information of reasoning steps within complex questions, serving as a guide to facilitate atomic retrieval on KG targeting the atomic-level simple questions at leaves of the tree. Additionally, we design strategies for atomic retrieval, which extract and retrieve question-relevant KG subgraphs to assist the few-shot LLM in answering atomic-level questions. Experiments on KGQA datasets demonstrate that our framework outperforms existing reasoning-based baselines. And in a low-cost setting without additional training or fine-tuning, our framework achieves competitive or superior results compared to most existing training-based baselines.

1 Introduction

To tackle challenges such as interpretability and hallucinations (Ji et al., 2023) in large language models (LLMs), some studies (Zhang et al., 2022b; Wang et al., 2023d) have sought to enhance LLM reasoning using knowledge graphs (KGs). These approaches offer domain-specific and real-time knowledge, aiding LLMs in performing interpretable reasoning and reducing hallucinations. In the evaluation of KG-enhanced LLM methods, knowledge graph question answering (KGQA) is an important task aiming to answer given questions based on factual knowledge in KG.

Previous works on KG-enhanced LLM for KGQA proposed retrieve-then-answer methods aiding LLM with KG retrieval in a single round interaction between LLM and KG , as shown in Figure 1 (top left). They retrieve question-relevant KG facts, which are then fed into the LLM all at once as references for answer generating (Baek et al., 2023a; Sen et al., 2023; Wu et al., 2023). However, these methods rely heavily on the KG retrieval results, using the LLM only as a tool for reasoning based on the logic in retrieval results and extracting the final answer from them. This limits the flexibility of LLM in reasoning (Yang et al., 2023) and doesn't fully leverage its strengths in question reasoning.

To enable the LLM to play a more active role in reasoning, existing methods enable multi-round LLM-KG interaction for LLMs to perform multi-hop reasoning on KG, as shown in Figure 1 (bottom left). Starting from the topic entities of the question, they prompt the LLM to select the most relevant next-hop relations or entities on KG. And iterate this process until LLM determines that the chain of entities and relations it traversed is sufficient to answer the question (Jiang et al., 2023a; Sun et al., 2024). However, these methods perform only one-hop retrieval of relations and entities on KG during each round of LLM-KG interaction, resulting in low efficiency. Additionally, each hop's retrieval is based solely on the whole original question, without making LLM aware of the specific sub-question that needs to be addressed at the current step for more targeted selection.

The methods above on KG-enhanced LLM for KGQA are all based on original questions without analyzing their logic explicitly during reasoning and KG retrieval. To tackle this limitation, we draw inspiration from a work training small language

[*]Corresponding author.

models on explainable question answering (Zhang et al., 2023), which integrate heterogeneous knowledge sources and analyze the semantics of complex questions with question decomposition tree. Our idea is to introduce this structure into a framework of KG-enhanced LLM. To realize our idea, we need to tackle two key problems. First, we need to design a framework capable of performing question decomposition and enabling atomic retrieval for atomic-level questions. Second, considering the shortcoming on efficiency in previous KG retrieval, it's also important to design the KG extracting and retrieving strategy for atomic retrieval.

To this end, we propose a Framework of **K**G-**E**nhanced **L**LM Based on Question **D**ecomposition **a**nd Atomic **R**etrieval (**KELDaR**). To excavate the information of reasoning logic in questions, we introduce the question decomposition tree as a framework for LLM reasoning, which classifies the questions according to their complexity and allows complex questions to be decomposed into a tree for multi-step reasoning, enabling atomic retrieval on KG for corresponding sub-question of each reasoning step. To improve the efficiency of the atomic retrieval, we further design efficient strategies for extracting and retrieving question-relevant KG subgraph of facts, expanding the candidate subgraph to a pruned two-hop range.

To evaluate the effectiveness of our framework, we conducted experiments on two commonly used KGQA datasets (Yih et al., 2016; Talmor and Berant, 2018). Experimental results demonstrate that our proposed framework significantly enhances the reasoning performance of the LLM. Our framework outperforms reasoning-based baselines without additional training or fine-tuning like us, and even achieves competitive or superior results compared to most training-based baselines.

Our main contributions are as follows:

- We introduce the question decomposition tree as a framework for LLM reasoning, enabling atomic fact retrieval on KG for each reasoning step.
- We design efficient strategies for atomic retrieval to extract and retrieve question-relevant KG subgraph of facts, expanding the candidate subgraph to a pruned two-hop range.
- Experimental results demonstrate that our framework effectively enhances the reasoning performance of LLMs in a low-cost setting without additional training or fine-tuning. Moreover, it outperforms existing state-of-the-art reasoning-based baselines.

2 Related Work

2.1 LLM Prompting

To stimulate the reasoning capability of LLMs, various prompting strategies have been proposed. The chain-of-thought (CoT) (Wei et al., 2022) is designed to enable LLMs to reason more reliably and interpretably. After this, some works (Yao et al., 2023a; Besta et al., 2024) further introduced other prompting structures to support LLM reasoning. Others (Wang et al., 2023b; Zhou et al., 2023; Khot et al., 2023) guide LLMs to generate plans to decompose tasks and solve them step-by-step. Additionally, some work (Wang et al., 2023c) improves the greedy decoding strategy in CoT by sampling a set of reasoning paths from the LLM and selecting the most consistent answer. In order to tackle the problem that LLMs often lack external knowledge and exhibit hallucinations, some works (Yao et al., 2023b; Wang et al., 2023a) combine LLM reasoning with external knowledge sources, gathering additional information to mitigate the hallucination and error propagation in CoT, thereby producing accurate and reliable reasoning.

2.2 Knowledge Graph Question Answering (KGQA)

Existing works on KGQA can be broadly categorized into two types: Semantic Parsing (SP)-based methods and Information Retrieval (IR)-based methods. **SP-based methods** (Sun et al., 2020; Chen et al., 2021; Ye et al., 2022) first parse the semantics of the input question to transform it into a logical form (LF) such as SPARQL or S-expression. Then execute LF on the KG to query for the answer. To avoid the inability to obtain an answer due to LFs that are not executable, DecAF (Yu et al., 2023) combines the direct answer from LLM to generate the final answer. **IR-based methods** retrieve question-relevant facts from the KG, and then reason to answer the question according to the retrieved facts. Early works (Miller et al., 2016; Sun et al., 2018, 2019; He et al., 2021) utilize structures like key-value memory network and graph neural network (GNN) to encode the knowledge in KG and perform KG reasoning and retrieval. Some works (Shi et al., 2021; Zhang et al., 2022a; Jiang et al., 2023c,b) leverage pretrained language models (PLMs) for tasks such as question encoding, KG relation retrieval, and semantic matching

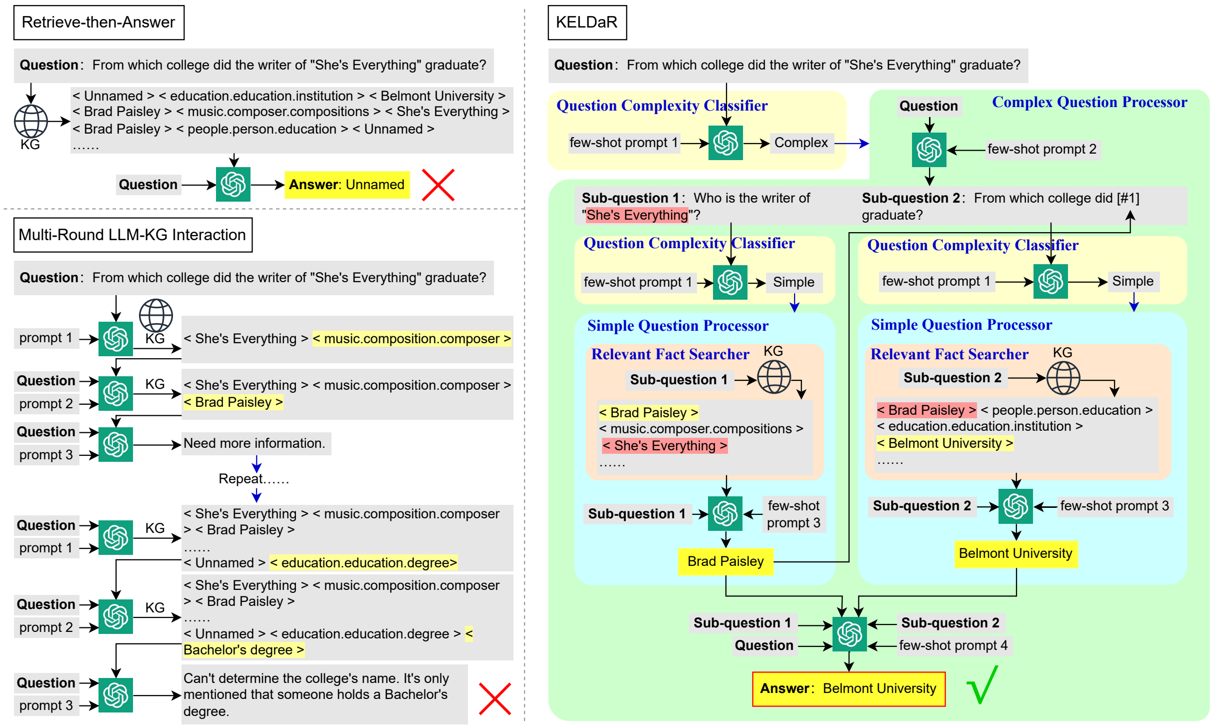

Retrieve-then-Answer
Question: From which college did the writer of "She's Everything" graduate?
< Unnamed > < education.education.institution > < Belmont University >
< Brad Paisley > < music.composer.compositions > < She's Everything >
< Brad Paisley > < people.person.education > < Unnamed >
KG
Question
Answer: Unnamed
Multi-Round LLM-KG Interaction
prompt 1
prompt 2
prompt 3
Question
KG
< She's Everything > < music.composition.composer >
< She's Everything > < music.composition.composer > < Brad Paisley >
Need more information.
Repeat……
< Unnamed > < education.education.degree>
< Unnamed > < education.education.degree > < Bachelor's degree >
Can't determine the college's name. It's only mentioned that someone holds a Bachelor's degree.
KELDaR
Question Complexity Classifier
few-shot prompt 1
Complex
Complex Question Processor
few-shot prompt 2
Sub-question 1: Who is the writer of "She's Everything"?
Sub-question 2: From which college did [#1] graduate?
Simple
Simple Question Processor
Relevant Fact Searcher
< Brad Paisley > < music.composer.compositions > < She's Everything >
< Brad Paisley > < people.person.education > < education.education.institution > < Belmont University >
few-shot prompt 3
Brad Paisley
Belmont University
few-shot prompt 4
Answer: Belmont University

Figure 1: Overview of previous methods and our proposed framework KELDaR.

between questions and relations, aiding in information retrieval and question reasoning. With the advancement of LLMs, recent works utilize LLMs for KG retrieval and subsequent reasoning. StructGPT (Jiang et al., 2023a) and ToG (Sun et al., 2024) design LLM-based methods for entity and relation retrieval on KG, and continue to invoke LLM for reasoning based on retrieval results. Answers to the questions are obtained through the retrieving and reasoning process iteratively. RoG (LUO et al., 2024) employs the LLM to generate KG relation chains as reasoning plans, which are then used for KG retrieval, followed by LLM-based reasoning to generate answers based on retrieval results.

3 Preliminary

Knowledge Graph (KG) can be formalized as a collection of factual triples composed of entities and relations, represented as $\mathcal{G} = \{(e_h, r, e_t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where e_h, r, and e_t denote the head entity, relation, and tail entity respectively; $\mathcal{E}$ and $\mathcal{R}$ denote the sets of all entities and relations in KG respectively.

Knowledge Graph Question Answering (KGQA) is a task that involves answering specific questions based on factual information from a KG. Given a question q and a KG $\mathcal{G}$, topic entity $t^q \in T^q$ can be extracted from the question q, and golden answers to the question is $A^q = \{a^q\}$. Both topic entities and golden answers correspond to entities in KG, i.e., $T^q, A^q \subseteq \mathcal{E}$. The goal of KGQA is to design a model f that predicts the answer a^q to the question q based on the knowledge in the KG $\mathcal{G}$, formulated as $a^q = f(q, \mathcal{G})$.

[Task description and requirements]
Question: [question of sample 1]
(Other items)
Answer: [answer of sample 1]
……
Question: [question to be processed]
(Other items)
Answer:

Figure 2: Illustration of the few-shot prompts for LLM in our proposed framework. The part enclosed by "( )" is optional.

4 Method

4.1 Overview

Our framework first employs the *Question Complexity Classifier* (see Section 4.2) to classify the complexity of input questions. For complex questions, the framework uses *Complex Question Processor* (see Section 4.3) to decompose them and solve the sub-questions with our framework iteratively, then integrate them to answer the original question. For simple questions, it directly infers the answer using the *Simple Question Processor* (see Section 4.4) with the assistance of the *Relevant Fact Searcher* (see Section 4.5) retrieving relevant KG facts. The detailed framework workflow is presented in Algorithm 1. Figure 1 (right) illustrates an example of how our framework processes a complex question. The structure of the few-shot prompts for LLM in our framework is shown in Figure 2, where the content at "Other items" is determined according to the needs of different modules.

4.2 Question Complexity Classifier

The Question Complexity Classifier determines the complexity of the input question q and categorizes it into a question type c^q, formalized as:

$$c^q = \text{Classifier}(q), \tag{1}$$

where $c^q \in \{\texttt{Simple}, \texttt{Complex}\}$. The backbone of *Classifier* is a few-shot prompted LLM. It generates a determination of either `Simple` or `Complex`, which directs the question to different subsequent modules for further processing. See Appendix A.2 for why we choose to implement *Classifier* by prompting LLM.

The "Answer" of each sample question in the prompt is its class annotated manually, which is guided by the SPARQL provided in the dataset and our own understanding of the question's semantics. The specific classification criteria are: if the question involves the logic of composition thus requires multi-step reasoning, it's complex. Otherwise, if it doesn't involve composition structures and only needs single-step reasoning, it's simple.

4.3 Complex Question Processor

We formalize the Complex Question Processor as:

$$a^q = \text{CompQProcessor}(q, \mathcal{G}), \tag{2}$$

where q is the input complex question, $\mathcal{G}$ is the KG, and the derived answer is a^q. This module is designed to process questions classified as `Complex` by the Question Complexity Classifier and involves three steps: decomposition, handling, and integration.

Complex Question Decomposition In this part, the input problem q is decomposed into a series of sub-questions Sq^q using multi-step reasoning, which can be formalized as:

$$Sq^q = \text{Decomposer}(q). \tag{3}$$

$Sq^q = [sq^q_1, ..., sq^q_l]$, where sq^q_i denotes the sub-question of the i-th reasoning step and l is the length of Sq^q. The backbone of *Decomposer* is a few-shot prompted LLM, which develops a multi-step reasoning plan for the given complex question in the form of sub-questions.

The "Answer" of each sample in the prompt is a sequence of sub-questions crafted manually. Each sub-question i conforms to $sq^q_i =< w_1, ..., w_{|sq^q_i|} >$, in which $w_k \in \mathcal{W} \cup \mathcal{L}_i$. Here, $\mathcal{W}$ represents the set of English words and punctuations; $\mathcal{L}_i = \{\texttt{[\#j]} | 1 \le j < i, j \in \mathbb{Z}\}$, where `[#j]` refers to the answer tag of previous sub-question j.

Sub-questions Handling Obtaining a series of sub-questions, they should be answered sequentially since the reasoning steps are ordered, which is formalized as:

$$\begin{aligned}
\widetilde{sq^q_1} &= sq^q_1, \\
a^{\widetilde{sq^q_1}} &= \text{KELDaR}(\widetilde{sq^q_1}, \mathcal{G}), \\
\widetilde{sq^q_2} &= \mathit{replace}(sq^q_2, Sa^q_{<2}), \\
a^{\widetilde{sq^q_2}} &= \text{KELDaR}(\widetilde{sq^q_2}, \mathcal{G}), \\
&\vdots \\
\widetilde{sq^q_l} &= \mathit{replace}(sq^q_l, Sa^q_{<l}), \\
a^{\widetilde{sq^q_l}} &= \text{KELDaR}(\widetilde{sq^q_l}, \mathcal{G}),
\end{aligned} \tag{4}$$

where $Sa^q = [a^{\widetilde{sq^q_1}}, ..., a^{\widetilde{sq^q_l}}]$, representing the list of answers to sub-questions. l is the length of sub-question list.

Since the sub-question might depend on answers to previous sub-questions, it needs to be preprocessed with the function $\mathit{replace}$. It replaces each tag `[#k]` in sub-question sq^q_i with corresponding answer $a^{\widetilde{sq^q_k}}$ in answers $Sa^q_{<i}$ to previous sub-questions, yielding a comprehensible natural language question $\widetilde{sq^q_i}$. It is then fed to our framework KELDaR as a new question to infer its answer $a^{\widetilde{sq^q_i}}$.

Sub-questions Integration In this part, we integrate all sub-questions and their answers to infer the answer a^q to the original complex question q. This process can be formalized as:

$$a^q = \text{Integrator}(q, \widetilde{Sq^q}, Sa^q), \tag{5}$$

where the list of sub-questions is $\widetilde{Sq^q} = [\widetilde{sq^q_1}, ..., \widetilde{sq^q_l}]$ and a corresponding list of answers is $Sa^q = [a^{\widetilde{sq^q_1}}, ..., a^{\widetilde{sq^q_l}}]$. l is the length of sub-question list. *Integrator* adopts a few-shot prompted LLM as its backbone, whose prompt contains samples of "question - sub-questions and answers - reasoning process and results". Given the sub-QAs for a new question, LLM constructs the reasoning chain and generates the answer a^q to the original complex question.

We find that there are inevitably some problems with the results generated by the *Decomposer* and sub-question handling. To avoid the bad effects to our framework caused by the error propagation through two previous steps, we design several examples in the LLM prompt of this part to handle these errors, guiding LLM to select useful references and combine its own knowledge when necessary to complete the reasoning.

4.4 Simple Question Processor

This module is designed to handle questions classified as `Simple` by the Question Complexity Classifier. It can be formalized as:

$$a^q = \text{SimpQProcessor}(q, \mathcal{G}), \tag{6}$$

where q is the input simple question, $\mathcal{G}$ is the KG, and a^q is the derived answer. For input questions, this module first invokes the Relevant Fact Searcher (see Section 4.5) to retrieve a set of relevant facts Ref^q from the KG. These facts serve as references to aid reasoning, formalized as:

$$a^q = \text{Answerer}(q, Ref^q). \tag{7}$$

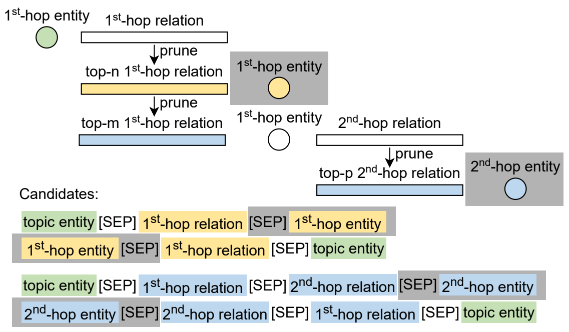
The backbone of *Answerer* is a few-shot prompted LLM whose prompt contains samples of "question - relevant KG facts - reasoning process and answer". For new questions, LLM generates a single-step reasoning process and the answer a^q based on the provided relevant facts.

To deal with cases that facts retrieved by the Searcher are insufficient to infer the answer, we include several examples in the LLM prompt with imperfect results of Searcher. These examples guide the LLM to assess the usability of the reference facts and, when necessary, draw on LLM's own knowledge to complete reasoning.

4.5 Relevant Fact Searcher

The Relevant Fact Searcher queries the KG $\mathcal{G}$ for facts relevant to the input question q to provide reference knowledge for the Simple Question Processor. It can be formalized as:

$$Ref^q = \text{Searcher}(q, \mathcal{G}), \tag{8}$$



1st-hop entity
1st-hop relation
prune
top-n 1st-hop relation
1st-hop entity
prune
top-m 1st-hop relation
1st-hop entity
2nd-hop relation
prune
top-p 2nd-hop relation
2nd-hop entity
Candidates:
topic entity [SEP] 1st-hop relation [SEP] 1st-hop entity
1st-hop entity [SEP] 1st-hop relation [SEP] topic entity
topic entity [SEP] 1st-hop relation [SEP] 2nd-hop relation [SEP] 2nd-hop entity
2nd-hop entity [SEP] 2nd-hop relation [SEP] 1st-hop relation [SEP] topic entity

Figure 3: Illustration of two strategies of extracting candidate subgraph of facts. The **triple/quadruple-based strategy** can be represented by all parts of the figure, resulting in candidates of triples and quadruples with first and second-hop entities. The **relation-based strategy** can be represented by the remaining parts of the figure excluding the greyed-out areas, leading to candidates that don't include first and second-hop entities.

where Ref^q represents the set of retrieved facts. *Searcher* operates in three steps: topic entity extraction, candidate subgraph extraction, and relevant fact retrieval.

Topic Entity Extraction To a single question, the majority of the data in KG is irrelevant. They introduce noise into KG retrieval and increase its time and space cost, interfering with the retrieval and subsequent reasoning (Ding et al., 2024). Therefore, we first extract topic entities of question q as the source of candidate subgraph extraction from the extensive KG, which is formalized as:

$$T^q = \text{TopicExtractor}(q), \tag{9}$$

where $T^q = \{t^q\} \subseteq \mathcal{E}$ is the extracted topic entity set, with each entity corresponds to one of KG entities $\mathcal{E}$.

We implement two methods for topic entity extraction depending on whether golden topics are used. For method not using golden topics, we employ existing entity linker for extraction. For method using golden topics, we extract all entity IDs from the SPARQL annotated in the dataset as the golden topic set. Considering that this golden set doesn't cover intermediate entities in multi-step reasoning for complex questions, the topics of sub-questions may not be included. So we further combine results of entity linker with the golden set.

Candidate Subgraph Extraction Based on the topic entities, this part focuses on extracting a subgraph from the KG, which contains question-relevant facts and should be as small as possible, serving as the set of candidate facts. This part can be formalized as:

$$Cand^q = \text{CandExtractor}(q, T^q, \mathcal{G}), \qquad (10)$$

where $Cand^q$ is the set of candidate facts extracted from the KG neighborhoods of topic entities T^q.

We designed two fact extraction strategies for different retrieval objects: triple/quadruple-based and relation-based. Both strategies extract relations and entities within a two-hop range of the topic entities, for reasons discussed in Appendix A.3.

The triple/quadruple-based extraction strategy is illustrated in Figure 3. From the topic entity t^q, we prune first and second-hop relations in KG according to their relevance to question q. Note that only for the first-hop entities without natural language names, we further extract the second-hop. With the format of triple/quadruple in Figure 3, we construct the two-hop pruned relations and entities to form the candidates $Cand^q_{t^q}$. By merging the candidates extracted for each topic entity, we obtain the candidate subgraph in the form of a fact list, denoted as $Cand^q = \cup Cand^q_{t^q}, \forall t^q \in T^q$.

The relation-based extraction strategy focuses more on relations, illustrated by parts excluding the greyed-out areas in Figure 3, which doesn't include first and second-hop entities in the extracted facts. The rest of the extraction process follows similar steps as the triple/quadruple-based strategy.

Relevant Fact Retrieval From the candidate facts $Cand^q$, we further retrieve a few facts that are most relevant to the input question q, providing precise references for LLM's reasoning. We follow the method of "retrieval-then-reranking" proposed by Baek et al. (2023b). With a retriever and a reranker, we retain the top K_{rt} and K_{rr} relevant facts to question q respectively, as represented in equations (11).

$$\begin{aligned} Res^q &= \text{Retriever}(q, Cand^q, K_{rt}), \\ Ref^q &= \text{Reranker}(q, Res^q, K_{rr}). \end{aligned} \qquad (11)$$

The retrieval results $Ref^q = \{ref^q_1, ..., ref^q_{K_{rr}}\}$ are used as references for further reasoning.

For the relation-based retrieval, it's necessary to complement each retrieved fact with the corresponding first or second-hop entities to construct a complete reference fact.

5 Experiments

5.1 Datasets and KG

To evaluate the performance of KELDaR, we employed two commonly used KGQA datasets: WebQuestionsSP (WebQSP) (Yih et al., 2016) and ComplexWebQuestions (CWQ) (Talmor and Berant, 2018). In WebQSP and CWQ, each question is annotated with a SPARQL, which can be executed on KG to query for its golden answers. WebQSP contains 4,737 QA pairs. CWQ is built on top of WebQSP by making the original questions more complex through methods such as expanding the entities in original questions into sub-questions and adding constraints to the answers of original questions. This results in 34,689 QA pairs of four combination types: composition, conjunction, comparative, and superlative. The statistics of the number of examples in the training and test sets of both datasets are shown in Table 1.

Dataset	Train	Test
WebQSP	3098	1639
CWQ	27639	3531

Table 1: Statistics of example distribution in the experimental datasets.

Both datasets are based on Freebase KG (Bollacker et al., 2008). We use the Freebase preprocessed with the method in Lan and Jiang (2020) for all experiments. This method only extracts triples with entities in Freebase ID, English text, or numeric format from the complete Freebase.

5.2 Baselines and Our Frameworks

For baselines, **training-based methods** optimize the proposed models on specific downstream training data by fine-tuning or training PLMs, LLMs, or retrievers. We select KV-Mem (Miller et al., 2016), GraftNet (Sun et al., 2018), PullNet (Sun et al., 2019), EmbedKGQA (Saxena et al., 2020), TransferNet (Shi et al., 2021), NSM (He et al., 2021), KGT5 (Saxena et al., 2022), SR+NSM+E2E (Zhang et al., 2022a), TIARA (Shu et al., 2022), KD-CoT (Wang et al., 2023a), UniKGQA (Jiang et al., 2023c), ReasoningLM (Jiang et al., 2023b), DecAF (Yu et al., 2023) and RoG (LUO et al., 2024). **Reasoning-based methods** directly utilize existing models including LLMs without any additional training or fine-tuning. We select IO (Brown et al., 2020), CoT (Wei et al., 2022), CoT+SC (Wang et al., 2023c), StructGPT (Jiang et al., 2023a), KB-BINDER (Li et al., 2023) and ToG (Sun et al., 2024).

According to our two proposed fact extraction and retrieval strategies, and whether to use golden topics, our framework is evaluated under four settings. **KELDaR** and **KELDaR*** employ the triple/quadruple-based strategy, while **KELDaR-rel** and **KELDaR-rel*** utilize the relation-based strategy. * indicates the usage of golden topics.

5.3 Evaluation Metric

Following previous work (Wang et al., 2023a; Jiang et al., 2023a; Li et al., 2023; Sun et al., 2024), we employs Exact Match (EM) as the evaluation metric in our experiments.

Considering that it's challenging to locate the answer entities precisely in LLM's outputs when it isn't marked explicitly with "{ }" as expected, we adopt the evaluation method in Sun et al. (2024). If LLM's output contains complete answer markers, we extract the answer directly and match it with the golden answer. Otherwise, the entire LLM output is considered as the searching scope. We regard the result as correct if it contains the golden answer.

5.4 Implementation Details

In the main experiments, we utilize gpt-3.5-turbo API (OpenAI, 2022) as the LLM in our proposed framework, with the temperature set to the default value of 1.0, and the maximum token length for generation set to 100. The numbers of samples in few-shot prompts is as follows: 20 samples each for simple and complex questions in Question Complexity Classifier, 6 for Decomposer and 10 for Integrator in Complex Question Processor, and 4 for Simple Question Processor. Additionally, to keep consistency with our framework's settings, the implementation of IO, CoT, and CoT+SC prompting strategies in main experiments all utilize the gpt-3.5-turbo API as the LLM and employ 4 shots.

In Relevant Fact Searcher, we employed ELQ (Li et al., 2020) as the entity linker, distilbert[1] as the retriever, and MiniLM[2] as the reranker. The number of facts K_{rt} retained by the retriever is set to 20, while the number of facts K_{rr} retained by the reranker is set to 5 by default.

For the maximum question decomposition depth in our framework, we set it to 1. For further implementation details, please refer to Appendix B.

5.5 Main Results

The main results of our experiments are shown in Table 2. Compared to other reasoning-based methods, our proposed frameworks in all four settings show significant improvements over both the IO prompting method, representing the original few-shot performance of GPT-3.5, and the previous state-of-the-art reasoning-based methods. Specifically, on WebQSP and CWQ, KELDaR-rel* improves accuracy by 14.6% and 19.2% respectively compared to the IO method, and by 5.0% and 12.0% compared to previous best reasoning-based methods. These results demonstrate that our proposed framework can effectively leverage the KG to enhance LLM's reasoning significantly, outperforming the state-of-the-art reasoning-based results. Particularly for the more challenging complex QA in CWQ, our framework mitigates the limitations of existing reasoning-based methods effectively.

Compared to most training-based methods, our framework achieves similar or superior performance without requiring additional training or fine-tuning. We directly invokes frozen models of LLM, entity linker, retriever and reranker for reasoning. This indicates that our framework is practical and competitive on KGQA, saving additional costs as-

Types	Methods	WebQSP	CWQ
Training-based	KV-Mem	46.7	18.4
	GraftNet	66.4	36.8
	PullNet	68.1	45.9
	EmbedKGQA*	66.6	-
	TransferNet	71.4	48.6
	NSM*	74.3	48.8
	KGT5*	56.1	36.5
	SR+NSM+E2E*	69.5	49.3
	TIARA*	75.2	-
	KD-CoT	68.6	55.7
	UniKGQA*	77.2	51.2
	ReasoningLM*	78.5	69.0
	DecAF	82.1	**70.4**
	RoG*	**85.7**	62.6
Reasoning-based	StructGPT*	72.6	-
	KB-BINDER	74.4	-
	ToG* ‡	63.2	29.2
	IO †	64.8	34.4
	CoT †	70.3	40.2
	CoT+SC †	71.4	41.6
	KELDaR	75.5	41.9
	KELDaR*	76.0	50.7
	KELDaR-rel	76.7	44.2
	KELDaR-rel*	**79.4**	**53.6**

Table 2: Main results (EM in percent) of our frameworks and baselines on KGQA. * indicates the usage of the golden topics. † indicates the results obtained by reproducing the methods. ‡ indicates the results obtained by running the source code provided by the work on the LLM used in our study. The best results in each type are in **bold** and second-best results are underlined.

[1]https://huggingface.co/sentence-transformers/msmarco-distilbert-base-v3

[2]https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-12-v2

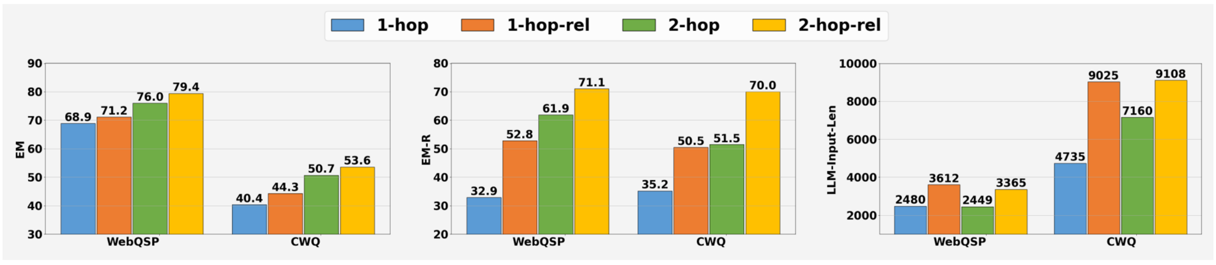


Figure 4: Experimental results on strategies of fact extraction and retrieval on KG. **1-hop** and **2-hop** indicate the triple/quadruple-based fact extraction and retrieval strategies with candidate extraction ranges of 1 hop and 2 hops respectively on KG. **1-hop-rel** and **2-hop-rel** indicate the relation-based strategies. **EM-R** represents the accuracy of the fact retrieval in the framework. **LLM-Input-Len** is the average length of LLM input tokens for all questions.

Methods	WebQSP	CWQ
KELDaR	**75.5**	**41.9**
w/o CQP	75.0	40.9
KELDaR*	**76.0**	**50.7**
w/o CQP	72.3	46.8
KELDaR-rel	76.6	**44.2**
w/o CQP	**77.9**	39.0
KELDaR-rel*	**79.4**	**53.6**
w/o CQP	79.1	47.2

Table 3: Ablation study results (EM in percent) on the strategy for complex question processing (CQP). The better results in each setting of our methods are in **bold**.

sociated with training and fine-tuning while still providing robust performance.

In order to see how our method works on different LLMs, we further conduct experiments with GPT-4 (see Appendix C).

5.6 Further Analysis

5.6.1 Analysis on the strategy of complex question processing

To analyze the impact of our complex question processing (CQP) strategy, we conduct an ablation study by removing the Complex Question Processor and fixing the output of Question Complexity Classifier to `Simple`. The performance of this modified framework is compared to our original framework. Results are presented in Table 3. Our CQP strategy improves the framework's performance generally, demonstrating its positive impact.

Considering that most questions are complex in CWQ while are simple in WebQSP, our strategy is naturally more suited for CWQ, confirmed by the results on CWQ in Table 3. For questions in WebQSP, they could sometimes be misclassified as `complex` by the Question Complexity Classifier, leading to redundant subsequent steps and additional errors. That's why the KELDaR-rel without CQP performs better on WebQSP. But in other three cases on WebQSP, the strategy has a positive effect. This shows the robustness of our framework in reducing the impact of Classifier's error, ensuring that our CQP strategy enhances the performance on complex questions while having minimal negative impact on simple QA.

5.6.2 Analysis on strategies of fact extraction and retrieval on KG

To reduce the impact of accuracy in topic entity extraction, we conducted experiments using the golden topics, in order to analyze the influence of our fact extraction and retrieval strategies.

As shown in Figures 4 (left and center), our proposed two-hop fact extraction strategies significantly outperforms the one-hop settings in both triple/quadruple-based and relation-based settings. This leads to a great increase in the accuracy of retrieving relevant facts, providing more valuable references for subsequent reasoning and enhancing the overall performance effectively. These findings manifest the critical importance of the extraction strategies in our frameworks.

Comparing two fact extraction and retrieval strategies we designed, the results in Figure 4 illustrate that the relation-based strategy, compared to the triple/quadruple-based strategy, requires longer LLM prompts but achieves better retrieval and overall performance. The relation-based strategy can contain more entities in references, resulting in longer LLM prompts to provide richer contextual information for the reasoning.

5.6.3 Analysis on effects of varying number K_{rr} of references

When answering simple questions, the number of references provided in the LLM prompt corre-

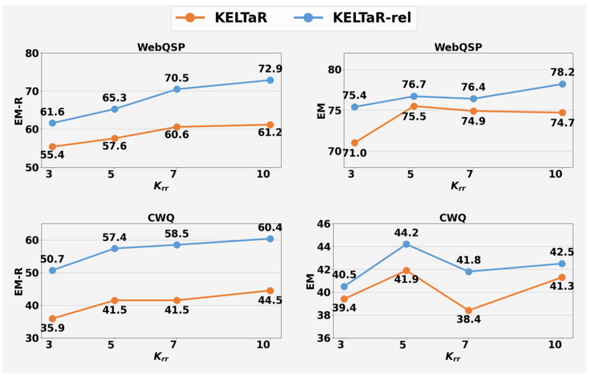


Figure 5: Experimental results under different number of references. **EM-R** represents the accuracy of the fact retrieval in the framework.

sponds to the number of facts retained after reranking in Relevant Fact Searcher. The results for different values of this parameter are shown in Figure 5. As the number of references increases, there is a steady rise in the accuracy of fact retrieval naturally. However, this does not necessarily lead to an improvement in overall accuracy, indicating that the additional correct references are not utilized by the LLM effectively and may even cause a decline in overall performance in some cases.

We suspect that this phenomenon can be attributed to the generative nature of the gpt-3.5-turbo used in the experiments. As the length of references for new question in the prompt increases, the distance between previous QA examples and the end of prompt also increases, which may cause LLM to forget patterns learned from examples on how to infer the answers to the questions based on references. Consequently, this leads to a negative impact on LLM's ability to answer the given question effectively.

5.6.4 Analysis on the Reasoning Time of Our Method

The reasoning time of our framework primarily depends on the time cost of the KG Relevant Fact Searcher and the number of LLM calls. Assuming that on average each complex question is decomposed into k sub-questions. In our experiments, approximately 20.5% and 96.3% of the questions in WebQSP and CWQ, respectively, are classified to be complex. The average number of sub-questions k for these complex questions was approximately 2.03 and 2.37 in WebQSP and CWQ, respectively. And we set the maximum depth D of the decomposition to 1 in our experiments, meaning each question is decomposed at most once, considering the limited complexity of questions in the two datasets.

Time Cost of Relevant Fact Searcher We found that in our experiments, each call to the Relevant Fact Searcher costs 27.3s on average. However, about 80% of the calls took less than 20s, with the remaining calls taking longer due to the larger number of relations and entities within two hops of the topic entity in the KG. For a complex question, each sub-question resulting from decomposition requires one call to the Searcher to retrieve KG facts, resulting in k calls in total. And a simple question requires only one call to the searcher.

Number of LLM Calls For complex questions, the LLM is first called once in the Question Complexity Classifier. Then, for solving each sub-question, the LLM is called once in both the Classifier and the Simple Question Processor, requiring a total of $2k$ LLM calls for all sub-questions. Finally, integrating the sub-questions and their answers to derive the original question's answer requires one more LLM call. Thus, complex questions require $2k+2$ LLM calls in total. Simple questions require only two LLM calls, one in the Classifier and one in the Simple Question Processor.

Our method does not require training and employs multithreading to reduce reasoning time. To complete the reasoning for 1639 questions in WebQSP and 3531 questions in CWQ, our framework took 2.83h and 14.82h, respectively. The longer time on CWQ is mainly due to the larger number of questions and their higher complexity compared to WebQSP.

6 Conclusion

This study introduces a framework of KG-enhanced LLM based on question decomposition and atomic retrieval, called KELDaR. We introduce the question decomposition tree as a framework for LLM reasoning, which allows atomic KG retrieval targeting each reasoning step. Additionally, we designed strategies of atomic retrieval to extract and retrieve question-relevant KG facts, expanding the range of candidate subgraph to two hops. Experimental results demonstrate that our proposed framework significantly outperforms the state-of-the-art reasoning-based methods.

Limitations

Although our proposed framework KELDaR demonstrates excellent performance in KGQA datasets, it still has some limitations. First, the performance of our framework varies across different composition types of complex questions, particularly struggling with superlative questions. Future work could explore new question decomposition strategies to improve the decomposition paradigms for these challenging types. Second, it's also required to investigate more controllable methods on complex question decomposition than that in our framework, aiming to make the sub-questions generated by the Decomposer more reliable. Additionally, due to the high cost of the GPT-4 API, we hasn't conduct complete experiments on GPT-4. Thus, broader experiments are required to evaluate the applicability of our framework with GPT-4 and other LLMs.

Ethics Statement

Our proposed framework KELDaR perform question answering with LLM reasoning and KG retrieval, achieving excellent performance. However, its question answering and reasoning capabilities are still not perfect, and it may make mistakes in KG retrieving by retrievers and answer generating by the LLM. Consequently, as for domains with high precision requirements, the results of our framework should be used carefully, in order to avoid the risk of adverse consequences. It is required to further verify the results generated by the framework to correct the errors for application in areas that need high precision.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (Grant No. 2022YFC3302100) and the National Natural Science Foundation of China (Grant No. 62476025).

References

Jinheon Baek, Alham Aji, and Amir Saffari. 2023a. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *Proceedings of the First Workshop on Matching From Unstructured and Structured Data (MATCHING 2023)*, pages 70–98.

Jinheon Baek, Alham Fikri Aji, Jens Lehmann, and Sung Ju Hwang. 2023b. Direct fact retrieval from knowledge graphs without entity linking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10038–10055, Toronto, Canada. Association for Computational Linguistics.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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*, volume 33, pages 1877–1901. Curran Associates, Inc.

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. ReTraCk: A flexible and efficient framework for knowledge base question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 325–336, Online. Association for Computational Linguistics.

Wentao Ding, Jinmao Li, Liangchuan Luo, and Yuzhong Qu. 2024. Enhancing complex question answering over knowledge graphs through evidence pattern retrieval. In *Proceedings of the ACM on Web Conference 2024*, WWW '24, page 2106–2115, New York, NY, USA. Association for Computing Machinery.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 553–561, New York, NY, USA. Association for Computing Machinery.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023a. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Xin Zhao, Yaliang Li, and Ji-Rong Wen. 2023b. 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*, pages 3721–3735, Singapore. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023c. UniKGQA: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974, Online. Association for Computational Linguistics.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6433–6441, Online. Association for Computational Linguistics.

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

LINHAO LUO, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Alexander 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*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. Gpt-4.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828, Dublin, Ireland. Association for Computational Linguistics.

Apoorv Saxena, Aditay Tripathi, and Partha 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*, pages 4498–4507, Online. Association for Computational Linguistics.

Priyanka Sen, Sandeep Mavadia, and Amir Saffari. 2023. Knowledge graph-augmented language models for complex question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 1–8.

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

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

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William 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*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph:

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

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. Sparqa: Skeleton-based semantic parsing for complex questions over knowledge bases. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8952–8959.

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

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

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023b. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

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

Yujie Wang, Hu Zhang, Jiye Liang, and Ru Li. 2023d. Dynamic heterogeneous-graph reasoning with language models and knowledge representation learning for commonsense question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14048–14063, Toronto, Canada. Association for Computational Linguistics.

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

Yike Wu, Nan Hu, Sheng Bi, Guilin Qi, J. Ren, Anhuan Xie, and Wei Song. 2023. Retrieve-rewrite-answer: A kg-to-text enhanced llms framework for knowledge graph question answering. *ArXiv*, abs/2309.11206.

Lin F. Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. 2023. Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling. *IEEE Transactions on Knowledge and Data Engineering*, 36:3091–3110.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

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)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

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

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Jiajie Zhang, Shulin Cao, Tingjian Zhang, Xin Lv, Juanzi Li, Lei Hou, Jiaxin Shi, and Qi Tian. 2023. Reasoning over hierarchical question decomposition tree for explainable question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14556–14570, Toronto, Canada. Association for Computational Linguistics.

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

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022b. GreaseLM: Graph REASoning enhanced language models. In *International Conference on Learning Representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Additional Details and Discussions of Our Method

A.1 Algorithm of Our Proposed Framework KELDaR

The algorithm of our proposed framework KELDaR is shown in Algorithm 1.

Algorithm 1 KELDaR

Input: question q, KG $\mathcal{G}$, maximum depth D of question decomposition
Output: answer entity a^q

```
 1: $depth \leftarrow 0$
 2: $a^q \leftarrow$ KELDaR$(q, \mathcal{G}, D, depth)$
 3:
 4: function KELDaR$(q, \mathcal{G}, D, depth)$
 5:     $c^q \leftarrow$ Classifier$(q)$
 6:     if $c^q =$ "Simple" OR $depth = D$ then
 7:         $Ref^q \leftarrow$ Searcher$(q, \mathcal{G})$
 8:         $a^q \leftarrow$ Answerer$(q, Ref^q)$
 9:     else
10:         $Sq^q \leftarrow$ Decomposer$(q)$
11:         if $len(Sq^q) = 0$ then
12:             $Ref^q \leftarrow$ Searcher$(q, \mathcal{G})$
13:             $a^q \leftarrow$ Answerer$(q, Ref^q)$
14:         else
15:             $\widetilde{Sq^q} \leftarrow [\,]$
16:             $Sa^q \leftarrow [\,]$
17:             for $i = 0 \rightarrow len(Sq^q) - 1$ do
18:                 $sq_i^q \leftarrow Sq^q[i]$
19:                 $\widetilde{sq_i^q} \leftarrow replace(sq_i^q, Sa_{<i}^q)$
20:                 $\widetilde{Sq^q}.append(\widetilde{sq_i^q})$
21:                 $a^{\widetilde{sq_i^q}} \leftarrow$ KELDaR$(\widetilde{sq_i^q}, \mathcal{G}, D, depth + 1)$
22:                 $Sa^q.append(a^{\widetilde{sq_i^q}})$
23:             end for
24:             $a^q \leftarrow$ Integrator$(q, \widetilde{Sq^q}, Sa^q)$
25:         end if
26:     end if
27:     return $a^q$
28: end function
```

A.2 Selection of the Implementation Method for the Question Complexity Classifier

We choose to implement the Question Complexity Classifier by prompting LLM, because the classification task in our Classifier is not suitable to be completed through training a simple classifier. It is difficult to obtain labeled training data for this task. This task is used to determine the complexity of a question, with the criterion being whether the input question can be resolved through single-step or multi-step reasoning. This task requires classification based on the understanding and analyzing of the question's semantics, and does not exactly correspond to the single/multi-hop question classification standard and results. This is because "single-step" reasoning can correspond to either single-hop or multi-hop in a KG. For example, the questions "What do Jamaican people speak?" and "Who was Richard Nixon married to?" are both semantically understood as simple questions that can be resolved through single-step reasoning. However, the former corresponds to a single-hop fact in the KG, while the latter corresponds to a two-hop fact. Therefore, it is not feasible to label the complexity we need based on the single/multi-hop attribute of each question. Consequently, it is difficult to provide accurate "complexity" labels for each question as training data for the first step classification task. Hence, this classification step cannot be effectively completed by training a simple classifier, and we opted to use a prompted LLM instead.

A.3 Selection of the Fact Extraction Range in KG

In the Relevant Fact Searcher, we designed two strategies for extracting subgraphs of facts, with the extraction range set within two hops from the topic entity. This is because the questions to whom the Searcher provides reference facts are those classified as simple questions, which can be solved through one-step reasoning. The facts within two hops from the topic entity in the KG typically suffice to provide the necessary reference knowledge for such one-step reasoning questions.

Additionally, the reason for not extracting just one-hop facts lies in the presence of compound value type (CVT) entities in the KG used in this study. CVTs are used to represent n-ary relations in KG and don't have natural language names. Some one-step reasoning questions require traversing a CVT to reach the answer, necessitating an additional hop. For instance, as illustrated in Figure 6, the KG doesn't directly connect "Richard Nixon" and "Pat Nixon" through a spouse relation. Instead, it requires traversing two relations, "people.person.spouse_s" and "people.marriage.spouse", via a CVT to arrive at the answer. Therefore, to ensure that the reference facts necessary for simple questions are included in the candidate subgraph, we need to extract at least two-hop facts from the topic entity.

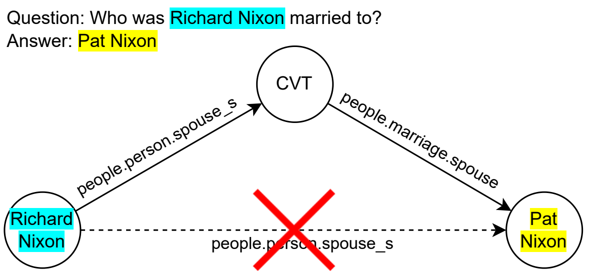


Figure 6: An example of the Compound Value Type (CVT) in KG.

B Additional Implementation Details of Experiments

For the entity linker ELQ (Li et al., 2020) used in our experiments, we set its threshold to -5 following the settings in Ye et al. (2022). In the candidate extraction step of the Searcher, we utilize distilbert as the retriever for the relation pruning based on its similarity with the question. During the pruning process, we retain $n = 10$ first-hop relations for candidate extraction, where $m = 5$ of them are used to continue the search for second-hop. The maximum number of entities linked to each first-hop relation for second-hop search was limited to 200. If it exceeds 200, we randomly select 200 entities. When pruning second-hop relations, we retained $p = 5$ relations. For the relation-based extraction and retrieval strategy, after reranking, when querying first-hop or second-hop entities based on the topic entities and relations in obtained facts, we set the maximum entity count to 20. If the length of the query result exceeded 20, we randomly select 20 entities while prioritizing those with natural language names, to control the length of the subsequent prompts provided to the LLM.

For the maximum question decomposition depth in our framework, we set it to 1. This decision was made based on the complexity of the questions in the two KGQA datasets used in our experiments. The questions in these datasets may involve up to 4-hops reasoning on KG, but there are compound value types (CVTs) in Freebase, and each reasoning step in our framework can go through CVTs and reference two hops in the KG. According to the decomposition trees corresponding to the SPARQL annotations in the datasets, complex questions in the datasets are all able to be decomposed into a series of simple questions through one decomposition step. Therefore, to avoid wasting resources, we set the maximum question decomposition depth D to 1, indicating that complex questions can be decomposed at most once.

Methods	**WebQSP**	**CWQ**
IO w/ GPT-3.5	64.8	34.4
IO w/ GPT-4	67.0	43.7
KELDaR-rel w/ GPT-3.5	76.7	44.2
KELDaR-rel* w/ GPT-3.5	79.4	53.6
KELDaR-rel w/ GPT-4	82.0	51.3
KELDaR-rel* w/ GPT-4	84.7	63.0

Table 4: Performance (EM in percent) of IO prompting method and our framework KELDaR-rel on different LLMs. * indicates the usage of the golden topics.

C Additional Experimental Results on GPT-4

In order to examine the effectiveness of our method on different LLMs, we conduct additional experiments using GPT-4 (gpt-4-turbo) (OpenAI, 2023). Due to the high cost of the GPT-4 API, we randomly selected 300 examples from each of the two datasets. And the experiments are conducted using both the IO prompt, representing the original few-shot performance of GPT-4, and our KELDaR-rel with the relation-based strategy. The results are shown in Table 4, which demonstrate that our framework achieves better performance on the superior GPT-4 model.

Moreover, by comparing with the performance of KELDaR-rel on GPT-3.5, we observe that even with the less powerful GPT-3.5, our KELDaR-rel outperforms the IO prompting method with the more powerful GPT-4. This further highlights the effectiveness of our framework in enhancing the performance of LLMs.