Finetuned Large Language Models as Decomposers: Step-by-Step Reasoning for Knowledge Base Question Answering

Anonymous ACL submission

Abstract

As semantic parsing and complex reasoning are fundamental to tackling complex question answering over knowledge bases (KBQA), the growing trend is to leverage large language models (LLMs), which exhibit outstanding semantic understanding and logical reasoning abilities, for this task. However, most of the existing LLM-based KBQA systems still operate as black boxes, unable to provide explanations for the derived results, motivating us to develop an interpretable and trustworthy KBQA system. In this paper, we innovatively introduce question templates as intermediary outcomes for the logical reasoning of LLMs to make the multi-step reasoning process of the existing KBQA system interpretable. Specifically, our method, named *Keqing*, first decomposes complex questions into simpler sub-questions according to predefined question templates using LLMs, and then addresses each sub-question by retrieving relevant information from knowledge bases or performing logical reasoning to achieve the final answer. To make *Keqing* more practical and trustworthy, we develop an automatic pipeline for question template construction to scale up the number of question templates at a low cost, and also incorporate the uncertainty estimation technique to provide confidence levels for the reasoning answers. Extensive experiments demonstrate that *Keqing* can achieve comparable performance to previous state-of-the-art methods and has better interpretability by rendering a step-by-step reasoning process.

1 Introduction

Aimed at locating the answer candidates for natural language questions from a specified knowledge base (KB), knowledge base question answering (KBQA) offers an attractive alternative for users to access vast amounts of structured information within large-scale KBs (Bollacker et al., 2008; Hoffart et al., 2011; Vrandečić and Krötzsch, 2014; Lehmann et al., 2015), and thus has gained widespread attention in both academic and industrial applications. As answering simple factoid questions becomes increasingly straightforward (Bordes et al., 2015; Hu et al., 2017; Petrochuk and Zettlemoyer, 2018) due to the advancements in deep neural models (Yin et al., 2016; Miller et al., 2016; Hao et al., 2017; Yu et al., 2017), the focus has shifted towards tackling complex questions that require multi-hop reasoning (Trivedi et al., 2017; Luo et al., 2018; Talmor and Berant, 2018; Cui et al., 2019; Cao et al., 2020), bringing new challenges of requiring sophisticated reasoning. Over the past years, approaches to solving complex KBQA have evolved into two dominant paradigms: 1) semantic parsing (SP)-based methods (Yih et al., 2015; Lan and Jiang, 2020; Chen et al., 2021c; Ye et al., 2022; Das et al., 2021) aim to translate the question into a logical form that can be executed against KBs, which typically relies on a powerful semantic parser implemented by advanced neural models; 2) information retrieval (IR)-based methods (Sun et al., 2018; He et al., 2021; Oguz et al., 2020) focus on extracting a question-related subgraph from KBs and subsequently perform complex reasoning over this subgraph.

Recently, with large language models (LLMs) (Brown et al., 2020; Chen et al., 2021a; Chowdhery et al., 2022) exhibiting outstanding capabilities in various reasoning-related tasks (Wei et al., 2022; Wang et al., 2022), there is an emerging trend to leverage them for KBQA. Capitalizing on the excellent generalization ability of LLMs to perform zero-shot and few-shot learning, a straightforward approach under the IR paradigm is to prepend the retrieved facts (*i.e.*, triples in the subgraph) from KBs to the target question (Baek et al., 2023), thereby forming a prompt to be forwarded to LLMs for generating the answer. Another approach under the SP paradigm leverages in-context learning to enable LLMs to generate valid logical forms for the target question by imitating a few demonstrations

(Gu et al., 2022; Li et al., 2023a; Tan et al., 2023). Both of these schemes are training-free and versatile across different knowledge domains, yet their performance cannot rival SOTA results without explicitly adapting LLMs to the formal structure of KBs. Therefore, some recent studies (Luo et al., 2023; Xu et al., 2023; Li et al., 2024) tend to focus on aligning the inherent knowledge of LLMs with the structured knowledge of KBs by fine-tuning LLMs on a moderate quantity of data consisting of pairs of complex questions and their formal programs. These fine-tuning based methods usually perform more competitively and generalize better to novel schema items (*e.g.*, relations) in KBs.

Despite achieving promising performance, the black-box nature of the working mechanisms behind these LLM-based approaches makes the reasoning process difficult to interpret. This hinders users from interacting with them to gain deeper insights into the decision-making process. In addition, the lack of confidence estimation for the deduced answers diminishes the reliability of these systems when deployed in real-world applications. Aimed at constructing an interpretable and trustworthy LLM-based KBQA system, we innovatively introduce question templates as intermediary outcomes for reasoning. This novel KBQA pipeline enables the decomposition of each complex question into a series of sub-questions and then we can perform logical reasoning on the knowledge graph by adjusting the logical chains used to solve these sub-questions, ultimately deriving the answer. Our contributions are summarized as follows:

- To make the multi-step reasoning process of the KBQA system interpretable, we innovatively introduce the question templates as intermediary outcomes for logical reasoning of LLMs and develop a novel interpretable KBQA system, named *Keqing*.
- To scale up the number of question templates and make it align with the huge amount of logical chains in KG, we develop an automatic pipeline to construct a collection of question templates that facilitate the mapping of natural language queries to the logical chains.
- We introduce the uncertainty estimation technique into *Keqing* to provide confidence levels for the reasoning answers, thereby making the reasoning process more trustworthy and facilitating friendly interaction with users.
- Abundant experiments demonstrate the superiority and effectiveness of our method over previous LLM-based approaches.

2 Related work

Existing literature (Lan et al., 2022) typically categorizes KBQA approaches into two mainstreams, *i.e.*, semantic parsing-based (SP-based) methods and information retrieval-based (IR-based) methods. Due to the page limitation, we have moved this section to the Appendix A.

3 *Keqing* for *K*nowledg*e* base *Q*uestion answer*ing*

3.1 Motivation of *Keqing*

The motivation of our work includes two folds:
1) Align the reasoning logic of LLMs with the logical chains of knowledge graph. To align the inherent knowledge of LLMs with the human-induced knowledge in knowledge graphs (KGs), existing methods either convert KGs into corpora, aiming for LLMs to acquire underlying structured knowledge through supervised fine-tuning (Oguz et al., 2020), or employ demonstration to enable LLMs to mimic the reasoning logic of exemplars (Li et al., 2023a). However, neither of these two approaches can guarantee an exact match between the reasoning logic of LLMs and the logical chains of KGs. This is because LLMs without logical alignment often treat multi-hop questions as single-hop ones or unnecessarily break down single-hop questions into multiple hops. While this may not always affect the inference result, it can lead to inconsistencies with human-induced reasoning logic and reduce interpretability.

Thus, to achieve logical alignment at the semantic level, rather than at the symbolic level where the reasoning logic in SQL form could be difficult for LLMs to understand (Cheng et al., 2022), we introduce question templates as intermediary outcomes for reasoning so that each complex question can be decomposed into a series of sub-questions. Then, by adjusting the logical chains of solving these sub-questions, we can perform logical reasoning on KG and ultimately derive the answers. The primary challenge we face is scaling up the number of question templates to make it align with the huge amount of logical reasoning chains in KG, which is also the issue to be addressed in Section 3.3.
2) Trustworthy/Interpretable LLM-based KBQA system. Existing LLM-based KBQA systems heav-

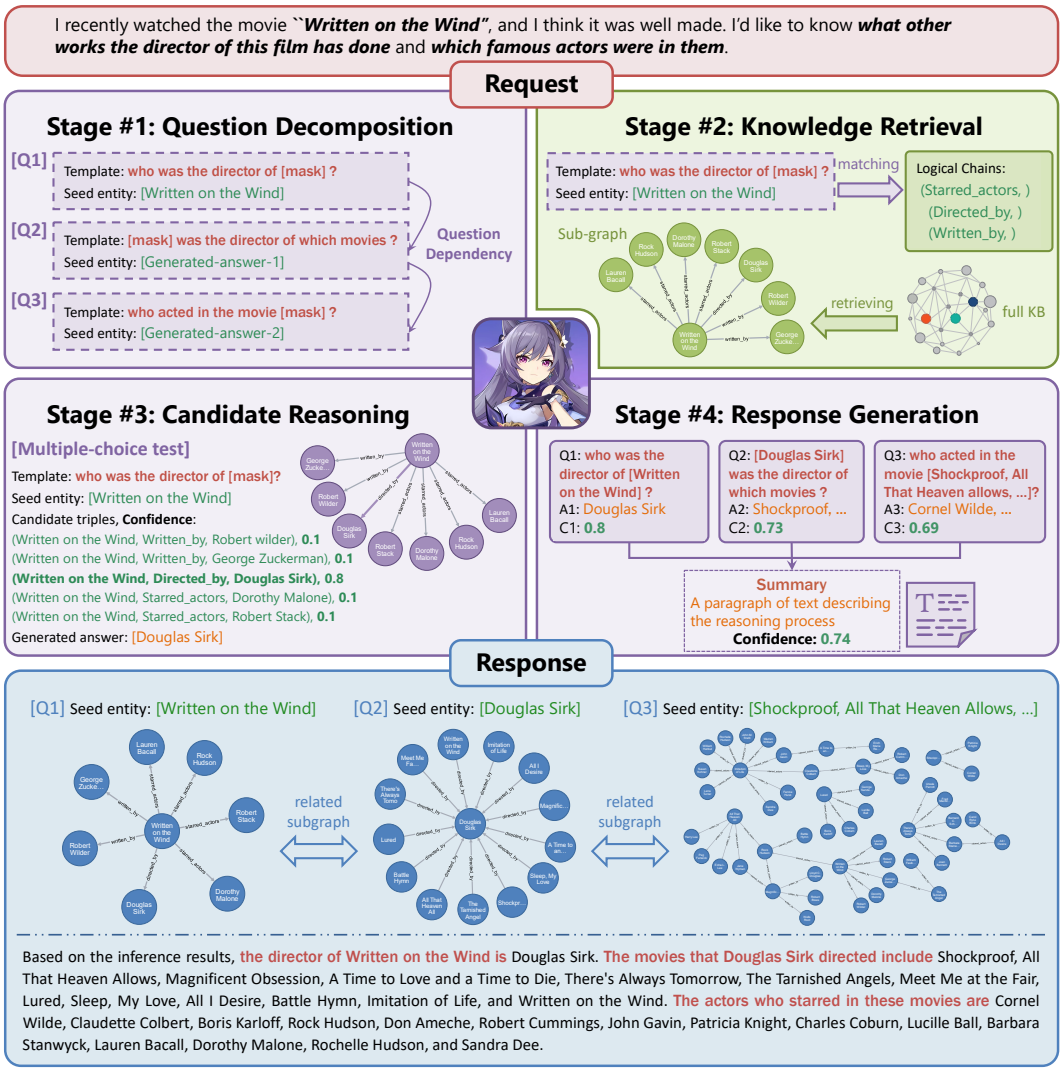

Figure 1: The workflow of *Keqing* applied for KBQA mainly consists of four stages: **#1** *Question Decomposition*: decompose a complex question into several sub-questions similar to predefined question templates; **#2** *Knowledge Retrieval*: retrieve candidate entities on the KG by aligning decomposed sub-questions to pre-collected logical chains; **#3** *Candidate Reasoning*: select the correct answer from the candidate answers to solve each sub-question; **#4** *Response Generation*: generate response by summarizing multiple rounds of questions and answers.

ily rely on the black-box logical reasoning of LLMs (Tan et al., 2023), which makes it difficult to interpret the reasoning process or even to provide confidence levels for the inference results. Moreover, if the causes of errors in KBQA system remain uninterpretable, *e.g.* the LLM only fails to understand the solution of a particular step in a multi-step reasoning chain, it becomes impossible for humans to make targeted adjustments to the current KBQA system, and current methods tend to continuously increase training data in hopes that the LLM will be able to perform automatic corrections (Gu and Su, 2022), which usually leads to a significant waste of computational resources.

As for *Keqing*, by introducing question templates for logical reasoning and then mapping each sub-question to the corresponding logical chain, it can intuitively explain the reasoning process of KBQA to users. This also facilitates user error identification, allowing them to determine whether errors are due to faults in LLMs' semantic-level logical reasoning or inappropriate logical chain design for the sub-questions. Moreover, *Keqing* can also provide confidence levels for the reasoning answers through the uncertainty estimation techniques in Section 3.3, making it more trustworthy.

3.2 Workflow of *Keqing*

Under the scenario of KBQA, given a natural language query q, the target of KBQA is to retrieve an answer list $\mathcal{A}$ from a symbolic KG denoted as $\mathcal{K}$ for the query q. Assuming a training set $\mathcal{D} = \{(q_i, \mathcal{A}_i)\}_{i=1}^{N}$ consisting of N question-answer pairs, an ideal KBQA model is supposed

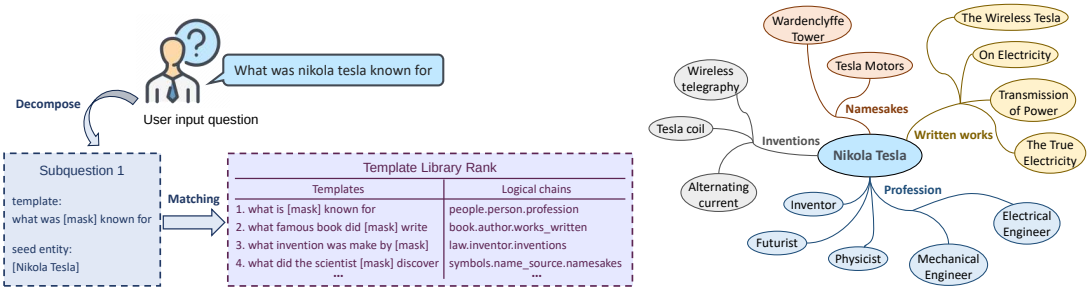

Figure 2: The pipeline of aligning decomposed sub-questions to executable logical chains on KG, where each sub-question will be mapped to a set of logical chains corresponding to top-K relevant question templates.

to learn reasoning patterns (*a.k.a.* logical chains), each of which is a subset of KG edges, from given QA pairs, and then select reasonable logical chains to deduce the answer to the query q (Lan et al., 2021). Thus, as shown in Fig. 1, the workflow of *Keqing* mainly consists of four modules, specifically *Question Decomposition*, *Knowledge Retrieval*, *Candidate Reasoning*, and *Response Generation*, and we will introduce the technical details of each module in the following parts.

1) Decompose complex questions through slot filling. The advantages of introducing the module of *Question Decomposition* are two folds: 1) compared to the code form of SQL instructions, the text form of decomposed sub-questions are much easier to be learned by LLMs, most of whose pretraining corpus is still in text form (Touvron et al., 2023; Chiang et al., 2023); 2) for each practical question in our daily life, especially in the field of *math* or *science*, multiple solutions could exist for the same question, where sufficient pre-collected logical chains for each question template can generate multiple potential answer candidates. As a result, more tolerance could be provided for the following reasoning procedure.

Formally, given a complex question (query) q_i from the user and a set of predefined sub-question templates $\mathcal{Q} = \{q^{(k)}\}_{k=1}^{K}$, the target of the *Question Decomposition* module in *Keqing* is to decompose the given query q_i into T sub-questions through the generation of LLMs, formulated as:

$$\{q_{i,t}\}_{t=1}^{T} = \mathbf{LLM}(q_i), \quad q_{i,t} \in \{q^{(k)}\}_{k=1}^{K}, \quad (1)$$

where the training objective of each sub-question $q_{i,t}$ is to be exactly matched with one of K predefined question templates. As the formulation of prompt and instruction shown in Table 6, taking the original question q_i as the input query, LLMs are finetuned to filling the slots of sub-questions $q_{i,t}$ by generation, as well as corresponding seed entities and dependencies.

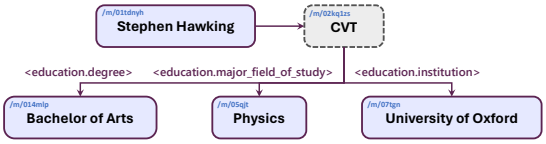

Figure 3: The *compound value types* (CVTs) of the Freebase dataset, where each triplet (s, r, o) can be converted to text by serializing their text surface forms.

For instance, to solve the 3-hop MetaQA question in Fig. 1, specifically "*..., what other works the director of Written on Wind has done and which famous actors were in them?*", *Keqing* is supposed to answer the following questions sequentially: 1) "*who was the director of [mask]?*", 2) "*[mask] was the director of which movies?*", and 3) "*who acted in the movie [mask]?*". Besides, *Keqing* also automatically detects the seed entity "*Written on Wind*" and then forward it coupled with the first question "*who was the director of [mask]?*" to the following procedures to obtain the answer entities. These answer entities are treated as seed entities of the second question "*[mask] was the director of which movies?*". Then, the final answers are iteratively acquired based on the question dependency.

2) Retrieve candidate entities on KG. Considering it is not guaranteed that the generated sub-questions will exactly match the predefined question templates during the inference phase, we introduce an additional question-matching procedure to fill this gap, as shown in Fig. 2. With the same notation in Eq. (1) denoting the generated sub-questions as $\{q_{i,t}\}_{t=1}^{T}$ and predefined question templates as $\{q^{(k)}\}_{k=1}^{K}$, the template-matching process aims to map each sub-question $q_{i,t}$ to its most relevant question templates, resulting in a set of logical chains to be executed on the KG for retrieving potential answer candidates.

Formally, inspired by recent works (Das et al., 2022), we propose to use RoBERTa (Liu et al., 2019), a popular variant of BERT (Devlin et al., 2018), to encode both the decomposed

sub-questions $\{q_{i,t}\}_{t=1}^{T}$ and question template $\{q^{(k)}\}_{k=1}^{K}$ to the same latent space. Subsequently, we measure their semantic distances with cosine similarity, specifically:

$$h_{q_{i,t}} = \textbf{BERT}(q_{i,t}), h_{q^{(k)}} = \textbf{BERT}(q^{(k)}),$$
$$\text{Sim}(q_{i,t}, q^{(k)}) = \frac{h_{q_{i,t}}^{T} h_{q^{(k)}}}{||h_{q_{i,t}}||||h_{q^{(k)}}||}. \quad (2)$$

According to the obtained similarity scores, we can rank the relevance between $\{q^{(k)}\}_{k=1}^{K}$ and $q_{i,t}$, and assign the most relevant question template to $q_{i,t}$. Noticeably, selecting top-K relevant question templates for each sub-question seems more attractive due to the benefit of extending the scope of retrieved answer candidates, but at the cost of increasing the difficulty of the following reasoning procedure. Then, for each question template $q^{(k)}$, we will collect a set of logical chains from KBQA dataset to answer this question, where the quality of the projection from question template to the set of collected logical chains will directly influence the performance of *Keqing*.

After obtaining the seed entity and matching the decomposed sub-questions to the corresponding logical chains, the target of *Knowledge Retrieval* module is to search the answer candidates along the logical chains on the KG. Formally, given the sub-question $q_{i,t}$ marked with seed entity $s_{i,t}$ and the set of collected logical chains $R_{i,t} = \{r_{i,t}^{(l)}\}_{l=1}^{L_{i,t}}$, where each $r_{i,t}^{(l)}$ defines an executable single hop reasoning path on the KG. Starting from the seed entity s, we can perform logical reasoning along $r_{i,t}^{(l)}$ and obtain the resulting triplets:

$$(s, r, o) := (subject, relation, object), \quad (3)$$

which represents that the subject has the relation to the object, resulting in a set of triplets including potential answer candidates denoted as $C_{i,t} = \{(s_{i,t}, r_{i,t}^{(l)}, o_{i,t}^{(l)})\}_{l=1}^{L_{i,t}}$.

3) Answer questions with retrieved candidate entities. With the retrieved answer candidates $C_{i,t} = \{(s_{i,t}, r_{i,t}^{(l)}, o_{i,t}^{(l)})\}_{l=1}^{L_{i,t}}$ in hand, the target of *Candidate Reasoning* is to select the correct entities to answer the current question $q_{i,t}$, where the challenge lies in how to enable LLMs to understand the triplets and process the reasoning procedure.

The most straightforward way is to directly convert the triplet into text using simple heuristics, such as serializing the triplet (s, r, o) by concatenating the text surface forms of subject, relation and object, as shown in Fig. 3. Then, the reasoning capability of LLMs can be used to select the correct answers. However, in practice, we find that even the most advanced LLMs, including ChatGPT, will often overlook certain correct answer candidates. Therefore, we employ LLMs to selectively identify the most reliable reasoning logic, denoted as $r_{i,t}^{*}$, from numerous executable logical chains $\{r_{i,t}^{(l)}\}_{l=1}^{L_{i,t}}$. Thus, given the answer candidates $C_{i,t} = \{(s_{i,t}, r_{i,t}^{(l)}, o_{i,t}^{(l)})\}_{l=1}^{L_{i,t}}$ and input question $q_{i,t}$, *Keqing* is forced to read the context by adding the prompt on the front, as shown in Table 6. Subsequently, the correct answers selected by LLMs can be formulated as

$$C_{i,t}^{*} = \textbf{LLM}(q_{i,t} | C_{i,t} = \{(s_{i,t}, r_{i,t}^{(l)}, o_{i,t}^{(l)})\}_{l=1}^{L_{i,t}}), \quad (4)$$

where $C_{i,t}^{*} = \{(s_{i,t}, r_{i,t}^{(l)}, o_{i,t}^{(l)}) | r_{i,t}^{(l)} = r_{i,t}^{*}\}_{l=1}^{L_{i,t}}$ denotes the subset of retrieved answer candidates satisfying the most reliable reasoning logic selected by LLMs.

For the selection of LLMs to implement the *Candidate Reasoning* module, we can choose ChatGPT due to its excellent capability of logical reasoning to select correct answers from context and zero-shot generalization to solve unseen questions. Another solution is finetuning open-source LLMs, following the same way as *Question Decomposition*, which is more suitable for domain-specific KBQA.

4) Generate response by summarizing question answers. After multiple rounds of questions and answers, for each complex question q_i, we obtain the decomposed sub-questions $\{q_{i,t}\}_{t=1}^{T}$ and corresponding generated answers $\{C_{i,t}^{*}\}_{t=1}^{T}$, which can be treated as an execution log. To allow users to understand the logic of KBQA more intuitively, we introduce a *Response Generation* module to summarize the inference process of *Keqing*, by introducing the prompt "with the task execution logs, the AI assistant needs to describe the process and inference results..." shown in Table 6, equipped with the execution log as input. Finally, *Keqing* can generate a comprehensive response, as shown in the response part of Fig. 1.

3.3 Technical Contributions

Incorporating question templates into the workflow of the KBQA system, *Keqing* is equipped with the ability to explain the logical reasoning process to users and facilitate error identification, as well as provide confidence levels for the reasoning answers

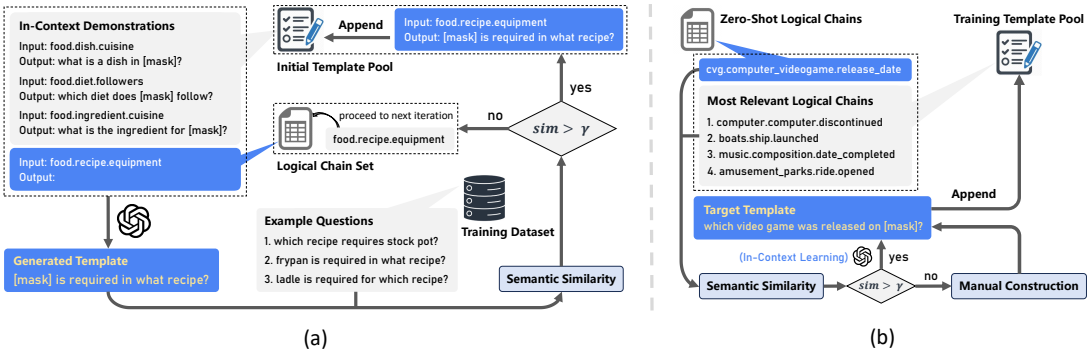

Figure 4: Illustration of the question template construction pipeline. (a) Generating question templates corresponding to the logical chains involved in the training set through multiple iterations; (b) Generating question templates for zero-shot logical chains not covered in the training set.

through the application of uncertainty estimation techniques (Lin et al., 2023). In this section, we will focus on elaborating the technical contributions of *Keqing* in the following parts.

1) Manually intervened question template construction pipeline. To scale up the number of question templates to make it align with the huge amount of logical reasoning chains in KG, we develop a pipeline for manually intervened question template construction, as shown in Fig. 4. The target of the whole pipeline is to assign a suitable question template $q^{(k)} \in \mathcal{Q}$ to each logical chain $r^{(k)} \in \mathcal{R}$ on the KG, mainly including four stages: 1) Initialization: the initial pool is comprised of a limited number of manually constructed pairs consisting of logical chains and question templates; 2) Construction: for each logical chain $r^{(k)}$ in training data, we select several semantically closest pairs from the pool as demonstrations, and then utilize ChatGPT to generate the corresponding question template $q^{(k)}$ through in-context learning; 3) Evaluation: for each generated question template $q^{(k)}$, we evaluate its similarity to the questions existing in the training set. the generation of the template highly matching with the question is considered as successful. If not, it will be required to wait for the next generation after the pool is updated. After multiple iterations, most of templates that highly match with questions in the training set can be generated successfully. Only a few failure templates will be manually intervened and finally stored back in the pool. 4) Extrapolation: for those logical chains not present in the training set, if a similar logical chain can be found in the pool, its question template will be generated using ChatGPT. Otherwise, the corresponding question template will be manually constructed by humans.

2) Uncertainty estimation of KBQA process in Keqing. Uncertainty estimation has become a popular research direction for LLMs to make the generated responses trustworthy (Lin et al., 2023), but few studies have been conducted in the field of LLM-based KBQA systems. With the workflow described in Section 3.2, for each complex question q_i, *Keqing* is able to decompose it into a series of sub-questions with *Question Decomposition* module, denoted as $\{q_{i,t}\}_{t=1}^{T}$, and reach out an optimal logical chain with *Candidate Reasoning* module, denoted as $\{r_{i,t}^{*}\}_{t=1}^{T}$, resulting in the final set of answer candidates candidates $C_{i,T}^{*}$. To measure the confidence level of the final answer set, we perform the uncertainty evaluation by repeatedly executing the workflow of *Keqing*. Assuming a total of M times execution, the confidence level of the set of inference results, denoted as $C_{i,T}^{*(m)}$, for each execution can be assessed as follows:

$$\text{Conf}(C_{i,T}^{*(m)}) = \frac{1}{T}\sum_{t=1}^{T}\text{Sim}(q_{i,t}^{(m)}, r_{i,t}^{*(m)}) \times \text{Conf}(r_{i,t}^{*(m)}) \quad (5)$$

where $\text{Sim}(q_{i,t}^{(m)}, r_{i,t}^{*(m)})$ can be estimated with the defination in Eq. (2). Since the question template $q_{i,t}^{*(m)}$ can be treated as a query of the logical chain $r_{i,t}^{*(m)}$ in semantic level, $\text{Conf}(r_{i,t}^{*(m)})$ can be directly provided by ChatGPT when selectively identifying the most reliable reasoning logic in Eq. (4). After M rounds of confidence level of KBQA process estimation, denoted as $\{\text{Conf}(C_{i,T}^{*(m)})\}_{m=1}^{M}$, we can sum up the confidence levels for the same answer candidates and ultimately obtain the most confident set of inference results.

4 Experiments

4.1 Experimental setup

Datasets. To evaluate the performance of *Keqing*, we have conducted experiments on three public

Method	MetaQA		
	1-hop	2-hop	3-hop
KVMemNN (Miller et al., 2016)	96.2	82.7	48.9
VRN (Zhang et al., 2018)	97.5	89.9	62.5
GraftNet (Sun et al., 2018)	97.0	94.8	77.7
PullNet (Sun et al., 2019)	97.0	**99.9**	91.4
EmbedKGQA (Saxena et al., 2020)	97.5	98.8	94.8
NSM (He et al., 2021)	97.2	**99.9**	98.9
CBR-SUBG (Das et al., 2022)	97.1	99.8	99.3
ChatGPT (Jiang et al., 2023)	61.9	31.0	43.2
StructGPT (Jiang et al., 2023)	94.2	93.9	80.2
KB-BINDER (Li et al., 2023a)	93.5	99.6	96.4
Keqing-Llama2 (Ours)	**98.4**	**99.9**	**99.6**

Table 1: Performance comparison of different methods on the MetaQA benchmark (Hits@1 in percent).

Method	F1
GraftNet (Sun et al., 2018)	62.8
QGG (Lan and Jiang, 2020)	74.0
ReTraCk (Chen et al., 2021b)	71.0
NSM (He et al., 2021)	69.0
CBR-SUBG (Das et al., 2022)	72.8
TIARA (Shu et al., 2022)	76.7
DecAF (Yu et al., 2022)	**78.8**
FlexKBQA-Codex (Li et al., 2023b)	60.6
Pangu-Codex (Gu et al., 2022)	68.3
Pangu-T5 (Gu et al., 2022)	79.6
KB-BINDER-Codex (Li et al., 2023a)	74.4
Keqing-Llama2 (Ours)	78.2
Keqing-ChatGPT (Ours)	**82.3**

Table 2: Performance comparison of different methods on the WebQSP benchmark.

Method	Overall		IID		Compositional		Zero-shot	
	EM	F1	EM	F1	EM	F1	EM	F1
QGG (Lan and Jiang, 2020)	-	36.7	-	40.5	-	33.0	-	36.6
GloVE+Transduction (Gu et al., 2021)	17.6	18.4	50.5	51.6	16.4	18.5	3.0	3.1
GloVE+Ranking (Gu et al., 2021)	39.5	45.1	62.2	67.3	40.0	47.8	28.9	33.8
BERT+Transduction (Gu et al., 2021)	33.3	36.8	51.8	53.9	31.0	36.0	25.7	29.3
BERT+Ranking (Gu et al., 2021)	50.6	58.0	59.9	67.0	45.5	53.9	48.6	55.7
RnG-KBQA (Ye et al., 2022)	68.8	74.4	86.2	89.0	63.8	71.2	63.0	69.2
DecAF (Yu et al., 2022)	68.4	**78.8**	84.8	89.9	**73.4**	**81.8**	58.6	72.3
TIARA (Shu et al., 2022)	**73.0**	78.5	**88.4**	**91.2**	66.4	74.8	**73.3**	**80.7**
Pangu (Gu et al., 2022)	**73.7**	**79.9**	**82.6**	**87.1**	**74.9**	**81.2**	69.1	76.1
KB-BINDER (Li et al., 2023a)	50.6	56.0	51.9	57.4	50.6	56.6	49.9	55.1
FlexKBQA (Li et al., 2023b)	62.8	69.4	71.3	75.8	59.1	65.4	60.6	68.3
Keqing-Llama2 (Ours)	72.5	78.7	81.4	86.7	68.4	75.6	**72.9**	**77.8**

Table 3: Performance comparison of different methods on the GrailQA dev set.

KBQA datasets with varying levels of difficulty, including MetaQA (Zhang et al., 2018), WebQuestionsSP (WebQSP) (Yih et al., 2016), and GrailQA (Gu et al., 2021). A detailed description of these datasets can be found in Appendix B.

Baselines. Among the baselines for comparison, we have included a diversity of competitive KBQA methods, ranging from graph neural network (GNN)-based systems to the ones capitalizing on pre-trained language models; we inherit their results from the paper directly with the same evaluation metric. Notably, the main competitors of *Keqing* are those approaches driven by LLMs such as ChatGPT (Jiang et al., 2023), StructGPT (Jiang et al., 2023), Pangu (Gu et al., 2022), KB-BINDER (Li et al., 2023a), and FlexKBQA (Li et al., 2023b).

Evaluation Metrics. Following previous work (He et al., 2021; Gu et al., 2022), we use **Hits@1**, **F1 score**, and **EM** as our evaluation metrics, which refer to accuracies of the single top-ranked answer, coverage of all the answers, and strict exact-match answer, respectively.

Implementation details. To achieve *question decomposition*, we employed Llama2 (Touvron et al., 2023), one of the most popular open-source LLMs, as the base model and fine-tuned a version of 7 billion parameters with LoRA (Hu et al., 2021). We set the rank r of LoRA to 16 and apply LoRA to the query and value weights in attention, resulting in about only 8 million trainable parameters. We train different numbers of epochs for different datasets, with an initial learning rate of 3e-4 adjusted by the *cosine* scheduler. Appendix C provides more implementation details.

4.2 Experimental results

In this section, we present an in-depth analysis of our experimental results to illustrate the superiority of *Keqing*, mainly focusing on three questions:
1) Does Keqing perform competitively enough compared to other KBQA methods? Table 1, Table 2, and Table 3 show the performance comparison of different approaches on MetaQA, WebQSP,

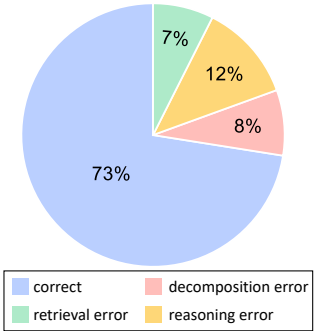

Figure 5: Correct and errors prediction results in the GrailQA dataset.

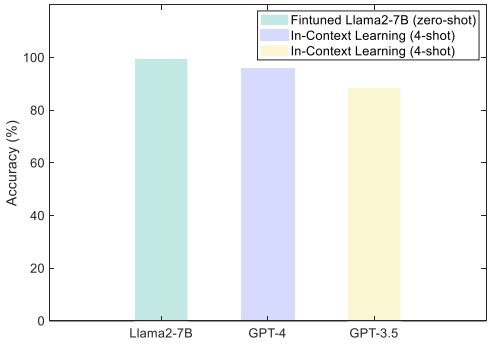

Figure 6: Comparison of decomposition performance of different LLMs.

and GrailQA, respectively. It can be found that *Keqing* consistently outperforms all baseline methods on MetaQA and WebQSP, demonstrating its effectiveness in addressing complex KBQA tasks requiring multi-step inference. Meanwhile, we note in Table 2 that *Keqing*-ChatGPT achieves better perform- ance than *Keqing*-Llama2, suggesting that ChatGPT offers superior reasoning ability than Llama2 in deciding which logic chain can be used to work out the answer to a sub-question.

As for the performance on GrailQA, we observe from Table 3 that *Keqing*-Llama2 is on par with the state-of-the-art results. Although *Keqing*'s overall score is slightly lower than Pangu, it yields better performance in the zero-shot setting, which we attribute to the fact that LLMs are more capable of generalizing at the semantic level than at the programmatic level. Therefore, decomposing complex questions can generalize more effectively than mapping them into logical forms.

2) Why use fine-tuned LLMs instead of the more powerful ChatGPT to realize question decomposition? As discussed in Section 3.1, fine-tuning plays a crucial role in bridging the gap between the internal knowledge of LLMs and the structured knowledge of KG, with the objective that each of the decomposed sub-questions can be addressed in just one step of reasoning. However, since question decomposition operates at the semantic level, one can reasonably guess that the more powerful ChatGPT or GPT4 would perform equally well with the benefit of in-context learning.

To test this hypothesis, we conduct the relevant experiment, with the result displayed in Fig. 6. Note that our fine-tuned Llama2-7B model just utilizes a small subset (10%) of all training data, surpassing both GPT-4 and GPT-3.5, even though they draw on the in-context learning.

3) Does Keqing enjoy better interpretability and may be further improved by user feedback? To prove the interpretability of *Keqing*, we provide case studies in Appendix F, which show the successful reasoning paths of some questions involving different levels of generalization. In addition, we also analyzed the different error types of *Keqing* on GrailQA, as exhibited in Fig 5. According to the statistics, reasoning errors are the most common (12%) one among all errors, this is primarily due to the LLM selecting the logical chain as the final reasoning path based on confidence, which leads to a diversification of reasoning paths and increases the reasoning error rate. Retrieval errors (7%) and decomposition errors (8%) appear in similar proportions because these two errors are often coupled; decomposition errors directly lead to retrieval errors. However, some logical chains with retrieval errors can still answer zero-shot questions in the test set, resulting in a slightly lower rate of retrieval errors. Refining these major error cases is promising for specifically improving *Keqing*'s performance.

More experimental results and analysis can be found in Appendix E, F, and G.

5 Conclusion

In this paper, we present Keqing, a novel method for developing an interpretable and trustworthy KBQA system. Our approach begins by decomposing complex questions into simpler sub-questions using predefined templates, followed by employing LLMs for logical reasoning. To enhance Keqing's practicality and credibility, we developed an automatic pipeline for low-cost template construction and introduced uncertainty estimation techniques to assign confidence levels to reasoning answers. Extensive experiments demonstrate that Keqing not only matches the performance of state-of-the-art methods but also offers superior interpretability.

6 Limitation

As discussed in Appendix, the performance of *Keqing* on KBQA is still limited by a series of issues, like the question decomposition and answer extraction capability of used LLM, and recall rate of retrieval module, either of which can be further improved. Moreover, a more serious question in practice is how to construct a precise and comprehensive knowledge graph for specific fields.

References

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

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*, pages 1247–1250.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2020. Kqa pro: A dataset with explicit compositional programs for complex question answering over knowledge base. *arXiv preprint arXiv:2007.03875*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021a. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021b. 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.

Yongrui Chen, Huiying Li, Yuncheng Hua, and Guilin Qi. 2021c. Formal query building with query structure prediction for complex question answering over knowledge base. *arXiv preprint arXiv:2109.03614*.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. 2022. Binding language models in symbolic languages. *arXiv preprint arXiv:2210.02875*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2019. Kbqa: learning question answering over qa corpora and knowledge bases. *arXiv preprint arXiv:1903.02419*.

Rajarshi Das, Ameya Godbole, Ankita Naik, Elliot Tower, Manzil Zaheer, Hannaneh Hajishirzi, Robin Jia, and Andrew McCallum. 2022. Knowledge base question answering by case-based reasoning over subgraphs. In *International conference on machine learning*, pages 4777–4793. PMLR.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

Yu Gu and Yu Su. 2022. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. *arXiv preprint arXiv:2204.08109*.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), pages 221–231.

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

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. 2011. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*, pages 229–232.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2017. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):824–837.

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

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. *arXiv preprint arXiv:2105.11644*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. Association for Computational Linguistics.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023a. Few-shot in-context learning for knowledge base question answering. *arXiv preprint arXiv:2305.01750*.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2023b. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. *arXiv preprint arXiv:2308.12060*.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18608–18616.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2023. Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187*.

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

Haoran Luo, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, et al. 2023. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. *arXiv preprint arXiv:2310.08975*.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2185–2194.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2020. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. *arXiv preprint arXiv:2012.14610*.

Michael Petrochuk and Luke Zettlemoyer. 2018. Simplequestions nearly solved: A new upperbound and baseline approach. *arXiv preprint arXiv:1804.08798*.

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.

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 bases. *arXiv preprint arXiv:2210.12925*.

Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*.

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. *arXiv preprint arXiv:1809.00782*.

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

Chuanyuan Tan, Yuehe Chen, Wenbiao Shao, and Wenliang Chen. 2023. Make a choice! knowledge base question answering with in-context learning. *arXiv preprint arXiv:2305.13972*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. Lc-quad: A corpus for complex question answering over knowledge graphs. In *The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16*, pages 210–218. Springer.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Silei Xu, Shicheng Liu, Theo Culhane, Elizaveta Pertseva, Meng-Hsi Wu, Sina Semnani, and Monica Lam. 2023. Fine-tuned llms know more, hallucinate less with few-shot sequence-to-sequence semantic parsing over wikidata. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

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 Annual Meeting of the Association for Computational Linguistics (ACL)*.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*.

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

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. *arXiv preprint arXiv:1606.03391*.

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

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. *arXiv preprint arXiv:1704.06194*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

A Related Works

Existing literature (Lan et al., 2022) typically categorizes KBQA approaches into two mainstreams, *i.e.*, semantic parsing-based (SP-based) methods and information retrieval-based (IR-based) methods. Specifically, IR-based methods extract a question-specific subgraph from KB and then rank the candidate entities in the subgraph to derive the final answer. While SP-based methods represent a more favored direction by parsing questions into executable KB queries. Traditional methods (Yih et al., 2015) generate the query graph in stages, often resulting in a large number of noisy candidates. To address this, methods like QGG (Lan and Jiang, 2020) and AQG (Chen et al., 2021c) prune the search space. Recent works leveraging natural language generation approach KBQA as a Seq2Seq task. CBR-KBQA (Das et al., 2021) uses T5 to directly transform questions to SPARQL. Recently, LLMs have demonstrated strong few-shot learning abilities across various tasks, leading to increasing exploration (Gu et al., 2022; Li et al., 2023a; Tan et al., 2023) of the few-shot setting on the KBQA task.

In this paper, we propose *Keqing* following the promising SP-based paradigm. By incorporating question decomposition into the KBQA system, our approach enjoys better interpretability in the way of rendering the reasoning process transparent, thus facilitating user interaction for useful feedback.

B Dataset Descriptions

MetaQA encompasses a movie ontology derived from the WikiMovies Dataset along with three sets of question-answer pairs categorized by varying levels of difficulty.

WebQSP comprises questions sourced from WebQuestions, answerable through Freebase, and it assesses i.i.d. generalization primarily on straightforward questions.

GrailQA represents a diverse KBQA dataset constructed on Freebase, spanning 32,585 entities and 3,720 relations across 86 domains. Its design aims to evaluate KBQA models' generalization across three tiers: I.I.D., compositional, and zero-shot.

Table 4 lists the statistics for the train/dev/test splits of these datasets.

Table 4: Dataset statistics.

Dataset	Train	Dev	Test
GrailQA	44337	6763	13231
WebQSP	3098	-	1639
MetaQA-1hop	96106	9992	9947
MetaQA-2hop	118680	14872	14872
MetaQA-3hop	114196	14274	14274

C Runtime and Memory Complexity

All our experiments were done on two NVIDIA RTX A6000 GPUs (with 48GB RAM each), and the results were averaged from five randomly seeded experiments. Besides, the specific versions of ChatGPT and GPT4 we called through the OpenAI API were `gpt-3.5-turbo-1106` and `gpt-4-turbo`.

As presented in Figure 1, the workflow of *Keqing* mainly consists of four stages, where **#1** *Question Decomposition*, **#3** *Candidate Reasoning*, and **#4** *Response Generation* are all performed with the powerful capabilities of LLMs, while **#2** *Knowledge Retrieval* is a self-contained module that serves the purpose of searching for facts relevant to each sub-question from the given KB, which can be incorporated into any existing advanced retrieval technique.

Although we can use off-the-shelf LLMs to complete *Question Decomposition* and *Candidate Reasoning*, we instead employed a fine-tuned LLM in our experiments to achieve better performance. Concretely, we chose to train the Llama2 model with 7 billion parameters (Llama2-7B (Touvron et al., 2023)) using a parameter-efficient fine-tuning technique, *i.e.*, LoRA (Hu et al., 2021), which we found to achieve reasonably good performance, finished on two NVIDIA RTX A6000 graphics cards with 48G memory for each. The detailed information about runtime and memory usage are listed in Table 5.

D Prompt Used in Keqing

The prompt plays a crucial role in harnessing the power of large language models. By carefully crafting the prompt, users can effectively communicate their tasks and intentions, thereby guiding the LLM to generate responses that are accurate and aligned with task goals, which improves the overall usability and effectiveness of the model. Here, we present the prompt we used in each of the necessary modules of *Keqing* in Table 6.

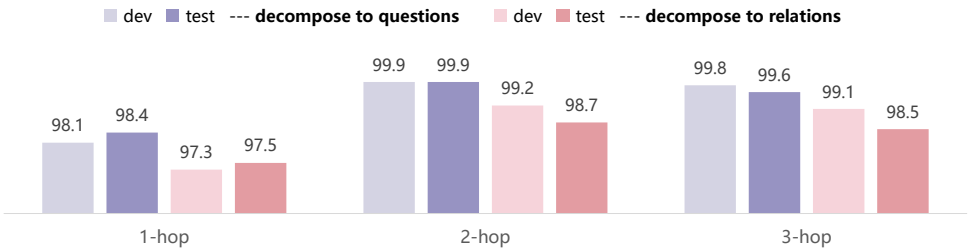

Figure 7: Performance comparison of decomposing KBQA questions into sub-questions and logical chains by finetuning Llama2 on MetaQA dataset.

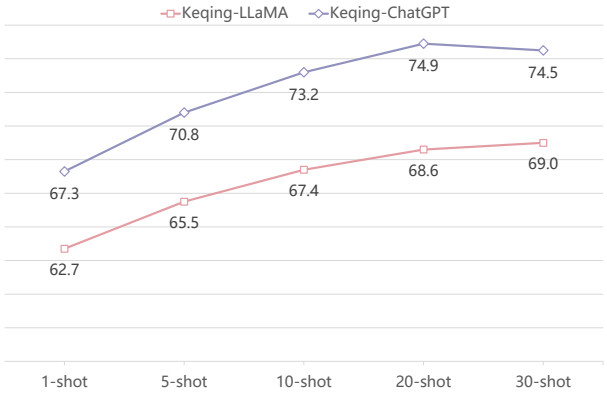

Figure 8: Performance of *Keqing* on WebQSP using different numbers of question templates to match each sub-question.

E Ablation Study

Effect of the number of retrieved question templates: As claimed in Section 2, *Keqing* will select top-K relevant question templates for each sub-question to extend the scope of candidate answers retrieved, and here we investigate the influence of the number of retrieved question templates. From the results shown in Fig. 8, we can observe that the performance of *Keqing* generally improves as the increase of the number of retrieved question templates, indicating that sufficient answer candidates can provide tolerance for the following procedure of answer reasoning. Moreover, this gain of performance gradually decays with the increase of the number of retrieved question templates, reflecting the fact that excessive context can cause misunderstandings of LLMs used for *Candidate Reasoning*.

For the ablation study, we mainly focus on investigating the factors that will influence the performance of *Keqing* to answer the following questions, 1) will decomposing complex problems into sub-problems using LLMs perform better than directly predicting logical chains? 2) how the number of question templates retrieved for each sub-question affects the performance of *Keqing*?

Generate sub-questions *v.s.* generate logical chains: As shown in Fig. 7, we conduct the performance comparison of decomposing complex questions into sub-questions and logical chains on MetaQA dataset, where the only modification is to repalce *Question Decomposition* and *Knowledge Retrieval* modules in *Keqing* with LLMs that are finetuned to directly predict logical chains. From the results, we can find that the performance of *Keqing* to accomplish KQBA tasks by generating sub-questions comprehensively outperforms the other one targeted at generating logical chains, reflecting the fact that the logic of decomposing questions in text form could be easier to be captured by pre-trained LLMs than that in SQL form.

F Qualitative Visualization

Case study on various KBQA benchmarks: To demonstrate the effectiveness of *Keqing*, we conduct a comprehensive case study that covers examples involving different levels of generalization, as shown in Fig. 9. For instance, analyzing the *i.i.d* test case from MetaQA, we can see that *Keqing* precisely breaks the input question into three simple sub-questions and finally obtains the correct answer by iteratively answering each sub-question. For the *zero-shot* test case from WebQSP, even though the gold logic chain "original_idea.innovator" has not appeared in the train-

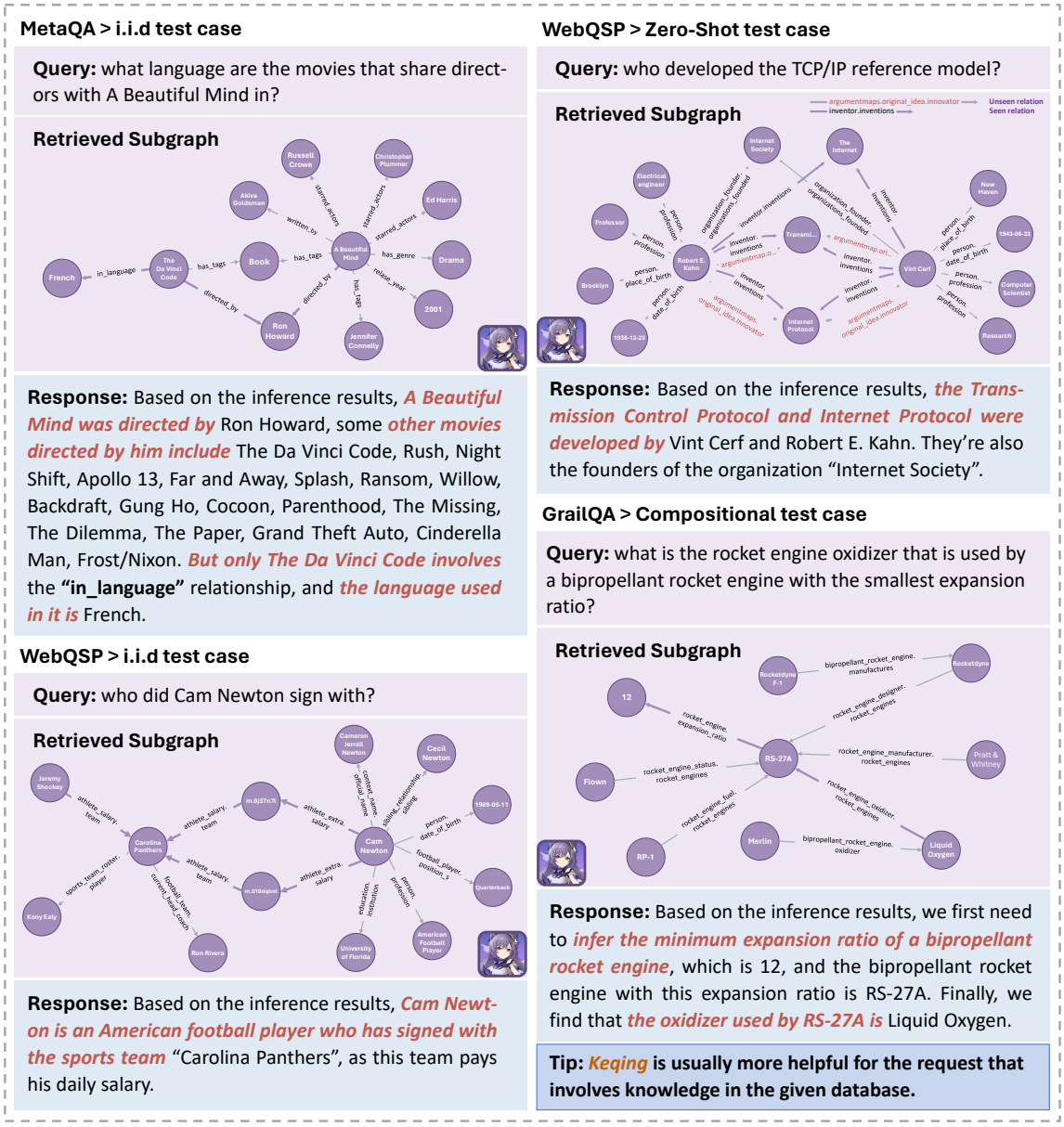

Figure 9: Case study of evaluating *Keqing* on the testing samples of various KBQA benchmarks.

ing set, surprisingly, *Keqing* still arrives at the right answer by matching a semantically similar logic chain "inventor.inventions". For the compositional test case from GrailQA, *Keqing* demonstrates its ability to solve combinatorial problems that did not appear in the training set by utilizing the logical chains to solve sub-questions.

G Failure examples

In this section, we further analyzed the potential causes leading to *Keqing*'s failure to produce the correct answer, which can be roughly categorized into three cases: *i*) the LLM failed to decompose the original question properly; *ii*) the retrieved knowledge does not contain the correct answer to the question; *iii*) the LLM failed to extract the right answer from the given context. In the following, we display a representative example of each of the three failures.

H Retrieval Process

As shown in Fig 10, user questions are decomposed by Large Language Models (LLMs) to generate sub-questions similar to those in the Knowledge Graph (KG). Based on the generated sub-questions, Top-K relational matching is performed within the template repository. Then, a SPARQL query is constructed by selecting the relations with a high confidence level, as determined by the LLMs, to query the database and retrieve the answer.

Stage	Dateset	Trainingexamples	LoRABS	GPUusage	Trainingepoch	Trainingtime	Precision
[c]*QuestionDecomposition*	MetaQA	32927(10%)	64	18.5G	1	1.1h	bf16
	WebQSP	3098	16	27.8G	10	11.8h	bf16
	GrailQA	44337	32	37.9G	3	6.8h	bf16
[c]*CandidateReasoning*	MetaQA	24695(10%)	16	15.3G	3	4.5h	bf16
	WebQSP	3098	4	23.7G	10	5.7h	bf16
	GrailQA	30496	4	23.8G	3	7.6h	bf16

Table 5: Basic statistics of the required memory of GPUs, tuning time, batch size (denoted as BS) per device using LoRA tuning.

Decomposition Failure

Question: what unit of measurement system of electric field of strenght does watt per square metre per steradian have?

Gold answer:
[{"question": "which unit of measurement system uses [MASK] as a unit of electric field of strength?", "id": 0, "dep": [-1], "seed_entity": ["watt per square metre per steradian"], "function": "none"}]

Generated response:
[{"question": "what is the electric field strength of [MASK]?", "id": 0, "dep": [-1], "seed_entity": ["watt per square metre per steradian"], "function": "none"}, {"question": "what unit of measurement system is used for [MASK]?", "id": 1, "dep": [0], "seed_entity": ["<GENERATED>-0"], "function": "none"}]

Retrieval Failure

Question: what is the xbox 360 controller game version that supports the europe region?

Gold Logical Chain:
["computer.computer_peripheral.supporting_game_versions"],
["cvg.game_version.regions"]
Gold answer: ["Sonic & Sega All-Stars Racing with Banjo-Kazooie"]

Retrieved Logical Chain:
['cvg.computer_game_region.versions_released_in_this_region",
"cvg.cvg_platform.games_on_this_platform",
"cvg.computer_videogame.versions",
"cvg.game_version.game",
"cvg.game_series.games_in_series_inv"],

["cvg.computer_videogame.peripherals_supported",
"cvg.computer_game_distribution_system.platforms_supported_inv",
"cvg.game_version.peripheral_classes_supported",
"cvg.computer_videogame.processors_supported",
"cvg.cvg_platform_family.platforms_inv",
"cvg.computer_videogame.game_series"]
Generated response: [" "]

Reasoning Failure

Question: what is the event that shares the conference venue associated with electronic entertainment expo 2009?
Gold answer: ["Electronic Entertainment Expo 2011", "Electronic Entertainment Expo 2010", "Electronic Entertainment Expo 2012", "Electronic Entertainment Expo 2013", "Electronic Entertainment Expo 2014"]

Gold Logical Chain:
["conferences.conference_venue.conferences", "conferences.conference.venue"]

Retrieved Logical Chain:
["conferences.conference_venue.conferences",
"basketball.basketball_conference.league",
"conferences.type_of_conference.conferences_of_this_type",
"exhibitions.exhibition_run.venue_inv",
"conferences.conference.proceedings",
"conferences.conference_subject.specific_conferences_about_this"],

["conferences.conference.focus",
"conferences.conference.proceedings",
"basketball.basketball_conference.league",
"conferences.conference_venue.conferences",
"conferences.conference.venue"
"conferences.type_of_conference.conferences_of_this_type"]

Reasoning Logical Chain:
["conferences.conference_subject.specific_conferences_about_this",
"conferences.conference.focus"]

Generated response:
["Electronic Entertainment Expo 2011", "Electronic Entertainment Expo 2012", "Electronic Entertainment Expo 2009", "Electronic Entertainment Expo 2010"]

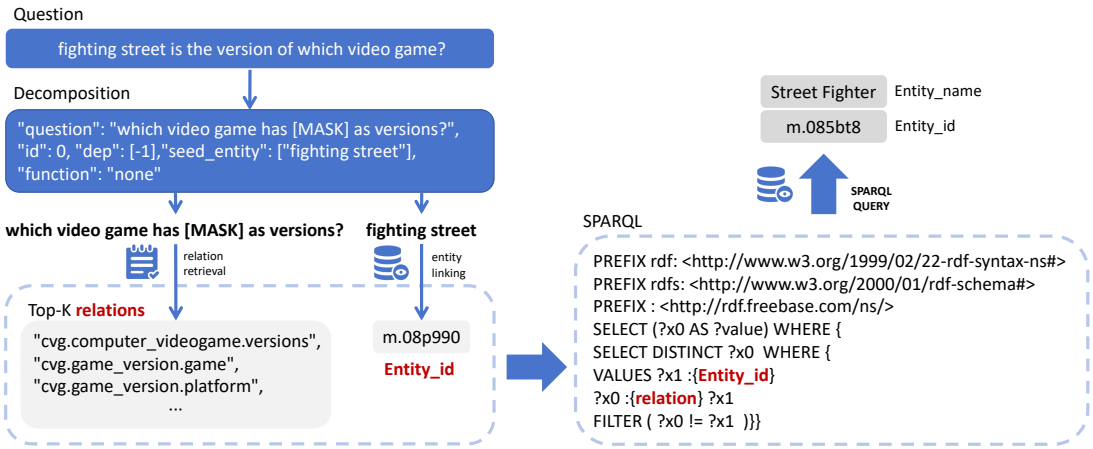

Figure 10: Retrieve Candidate Entities on Knowledge Graph

Table 6: The details of the prompt design of each module in *Keqing*. The *Execution Logs* in *Response Generation* module indicates the record of multiple rounds of questions and answers.

Module Name	Prompt Templates
Question Decomposition	**Instruction:** Parse the user input question to several subquestions: [{"question": subquestion, "id": subquestion_id, "dep": dependency_subquestion_id, "seed_entity": seed_entity or <GENERATED>-dep_id }]. The special tag "<GENERATED>-dep_id" refer to the generated answer of the dependency subquestion and "dep_id" must be in "dep" list. The "dep" field denotes the ids of the previous prerequisite subquestions which generate a new answer entity that the current subquestion relies on. Think step by step about all the subquestions needed to resolve the user's request. Parse out as few subquestions as possible while ensuring that the answer to the input question can be derived. Pay attention to the dependencies and order among subquestions. If the user input question can't be parsed, you need to reply empty JSON []. **Input:** I recently watched the movie Written on the Wind, and I think it was well made. I'd like to know what other works the director of this film has done and which famous actors were in them. **Output:** {"question": "who was the director of [mask]?", "id": 0, "dep": [-1], "args": {"seed_entities": ["Written on the Wind"]}}, {"question": "[mask] was the director of which movies?", "id": 1, "dep": [0], "args": "seed_entities": ["<GENERATED>-0"]}}, {"question": "who acted in the movie [mask]?", "id": 2, "dep": [1], "args": "seed_entities": "<GENERATED>-1"]}}]
Candidate Reasoning	**Instruction:** Choose the most plausible relation to answer the user's question: ["relation"]. If you don't know the answer, say you don't know, don't try to make up the answer. The relation must be selected from the following options: {"written by", "directed by", "starred actors" }. **Input:** Who was the director of Written on the Wind? **Output:** ["directed by"]
Response Generation	**Instruction:** With the task execution logs, the AI assistant needs to describe the process and inference results. Please first think cafefully and directly answer my request based on the inference results. Then please detail your workflow step by step including the used models and inference results for my request in your friendly tone. Please filter out information that is not relevant to my request. If there is nothing in the results, please tell me you can't make it. **Input:** Execution Logs. **Output:** Based on the inference results, the director of Written on the Wind is Douglas Sirk. The movies that Douglas Sirk directed include Shockproof, All That Heaven Allows, Magnificent Obsession, A Time to Love and a Time to Die, There's Always Tomorrow, The Tarnished Angels, Meet Me at the Fair, Lured, Sleep, My Love, All I Desire, Battle Hymn, Imitation of Life, and Written on the Wind. The actors who starred in these movies are Cornel Wilde, Claudette Colbert, Boris Karloff, Rock Hudson, Don Ameche, Robert Cummings, John Gavin, Patricia Knight, Charles Coburn, Lucille Ball, Barbara Stanwyck, Lauren Bacall, Dorothy Malone, Rochelle Hudson, and Sandra Dee.