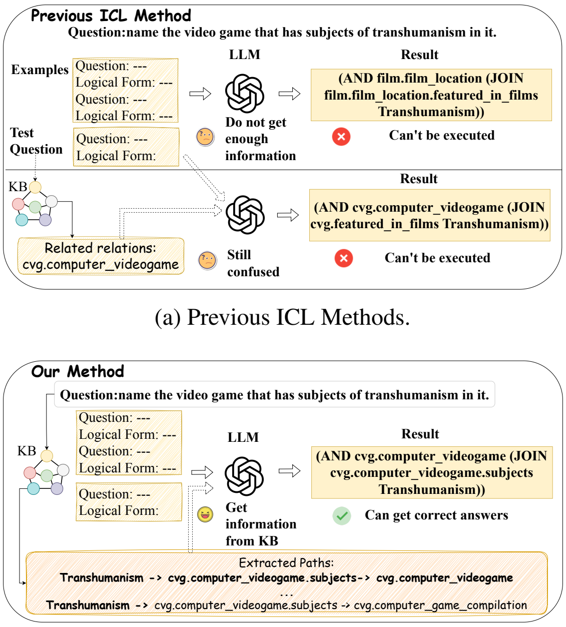
Augmenting Reasoning Capabilities of LLMs with Graph Structures in Knowledge Base Question Answering

Yuhang Tian[1], Dandan Song[1*], Zhijing Wu[1], Changzhi Zhou[1], Hao Wang[1], Jun Yang[1], Jing Xu[1], Ruanmin Cao[2], Haoyu Wang[2]
[1]School of Computer Science and Technology, Beijing Institute of Technology, China
[2]Shanghai Wonder Asset Management Ltd.
{tianyuhang,sdd,zhijingwu,zhou_changzhi97,wanghao,yangjun123,
xujing}@bit.edu.cn;caoruanmin@126.com; why_oneisall@163.com

Abstract

Recently, significant progress has been made in employing Large Language Models (LLMs) for semantic parsing to address Knowledge Base Question Answering (KBQA) tasks. Previous work utilize LLMs to generate query statements on Knowledge Bases (KBs) for retrieving answers. However, LLMs often generate incorrect query statements due to the lack of relevant knowledge in the previous methods. To address this, we propose a framework called **A**ugmenting **R**easoning Capabilities of LLMs with **G**raph Structures in **K**nowledge **B**ase **Q**uestion **A**nswering (**ARG-KBQA**), which retrieves question-related graph structures to improve the performance of LLMs. Unlike other methods that directly retrieve relations or triples from KBs, we introduce an unsupervised two-stage ranker to perform multi-hop beam search on KBs, which could provide LLMs with more relevant information to the questions. Experimental results demonstrate that ARG-KBQA sets a new state-of-the-art on GrailQA and WebQSP under the few-shot setting. Additionally, ARG-KBQA significantly outperforms previous few-shot methods on questions with unseen query statement in the training data. Our code is available at https://github.com/Maydaytyh/ARG-KBQA.


Previous ICL Method
Question:name the video game that has subjects of transhumanism in it.
Examples
Question: ---
Logical Form: ---
Question: ---
Logical Form: ---
Test Question
Question: ---
Logical Form:
LLM
Do not get enough information
Result
(AND film.film_location (JOIN film.film_location.featured_in_films Transhumanism))
Can't be executed
KB
Related relations: cvg.computer_videogame
Still confused
Result
(AND cvg.computer_videogame (JOIN cvg.featured_in_films Transhumanism))
Can't be executed
Our Method
Question:name the video game that has subjects of transhumanism in it.
KB
Question: ---
Logical Form: ---
Question: ---
Logical Form: ---
Question: ---
Logical Form:
LLM
Get information from KB
Result
(AND cvg.computer_videogame (JOIN cvg.computer_videogame.subjects Transhumanism))
Can get correct answers
Extracted Paths:
Transhumanism -> cvg.computer_videogame.subjects-> cvg.computer_videogame
...
Transhumanism -> cvg.computer_videogame.subjects -> cvg.computer_game_compilation

(a) Previous ICL Methods.

(b) Our Method.

Figure 1: The difference between our method and previous ICL methods.

1 Introduction

Knowledge Bases (KBs), such as Freebase (Bollacker et al., 2008), Wikidata (Vrandečić and Krötzsch, 2014), and DBpedia (Auer et al., 2007), storing a tremendous amount of world knowledge in the way of triples, is crucial in many domains as it can provide highly accurate information. Knowledge Base Question Answering (KBQA) (Berant et al., 2013; Yih et al., 2015), which retrieves answers from knowledge bases in response to natural language questions, has been widely studied in recent years. As the scale of KB continues to expand, the models (Ye et al., 2021; Das et al., 2021; Shu et al., 2022) used for KBQA often require large amounts of manually annotated data, extensive training processed, and complex model architectures. Therefore, finding ways to achieve good performance with limited data has become an urgent problem that needs to be addressed.

Recently, Large Language Models (LLMs) have demonstrated the ability to complete complex reasoning tasks, like generating executable code, with only minimal labeled data. This is due to their In-Context Learning (ICL) capabilities (Hasan et al., 2024; Cahyawijaya et al., 2024). Following the ICL paradigm, Li et al. (2023) and Nie et al. (2024) use a few (question, logical form) example pairs to guide LLMs in generating executable logical forms for retrieving answers from KBs. Since LLMs do not have direct access to KBs, these examples can

[*]Corresponding author.

Findings of the Association for Computational Linguistics: EACL 2024, pages 11967–11977
November 12-16, 2024 ©2024 Association for Computational Linguistics

provide helpful information for generating correct logical forms. Nonetheless, when the logical form involved in the test question has not appeared in the training set, and the examples provided in a few-shot scenario all come from the training set, they cannot offer directly useful information for the current problem. For example, as shown in Figure 1a, when the LLMs lack information relevant to the question, it may generate a logical form that is incorrect. Therefore, it is crucial to equip LLMs with comprehensive and relevant information to enhance their performance. Some works (Nie et al., 2024; Wang et al., 2023) retrieve relations or triples from KBs to provide more useful information. However, there are many questions requiring multi-hop reasoning on the knowledge base (KB) to get the answers. Therefore, merely providing relations or triples is still insufficient to solve the problem, as shown in Figure 1a.

To address this challenge, we propose an innovative method called **A**ugmenting **R**easoning Capabilities of LLMs with **G**raph Structures in Knowledge Base Question Answering (**ARG-KBQA**). Our method aims to provide LLMs with information more helpful for generating logical forms, specifically in the zero-shot generalization sceario mentioned above. Unlike previous works which retrieve relations or triples from KBs as supply information, we leverage the graph structures within the KB to enhance the performance of LLMs as shown in Figure 1b. To efficiently retrieve question-related graph structures from the KB, we design an unsupervised two-stage ranker that performs multi-hop beam search on the KB. In this process, we perform a two-stage ranking at each hop using a pre-trained Language Model without additional training, enhancing the efficiency and effectiveness of retrieving related graph structures (paths) from the KB. Furthermore, to facilitate the understanding of LLMs, we convert the extracted graph structures (paths) into logical forms that match the target format.

We conduct extensive experiments on two datasets, GrailQA and WebQSP. ARG-KBQA sets a new state-of-the-art (SOTA) in the few-shot setting, demonstrating that retrieved graph structures provide LLMs with sufficient information. When it comes to the questions unseen in training set of GrailQA, ARG-KBQA significantly surpasses the existing SOTA, with improvements of **8.1** in EM and **7.4** in F1. These results suggest that related graph structures can enhance LLMs' reasoning capabilities, especially when the KB schema in the correct logical form is absent from the training set.

Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to propose integrating the graph structure of the KB to augment reasoning capabilities of LLMs for KBQA.
- We introduce an unsupervised two-stage ranker to extract graph structures from the KB, efficiently retrieving question-relevant graph structures.
- Through extensive experiments on GrailQA and WebQSP, we demonstrate that ARG-KBQA sets a new state-of-the-art (SOTA) in the few-shot setting.

2 Related Work

2.1 Knowledge Base Question Answering.

Current KBQA methods are divided into Information Retrieval (IR-based) and Semantic Parsing-based (SP-based).

IR-based methods retrieve relevant subgraphs from the KB and then use ranking algorithms to determine the answer. SR (Zhang et al., 2022) emphasizes the importance of subgraph retrieval accuracy and introduces a trainable, decoupled subgraph retriever that enhances the performance of subgraph-oriented KBQA models. UniKGQA (Jiang et al., 2022) proposes an unified framework that integrates retrieval and reasoning across learning parameters and architectures. EPR (Ding et al., 2024) accelerates the retrieval of atomic patterns by indexing the atomic adjacency patterns of resource pairs and introduced an algorithm for constructing evidence patterns.

SP-based methods aim to convert natural language questions into query statements for KB, such as SPARQL or S-expressions. RnG-KBQA (Ye et al., 2021), TIARA (Shu et al., 2022), and DECAF (Yu et al., 2022) use sequence-to-sequence models to entirely create S-expressions and provide multiple improvements to the semantic parsing procedure. FC-KBQA (Zhang et al., 2023) extracts relevant fine-grained knowledge components from KB and reformulate them into middle-grained knowledge pairs for generating the final logical expressions. Besides, KB-BINDER (Li et al., 2023), and KB-Coder (Nie et al., 2024) utilize few-shot in-context learning to enable the large language model to generate the logical form for a given question.

2.2 KG-enhanced LLM for KBQA.

Considering the few-shot and in-context learning capabilities of LLMs, current studies (Gu et al., 2023; Li et al., 2023; Nie et al., 2024; Jiang et al., 2023; Xiong et al., 2024; Sun et al., 2023) utilize LLMs for reasoning on knowledge bases (KB). Pangu (Gu et al., 2023) uses LLM as an agent to explore the environment(KB) to construct valid plans incrementally. KB-BINDER (Li et al., 2023) provides the (question, logical form) pairs to the LLM as information from KB. KB-Coder (Nie et al., 2024) additionally retrieves similar relations from the KB and adds them to the input of LLM for each question. StructGPT (Jiang et al., 2023) constructs the specialized function to collect relevant evidence from structured data (e.g., KB) and lets LLMs concentrate the reasoning task based on the collected information. Interactive-KBQA (Xiong et al., 2024) develops three generic APIs for LLMs to interact with KB to generate logical forms. ToG (Sun et al., 2023) proposes a paradigm called "LLM $\otimes$ KG" which use the LLM as an agent to explore related entities and relations on KB interactively and performs reasoning based on the retrieved knowledge.

3 Preliminaries

In this section, we introduce two crucial concepts: Knowledge Base (KB) and Logical Form.

3.1 Knowledge Base

Knowledge Base (KB) stores structured knowledge as a collection of triples $\mathcal{G} = \{(h, r, t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$, where h, r and t denote head, relation, and tail , respectively, and $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations. It should be noted that each entity in KB has a machine identifier (*mid*) and a corresponding surface name. For example, there is an entity whose *mid* is `m.03ck28v` and the surface name is "Sunnyside". Obviously, different entities can share the same surface name.

3.2 Logical Form

A logical form is a statement used to query a KB to obtain the answer given a problem. Common types of logical forms include SPARQL and S-expressions (Gu et al., 2021). SPARQL is a query language for RDF databases that is similar in syntax to SQL. Its core idea is to extract a subject or object from a triple based on a given predicate verb. The format of SPARQL can typically be shown as follows.

```
SELECT <variables>
WHERE {
    <graph pattern>
}
```

The structure of SPARQL is relatively complex. To improve compactness, compositionality, and readability, Gu et al. (2021) proposed S-expressions, a linearized version of query statements. S-expressions include operators such as `JOIN`, `AND`, `COUNT`, `ARGMAX`, and `ARGMIN`, enabling basic operations for querying a knowledge base. In this article, we discuss and use S-expressions.

4 Methodology

Overall, our proposed ARG-KBQA model, as illustrated in Figure 2, aims to generate a logical form for a given question to access the KB and retrieve the answers. The ARG-KBQA model consists of three modules: Multi-hop Path Enumeration, Target Logical Form Generation, and Alignment and Answer Extraction. Next, we will introduce these three modules separately.

4.1 Multi-hop Path Enumeration

In this module, we perform a multi-hop beam search using an unsupervised two-stage ranker on the KB to extract question-related graph structures, which are actually ranked question-related paths in our work. Furthermore, we convert the top k paths into logical forms that are consistent with the target format. Specifically, since most of the questions are either one-hop or two-hop (We provide a detailed analysis in the appendix A.1), we perform a two-hop beam search here.

For a given question, there may be entities or literals (which are usually "integer", "float", "year", or "date") present. For example, in the question " the distance of at least 57.05 is attained by which star system? ", the entity is "John Elliott" and the literal is "57.05". Given a question, we first identify all entities and literals within it. To ensure a fair comparison, we use the same settings as KB-Coder (Nie et al., 2024), utilizing the question's golden entities. We refer to the set containing all entities and literals within a question as E. For each entity or literal in E, we then retrieve all one-hop relations, referred to as R_1. Next, we perform a two-stage sorting of all the relations in R_1.

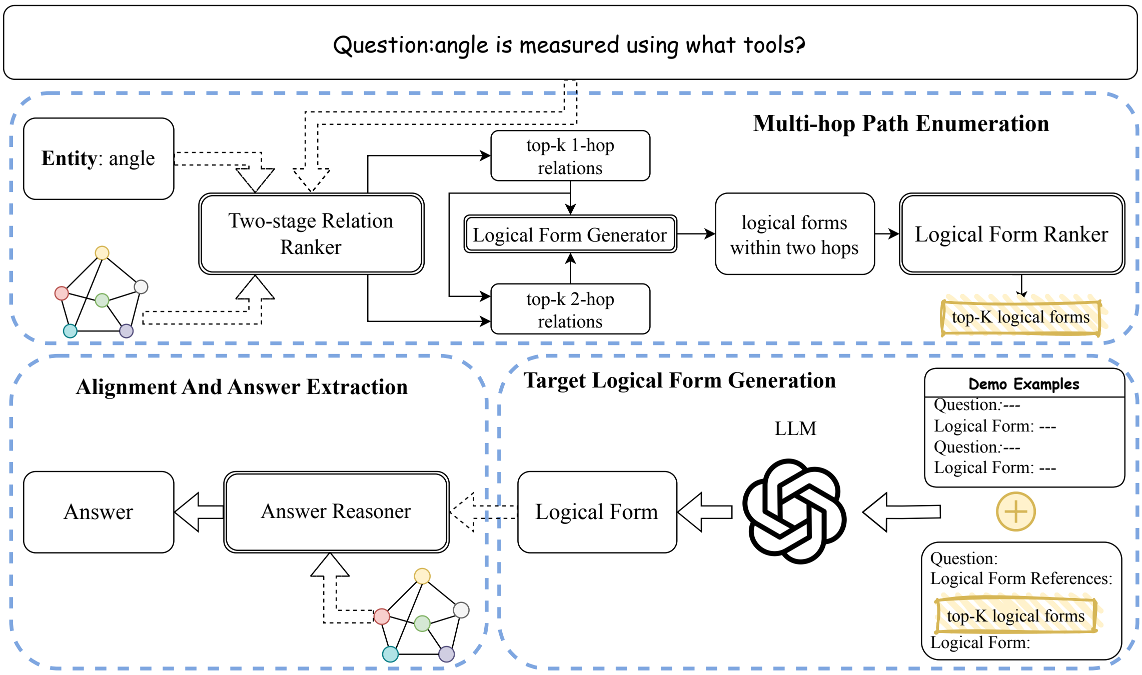


Figure 2: The overview of our methods.

First-hop Stage In this stage, we extract k first-hop relations for all relations or literals in E. Specifically, we use the relation-encoder R_{enc}, which is a pre-trained encoder, to encode all relations r, such that $\mathbf{r} = R_{\text{enc}}(r)$. At the same time, we also use R_{enc} to encode the question, such that $\mathbf{q} = R_{\text{enc}}(q)$. Then, we use Equation 1 to calculate the similarity between the question and the relation, using cosine similarity as follows.

$$\text{similarity} = \frac{\mathbf{q} \cdot \mathbf{r}}{\|\mathbf{q}\| \|\mathbf{r}\|} \quad (1)$$

Based on the similarity between the relation and the question, we select the top $2k$ relations, referred to as $R_1^{(1)}$. Then, for each relation r within $R_1^{(1)}$, we concatenate it with the question as s as follows.

```
Question: question q
Relation: relation r
```

We then use a sentence encoder S_{enc} to encode the sentence s (i.e., $\mathbf{h} = S_{\text{enc}}(s)$) and add a linear projection layer to map it to a score. These scores are then sorted to identify the top-k relations, referred to as $R_1^{(2)}$.

Second-hop Stage In this stage, we extract k next-hop relation from the relations in $R_1^{(2)}$. Using $R_1^{(2)}$ as the first-hop relations, we expand (How this expansion is achieved is explained in detail in the appendix A.2) each relation to obtain the two-hop relations, R_2. To better ensure semantic integrity, we concatenate the previous-hop relations of the current relation as follows and encode them as r_2 using R_{enc}.

```
One-hop Relation: previous relation
Relation: current relation
```

Similarly to the one-hop stage, we select top $2k$ two-hop relations as $R_2^{(1)}$ by the similarity between the r_2 and q. Then, for each relation r in $R_2^{(1)}$, we concatenate it with its previous relation and the question as follows.

```
Question: question q
One-hop Relation: previous relation
Relation: current relation
```

Then, we use the same manner as the first-hop stage to score and r in $R_2^{(1)}$. And the top k relations are selected as $R_2^{(2)}$. Thus, we obtain all the two-hop paths starting from the entities or literals in the question.

Then, these paths within two-hops are transformed into logical forms. After obtaining all the logical forms, we concatenate the question with each logical form as follows.

```
Question: question q
Logical form: logical form
```

Then, we use the same sentence encoder S_{enc} as before to score them, selecting the top k as logical form references.

4.2 Target Logical Form Generation

In this module, we employ few-shot in-context learning to generate the logical form for a given question. For each question, we use two settings to get example questions from the training set: (1) use fixed randomly selected questions, which is **ARG-KBQA**. (2) use BM25 to select N most similar questions, which is **ARG-KBQA(R)**. Following the principles of few-shot in-context learning, we provide the logical form for each example question. For example, for the question “How many game expansions has John Elliott released?”, the corresponding logical form should be as follows.

```
COUNT (AND cvg.computer_game_expansion
      JOIN (R cvg.cvg_publisher.games_published)
      m.0774ytb)
```

In this expression, $m.0774ytb$ is the *mid* for the entity named “John Elliott”" Since LLMs cannot directly access the knowledge base (KB), they do not recognize or generate specific mids as $m.0774ytb$. However, they can understand and generate the surface names associated with these entities. Therefore, we replace the mid with its corresponding surface name in the logical form of each example question. As a result, the above expression should be modified to the following format.

```
COUNT (AND cvg.computer_game_expansion
      JOIN (R cvg.cvg_publisher.games_published)
      John Elliott)
```

Similarly, the mids in the logical form references are replaced with their corresponding surface names.

In general, the input format for the LLM consists of N example questions, each substituted with the *mid* logical form. Additionally, for each test question, we incorporate k previously generated logical form references. Subsequently, we utilize the LLM to generate the logical form for the given question.

4.3 Alignment And Answer Extraction

In this module, we reason on the KB using the LLM's output to obtain the final answers. During execution, we align the entities and relations in the logical form with the KB, as the generated logical form is not directly executable.

Initially, we extract all entity names from the logical form and then search for entities with the surface names within the KB. If we locate entities with the same surface names, we choose the top N_e entities based on the popularity score FACC1 (Gabrilovich et al., 2013). If there is no exact match, we use BM25 to retrieve similar entities from the knowledge base and subsequently select the top N_e most similar ones. Similarly, for relations, if an exact match is found, we use it directly. Otherwise, we first use BM25 to identify the top N_r most similar relations. These relations are then scored in conjunction with the question, and the most relevant relation is ultimately selected.

Following that, we generate all possible combinations and permutations of the enumerated relations and entities. Execution halts upon finding an answer that satisfies the specified conditions, which becomes the final answer. The logical form that produces this answer is also regarded as the final logical form.

5 Experiments

In this section, we demonstrate the experimental validation of ARG-KBQA, including dataset descriptions, implementation details, results, and a detailed analysis.

5.1 Datasets

We use two mainstream KBQA datasets for our experiments, as shown below:

GrailQA (Gu et al., 2021) is a challenging KBQA dataset based on Freebase, containing 64,331 question-logical form pairs, 32,585 entities, and 3,239 literals. It focuses on evaluating the generalization capabilities by introducing three levels of questions: I.I.D., compositional, and zero-shot. Consistent with the KB-Coder (Nie et al., 2024), we evaluate our method on the dev portion of the GrailQA dataset.

WebQSP (Yih et al., 2016) is another popular KBQA dataset based on Freebase, containing 4,737 natural language questions. The primary goal of this dataset is to evaluate the generalization capability within an i.i.d. setting, given that the training and testing data include the same entities and relations.

5.2 Baselines

For a comprehensive comparison, we select baselines that primarily include two categories: fully supervised learning and in-context learning. The fully supervised methods include Rng-KBQA (Ye et al., 2021), DecAf (Yu et al., 2022), and TIARA (Shu et al., 2022). The methods of in-context learning include KB-BINDER (Li et al.,

Methods	I.I.D.		Compositional		Zero-shot		Overall	
	EM	F1	EM	F1	EM	F1	EM	F1
Full Supervised on the Entire Training set								
RnG-KBQA (Ye et al., 2021)	86.7	89.0	61.7	68.9	68.8	74.7	69.5	76.9
DecAF (Yu et al., 2022)	**88.7**	**92.4**	**71.5**	**79.8**	65.9	74.7	72.5	81.4
TIARA (Shu et al., 2022)	88.4	91.2	66.4	74.8	**73.3**	**77.3**	**75.3**	**81.9**
In-Context Learning (Training-Free)								
KB-BINDER (1)†	40.0	43.3	33.9	36.6	40.1	44.0	38.7	42.2
KB-Coder (1)†	40.6	45.5	34.5	38.6	42.2	47.3	40.1	44.9
ARG-KBQA(1)	**46.6**	**51.5**	**36.4**	**41.8**	**46.6**	**52.1**	**43.8**	**48.5**
KB-BINDER (6) †	43.6	48.3	**44.5**	48.8	37.5	41.8	45.7	50.8
KB-Coder (6) †	43.6	49.3	44.0	**49.6**	37.7	43.2	45.9	**51.7**
ARG-KBQA(6)	**48.5**	**52.4**	43.5	46.8	**48.4**	**52.4**	**47.5**	51.5
KB-BINDER (1)-R †	74.7	79.7	44.6	48.5	37.1	40.8	47.6	51.7
KB-Coder (1)-R †	76.2	**80.2**	**50.4**	**54.8**	45.8	50.6	54.0	58.5
ARG-KBQA(1)-R	**76.4**	78.5	47.0	52.7	**53.9**	**58.0**	**57.7**	**61.7**
w/ gpt-3.5-0125	79.2	81.8	48.9	55.2	54.4	59.3	58.9	63.7
w/ GPT4	79.2	81.6	53.0	59.0	52.2	57.7	58.8	63.7
KB-BINDER (6)-R †	75.8	80.9	48.3	53.6	45.4	50.7	53.2	58.5
KB-Coder (6)-R †	76.9	81.0	**52.7**	**57.8**	48.9	54.1	56.3	61.3
ARG-KBQA(6)-R	**79.0**	**81.5**	48.4	55.8	**55.9**	**61.4**	**59.6**	**64.9**
w/ gpt-3.5-0125	76.6	79.1	48.3	55.1	55.6	61.4	58.9	64.1
w/ GPT4	79.9	82.4	55.7	62.3	55.7	61.5	61.4	64.1

Table 1: 40-shot results of ARG-KBQA and baselines on the local dev set of GrailQA. **Bold** numbers indicate the best performance, and underlined numbers indicate the second-best performance. (1) and (6) denote using the top 1 and top 6 logical forms generated by the LLM, respectively. -R denotes the example questions are retrieved by BM25. † denotes the results come from Nie et al. (2024).

2023) and KB-Coder (Nie et al., 2024). In particular, the results of KB-Coder and KB-BINDER are both taken from the KB-Coder paper with using the `gpt-3.5-turbo-0613` model. To maintain a fair comparison, we use the same model `gpt-3.5-turbo-0613` in this paper.

5.3 Evaluation Metric

Consistent with previous work (Ye et al., 2021; Li et al., 2023; Nie et al., 2024), we use F1 Score as evaluation metrics for WebQSP, while Exact Match (EM) and F1 Score for GrailQA.

5.4 Implementation Details

In the Multi-hop Path Enumeration module, the two-stage ranker employs deberta-v3-large (He et al., 2020) as the R_{enc} and instructor-large (Su et al., 2023) as the S_{enc}, respectively. In the two-stage ranker, the number of relations selected, denoted as k, is set to 20. In the Target Logical Form Generation module, we conduct 40-shot experiments on GrailQA and 100-shot experiments on WebQSP, consistent with previous work (Li et al., 2023; Nie et al., 2024). We leverage the `gpt-3.5-turbo-0613` model from the OPENAI API[1] to generate K logical forms for each question (where K is set to 1 and 6), and we conduct experiments using ARG-KBQA and ARG-KBQA(R), respectively. Additionally, we test `gpt-3.5-turbo-0125` and `gpt-4-turbo` on GrailQA, and `gpt-3.5-turbo-0125` on WebQSP. In the Alignment and Answer Extraction module, the top 15 entities ($N_e = 15$) and the top 10 relations ($N_r = 10$) in the logical form are selected as candidates.

5.5 Main Result

we report the performance of ARG-KBQA , ARG-KBQA(R), and other baselines on GrailQA and WebQSP in Table tables 1 and 2, respectively. We report the results of ARG-KBQA(1) and ARG-KBQA(6), representing using the top 1 and top 6 logical forms generated by the LLM, respectively.

Results on GrailQA From the results in the In-Context Learning section, it is obvious that the ARG-KBQA(1) and ARG-KBQA(6) outperforms other methods in the same setting. These

[1] `https://openai.com/api`

Methods	F1
Full Supervised on the Entire Training set	
RnG-KBQA (Ye et al., 2021)	75.6
DecAF (Yu et al., 2022)	76.7
TIARA (Shu et al., 2022)	**78.7**
In-Context Learning (Training-Free)	
KB-BINDER (1)†	52.6
KB-Coder (1)†	55.7
ARG-KBQA(1)	**58.8**
KB-BINDER (6)†	56.6
KB-Coder (6)†	60.5
ARG-KBQA(6)	**62.7**
KB-BINDER (1)-R†	68.9
KB-Coder (1)-R†	72.2
ARG-KBQA(1)-R	**72.5**
w/ gpt-3.5-0125	71.6
KB-BINDER (6)-R†	71.1
KB-Coder (6)-R†	75.2
ARG-KBQA(6)-R	**75.6**
w/ gpt-3.5-0125	73.9

Table 2: 100-shot results of ARG-KBQA and baselines on the test set of WebQSP. **Bold** numbers indicate the best performance, and underlined numbers indicate the second-best performance.

improvements, particularly in the zero-shot level of GrailQA, indicate that the provided logical form references play a crucial role when the examples are not related to the test question. When using more effective BM25, ARG-KBQA(1)-R shows significant overall performance improvement, especially in the zero-shot scenario, outperforming the SOTA (KB-Coder(1)-R) by 8.1% in EM and 7.4% in F1. However, in the compositional scenario, its performance is better than that of KB-BINDER but slightly lower than that of KB-Coder. This could be because, for compositional scenarios, the correct logical form is often a combination of logical forms from multiple training data, thus making demo examples more effective. Consequently, our generated logical form references may have caused some misleading results in certain situations.

When using the latest LLMs, such as `gpt-3.5-turbo-0125` and `gpt-4-turbo`, the results show that ARG-KBQA performs similarly on these two models, both of which outperform its performance on `gpt-3.5-turbo-0613`. This indicates that as the performance of LLMs improves, the executability and accuracy of the generated logical forms increase. However, in the zero-shot level of GrailQA, the performance of `gpt-4-turbo` is actually weaker than that of `gpt-3.5-turbo-0613` and `gpt-3.5-turbo-0125`. We conduct a detailed analysis of the experimental results and find that one possible reason is that as the LLM's performance improves, it may be more inclined to rely on its internal knowledge and ignore the prompt's provided information when the examples are not useful. We present an example in the appendix A.3.

As for using majority vote, the results indicate that in both I.I.D and zero-shot scenarios, our method significantly outperforms the KB-Coder. This demonstrates the effectiveness of our approach. Interestingly, similar to the results of ARG-KBQA(1)-R on the zero-shot scenario, as the performance of the LLM improves, the experimental results do not improve but rather slightly decline, likely due to the aforementioned reason.

Results on WebQSP As shown in Table 2, we achieve SOTA performance in all settings on WebQSP. Specifically, when using majority vote setting, the performance of ARG-KBQA is comparable to that of fully supervised methods. Similarly, we test the performance of ARG-KBQA on `gpt-3.5-turbo-0125`, and the results indicate that the improved capabilities of the LLM do not lead to performance gains on WebQSP.

Overall, according to the presented experiment results, augmenting LLMs with the graph structure of knowledge bases could significantly enhance the performance of few-shot approaches for KBQA.

5.6 Ablation Study

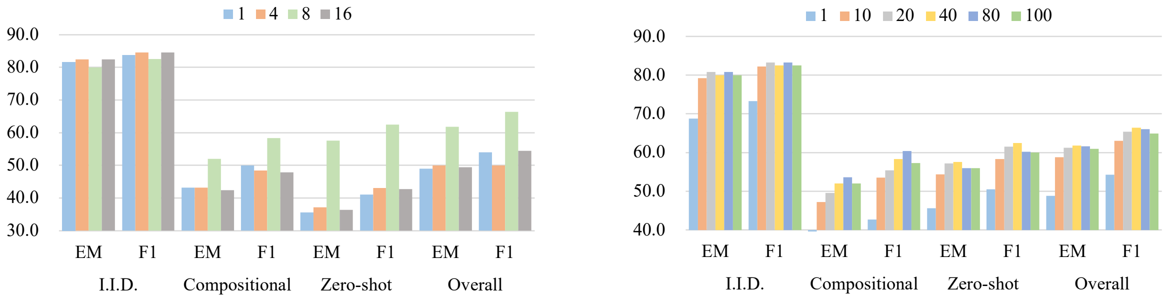
To evaluate the effectiveness of various components of ARG-KBQA, we randomly select 500 samples from the GrailQA (Gu et al., 2021) dev set, maintaining the original dataset's proportions: i.i.d.:compositional:zero-shot at a ratio of 1:1:2. Specifically, we consider three settings in the ablation study: (1) removing logical form references generated by section 4.1. (-w/o lfrs) (2) removing demo examples (-w/o examples) (3) adding randomly selected logical form references (-w random_lfr) (4) using flan-t5-xl as the generation model (-w flan-t5-xl)

From the results shown in Table 3, we can observe the following:

- Compared to the complete model, the model without logical form references (*w/o* lfrs) performs significantly worse overall, with the Exact Match (EM) score being 14% lower and the F1 score 12.3% lower in the zero-shot level. This highlights the effectiveness of the logical form references we introduced,

Method	I.I.D.		Compositional		Zero-shot		Overall	
	EM	F1	EM	F1	EM	F1	EM	F1
complete model	80.8	83.3	**49.6**	**55.4**	**57.2**	**61.5**	**61.2**	**65.4**
w/o lfrs	**84.0**	**87.0**	42.4	48.8	43.2	49.2	53.4	58.5
w/o examples	32.8	36.1	28.8	34.1	40.4	43.3	39.2	32.8
w random_lfr	82.3	84.0	40.0	46.5	37.6	43.4	54.5	82.4
w flan-t5-xl	31.2	35.6	18.4	26.4	21.6	25.9	23.3	28.5

Table 3: Ablation study results on GrailQA.



(a) Effect of the number of logical form. references

(b) Effect of the shot number.

Figure 3: Parameters experiments results of two parameters: shot number and the number of logical form references.

specifically when the example questions are not useful.

- The *-w/o* examples setting shows an even more substantial performance decline, highlighting the importance of examples in in-context learning for LLMs to learn and solve new tasks.
- In the *-w* random_lfr setting, where logical form references are randomly selected from all possible logical forms with two hops, the performance is worse than -w/o lfrs. This underscores the importance of our two-stage ranker, which selects the most relevant logical forms for the questions.
- In the *-w* flan-t5-xl setting, we use the exact same input as the complete model, only replacing the generative model with flan-t5-xl. It can be observed that the performance significantly declines. This indicates that using LLMs is necessary in our experiments, and smaller models like flan-t5-xl do not possess sufficient contextual learning and generation capabilities.

5.7 Parameters Experiments

In this section, we analyze the effects of two parameters, including shot number and the number of logical form references. We use the same setting as the ablation study, randomly selecting 500 samples from the GrailQA dev set, with the proportions of the three levels of data maintained at 1:1:2. To accurately demonstrate the impact of different parameters on different levels of data, we present the results for each of the three levels as shown in Figure 3.

Results on the logical form references We test the impact of the number of logical form references, as shown in Figure 3a. The results indicate that as the number of logical form references increases, both EM and F1 improve across all levels, with the effect being particularly pronounced in the zero-shot scenario. This is because more examples provide the LLM with more knowledge. However, it is important to note that too many references can lead to a performance decline, possibly due to the lower-ranked logical form references misleading the LLM. Additionally, the number of logical form references has little impact on the results in the I.I.D. scenario, as the LLM might rely more on the information provided by the examples for independently and identically distributed data.

Results on the ICL examples Additionally, we test the impact of varying the number of in-context learning examples from 1 to 100 on the experimental results, as shown in Figure 3b. Notably, EM and F1 scores increase across all levels with the number of examples. The effect of the number of examples is relatively consistent across all levels. However, the performance gap between using 1 example and 10 examples is the smallest in the zero-shot level.

This indicates that with a very small number of examples, the provided logical form references can effectively guide the LLM.

6 Conclusion

In conclusion, our proposed framework, ARG-KBQA, significantly enhances the reasoning capabilities of Large Language Models (LLMs) for Knowledge Base Question Answering (KBQA). By leveraging graph structures within Knowledge Bases (KBs) through an innovative unsupervised two-stage ranker, ARG-KBQA effectively retrieves relevant information for accurate logical form generation. Experimental results on GrailQA and WebQSP datasets highlight the framework's effectiveness, setting new state-of-the-art benchmarks in the few-shot setting and excelling in zero-shot generalization scenarios. These findings underscore the importance of integrating graph structures to augment LLM reasoning capabilities, especially for unseen query statements in training data.

Limitations

Despite its significant advancements, ARG-KBQA has some limitations. On the one hand, the prerequisite for the two-stage ranker to obtain results is to find nodes in the knowledge base (KB). If the corresponding nodes are not found or the wrong nodes are identified, it will greatly affect the accuracy of the subsequent steps. On the other hand, the framework requires an alignment process to map the logical form output by the LLM to the KB. This process is time-consuming and prone to errors. Therefore, it is crucial to improve the precision of the logical form generated by the LLM or to achieve more efficient and accurate reasoning. We will consider these limitations and address them in future work.

Acknowledgements

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

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. *semantic web*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

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.

Samuel Cahyawijaya, Holy Lovenia, and Pascale Fung. 2024. Llms are few-shot in-context low-resource language learners. *CoRR*, abs/2403.16512.

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. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611, Online and Punta Cana, Dominican Republic. 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*, pages 2106–2115.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

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

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.

Md. Arid Hasan, Shudipta Das, Afiyat Anjum, Firoj Alam, Anika Anjum, Avijit Sarker, and Sheak Rashed Haider Noori. 2024. Zero- and few-shot prompting with llms: A comparative study with fine-tuned models for bangla sentiment analysis. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 17808–17818. ELRA and ICCL.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

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

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. *arXiv preprint arXiv:2212.00959*.

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.

Zhijie Nie, Richong Zhang, Zhongyuan Wang, and Xudong Liu. 2024. Code-style in-context learning for knowledge-based question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18833–18841.

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*.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121, Toronto, Canada. Association for Computational Linguistics.

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

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

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

Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-kbqa: Multi-turn interactions for knowledge base question answering with large language models. *Preprint*, arXiv:2402.15131.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2021. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. *arXiv preprint arXiv:2109.08678*.

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.

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*.

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

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

A Appendix

A.1 Analysis of Datasets in Terms of the Number of Hop

We conduct a hop count analysis on the questions from the WebQSP and GrailQA datasets, and the results are presented in the table 4.

Dataset	1 hop	2 hop	≥ 3hop
WebQSP	65.49%	34.51%	0.00%
GrailQA	65.49%	34.51%	5.25%

Table 4: Analysis of Datasets in Terms of the Number of Hop

A.2 How Is the Expansion From 1-Hop Relation to 2-Hop Relation Done

The expansion refers to leveraging the top-k 1-hop paths obtained previously to directly use relevant SPARQL queries to reach all possible 2-hop relations from the KB. That means after obtaining the 1-hop relation, we use the following two SPARQL queries to obtain the forward and reverse 2-hop relations respectively.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax
     -ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://rdf.freebase.com/ns/>
SELECT distinct ?y as ?r1 WHERE {
"""
'?x1 :' +relation+ ' :' + entity + '. '
"""
?x2 ?y ?x1 .
FILTER regex(?y, "http://rdf.freebase.com/ns/")
}
```

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax
     -ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://rdf.freebase.com/ns/>
SELECT distinct ?y as ?r1 WHERE {
"""
'?x1 :' +relation+ ' :' + entity + '. '
"""
?x1 ?y ?x2 .
FILTER regex(?y, "http://rdf.freebase.com/ns/")
}
```

A.3 Examples of GPT-4 Performing Worse Than GPT-3.5

Take the following question as an example: "Which unit of electric current does not exceed 3.479e+25 current in amperes?". The correct logical form is as follows.

```
(AND measurement_unit.current_unit
    (le measurement_unit.current_unit.
        current_in_amperes
    3.479e+25^^http://www.w3.org/2001/XMLSchema#
        float))
```

Our provided logical form reference is as follows.

```
(AND measurement_unit.current_unit (le
     measurement_unit.current_unit.
     current_in_amperes 3.479e+25^^http://www.w3.org
     /2001/XMLSchema#float))
...
(AND meteorology.beaufort_wind_force (JOIN
     meteorology.beaufort_wind_force.
     minimum_wind_speed_km_h 3.479e+25^^http://www.
     w3.org/2001/XMLSchema#float)
```

but the logical form generated by GPT-4 is as follows.

```
(AND electrical.current_unit
    (lt electrical.current_unit.amperes
    3.479e+25^^http://www.w3.org/2001/XMLSchema#
        float))
```

And the logical form generated by gpt-3.5-turbo-0613 is as follows.

```
(AND measurement_unit.current_unit
    (NOT (lt physics.electric_current_unit.
        current_in_amperes 3.479e+25^^http://www.w3
        .org/2001/XMLSchema#float)))
```

It can be observed that **GPT-4 did not use the correct relation from the logical form reference**. The logic form generated by GPT-3.5-turbo-0613, although not entirely correct, includes the correct relation.