No Need for Large-Scale Search: Exploring Large Language Models in Complex Knowledge Base Question Answering

Shouhui Wang, Biao Qin*
School of Information, Renmin University of China, Beijing, China
{wsh_inf, qinbiao}@ruc.edu.cn

Abstract

Knowledge Base Question Answering (KBQA) systems play a pivotal role in the domain of natural language processing and information retrieval. Its primary objective is to bridge the gap between natural language questions and structured knowledge representations, especially for complex KBQA. Despite the significant progress in developing effective and interconnected KBQA technologies, the recent emergence of large language models (LLMs) offers an opportunity to address the challenges faced by KBQA systems more efficiently. This study adopts the LLMs, such as Large Language Model Meta AI (LLaMA), as a channel to connect natural language questions with structured knowledge representations and proposes a **T**hree-step **F**ine-tune **S**trategy based on large language model to implement the KBQA system (TFS-KBQA). This method achieves direct conversion from natural language questions to structured knowledge representations, thereby overcoming the limitations of existing KBQA methods, such as addressing large search and reasoning spaces and ranking massive candidates. To evaluate the effectiveness of the proposed method, we conduct experiments using three popular complex KBQA datasets. The results achieve state-of-the-art performance across all three datasets, with particularly notable results for the WebQuestionSP dataset, which achieves an $F1$ value of 79.9%.

Keywords: knowledge base question answering, large language models, information retrieval

1. Introduction

Knowledge Base Question Answering (KBQA) systems have become a prominent research area in the field of natural language processing (NLP) and information retrieval (IR) (Deng et al., 2020). These systems aim to bridge the gap between unstructured natural language questions and structured knowledge representations, thereby enabling users to retrieve precise and meaningful answers from vast knowledge bases (KBs), such as WordNet (Miller, 1995), DBpedia (Auer et al., 2007), Freebase (Bollacker et al., 2008), and Yet Another Great Ontology (YAGO) (Suchanek et al., 2007). The existing KBQA methods primarily comprise information retrieval-based (IR-based) approaches (Saxena et al., 2020; Zhang et al., 2022; Shi et al., 2021) and semantic parsing-based (SP-based) approaches (Das et al., 2021; Chen et al., 2021; Ye et al., 2022; Gu and Su, 2022; Shu et al., 2022). The former faces a large reasoning space. The latter, in the context of multi-hop question answering (QA), must rank and find targets from numerous candidates (e.g., candidate logical forms and candidate schemas), which is a challenging task. Furthermore, owing to errors in entity linking, a valid target logical form may not exist among the candidates (Ye et al., 2022). Obviously, this is not an easy task, particularly for multi-hop QA. It requires the support of high performance hardware and has a semantic gap between natural language questions and structured knowledge representations.

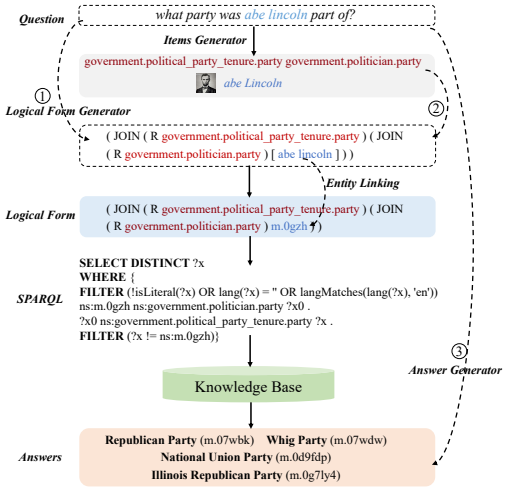


Figure 1: The objective of the semantic parsing-based (SP-based) method is to map questions into target logical forms and subsequently transform them into executable programs, such as SPARQL, to retrieve the desired answers. To achieve this goal, we introduce TFS-KBQA, a **T**hree-step **F**ine-tune **S**trategy based on large language model for KBQA, which employs three strategies.

Recently, there has been a rapid advancement of large language models (LLMs), such as GPT-3 (Brown et al., 2020), Pattern and Language Model (PaLM) (Chowdhery et al., 2022), Galacica (Taylor et al., 2022), and Large Language Model Meta AI (LLaMA) (Touvron et al., 2023a). These models have demonstrated a robust ability to comprehend and address various complex NLP tasks (Zhao

*Corresponding author

LREC-COLING 2024, pages 12288–12299

et al., 2023). For example, ChatGPT [1], as one of the most prominent applications of LLMs, demonstrates its remarkable language prowess. The impressive language comprehension ability of LLMs demonstrates their capability to address the challenges in KBQA tasks and holds promise in bridging the gap between natural language questions and structured knowledge representations.

Therefore, this study introduces a **T**hree-step **F**ine-tune **S**trategy based on LLMs for implementing the KBQA system called TFS-KBQA, which exploits the power of LLMs to effectively address KBQA tasks. Herein, we employ LLaMA, a well-known LLM with superior performance across numerous NLP tasks, as the chosen model for our strategy. In our implementation of the KBQA task, we employ three distinct strategies that work in tandem to determine the final answer to the question and significantly enhance the overall performance, as depicted in Figure 1. The first strategy leverages the powerful translation ability of LLaMA during fine-tuning, enabling us to directly convert the question into the corresponding logical form using the logical form generator guided by the provided prompt (see Strategy ① illustrated in Figure 1). The second strategy starts by utilizing the translation ability of LLaMA during fine-tuning to acquire the item generator. This allows for direct conversion of the question into relevant items, such as schema (structured knowledge representation) and entities, guided by the given prompt. Subsequently, we capitalize on the generation ability of LLaMA through fine-tuning to generate the logical form according to the obtained schemas and entities (see Strategy ② shown in Figure 1). Upon obtaining the logical forms using Strategies 1 and 2, we proceed to link the entity mentions to the corresponding entities in the KB through the process of entity linking, thus obtaining the target logical form. Subsequently, we effortlessly convert the target logical form into an executable program (e.g., SPARQL) and execute it in the KB to retrieve the answer to the question. In Strategy 3, the model relies on the provided prompt to harness the generation ability of LLaMA during fine-tuning, thereby effectively mapping the question to the answer. This enables us to directly obtain the answer from the model without additional conversion (see Strategy ③ depicted in Figure 1).

By adopting these three strategies, we smoothly convert the question into a logical form, which is then transformed into an executable query, thereby effectively deriving the answer. Consequently, this approach overcomes the limitations of previous methods, such as addressing large reasoning spaces, the challenge of ranking numerous candidates, and finding the target logical form. Moreover, these three strategies capitalize on the diverse capabilities of LLaMA, effectively complementing each other and resulting in better overall performance in the KBQA task. Using TFS-KBQA, we achieved competitive performance on three widely recognized complex KBQA datasets: WebQuestionSP (WebQSP) (Yih et al., 2016), ComplexWebQuestions (CWQ) (Talmor and Berant, 2018), and ComplexQuestions (CQ) (Bao et al., 2016). In particular, we obtained an F1 value of 79.9% for the WebQSP dataset. This result further validates the feasibility and effectiveness of utilizing LLMs for KBQA tasks, particularly in bridging the semantic gap between natural language questions and structured knowledge representations. For example, consider the question "*Where was Johannes Messenius born?*". Although the semantic expressions of the keywords "*where...born*" and schema "*place_of_birth*" are similar, they may not match exactly at the character level, indicating a semantic gap between natural language questions and structured knowledge representations. However, the success of our method in generating logical forms and schemas highlights the remarkable ability of LLMs to understand natural language. Our first attempt to use LLaMA in KBQA tasks is successful, and we hope that our study will serve as inspiration for further research in this direction.

The main contributions of this study are summarized as follows: (1) We leverage the powerful natural language understanding ability of LLM to explore the application of LLaMA in KBQA tasks, thereby effectively bridging the semantic gap between natural language questions and structured knowledge representations. (2) Our TFS-KBQA employs three strategies to implement the KBQA task, which facilitates the mapping transformation from natural language questions to logical forms, thereby overcoming challenges related to handling large reasoning spaces and numerous candidates. Additionally, these strategies complement each other to ultimately improve the performance of the KBQA system. (3) In Strategy 2, we achieve the transformation from question to logical form through two gradual steps. Each step reduces the formal difference between the input and output, thus obtaining a more accurate logical form. (4) We conduct a comprehensive evaluation of our TFS-KBQA on three widely recognized complex KBQA datasets, and obtain highly competitive results.

2. Related Works

2.1. Knowledge Base Question Answering

The current KBQA methods primarily focus on IR-based and SP-based methods. The IR-based method compares the semantic similarity between

[1] https://openai.com/blog/chatgpt

questions and their candidate answers (subgraphs) and selects the candidates with the highest scores as the final answers to the questions. The main objective is to retrieve a question-relevant subgraph and perform reasoning on it, resulting in a smaller reasoning space compared with retrieving the answer to the question across the entire KB (Zhang et al., 2022; Zhou et al., 2021; He et al., 2021; Shi et al., 2021).

SP-based methods first convert the question into a logical form, then further transform it into a structured query, and finally obtain the answer to the question. Ye et al. (2022) introduced a rank-and-generate KBQA approach, which initially ranks the candidate logical forms using a contrastive ranker and then generates the final logical form based on the top-K logical forms. However, generating candidate logical forms, particularly for complex questions, may lead to numerous candidates. This could make it challenging for the ranker to identify the target logical form from the candidates and can increase hardware demands. Moreover, because of errors in entity linking, the candidates might not contain the target logical form. Based on the study conducted by Ye et al. (2022), Shu et al. (2022) further explored the use of pre-trained language models (PLMs) by improving zero-shot mention detection, matching entity-independent semantics with schema retrievers, and reducing generation errors with constrained decoding.

Similarly, Chen et al. (2021) and Das et al. (2021) adopted a two-step approach to transform questions into logical forms. The former employed a retriever to obtain relevant KB items (e.g., entities and schema) and then utilized a transducer to generate the logical form. Moreover, the latter used a neural retriever to retrieve similar queries (and their logical forms) and generated a logical form based on the retrieved cases. To reduce the search space, Gu and Su (2022) proposed a generation-based model that directly transforms the question into a logical form using the seq2seq method. This method effectively addresses the challenges of a large search space and schema linking in KBQA through dynamic program induction and dynamic contextualized encoding. Additionally, Lan and Jiang (2020) incorporated constraints into query graphs to prune the search space and introduced a modified staged query graph generation method to handle complex questions with multi-hop relations and constraints. Zhang et al. (2023) extracted relevant fine-grained knowledge components from the KB and reformulated them into middle-grained knowledge pairs to generate the final logical expressions. Gu et al. (2023) proposed a generic framework for grounded language understanding that capitalizes on the discriminative ability of language models and achieved success in the field of KBQA.

2.2. Pre-trained Language Model on KBQA

Following the emergence of neural language models, the rapid development of PLMs has significantly advanced the field of KBQA, making them the primary choice for solving KBQA tasks (Saxena et al., 2020; Chen et al., 2021; Das et al., 2021; Gu and Su, 2022; Cao et al., 2022; Yan et al., 2021; Yu et al., 2022a). PLMs, which are language models pre-trained on large-scale corpora, have demonstrated remarkable capabilities in understanding natural language and handling complex QA tasks. For example, Ye et al. (2022) proposed the Rank-and-Generate KBQA (RnG-KBQA) method, which utilizes BERT in the logical form ranking stage and employs the sequence-to-sequence (seq2seq) generation method (T5) to convert query sequences into logical form sequences during the target logical form generation stage. Similarly, Shu et al. (2022) introduced the TIARA method, where BERT was used in the schema retrieval phase, and T5 was applied in the target logical form generation phase.

With the continuous advancement in hardware and the availability of large-scale corpora, researchers have been actively developing LLMs, such as GPT-X, LLaMA, and PaLM. These models use a significant number of parameters and demonstrate robust language understanding and generation capabilities, thereby enabling their wide application in various NLP tasks. LLMs have demonstrated remarkable success in various tasks, including text classification, machine translation, question answering, and dialogue systems, making them a prominent and mainstream approach in the field of NLP (Yu et al., 2022b; Kasneci et al., 2023; Aher et al., 2023; Zhu et al., 2023; Singhal et al., 2023; Tinn et al., 2023). This has created an opportunity to leverage LLMs for KBQA and to address the challenges unique to this domain. Consequently, we opt to utilize LLaMA, which currently exhibits superior performance in various NLP tasks, for KBQA and introduce the TFS-KBQA, which can effectively overcome the challenges associated with retrieving target candidates from a massive candidate pool and address a large search space.

3. Our Approach

Our TFS-KBQA leverages the potential of LLaMA in NLP tasks and implements the KBQA task using three different strategies, as shown in Figure 1. To comprehensively demonstrate the design and achievement of TFS-KBQA, we present a detailed and systematic investigation of each of the three strategies in subsequent sections.

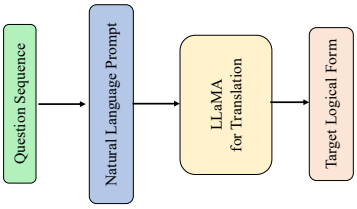
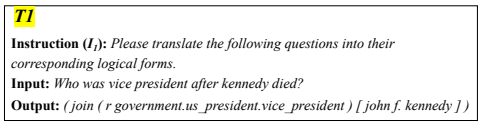


Figure 2: The architecture of Strategy 1 implements the direct conversion from the question to the target logical form.



3.1. Preliminaries

A KB generally contains a substantial number of triples, which are represented in the form of <*Subject*, *Predicate*, *Object*>. Each of these triples is referred to as a fact. We generally represent facts as <*s*, *r*, *o*>, where *s* denotes an entity; *o* denotes an entity or literal, such as data time and integer values; and *r* denotes the binary relation between *s* and *o*. For example, consider the following fact: <*m.01b716, people.person.place_of_birth, m.01m8_w*>, where "*m.01b716*" and "*m.01m8_w*" represent entities and "*people.person.place_of_birth*" represents the binary relation.

In our task, the primary objective is to convert a natural language question Q into an equivalent logical form L, which can easily be converted into a SPARQL query to retrieve the answer. Therefore, we adopt s-expressions to represent the KB query, following the precedent set by previous studies (Gu et al., 2021; Ye et al., 2022; Gu and Su, 2022; Shu et al., 2022). S-expressions strike a balance between compactness, composition, and readability, making them conducive to effectively utilizing the seq2seq model for the Q-to-L transformation (Gu et al., 2021). For example, for the question "*Who was the vice president after Kennedy died?*", the equivalent logical form is "*(join (r government.us_president.vice_president) m.0d3k14)*."

3.2. Strategy 1

By leveraging the advantages of LLMs in the field of NLP, instruction-based fine-tuning for addressing downstream tasks is a technique that has demonstrated significant effectiveness in recent LLM developments (Taori et al., 2023; Wang et al., 2022). In Strategy 1, we define a task $T1$, which comprises a given dataset $D_1 = \{(X_i^1, Y_i^1)\}_{i=1}^N$ containing N input–output instances and a specific task instruction I_1. After fine-tuning, we obtain model M_1, which can directly convert a given natural language question into the target logical form. For model M_1, when provided with the task instruction I_1 and the corresponding input X_i^1, we can obtain the target logical form Y_i^1, where $M(I_1, X_i^1) = Y_i^1$, for $i \in \{1, 2, ..., N\}$. As an illustration, we utilize the pre-trained general LLaMA, its parameters, and a training dataset, along with our defined instruction "*Please translate the following questions into their corresponding logical forms.*", to perform fine-tuning. This process results in a logical form generator model. After the model undergoes instruction-based fine-tuning and receives the question "*Who was vice president after Kennedy died?*", the corresponding output of the model is "*(join (r government.us_president.vice_president) [john f. kennedy])*" (see Task $T1$).

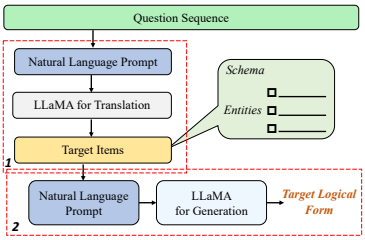
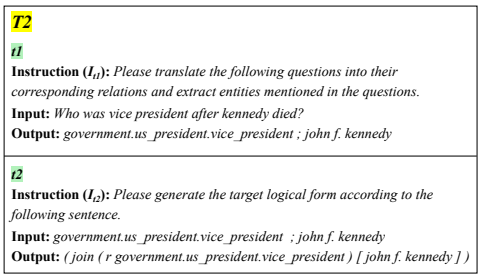


Figure 3: The architecture of Strategy 2 involves transforming the question into the target logical form in two steps.



3.3. Strategy 2

In Strategy 1, we achieve a direct transformation from natural language questions to logical forms. We believe that the span of this transformation from the question to the logical form is quite significant, which rigorously tests the ability of LLaMA to comprehend and generate natural language. Even a minor error in the model can lead to incorrect logical forms, potentially affecting the performance of the KBQA task. Therefore, in Strategy 2, we propose dividing the conversion process from a question to a logical form into two steps. The first step involves converting the question into the following KB relevant items: schemas and entities. By adopting this approach, the LLaMA only needs correctly understand the question and perform the transformation to items at the semantic level, without generating any semantically unrelated characters, such as "*)*," "*[*," "*r*," "*join*," and "*argmax*." In Step 2, we generate

the target logical form based on the items obtained in the first step. During this phase, LLaMA only needs understand the structure of the logical form and make necessary additions to the items to arrive at the final target logical form. By dividing the generation of the target logical form into these two steps, we gradually reduce the formal differences between the input and output at each stage. This enables us to obtain more accurate logical forms and enhances the overall performance of the KBQA.

In Strategy 2, we decompose Task $T2$ into two subtasks: $t1$ and $t2$. Below, we provide a detailed introduction to these two subtasks. For task $t1$, we construct a dataset $D_{t1} = \{(X_i^{t1}, Y_i^{t1})\}_{i=1}^{N}$, which contains N [2] input–output instances along with the specific task instruction I_{t1}. Using the LLaMA and performing fine-tuning on the dataset D_{t1}, we obtain the model M_{t1}, which can effectively transform natural language questions into relevant KB items. For instance, given the instruction "*Please translate the following questions into their corresponding relations and extract entities mentioned in the questions*," after fine-tuning the dataset D_{t1}, we obtain the item generator model M_{t1}. When the model receives the input "*Who was vice president after Kennedy died?*", the corresponding output of the model is "*government.us_president.vice_president; John F. Kennedy*," where "*government.us_president.vice_president*" denotes the schema and "*john f. kennedy*" denotes the entity mentioned in the question (see Task $T2$ ($t1$)). Compared with task $T1$, task $t1$ reduces the formal difference between the input and output, thereby enabling a more effective transformation from natural language questions to semantically relevant KB items. This improvement enhances the performance of task $t1$, which is the premise for obtaining a more accurate target logical form.

In task $t2$, we construct a dataset $D_{t2} = \{(X_i^{t2}, Y_i^{t2})\}_{i=1}^{N}$ that comprises N input–output instances and a specific task instruction I_{t2}. The primary objective of this task is to complement the relevant KB items obtained in task $t1$ by introducing specific symbols and characters, such as "*)*," "*[*," and "*join*", to facilitate the conversion of the items into the target logical forms. For instance, given the task instruction "*Please generate the target logical form based on the following sentence*" and the dataset D_{t2}, we fine-tune the model on this dataset to obtain the model M_{t2}. When the model receives the input "*government.us_president.vice_president; john f. kennedy*", the corresponding output is "*(join (r government.us_president.vice_president) [john f. kennedy])*" (see Task $T2$ ($t2$)). In summary, this task complements the information obtained in task

[2]Across various tasks using the same dataset, the number of instances remains consistent.

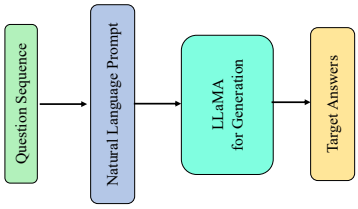


Figure 4: The architecture of Strategy 3 allows for the direct generation of the target answers to the question.

T3

Instruction (I_3): *Please provide answers to the following questions based on your understanding.*
Input: *Who was vice president after kennedy died?*
Output: *Lyndon B. Johnson*

$t1$ by incorporating specific symbols, resulting in the generation of the target logical form. This is a simplified and less complex task compared to task $T1$ within the LLaMA. Moreover, because simpler tasks are more suitable for LLaMA, we can obtain a more accurate target logical form, thereby improving the performance of KBQA tasks.

3.4. Strategy 3

Inspired by the study conducted by Yu et al. (2022a), we decided to use the powerful natural language understanding capabilities of LLaMA for direct question answering. Strategy 3 complements the question answers generated by the target logical forms in Strategies 1 and 2, thereby further improving the KBQA performance. In this strategy, we define Task $T3$ and construct the dataset $D_3 = \{(X_i^3, Y_i^3)\}_{i=1}^{N}$ that contains specific task instructions I_3 and N input–output instances. By fine-tuning the LLaMA on the dataset D_3, we obtain the answer generator model M_3, which efficiently provides answers when we input questions. For example, given the question "*Who was the vice president after Kennedy died?*" and the task instruction"*Please provide answers to the following questions based on your understanding.*", our model M_3 yields the following answer to the question: *Lyndon B. Johnson* (see Task $T3$). Overall, leveraging the capabilities of LLaMA and implementing Strategy 3 allow us to directly obtain accurate answers to questions, thereby improving the overall performance of KBQA.

3.5. Entity Linking

In Freebase, the Machine-Generated Identifier (*MID*) [3] is an important element for uniquely identifying the entities; its form is similar to "*m.012_0k9*". Entity linking maps an entity mention to its corresponding *MID* in Freebase. For example, given an entity "*john f. kennedy*," we map it to its corresponding *MID* "*m.0d3k14*" in Freebase.

[3]https://developers.google.cn/freebase?hl=zh-cn

Although obtaining the final target logical form is critical in solving the KBQA task, the entity linking process is also very important. For example, in the RnG-KBQA method proposed by Ye et al. (2022), an incorrect result from entity linking can lead to the absence of the target logical form among the candidate logical forms, directly affecting the performance of the ranker model. To ensure improved entity linking accuracy, we employ two implementation methods. One of the methods involves utilizing the Google knowledge graph search application programming interface (API) [4], which is used in QGG (Lan and Jiang, 2020) to implement entity linking. The other approach utilizes the most advanced entity linking system, ReFinED [5], as proposed by Tom Ayoola (2022b,a). This system offers higher accuracy and significantly reduces costs compared to existing approaches. By employing these two methods, we enhance the accuracy of entity linking, thereby contributing to the overall effectiveness of the KBQA system.

4. Experiments

4.1. Datasets

We validate our TFS-KBQA on three representative complex KBQA datasets.

WebQSP was created by Yin et al. (2016) to evaluate the costs and benefits of gathering semantic parse labels, including semantic parses for questions. It comprises a training set with 3,298 samples and a test set with 1,639 samples.

CWQ was introduced by Talmor and Berant (2018) and is an extended version of WebQSP with more hops and constraints. Following previous practices, we split the training, validation, and test sets into 27,623, 3,518, and 3,531 samples, respectively.

CQ was released by Bao et al. (2016) and includes 2,100 QA pairs, with 1,300 and 800 for training and testing, respectively. Notably, this dataset does not provide ground truth query graphs. Therefore, we assess the success of our fine-tuned model when applied to this dataset.

4.2. Evaluation Metrics

For WebQSP, we use $F1$ and Hits@1 as metrics, where $F1$ is computed using the official evaluation script. Hits@1 is calculated according to the method provided by Shu et al. (2022), where we randomly select an answer for each question 100 times and calculate the average Hits@1 [6]. For the other two datasets, we follow standard practice and report the $F1$ score, which is calculated based on the predicted results and the gold answer set.

4.3. Implementation Details

Our models are based on LLaMA foundation models. For our experiment, we utilized the latest version of LLaMA-2, which was developed and publicly released by Meta. LLaMA-2 is available in different sizes: 7 billion, 13 billion, and 70 billion parameters. In our study, we fine-tuned the 7 billion and 13 billion parameter LLaMA variants (**meta-llama/Llama-2-7b-hf** [7] and **meta-llama/Llama-2-13b-hf** [8]), following the approach outlined by Touvron et al. (2023b). During the training process, we adopted the LoRA partial parameter fine-tuning method, used the $AdamW$ optimization method (Loshchilov and Hutter, 2018) for 3 epochs, and utilized one NVIDIA v100-32G GPU with a batch size of 8. The peak learning rate was set to $3e$-4, and the maximum input length was limited to 128 tokens. We performed continuous training based on the fine-tuned (LoRA) model. This process involved merging the fine-tuned LoRA model with the base model to generate a new base model, followed by another round of fine-tuning. Additionally, to further enhance the KBQA performance, we comprehensively considered the results obtained from both the 13B and 7B parameter models. Our experiments showed that if the results from the former model are incorrect, the latter model may produce the correct results, indicating a degree of complementarity between the two. For further details on other parameter settings, please refer to the source code available on GitHub [9].

4.4. Baselines for Comparison

We comprehensively compared our TFS-KBQA with state-of-the-art (SOTA) models on three datasets. Our main focus is to compare the **RnG-KBQA** (Ye et al., 2022), **ReTraCk** (Chen et al., 2021), **ArcaneQA** (Gu and Su, 2022), **TIARA** (Shu et al., 2022), and **FC-KBQA** (Zhang et al., 2023) methods with TFS-KBQA because they share common characteristics with it. In particular, these methods first obtain the target logical form and then convert it into an executable program to execute the query, which is a widely used strategy in the current advanced methods. However, the most significant difference between them depends on the approaches and strategies used to obtain the target logical form. For example, **RnG-KBQA** employs BERT to rank candidate logical forms and

[4] https://developers.google.com/knowledge-graph

[5] https://github.com/amazon-science/ReFinED/tree/main

[6] SP-based methods yield unordered answers, which are usually evaluated by $F1$ rather than Hits@1.

[7] https://huggingface.co/meta-llama/Llama-2-7b-hf

[8] https://huggingface.co/meta-llama/Llama-2-13b-hf

[9] https://github.com/shouh/TFS-KBQA

T5 to generate the target logical forms based on the top-K candidates. **ArcaneQA** utilizes dynamic program induction and dynamic-contextualized encoding strategies to generate the target logical form using their generation-based model. **FC-KBQA** is a fine-to-coarse composition framework designed to extract knowledge components and subsequently generate the final logical expressions. Additionally, there are other methods, such as **QGG** (Lan and Jiang, 2020), that focus on query-graph generation, **SR** (Zhang et al., 2022), which relies on subgraph retrieval, **Pangu** (Gu et al., 2023), which is a generic framework for grounded language understanding, and **TransferNet** (Shi et al., 2021) infers the answer by transferring entity scores along relation scores over multiple steps.

4.5. Results

In this section, we present the results of our TFS-KBQA on three datasets. Tables 1, 2, and 3 demonstrate that TFS-KBQA outperforms the previous SOTA approach (**Pangu**) in the WebQSP dataset, achieving higher $F1$ score at 79.9% $(+0.3)$. The Hits@1 is 79.8%, which does not exceed the IR-based method (**SR**) [10]. In the CWQ dataset, our TFS-KBQA surpasses the current SOTA approach (**Program Transfer**) by a significant margin, achieving an $F1$ score of 63.6% $(+4.9)$. Similarly, in the CQ dataset, our TFS-KBQA outperforms the current SOTA method (**QGG**) with a competitive $F1$ score of 44.0% $(+0.7)$. These experimental results demonstrate that the strong semantic understanding ability of LLMs enables our method to handle more complex questions. For example, CWQ has more hops and constraints. Moreover, our TFS-KBQA achieves an improvement in KBQA performance through the mutual promotion and supplementation of various strategies.

We further conducted an ablation experiment to assess the effect of the three proposed strategies on the final KBQA task. Analyzing the experimental results across the three datasets, it is evident that the removal of any of these strategies decreases KBQA performance. This demonstrates that when simultaneously employing these three strategies for KBQA tasks, they complement each other. If a question cannot be answered by Strategy 1, the result of Strategy 2 can be used as an answer. Similarly, when Strategies 1 and 2 fail to provide answers, Strategy 3 can directly address the question. Notably, the omission of Strategy 3 results in a significant decrease in KBQA performance. For instance, in the WebQSP dataset, the removal of Strategy 3 causes the $F1$ score to decrease from 79.9% to 77.4%. This underscores the significance of directly generating answers using the instruction + fine-tuning method, which serves as a valuable supplement for effectively addressing complex KBQA tasks.

[10]The answers to the questions using the IR-based method are ordered, and the top-1 answer is considered the best. In contrast, answers obtained using SP-based methods are typically unordered. Hence, the two methods differ in terms of Hits@1.

Method	F1	Hits@1
IR-based Methods		
TransferNet (Shi et al., 2021)	–	71.4
NSM (He et al., 2021)	67.4	74.3
SR (Zhang et al., 2022)	74.5	**83.2**
SP-based Methods		
QGG (Lan and Jiang, 2020)	74.0	–
ReTraCk (Chen et al., 2021)	71.0	71.6
RnG-KBQA (Ye et al., 2022)	75.6	–
ArcaneQA (Gu and Su, 2022)	72.8	–
Program Transfer (Cao et al., 2022)	76.5	74.6
TIARA (Shu et al., 2022)	76.7	73.9
Pangu (Gu et al., 2023)	**79.6**	–
FC-KBQA (Zhang et al., 2023)	76.9	–
TFS-KBQA (Ours)	**79.9**	79.8
-Only use LLaMA-2-13b	79.5	79.5
-Only use LLaMA-2-7b	78.5	78.4
-w/o ReFinED entity linking	78.5	78.6
-w/o ELQ entity linking Δ	79.2	79.2
-w/o Google API entity linking	78.2	78.3
-w/o Strategy 1	78.5	78.6
-w/o Strategy 2	79.7	79.6
-w/o Strategy 3	77.4	76.9

Table 1: Results of F1 and Hits@1 for TFS-KBQA and the compared baselines on WebQSP. The two blocks of the baselines are IR-based and SP-based methods. Δ: We add the ELQ entity linking results commonly used in many studies to this dataset. Please note that 'w/o' is the abbreviation of without.

Additionally, to ensure the accuracy of entity linking results, we employ two entity linking methods: ReFinED and Google API. It is important to highlight that we incorporated the entity linking results from ELQ (Li et al., 2020), which is a widely used entity linker in numerous studies, into the WebQSP dataset (Ye et al., 2022; Shu et al., 2022). By analyzing the experimental results presented in Table 1–3, it shows that the removal of any entity linker results in a decrease in the KBQA performance, indicating that different entity linkers can complement each other. For instance, consider the entity mention "*palace of knossos*." The ReFinED method yields the entity linking result "*m.0ksnn*," while the Google API yields "*m.0123zb8k*," with "*m.0ksnn*" being the correct entity (see the case study in the Appendices for detail.). Conversely, there are instances in which the entity linking result of the ReFinED method is incorrect, and the Google API

Method	F1
PullNet (Sun et al., 2019)	47.2
QGG (Lan and Jiang, 2020)	40.4
NSM (He et al., 2021)	48.8
TransferNet (Shi et al., 2021)	48.6
RnG-KBQA (Ye et al., 2022)	42.3
Program Transfer (Cao et al., 2022)	**58.7**
FC-KBQA (Zhang et al., 2023)	53.1
TFS-KBQA (Ours)	**63.6**
-Only use LLaMA-2-13b	62.9
-Only use LLaMA-2-7b	62.2
-w/o ReFinED entity linking	61.1
-w/o Google API entity linking	61.4
-w/o Strategy 1	60.0
-w/o Strategy 2	62.7
-w/o Strategy 3	49.1

Table 2: Comparison between our TFS-KBQA and the baselines on CWQ.

Method	F1
Constraint-Based (Bao et al., 2016)	42.3
Embedding-Based (Luo et al., 2018)	42.8
QGG (Lan and Jiang, 2020)	**43.3**
TFS-KBQA (Ours)	**44.0**
-Only use LLaMA-2-13b	43.7
-Only use LLaMA-2-7b	41.1
-w/o ReFinED entity linking	42.8
-w/o Google API entity linking	42.7
-w/o Strategy 1	43.4
-w/o Strategy 2	43.3
-w/o Strategy 3	30.4

Table 3: F1 on CQ compared to baseline methods. Because this dataset does not provide the ground truth query graphs, we first employ the model that has been fine-tuned on the CWQ dataset to obtain the target logical form in Strategies 1 and then obtain 440 correct samples out of the initial 1,300 samples. We then use these correct samples to fine-tune our model.

provides the correct entity. This indicates the significance of entity linking in solving KBQA tasks because there are numerous cases in which our method correctly obtains the logical form, but entity linking errors result in inaccuracies in the final logical form.

Furthermore, we conducted experiments using both the LLaMA-2-7b model with 7-billion parameters and the LLaMA-2-13b model with 13-billion parameters to validate the enhanced capabilities of larger LLaMA-2 models in natural language understanding and processing. Based on the results presented in Tables 1, 2, and 3, we observed that compared to LLaMA-2-13b, utilizing only LLaMA-2-7b resulted in varying degrees of performance degradation in KBQA tasks. Particularly for CQ datasets, its $F1$ value dropped from 44.0% to 41.1%. This discrepancy can be attributed to the increased complexity of the CQ datasets compared with that of the WebQSP datasets, which highlights the fact that larger LLMs possess superior language understanding and reasoning abilities, making them advantageous for tackling more intricate questions. It is essential to note that larger LLMs also requires higher performance hardware. Deploying LLaMA-2-7b requires an NVIDIA TITAN-RTX-24G GPU, while LLaMA-2-13b requires an NVIDIA V100-32G GPU. For the larger LLaMA-2-70b model, an NVIDIA A800-80G GPU is essential. Therefore, when the hardware conditions permit, opting for a larger LLM may yield benefits. Therefore, under favorable hardware conditions, it can be inferred that using a larger LLaMA-2-70b model could potentially further enhance KBQA performance. Furthermore, owing to the absence of ground truth query graphs in the CQ dataset, we employed a fine-tuned model trained on WebQSP and CWQ to process the training dataset. This process yielded 440 correct samples out of the initial 1,300 samples, thereby enabling us to fine-tune various models and surpass the current SOTA method (**QGG**). Consequently, we anticipate that incorporating more training samples may lead to further improvements in the performance on this dataset.

In the following analysis, we will examine various prediction cases. We observe that Strategy 1, which directly converts natural language questions into logical forms, performs better on simple questions (one-hop or two-hop) than on complex questions (three hops or more). For complex questions, the predicted logical forms mostly agree with the golden logical forms, but deviations in certain details or schemas lead to final errors. For example, as shown in Table 4, our prediction and golden results differ in the form of the schema, resulting in KBQA errors. To improve KBQA performance, we utilize Strategy 2 to complement the prediction results of Strategy 1.

When the logical form obtained using Strategy 1 cannot be used for querying, we consider the results of Strategy 2. Strategy 2 involves two subtasks. The first subtask ($t1$) directly maps the question to relevant KB items. The second subtask ($t2$) generates a logical form based on the predicted items. Because we obtain the correct schema and entity in the first subtask, we eventually obtain the correct logical form. For example, the question "*What currency is used in the jurisdiction where the Cabinet of Peru is located?*" cannot be answered using Strategy 1, but through our experiment, we found that it can be resolved using Strategy 2. How-

Example
[*Strategy 1*]
Question: *What currency is used in the jurisdiction where the Cabinet of Peru is located?*
Predict: (join (r location.country.currency_used) (join government.governmental_jurisdiction.government [cabinet of peru]))
Golden: (join (r location.country.currency_used) (join government.governmental_jurisdiction.governing_officials (join government.government_position_held.governmental_body [cabinet of peru])))
[*Strategy 2*]
Question: *What languages are spoken where Haitian Creole is spoken?*
Predict (t1): location.country.languages_spoken location.country.languages_spoken ; haitian creole
Golden: location.country.languages_spoken location.country.official_language ; haitian creole
Predict (t2): (join (r location.country.languages_spoken) (join location.country.languages_spoken [haitian creole]))
Golden: (join (r location.country.languages_spoken) (join location.country.official_language [haitian creole]))
[*Strategy 3*]
Question: *What artist recorded Elf's Lament?*
Predict: singer, actor, film producer, songwriter, record producer, radio personality, music artist, artist, songwriter
Golden: actor, singer, songwriter
[*Entity Linking*]
Question: *What is the type of government practiced in the country where the Israeli Lira is used?*
Logical Form: (join (r location.country.form_of_government) (join location.country.currency_formerly_used [israeli lira]))
Target Logical Form: (join (r location.country.form_of_government) (join location.country.currency_formerly_used m.04r5x5 m.036g0k))

Table 4: The examples of three strategies on the CWQ dataset include a wrong example for each strategy. Red words represent the inconsistent schema items. The content contained in [ ] represents entity mentions. Blue word represents the wrong entity linking result, and brown word represents the golden entity linking result.

ever, if the first subtask prediction fails in Strategy 2, it will directly lead to an incorrect logical form in the second subtask. For example, as depicted in Table 4, for the question "*What languages are spoken where Haitian Creole is spoken?*", we obtain the wrong schema "*location.country.language_spoken*" in the first subtask. Although "*language_spoken*" and "*official_language*" in the golden schema are semantically the same, the difference in form results in the final error.

When both Strategies 1 and 2 fail, we resort to Strategy 3 to generate answers directly. However, as shown in the Table 4, we found that the answers we obtain may be more or less than the correct answers, or they may be entirely different. If the correct answers cannot be obtained through Strategy 3, then our TFS-KBQA cannot answer the question. Moreover, errors leading to the final KBQA results can arise from entity linking. When obtaining the logical form, we use entity linking technology to link entity mentions to the KB. If entity linking yields incorrect results, we will not obtain the correct answer to the question. For example, as shown in Table 4, given the question "*What is the type of government practiced in the country where the Israeli Lira is used?*", we obtain the correct logical form. However, when we link "*Israeli Lira*" to the KB through entity linking, we incorrectly connect it to entity "*m.04r5x5*" instead of the golden entity "*m.036g0k*", leading to the wrong target logical form, which eventually causes the KBQA error.

5. Conclusion

In this study, we utilize the latest LLaMA-2 to propose the TFS-KBQA method for KBQA tasks. This method implements KBQA tasks using three strategies, enabling it to use LLMs more effectively. Simultaneously, the three methods complement each other, synergistically contributing to the overall performance of our KBQA system. By evaluating the three widely used complex KBQA datasets, the experimental results demonstrate that employing LLMs for KBQA tasks is a straightforward, feasible, and universal approach. This study represents an initial exploration of employing LLMs for KBQA tasks. In the future, we aim to conduct more extensive studies based on this foundation. For example, we will explore how to obtain more appropriate instructions for a specific task.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61772534 and in part by the Public Computing Cloud, Renmin University of China.

Gati V Aher, Rosa I Arriaga, and Adam Tauman Kalai. 2023. Using large language models to simulate multiple humans and replicate human subject studies. In *International Conference on Machine Learning*, pages 337–371. PMLR.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735, Berlin, Heidelberg. Springer Berlin Heidelberg.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: technical papers*, pages 2503–2514.

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.

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, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8128–8140.

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

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*.

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.

Chaoyu Deng, Guangfu Zeng, Zhiping Cai, and Xiaoqiang Xiao. 2020. A survey of knowledge based question answering with deep learning. *Journal of Artificial Intelligence*, 2(4):157–166.

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.

Yu Gu and Yu Su. 2022. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731.

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.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274.

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

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *EMNLP*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

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.

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

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

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, pages 1–9.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

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

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

Robert Tinn, Hao Cheng, Yu Gu, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2023. Fine-tuning large neural language models for biomedical natural language processing. *Patterns*, 4(4).

Andrea Pierleoni Tom Ayoola, Joseph Fisher. 2022a. Improving entity disambiguation by reasoning over a knowledge base. In *NAACL*.

Joseph Fisher Christos Christodoulopoulos Andrea Pierleoni Tom Ayoola, Shubhi Tyagi. 2022b. ReFinED: An efficient zero-shot-capable approach to end-to-end entity linking. In *NAACL*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

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*.

Yuanmeng Yan, Rumei Li, Sirui Wang, Hongzhi Zhang, Zan Daoguang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Large-scale relation

learning for question answering over knowledge bases with pre-trained language models. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 3653–3660.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043.

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. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1746–1756.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2022a. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2022b. Generate rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations*.

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.

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Guangyou Zhou, Zhiwen Xie, Zongfu Yu, and Jimmy Xiangji Huang. 2021. Dfm: A parameter-shared deep fused model for knowledge base question answering. *Information Sciences*, 547:103–118.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.