FlexKBQA: A Flexible LLM-Powered Framework for Few-Shot Knowledge Base Question Answering

Zhenyu Li[1*], Sunqi Fan[1*], Yu Gu[2], Xiuxing Li[3, 4†], Zhichao Duan[1], Bowen Dong[1], Ning Liu[5], Jianyong Wang[1†]
[1]Tsinghua University
[2]The Ohio State University
[3]University of Chinese Academy of Sciences
[4]Key Laboratory of Intelligent Information Processing Institute of Computing Technology, CAS
[5] Shandong University

Abstract

Knowledge base question answering (KBQA) is a critical yet challenging task due to the vast number of entities within knowledge bases and the diversity of natural language questions posed by users. Unfortunately, the performance of most KBQA models tends to decline significantly in real-world scenarios where high-quality annotated data is insufficient. To mitigate the burden associated with manual annotation, we introduce ***FlexKBQA*** by utilizing Large Language Models (LLMs) as program translators for addressing the challenges inherent in the few-shot KBQA task. Specifically, ***FlexKBQA*** leverages automated algorithms to sample diverse programs, such as SPARQL queries, from the knowledge base, which are subsequently converted into natural language questions via LLMs. This synthetic dataset facilitates training a specialized lightweight model for the KB. Additionally, to reduce the barriers of distribution shift between synthetic data and real user questions, ***FlexKBQA*** introduces an execution-guided self-training method to iterative leverage unlabeled user questions. Furthermore, we explore harnessing the inherent reasoning capability of LLMs to enhance the entire framework. Consequently, ***FlexKBQA*** delivers substantial flexibility, encompassing data annotation, deployment, and being domain agnostic. Through extensive experiments on GrailQA, WebQSP, and KQA Pro, we observe that under the few-shot even more challenging zero-shot scenarios, ***FlexKBQA*** achieves impressive results with a few annotations, surpassing all previous baselines and even approaching the performance of supervised models, achieving a remarkable 93% performance relative to the fully-supervised models. We posit that ***FlexKBQA*** represents a significant advancement towards exploring better integration of large and lightweight models. Code is available at https://github.com/leezythu/FlexKBQA.

Introduction

Knowledge base question answering (KBQA) plays a crucial role in leveraging the substantial knowledge stored in knowledge bases and making it accessible to users (Berant et al. 2013; Yih et al. 2016; Gu et al. 2022). With the ever-growing size and complexity of knowledge bases (KBs), effective KBQA systems are becoming increasingly important. Many current KBQA systems depend on supervised training using large sets of manually annotated data. This approach brings about significant labor challenges for two main reasons: (1) The substantial size and comprehensive scope of the KB, coupled with its ever-evolving content, lead to a vast sample space (Berant et al. 2013; Gu et al. 2021). This complicates the process of collecting a representative and exhaustive set of training data. (2) The heterogeneity in KB schema and design protocols, combined with differences in the query languages, such as SPARQL (Su et al. 2016), S-expression (Gu et al. 2021), and KoPL (Cao et al. 2022a), requires tailored training processes for each KB. This intensifies the labor involved in developing and adapting models for each distinct KB. Considering these complexities, there

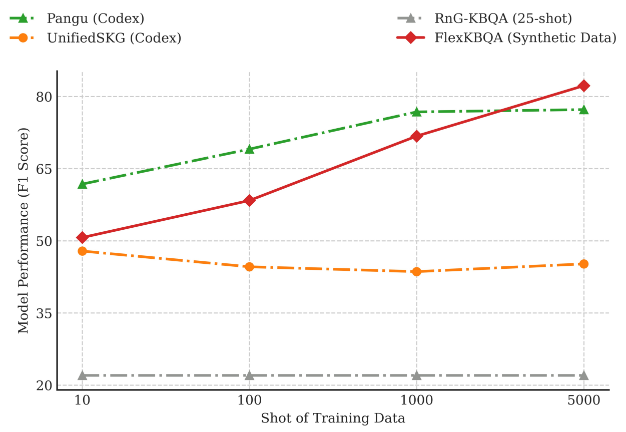

Figure 1: An analytical experiment on 500 random samples of GrailQA dev set (oracle entity linking). FlexKBQA's performance exhibits a consistently upward trend with the increasing synthetic data size, surpassing all in-context learning models with limited window length. Since our synthetic data are generated based on 25-shot real data, we also depict the performance of our underlying model (RnG-KBQA) trained by 25-shot real data as a baseline.

*These authors contributed equally.
†Corresponding author

emerges an imperative to forge *a flexible framework that can efficiently build adaptable KBQA models for different KBs and query languages, utilizing only a limited set of annotated examples*.

The emergence of large language models (LLMs) in recent times, such as Codex (Chen et al. 2021a) and GPT-4 (OpenAI 2023), suggests promising avenues for constructing such frameworks. LLMs have showcased versatility across diverse tasks and remarkable generalization with minimal demonstrations (Wei et al. 2022; Li et al. 2023a), as exemplified in Cheng et al. (2023), where Codex outperforms previous fine-tuning methods with only a handful of examples. Such feats hint at LLMs' potential in KBQA. However, research on LLMs' utility for few-shot KBQA remains scant. Initial explorations by Li et al. (2023a) and Gu, Deng, and Su (2023) employ in-context learning, allowing LLMs to transform questions into programs using minimal demonstrations. Nonetheless, this paradigm possess several limitations. First, LLMs inherently scuffle with constraints such as limited context window and considerable inference overhead. Furthermore, LLMs, despite their generalization capacities, face challenges in addressing the intricacies embedded within domain-specific KBs when not fine-tuned, leading to inaccuracies when generating target entities and relations (Li et al. 2023a). Additionally, without fine-tuning, the benefits of in-context learning diminish as more training data is introduced (Gu, Deng, and Su 2023).

In light of these challenges, we present FlexKBQA (see Figure 2), a flexible KBQA framework harnesses the advanced generative abilities of LLMs to produce synthetic data, which in turn aids in the training of lightweight models. Specifically, FlexKBQA first leverages collected templates of structured queries to sample a substantial number of programs (i.e., S-expressions) from the KB. These programs are then converted into coherent natural language questions using LLMs, ensuring both the accuracy of the programs and the fluency of the generated questions. Upon generating these program-question pairs, they serve as valuable resources for the fine-tuning of lightweight models. However, there could be a potential distribution shift between synthetic data and real user queries. To bridge this gap, we further introduce the execution-guided self-training (EGST) method, which employs the fine-tuned lightweight model to proactively annotate real user queries over the KB. These annotated queries subsequently serve as valuable training data, enabling self-improvement. FlexKBQA addresses the inherent constraints of the context window of LLMs, enabling the KBQA model to exhibit consistent improvement as more training data is incorporated (Figure 1). Additionally, by delegating the inference process to lightweight models rather than relying on the LLM directly, our approach ensures enhanced efficiency in inference.

We conduct extensive experiments on three representative datasets - GrailQA, WebQSP, and KQA Pro. FlexKBQA outperforms existing models by a large margin in the few-shot setting. Remarkably, with a mere 25 labeled examples, FlexKBQA surpasses the performance of all previous methods utilizing 100 shots on GrailQA. Furthermore, FlexKBQA's results are comparable to several fully supervised models on both GrailQA and WebQSP datasets.

The main contributions can be highlighted as follows:

- We present an efficient and flexible KBQA framework, capitalizing on the notable generative capabilities of LLMs.
- We introduce the execution-guided self-training strategy to address the distribution shift challenge by facilitating interaction between heterogeneous information and leveraging the inherent reasoning ability of LLM.
- Experimental results demonstrate that FlexKBQA significantly outperforms all baselines on real-world datasets under different settings.
- To the best of our knowledge, our work represents the inaugural endeavor in exploring the zero-shot KBQA.

Related Work

Few-Shot Language Understanding with LLMs

Researchers have been dedicated to studying the language understanding capabilities in the zero-shot or few-shot setting (Gao et al. 2019; Li et al. 2022, 2023b). In recent years, LLMs have demonstrated strong few-shot learning abilities across various tasks, such as question answering (Cheng et al. 2023), code generation (Ni et al. 2023), and embodied agents (Singh et al. 2023). Recently, Pangu (Gu, Deng, and Su 2023) and KB-BINDER (Li et al. 2023a) have pioneered the exploration of the few-shot setting on the KBQA task. They conduct experiments under the in-context learning paradigm, providing a few question-program pairs and allowing the large language model to leverage its completion capability to make new predictions. Despite the remarkable generalization ability of large language models, they are still limited by the inherent window size of in-context learning, which constrains their performance. Additionally, large models come with drawbacks in terms of cost and security. Therefore, in this paper, we explore a hybrid approach that combines large and lightweight models, aiming to achieve improved performance while being more deployable.

Combination of LLMs and Lightweight Models

Although LLMs have demonstrated strong capabilities across various tasks, they are computationally expensive, slow during inference, and difficult to deploy. Numerous research endeavors have been focused on exploring the interaction and complementary aspects between large-scale models and lightweight models, e.g., distilling the knowledge from large-scale models to lightweight models by matching the output distribution (Hinton, Vinyals, and Dean 2015). Despite the traditional knowledge distillation, researchers propose a new paradigm called teaching via data (TvD), in which they use an LLM-based "teacher" model to generate synthetic data for a specific task, then use the data to fine-tune a smaller "student" model (Ho, Schmid, and Yun 2023; Schick and Schütze 2021; Rosenbaum et al. 2022a,b; Meng et al. 2022; Ye et al. 2022a). The types of synthetic data generated by LLM are diverse. SYMGEN (Ye et al. 2023) first proposes using LLM to generate symbolic language with execution-based verification. Our work adopts the teaching via data paradigm but we are the first to utilize LLM as program translator, thereby addressing the challenge of annotating KBQA training data.

Methodology

Preliminaries

A knowledge base can be formally represented as $\mathcal{K} \in \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{E}$ represents the set of entities, $\mathcal{R}$ denotes the set of relations between entities, and $\mathcal{C}$ denotes the set of classes. The primary goal of KBQA is to answer user questions by leveraging the facts stored in the knowledge base. For accuracy and interpretability, most existing work adopt a semantic parsing framework, transforming natural language questions into programs, such as S-expression or SPARQL. These programs can be executed on the knowledge base to retrieve the final answer.

Most models utilize a framework comprising a ranking model, a generation model, or a combination of both. The ranking model is usually optimized as follows:

$$\mathcal{L}_{ranker} = -\frac{e^{s(q,p)}}{e^{s(q,p)} + \sum_{p_i \in P \wedge p_i \neq p} e^{s(q,p_i)}} \quad (1)$$

where p is the target program, and $s(q, p_i)$ is the similarity score between the question q and each program p_i from a candidate set P. The score is derived from a lightweight model like BERT (Devlin et al. 2019). On the other hand, a generation model employs advanced pre-trained generative models, such as T5, to generate the target program:

$$\mathcal{L}_{gen} = -\sum_{t=1}^{n} log(probability(p_t|p_{<t}; q; r)) \quad (2)$$

where n is the length of p, and r represents additional information like ranking results.

Automatic Program Sampling

The objective of program sampling is to generate valid (executable on KB) programs. We break this process into two crucial steps: template collection and step-wise grounding.

Template collection Taking a SPARQL query as an example, if we replace the entities and relations within it with variables (e.g., $ent0, rel0, ent1$), the resulting structural form is referred to as a "template". There are also previous works tackling KBQA tasks using template-driven methods directly (Unger et al. 2012; Zheng et al. 2018). By employing automated algorithms, we can generate a diverse set of templates to cover various question types that users may encounter in real-world scenarios. Compared to annotating programs for a large number of questions, the cost of collecting a handful of templates is negligible.

Step-wise grounding After obtaining a collection of diverse program templates, the next step is to perform entity and relation grounding. Leveraging the querying mechanism of SPARQL, we can directly treat the variables as query objects. However, directly executing the query with multiple variables can result in long execution times or errors, particularly when dealing with large-scale knowledge bases. To address this challenge, we introduce a "step-wise grounding" approach, where we iteratively determine the values of variables. Through empirical assessment, we note that this strategy efficiently narrows down the search space while maintaining diversity, ultimately enabling the derivation of a substantial number of programs.

Low-Resource Program Translation

As illustrated in Figure 2, conventional approaches often utilized language models to directly convert questions into programs via in-context learning. However, given that LLMs are primarily trained on natural language and lack specific training on programs, we argue that translating programs into natural language questions is a more efficient and intuitive approach, especially under the low-resource setting.

In this paper, we consider the LLM as a program translator that converts a program p_i^s into its corresponding natural language question q_i^s. The prompt is composed of (1) an instruction $Inst$ to guide the model in transforming programs into natural language questions. (2) a few seed pairs of golden programs and questions $\{(p_1^f, q_1^f), ..., (p_N^f, q_N^f)\}$ as demonstrations, where N denotes a N-shot setting, and $N = 0$ represents the zero-shot setting. Note that when constructing seed pairs, we should prioritize selecting diverse programs to cover a wide range of question types. Overall, the translation process can be formulated as:

$$q_i^s \leftarrow Translator(Inst; (p_1^f, q_1^f), ..., (p_N^f, q_N^f); p_i^s) \quad (3)$$

Execution-Guided Self Training

After generating a synthetic dataset using the aforementioned steps for training lightweight models, a potential issue still remains – the distribution discrepancy between synthetic and real-world questions. This discrepancy primarily arises from the limited overlap of entities and relations. Existing literature emphasizes that distribution shifts can have a substantial impact on the model's performance. For instance, a model trained on the GrailQA dataset might achieve only around 65% performance compared to being trained directly on the WebQSP (Gu et al. 2021). Therefore, in this paper, we propose using execution-guided self-training (**EGST**) to address this issue. Taking into account the ease of collecting user questions in real-world scenarios, we apply the self-training method to harness the information from these unlabeled user questions. Moreover, we introduce the utilization of execution results from structured queries as feedback information, substantially enhancing the purity of the training samples.

Specifically, the entire training process follows a teacher-student iterative training approach shown in Algorithm 1. Firstly, we train a teacher model using the synthetic data generated by the LLM. Subsequently, in each iteration, the teacher model generates pseudo programs for unlabeled real user questions, which undergo an execution-guided filtering mechanism to remove noisy data and obtain a cleaner training dataset. This filtered dataset, alongside the synthetic data, is employed to train the student model, which serves as the teacher model for the subsequent iteration. This iterative process continues until convergence is achieved.

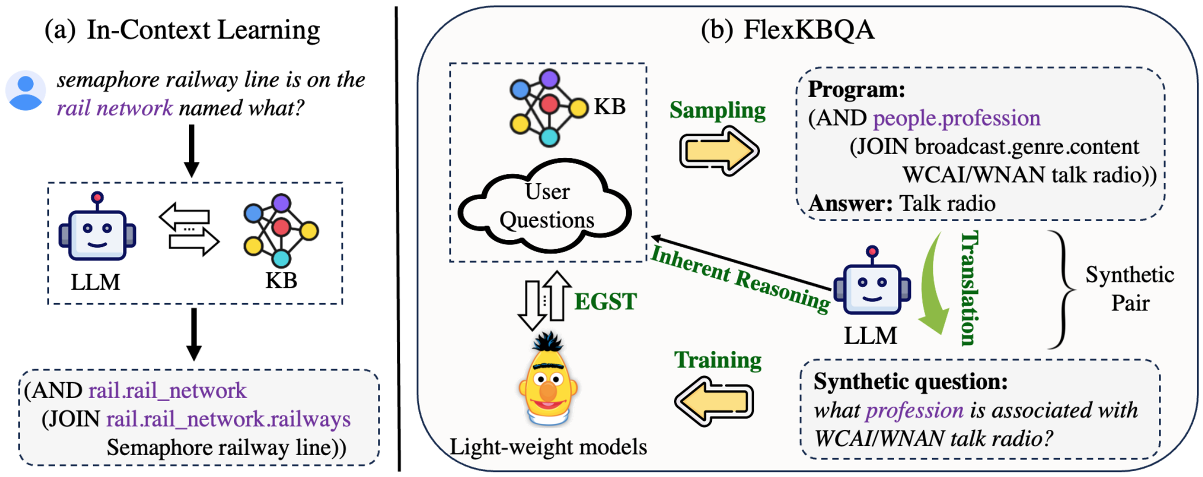
(a) In-Context Learning
semaphore railway line is on the rail network named what?
LLM
KB
(AND rail.rail_network (JOIN rail.rail_network.railways Semaphore railway line))
(b) FlexKBQA
KB
User Questions
Sampling
Program: (AND people.profession (JOIN broadcast.genre.content WCAI/WNAN talk radio))
Answer: Talk radio
Inherent Reasoning
Translation
Synthetic Pair
LLM
EGST
Training
Synthetic question: what profession is associated with WCAI/WNAN talk radio?
Light-weight models

Figure 2: A comparison between FlexKBQA and prior methods. (a): Prior approaches enable LLMs to directly ground the question to the knowledge base through in-context learning capabilities. (b): An illustration of FlexKBQA's innovative design: (1) Automatic Program Sampling module generates diverse and executable programs. (2) Low-Resource Program Translation module synthesizes high-quality data pairs. (3) Execution-Guided Self-Training module addresses distribution shift. (4) Inherent Reasoning module boosts the pipeline by leveraging inherent knowledge within LLMs.

For the execution-guided filtering mechanism, we employ the following filtering rules during each iteration:

- Error Filtering: we remove those data pairs where the predicted SPARQL queries either result in execution errors or fail to retrieve answers.
- Semantic Filtering: we employ an off-the-shelf pre-trained sentence-transformer (Reimers and Gurevych 2019) to calculate the semantic textual similarity between natural language questions and the relations in predicted programs, and filter out those data pairs with low similarity.
- Inherent Reasoning Filtering: We exclude samples whose pseudo answers do not align with the outcomes of Inherent Reasoning. Further details are provided in the next section.

Inherent Reasoning Augmentation

A widely accepted view is that LLMs encode knowledge in their parameters, allowing them to answer complex questions and directly respond to user queries. We refer this approach as Inherent Reasoning (**IR**). However, compared to transforming user questions into executable programs on KBs, Inherent Reasoning lacks interpretability and could exhibit limited effectiveness when dealing with domain-specific KBs. Thus, in this paper, we employ Inherent Reasoning as a data augmentation technique. It primarily operates in two stages. **(1)** In the execution-guided self-training stage, after annotating pseudo programs and obtaining answers for real user questions, we select samples where the answers align with those generated through Inherent Reasoning. By combining Inherent Reasoning with execution-guided filtering, we can obtain a training dataset with reduced noise. **(2)** Inherent Reasoning can serve as a complementary approach to semantic parsing. When the semantic parsing method fails to retrieve an answer (e.g. cannot enumerate ranking candidates or the predicted programs result in execution errors), we resort to using results from Inherent Reasoning as the final answers for the initially unanswerable questions. We believe that the integration of Inherent Reasoning and the semantic parsing approach holds significant value in the development of more accurate and interpretable KBQA systems.

In summary, with the aforementioned core components, FlexKBQA can achieve the following dimensions of flexibility: (1) Data Efficient: FlexKBQA requires only a small number of question-program pairs as prompts. (2) Domain-

Algorithm 1: Execution-Guided Self Training Procedure

```
Input: Unlabeled user questions $D^u = \{(q_i^u)\}$ ;
A few labeled question-program pairs $D^f = \{(q_i^f, p_i^f)\}$;
Synthetic pairs from LLM $D^s = \{(q_i^s, p_i^s)\}$;
Initial model parameters $\theta_{ini}$
Output: Model parameter $\theta_{final}$
1: Fine-tune teacher model $\theta_{tea}$ on synthetic data and
   small labeled data $D^s \cup D^f$ from $\theta_{ini}$
2: while not converged do
3:     //Generate pseudo-label for user questions
       $p_i^u \leftarrow f(q_i^u; \theta_{tea})$
4:     //Execution-guided filtering
       $\{(q_i^{uf}, p_i^{uf})\} \leftarrow Filter(KB, \{(q_i^u, p_i^u)\})$
5:     //Fine-tune student model $\theta_{stu}$ on all data
       $D = \{(q_i^s, p_i^s)\} \cup \{(q_i^f, p_i^f)\} \cup \{(q_i^{uf}, p_i^{uf})\}$
6:     //Update the teacher model
       $\theta_{tea} \leftarrow \theta_{stu}$
7: end while
8: $\theta_{final} \leftarrow \theta_{tea}$
```

Dataset	KB	Questions	Program Type
GrailQA	Freebase	64,331	S-expression
WebQSP	Freebase	4,737	SPARQL
KQA Pro	Wikidata	117,970	SPARQL

Table 1: Dataset Statistics

Agnostic: FlexKBQA is applicable to diverse knowledge bases, and alleviates common distribution shifting issues. (3) Deployable: Utilizing a lightweight model allows for cost-effective deployment compared to closed-source LLMs and enables seamless integration of domain-specific knowledge through fine-tuning.

Experimental Setup

Datasets

GrailQA (Gu et al. 2021) dataset is a large-scale dataset for knowledge base question answering that contains 64,331 question-logical form pairs. It has a broad coverage and many unique canonical logical forms. GrailQA introduces three levels of generalization for KBQA: *i.i.d.*, *compositional* and *zero-shot*. This dataset has attracted substantial research interest in recent years.

WebQSP (Yih et al. 2016) is another widely recognized KBQA dataset. The questions in this dataset are extracted from Google query logs, making them reflective of user preferences in real-world scenarios. It also offers SPARQL annotations that can be executed directly on Freebase. The main objective of this dataset is to evaluate the generalization capability in an *i.i.d.* setting, as the training and testing data share common entities and relations.

KQA Pro (Cao et al. 2022a) consists of around 120,000 diverse natural language questions that require various reasoning capabilities, such as multi-hop inference, attribute comparison, and set operations. KQA Pro is constructed based on a sub-knowledge base from Wikidata and does not assume an entity linking stage. Therefore, it requires models to memorize entities and relations, making it a more strong and challenging *i.i.d* task.

Underlying Model and Baselines

Note that FlexKBQA is model-agnostic. Considering performance and reproducibility, we choose a well performing RnG-KBQA (Ye et al. 2022b) as the underlying model for GrailQA and WebQSP. For KQA Pro, we select the BART-SPARQL (Cao et al. 2022a) model.

For baselines, we mainly evaluate FlexKBQA against Pangu (Gu, Deng, and Su 2023) and KB-BINDER (Li et al. 2023a). Both of them leverage the potent large language model Codex for in-context learning. Note that because the few-shot KBQA is a relatively novel task, our baseline choices are limited to these two methods. Since there are no available experimental results for Pangu and KB-BINDER on KQA Pro, we also re-implement an in-context learning model called LLM-ICL as an alternative for evaluation. We also provide the supervised results of several representative models for comparison.

Implementation Details

In experiments, we consider the original training dataset as the unlabeled real user questions. During the "step-wise grounding" stage, we not only employ random sampling but also gather the set of entities present in unlabeled user questions. These entities are then used as "topic entities" for constructing SPARQL queries. After removing duplicates, we obtain 6,184 synthetic pairs on Freebase and 5,017 on Wikidata. For a fair comparison, we use the same off-the-shelf entity linkers as Pangu (Gu, Deng, and Su 2023). And the LLM we utilize for program translation is gpt-3.5-turbo[1].

Results

Main Results

The results are presented in Table 2, 3 and 4. Compared to other baselines that also target the few-shot KBQA scenario, FlexKBQA demonstrates significant superiority. On the test set of GrailQA , it achieved an impressive Exact Match (EM) score of 62.8 and an F1 score of 69.4 with only 25 annotated samples, outperforming the previous state-of-the-art model Pangu by a significant margin of 6.7 points in terms of F1 score, despite Pangu utilizing more shots. Surprisingly, FlexKBQA also surpasses several supervised models, such as ReTraCk, which demand training with tens of thousands of samples. Since we use RnG-KBQA model as the underlying model of FlexKBQA, it's interesting to note that FlexKBQA attains a remarkable 93% performance relative to the fully-supervised RnG-KBQA model.

On WebQSP and KQA Pro datasets, FlexKBQA exhibits a similar trend of remarkable superiority over in-context learning methods. Specifically, when considering the F1 score, FlexKBQA achieves a significant 6.1-point improvement over Pangu on WebQSP under the 100-shot setting. It is worth noting that FlexKBQA's performance on the KQA Pro dataset shows a gap compared to the best-performing model. This difference can be attributed to the absence of an entity linking stage in KQA Pro. As a result, if the relations or entities in the test set were not present during training, FlexKBQA is more prone to producing semantic correct but unexecutable programs.

We also conduct experiments under a more challenging zero-shot setting, assuming no available annotated samples at all. This scenario has been rarely explored in previous research. We observe that, although the performance is behind the level achieved in the few-shot scenario, results on GrailQA show significant potential of our method. And we believe that this direction is worth further exploration.

EGST and IR

According to Tables 2, 3, and 4, FlexKBQA trained using synthetic data only demonstrates comparable performance when compared to previous in-context learning-based methods. The experimental results clearly illustrate the significant contribution of EGST to the model's performance.

[1]https://platform.openai.com/docs/models/gpt-3-5

		Overall		**I.I.D.**		**Compositional**		**Zero-shot**		**Dev Overall**	
	Model	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
Supervised	QGG (Lan and Jiang 2020)	-	36.7	-	40.5	-	33.0	-	36.6	-	-
	BERT+Ranking (Gu et al. 2021)	50.6	58.0	59.9	67.0	45.5	53.9	48.6	55.7	-	-
	ReTraCk (Chen et al. 2021b)	58.1	65.3	84.4	87.5	61.5	70.9	44.6	52.5	-	-
	RnG-KBQA (Ye et al. 2022b)	68.8	74.4	86.2	89.0	63.8	71.2	63.0	69.2	71.4	76.8
	ArcaneQA (Gu and Su 2022)	63.8	73.7	85.6	88.9	65.8	75.3	52.9	66.0	69.5	76.9
	DecAF (Yu et al. 2023)	68.4	78.7	84.8	89.9	73.4	81.8	58.6	72.3	-	81.4
Few-Shot (100 shots)	KB-BINDER (Li et al. 2023a)	50.6	56.0	-	-	-	-	-	-	-	-
	Pangu (Gu, Deng, and Su 2023)	53.3	62.7	54.7	62.9	54.5	63.7	52.3	62.2	-	-
Few-Shot (25 shots)	Fine-Tuning	16.6	21.3	19.0	24.8	17.3	21.8	15.2	19.4	16.4	21.6
	FlexKBQA	**62.8**	**69.4**	**71.3**	**75.8**	**59.1**	**65.4**	**60.6**	**68.3**	**65.5**	**71.1**
	***-w/o* IR**	**62.8**	68.0	**71.3**	75.3	**59.1**	64.1	**60.6**	66.4	**65.5**	70.6
	***-w/o* EGST**	52.4	57.7	56.9	61.8	49.4	54.4	51.7	57.3	57.0	62.1
Zero-Shot	**FlexKBQA**	61.9	68.9	72.1	77.0	58.4	65.2	58.9	66.9	64.6	70.3
	***-w/o* IR**	61.9	67.6	72.1	76.5	58.4	64.1	58.9	65.0	64.6	69.8
	***-w/o* EGST**	51.9	57.5	56.1	60.5	49.6	53.7	52.9	57.8	56.4	61.2

Table 2: Results on GrailQA

	Model	**F1**
Supervised	QGG (Lan and Jiang 2020)	74.0
	ReTraCk (Chen et al. 2021b)	71.0
	CBR (Das et al. 2021)	72.8
	Program Transfer (Cao et al. 2022b)	76.5
	RnG-KBQA (Ye et al. 2022b)	75.6
	DecAF (Yu et al. 2023)	78.8
Few-Shot (100 shots)	Fine-Tuning	25.6
	KB-BINDER (Li et al. 2023a)	53.2
	Pangu (Gu, Deng, and Su 2023)	54.5
	FlexKBQA	**60.6**
	***-w/o* IR**	58.2
	***-w/o* EGST**	51.1
Zero-Shot	**FlexKBQA**	46.2
	***-w/o* IR**	45.7
	***-w/o* EGST**	33.4

Table 3: Results on WebQSP

	Model	**Accuracy**
Supervised	EmbedKGQA (Saxena et al. 2020)	28.36
	RGCN (Schlichtkrull et al. 2018)	35.07
	RNN SPARQL (Cao et al. 2022a)	41.98
	BART+SPARQL (Cao et al. 2022a)	89.68
Few-Shot (100 shots)	Fine-Tuning	22.45
	LLM-ICL	31.75
	FlexKBQA	**46.83**
	***-w/o* IR**	33.32
	***-w/o* EGST**	23.10
Zero-Shot	**FlexKBQA**	33.28
	***-w/o* IR**	11.11
	***-w/o* EGST**	8.24

Table 4: Results on KQA Pro

It resulted in a substantial increase of 10.3 and 7.1 in F1 score on GrailQA and WebQSP, respectively, and a notable 10.2 increase in accuracy on KQA Pro. These findings provide strong evidence of the effectiveness of EGST in mitigating the impact of distribution shift. Inherent Reasoning can be regarded as an effective and flexible enhancement method that leverages the inherent knowledge of large models. On the GrailQA and WebQSP datasets, Inherent Reasoning leads to a F1 score improvement of 1.4 (EM score remains unchanged as it focuses solely on the consistency of structural queries) and 2.4, respectively. Remarkably, on the KQA Pro dataset, Inherent Reasoning achieves a accuracy increase of 13.5. This is because the absence of the entity linking stage results in numerous SPARQL execution errors. However, LLM was able to provide accurate answers directly in such cases.

In conclusion, FlexKBQA achieves superior performance compared to previous few-shot KBQA systems. Moreover, on certain datasets, it demonstrates competitive results when compared to supervised models. We claim that the superiority two main reasons. Firstly, it benefits from a large amount of synthetic data compared to in-context learning, leading to better convergence with respect to data distribution. Secondly, the innovative EGST and IR approaches equip FlexKBQA with the capability to harness unlabeled real user questions and leverage the inherent prowess of LLMs. These unique features distinguish FlexKBQA from other methods.

Beyond Few-Shot KBQA

In this section, we delve into its efficacy beyond the few-shot scenario. As demonstrated in Figure 3, with an increase in the number of samples, the model pretrained on our synthetic data consistently performs better. Even when there are 1000 real samples available, it still maintains an 8-point advantage over models trained solely on real samples. This ob-

Question I	*What type of art leonardo da vinci do?*
Pangu	(JOIN (R visual_art.visual_artist.associated_periods_or_movements) m.04lg6) (✗)
FlexKBQA	(JOIN (R visual_art.visual_artist.art_forms) m.04lg6) (✓)
Synthetic Data Pair	*What type of art did andy warhol create?* (JOIN (R visual_art.visual_artist.art_forms) m.0kc6)
Question II	*What airport do you fly into to get to destin fl?*
Pangu	(JOIN (R travel.travel_destination.tourist_attractions) m.0rp8x) (✗)
FlexKBQA *w/o* EGST	(JOIN (R travel.travel_destination.how_to_get_here) m.0rp8x) (✗)
FlexKBQA	(JOIN (R location.location.nearby_airports) m.0rp8x) (✓)
Pseudo Labeled Pair	*What airport is closer to downtown houston?* (JOIN (R location.location.nearby_airports) m.03l2n) (✓)

Table 5: Two typical questions from the test set of WebQSP that FlexKBQA succeeds while Pangu fails.

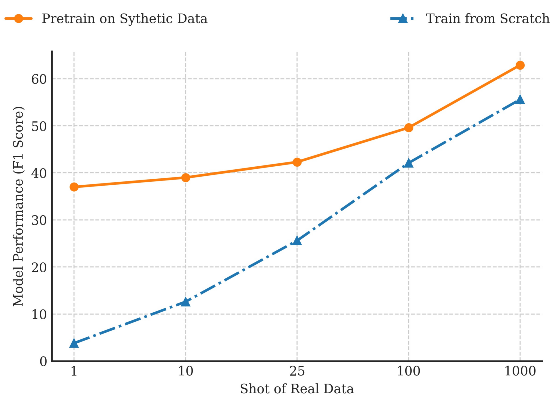

Figure 3: Results beyond few-shot setting. FlexKBQA consistently performs better with more annotated data.

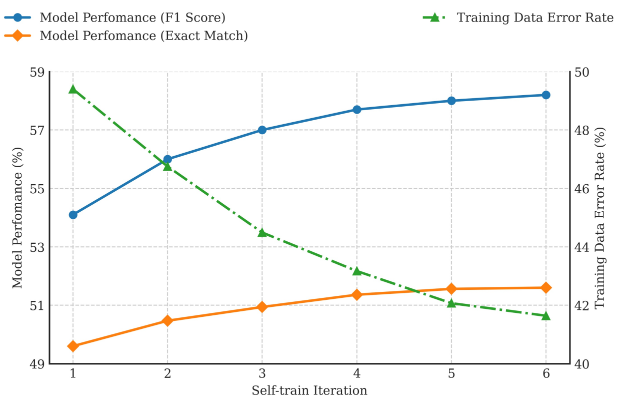

Figure 4: Variation of model performances and the error rate of pseudo-labeled programs with EGST Iterations.

servation underscores that our approach is not only applicable to few-shot scenarios but can also serve as a valuable data augmentation technique.

Ablation and Case Studies

In Figure 4, we delve into an ablation study to gauge the impact of EGST. We can observe that with each iteration, the model's performance steadily enhances, eventually converging around the 6th epoch. The underlying reason, illustrated by the green line, is the diminishing error rate of pseudo programs.

To better showcase the advantages of FlexKBQA, we conduct a comparison between FlexKBQA and Pangu, a state-of-the-art model. In Table 5, Question I illustrates a non-i.i.d scenario, where the relation involved in the question is not present in in-context examples. Consequently, Pangu struggles to answer correctly. However, due to FlexKBQA's inclusion of the relevant relation in its synthetic data, it manages to generate the correct answer. In an ideal situation where the synthetic data coverage is extensive enough, non-i.i.d scenarios like this could be eradicated. Question II demonstrates the impact of EGST. By utilizing information from a correctly pseudo-labeled user question, the model can successfully answer similar but ambiguous or more complex questions in the test set.

Conclusion and Future Work

In conclusion, this paper presented FlexKBQA, a framework that harnesses the power of LLMs as program translators and their inherent reasoning ability for KBQA tasks. Extensive experiments on diverse datasets showcase the effectiveness of FlexKBQA under the few-shot setting, surpassing all baseline methods and even approaching the performance of supervised models. Furthermore, we are pioneers in exploring zero-shot KBQA setting. FlexKBQA represents a significant advancement in the integration of large and lightweight models, offering three-fold flexibilities: data annotation, deployment, and being domain agnostic. Future research can explore the broader applications of this framework in more natural language understanding and reasoning tasks.

Acknowledgments

This work was supported in part by National Key Research and Development Program of China under Grant No.

2020YFA0804503, National Natural Science Foundation of China under Grant No. 62272264, and Beijing Academy of Artificial Intelligence (BAAI), and also supported by Independent Research Project of Medical Engineering Laboratory of Chinese PLA General Hospital (2022SYSZZKY23).

References

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1533–1544.

Cao, S.; Shi, J.; Pan, L.; Nie, L.; Xiang, Y.; Hou, L.; Li, J.; He, B.; and Zhang, H. 2022a. KQA Pro: A Dataset with Explicit Compositional Programs for Complex Question Answering over Knowledge Base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6101–6119.

Cao, S.; Shi, J.; Yao, Z.; Lv, X.; Yu, J.; Hou, L.; Li, J.; Liu, Z.; and Xiao, J. 2022b. 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)*, 8128–8140.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; de Oliveira Pinto, H. P.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; Ray, A.; Puri, R.; Krueger, G.; Petrov, M.; Khlaaf, H.; Sastry, G.; Mishkin, P.; Chan, B.; Gray, S.; Ryder, N.; Pavlov, M.; Power, A.; Kaiser, L.; Bavarian, M.; Winter, C.; Tillet, P.; Such, F. P.; Cummings, D.; Plappert, M.; Chantzis, F.; Barnes, E.; Herbert-Voss, A.; Guss, W. H.; Nichol, A.; Paino, A.; Tezak, N.; Tang, J.; Babuschkin, I.; Balaji, S.; Jain, S.; Saunders, W.; Hesse, C.; Carr, A. N.; Leike, J.; Achiam, J.; Misra, V.; Morikawa, E.; Radford, A.; Knight, M.; Brundage, M.; Murati, M.; Mayer, K.; Welinder, P.; McGrew, B.; Amodei, D.; McCandlish, S.; Sutskever, I.; and Zaremba, W. 2021a. Evaluating Large Language Models Trained on Code. arXiv:2107.03374.

Chen, S.; Liu, Q.; Yu, Z.; Lin, C.-Y.; Lou, J.-G.; and Jiang, F. 2021b. ReTraCk: A flexible and efficient framework for knowledge base question answering. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing: system demonstrations*, 325–336.

Cheng, Z.; Xie, T.; Shi, P.; Li, C.; Nadkarni, R.; Hu, Y.; Xiong, C.; Radev, D.; Ostendorf, M.; Zettlemoyer, L.; Smith, N. A.; and Yu, T. 2023. Binding Language Models in Symbolic Languages. arXiv:2210.02875.

Das, R.; Zaheer, M.; Thai, D.; Godbole, A.; Perez, E.; Lee, J. Y.; Tan, L.; Polymenakos, L.; and McCallum, A. 2021. Case-based Reasoning for Natural Language Queries over Knowledge Bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 9594–9611.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Gao, T.; Han, X.; Zhu, H.; Liu, Z.; Li, P.; Sun, M.; and Zhou, J. 2019. FewRel 2.0: Towards More Challenging Few-Shot Relation Classification. 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)*, 6250–6255.

Gu, Y.; Deng, X.; and Su, Y. 2023. Don't Generate, Discriminate: A Proposal for Grounding Language Models to Real-World Environments. arXiv:2212.09736.

Gu, Y.; Kase, S.; Vanni, M.; Sadler, B.; Liang, P.; Yan, X.; and Su, Y. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, 3477–3488.

Gu, Y.; Pahuja, V.; Cheng, G.; and Su, Y. 2022. Knowledge Base Question Answering: A Semantic Parsing Perspective. arXiv:2209.04994.

Gu, Y.; and Su, Y. 2022. ArcaneQA: Dynamic Program Induction and Contextualized Encoding for Knowledge Base Question Answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, 1718–1731.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531.

Ho, N.; Schmid, L.; and Yun, S.-Y. 2023. Large Language Models Are Reasoning Teachers. arXiv:2212.10071.

Lan, Y.; and Jiang, J. 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*, 969–974.

Li, T.; Ma, X.; Zhuang, A.; Gu, Y.; Su, Y.; and Chen, W. 2023a. Few-shot In-context Learning for Knowledge Base Question Answering. arXiv:2305.01750.

Li, X.; Li, Z.; Zhang, Z.; Liu, N.; Yuan, H.; Zhang, W.; Liu, Z.; and Wang, J. 2022. Effective few-shot named entity linking by meta-learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 178–191. IEEE.

Li, Z.; Li, X.; Duan, Z.; Dong, B.; Liu, N.; and Wang, J. 2023b. Toward a Unified Framework for Unsupervised Complex Tabular Reasoning. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 1691–1704. IEEE.

Meng, Y.; Huang, J.; Zhang, Y.; and Han, J. 2022. Generating Training Data with Language Models: Towards Zero-Shot Language Understanding. arXiv:2202.04538.

Ni, A.; Iyer, S.; Radev, D.; Stoyanov, V.; Yih, W.-t.; Wang, S.; and Lin, X. V. 2023. Lever: Learning to verify language-to-code generation with execution. In *International Conference on Machine Learning*, 26106–26128. PMLR.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv:1908.10084.

Rosenbaum, A.; Soltan, S.; Hamza, W.; Saffari, A.; Damonte, M.; and Groves, I. 2022a. CLASP: Few-Shot Cross-Lingual Data Augmentation for Semantic Parsing. arXiv:2210.07074.

Rosenbaum, A.; Soltan, S.; Hamza, W.; Versley, Y.; and Boese, M. 2022b. LINGUIST: Language Model Instruction Tuning to Generate Annotated Utterances for Intent Classification and Slot Tagging. arXiv:2209.09900.

Schick, T.; and Schütze, H. 2021. Generating Datasets with Pretrained Language Models. arXiv:2104.07540.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, 593–607. Springer.

Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; and Garg, A. 2023. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 11523–11530. IEEE.

Su, Y.; Sun, H.; Sadler, B.; Srivatsa, M.; Gür, I.; Yan, Z.; and Yan, X. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 562–572.

Unger, C.; Bühmann, L.; Lehmann, J.; Ngonga Ngomo, A.-C.; Gerber, D.; and Cimiano, P. 2012. Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*, 639–648.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837.

Ye, J.; Gao, J.; Li, Q.; Xu, H.; Feng, J.; Wu, Z.; Yu, T.; and Kong, L. 2022a. ZeroGen: Efficient Zero-shot Learning via Dataset Generation. arXiv:2202.07922.

Ye, J.; Li, C.; Kong, L.; and Yu, T. 2023. Generating Data for Symbolic Language with Large Language Models. arXiv:2305.13917.

Ye, X.; Yavuz, S.; Hashimoto, K.; Zhou, Y.; and Xiong, C. 2022b. 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)*, 6032–6043.

Yih, W.-t.; Richardson, M.; Meek, C.; Chang, M.-W.; and Suh, J. 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)*, 201–206.

Yu, D.; Zhang, S.; Ng, P.; Zhu, H.; Li, A. H.; Wang, J.; Hu, Y.; Wang, W.; Wang, Z.; and Xiang, B. 2023. DecAF: Joint Decoding of Answers and Logical Forms for Question Answering over Knowledge Bases. arXiv:2210.00063.

Zheng, W.; Yu, J. X.; Zou, L.; and Cheng, H. 2018. Question answering over knowledge graphs: question understanding via template decomposition. *Proceedings of the VLDB Endowment*, 11(11): 1373–1386.