Few-shot In-context Learning for Knowledge Base Question Answering

♠Tianle Li, ♠Xueguang Ma, ♠Alex Zhuang, ♥Yu Gu, ♥Yu Su, ♠,♣Wenhu Chen
♠University of Waterloo
♥The Ohio State University
♣Vector Institute, Toronto
{t29li,x93ma,a5zhuang,wenhuchen}@uwaterloo.ca, {gu.826,su.809}@osu.edu

Abstract

Question answering over knowledge bases is considered a difficult problem due to the challenge of generalizing to a wide variety of possible natural language questions. Additionally, the heterogeneity of knowledge base schema items between different knowledge bases often necessitates specialized training for different knowledge base question-answering (KBQA) datasets. To handle questions over diverse KBQA datasets with a unified training-free framework, we propose KB-BINDER, which for the first time enables few-shot in-context learning over KBQA tasks. Firstly, KB-BINDER leverages large language models like Codex to generate logical forms as the draft for a specific question by imitating a few demonstrations. Secondly, KB-BINDER grounds on the knowledge base to bind the generated draft to an executable one with BM25 score matching. The experimental results on four public heterogeneous KBQA datasets show that KB-BINDER can achieve a strong performance with only a few in-context demonstrations. Especially on GraphQA and 3-hop MetaQA, KB-BINDER can even outperform the state-of-the-art trained models. On GrailQA and WebQSP, our model is also on par with other fully-trained models. We believe KB-BINDER can serve as an important baseline for future research. Our code is available at https://github.com/ltl3A87/KB-BINDER

1 Introduction

Question answering over knowledge bases (KBQA) (Berant et al., 2013; Yih et al., 2015) has been a long-standing research problem in the AI community. It has attracted wide attention from the community with its significant role in making large-scale knowledge bases accessible to non-expert users (Wu et al., 2019; Lan et al., 2021; Gu et al., 2022). However, despite the fact that the increasing scale of knowledge bases can enable the retrieval with higher coverage on miscellaneous topics, it poses a great challenge for suppliers with limited resources, who rely on the models trained on certain knowledge bases or benchmarks. Concretely, the difficulties primarily lie in the following aspects: 1) Data intensiveness: larger knowledge bases require ever larger quantities of annotated data to allow fine-tuned models to generalize well over them. (Yih et al., 2016; Talmor and Berant, 2018; Gu et al., 2020). 2) Dataset specificity: For relatively small-scale KBQA datasets, the fully-trained models tend to overfit to a specific schema, and can hardly generalize to knowledge base questions in unseen domains (Su et al., 2016; Zhang et al., 2017; Sun et al., 2019). These challenges make it crucial to devise a new framework that can work in both low-resource and training-free settings in KBQA.

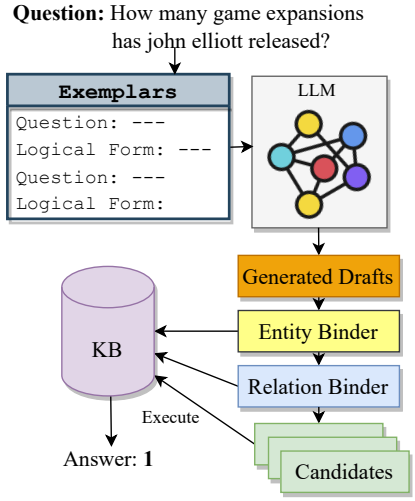

Figure 1: Overview of KB-BINDER pipeline. There are two primary stages in our method: 1) Generate the drafts as preliminary logical forms; 2) Bind the drafts to the executable ones with entity and relation binders grounded on the knowledge base. The final answer can be obtained after the execution of the final candidates.

Recently, large language models (LLMs) like GPT-3 and Codex (Brown et al., 2020a; Chen et al.,

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 6966–6980

2021a) have demonstrated their strong generalizability (Wang et al., 2022a; Wei et al., 2022b; Zhou et al., 2022b; Cheng et al., 2022; Zhou et al., 2022a; Suzgun et al., 2022) on a wide range of text, table, commonsense and even math QA tasks with few-shot in-context learning. Other works also validate that Codex (Chen et al., 2021a) can parse and transform unstructured instructions to structured and executable code with only a few dozen demonstrations (Gao et al., 2022; Chen et al., 2022). These works inspire us to tackle KBQA with LLMs, an under-explored area in the literature that is particularly challenging compared to other QA tasks because of the massive scale of modern KBs.

However, it is still unclear how to address KBQA with in-context learning. Unlike many other question-answering tasks, where the evidence is provided with a reasonable length limit, KBQA needs to condition on a massive graph containing millions of nodes and billions of edges. Evidently, it is impossible to feed the whole graph as-is to the language model. Even feeding a subgraph is extremely challenging as it requires splitting the monolithic graph into self-consistent and query-relevant chunks, which is itself an unaddressed research problem. Without feeding the knowledge graph as an additional input, language models become unaware of the schema of the KB. This problem makes it difficult to associate surface forms in the questions with the corresponding entities and relation types in a specific KB, not to mention generate executable logical forms with these linked entities and relations. These challenges make it hard to build in-context KBQA systems.

In this work, we propose KB-BINDER, which, for the first time, enables training-free few-shot in-context learning on KBQA. Our framework consists of two stages as shown in Figure 1. In the first stage, we demonstrate a few KBQA questions and their corresponding logical forms as the exemplary pairs for Codex to generate a **draft** of an unseen question. The **draft** is a 'preliminary' logical form likely to contain mistakes in both entities and relations. For example, due to a lack of information about the KB schema, Codex might generate a **draft** containing 'medicine.manufactured_drug.shape' while the true relation in the KB should be 'medicine.manufactured_drug_form.shape'. In the second stage, KB-BINDER binds the 'preliminary' entities to the true entity by using a lexicon-based similarity search over the whole KB. Once the entities are bound, we search through the vicinity of the bound entities to bind the 'preliminary' relations. We fill the bound entities and relations into the **draft** to generate a set of 'refined' logical forms. We execute these logical forms against the KB and return the executed results as the answer. To enhance KB-BINDER with more pertinent exemplars, we also propose a KB-BINDER-R with retrieved exemplars from the training set.

In general, previous works rely heavily on predefined heuristics for a target knowledge base to find the potential candidates (Ye et al., 2021; Gu and Su, 2022; Shu et al., 2022). KB-BINDER, however, does not need heuristics customized to specific KB schema due to the inherent generalizability of LLMs. We test the performance of our models under few-shot setting on four public datasets, WebQSP (Yih et al., 2016), GrailQA (Gu et al., 2020), GraphQA (Su et al., 2016) and MetaQA (Zhang et al., 2017). On GraphQA and 3-hop MetaQA, KB-BINDER achieves 39.5 F1 and 99.5% Hits@1 scores respectively, surpassing the previous SoTA by 7.7 on F1 score and 0.6% on Hits@1 correspondingly. On WebQSP, KB-BINDER-R can achieve 74.4% F1 score, only 4.4% lower than the SoTA model (Yu et al., 2022). These experimental results demonstrate the effectiveness of our approach.

Given the simplicity and generality of KB-BINDER, we believe it could serve as an important baseline for future KB research, especially in the low-resource setting.

2 Related Work

Knoweldge Base Question Answering. Most state-of-the-art KBQA models are based on semantic parsing (Lan et al., 2021; Gu et al., 2022), where a question is mapped onto a logical form over the KB. Locating the target logical form over the KB entails a massive search space (*e.g.*, FREEBASE (Bollacker et al., 2008) contains 45 million entities and 3 billion facts). Recent methods capitalize on the strong generalizability of LMs to generalize to the massive space unexplored during training (Chen et al., 2021b; Gu and Su, 2022; Ye et al., 2021; Shu et al., 2022). These methods are more data-efficient and can better handle the massive search space compared with earlier methods operating with an i.i.d. assumption (Yih et al., 2015; Dong and Lapata, 2016), however, they still require thousands of

labeled examples to fine-tune LMs. Despite being an appealing idea, few-shot KBQA has not been touched by existing work. It has been deemed highly non-trivial, if not impossible, to learn to handle the large search space in KBQA only with a handful of training data. One of the relevant works is Hua et al. (2020), which trains a meta-model to quickly adapt to a new question with a few training examples. However, they need 2,000 labeled questions to train the meta-model first, thus not a true few-shot setting. Finally, a concurrent work Gu et al. (2023) approaches to few-shot KBQA in a different way. They leverage the discriminative ability of LLMs instead of their generative ability. In this paper, we present the first effort to enable true few-shot learning for KBQA with LLMs in a generate-then-bind way, which may point to interesting opportunities for practical KBQA under low-data settings.

In-Context Learning with LLMs. In-context learning with large language models (Brown et al., 2020a) has shown strong few-shot performance in many NLP tasks, such as question answering (Cheng et al., 2022), information extraction (Dunn et al., 2022), and numerical reasoning (Lewkowycz et al., 2022). Analyses into the mechanisms behind this behavior are undertaken by Olsson et al. (2022); Xie et al. (2021). Empirically, Min et al. (2022) shows the effectiveness of constructing prompts using an input-label pairing format, and Liu et al. (2021) experiment with the number of examples provided, as well the idea of retrieving relevant examples to a test input to construct the prompt with. These results inform the prompt-construction methods used in our work. Lampinen et al. (2022) suggests that incorporating explanatory task instructions in context can improve performance, however, we leave a deeper exploration of this to future works.

Reasoning with LLMs. A number of methods have recently emerged to extend the reasoning capabilities of LLMs (Brown et al., 2020a; Kojima et al., 2022). Chain of Thought Prompting (CoT) (Wei et al., 2022a) showed that encouraging intermediate steps in model output can improve reasoning accuracy. Developing this idea, methods that involve a direct synthesis of formal programs that solve these tasks have shown further improvement (Chen et al., 2022; Nye et al., 2021; Gao et al., 2022; Cheng et al., 2022). The most relevant work to the QA setting is Binder (Cheng et al., 2022), where the LLM is prompted to conduct text-to-SQL generation and further answer questions using information retrieved from an SQL database. However, while SQL table headers demonstrated in examples can help an LLM generate reasonable SQL commands, the thousands of relations and millions of entities in a KB represent a much larger search space that cannot be captured as easily by the prompting an LLM. KB-BINDER solves this challenge using a draft generation and schema binding pipeline.

3 Methodology

Given a new question, KB-BINDER leverages an LLM to generate a preliminary logical form as a draft. A draft is not guaranteed to be executable, as it is generated by the LLM without being explicitly restricted to the candidates' vocabulary and knowledge graph structure. However, with the demonstration of in-context prompting, drafts can reveal the structural relationships among mentioned entities in a semantically reasonable way. As a result, the generated drafts can simplify the search space needed to retrieve real entities and schema terms. These entities and terms are then used to revise the draft to a real executable logical form for a given question. This process is illustrated in Figure 2.

3.1 Drafts Generator

We leverage the in-context learning capability of Codex to generate logical form drafts for unseen questions. Specifically, we randomly sample N examples from the training set as the exemplars, which are shown to the LLM in the form of <Question, Logical form> pairs. However, it is worth noting that the **MIDs** (i.e., machine identifier) in the original logical form are not easy to interpret and imitate. For instance, the raw logical form of the question "data compression is the genre of which file format?" is:

```
(AND computer.file_format (JOIN
computer.file_format.genre m.0279m))
```

where *m.0279m* is the MID of the entity "data compression" from FreeBase. The raw format of MIDs with no semantic meaning can hardly assist the large language model to understand and imply the latent relationships among schema items. Therefore, naturally, we substitute the MIDs in the original logical forms with their surface names in the prompting demonstrations. Consequently, the final

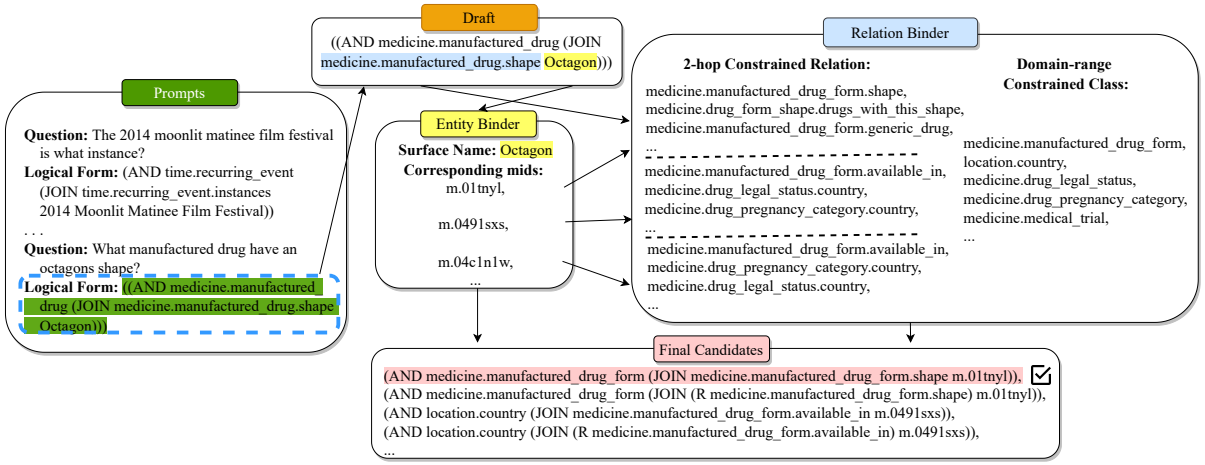

Figure 2: KB-BINDER framework: Given a question, the LLM will first generate its corresponding preliminary logical forms as the drafts, imitating the exemplary demonstration. Then the entity and relation binders will operate on the drafts to ground the entities and relations on KB respectively, which produces the final candidates.

processed logical form fed to Codex for the above example will become:

```
(AND computer.file_format (JOIN
computer.file_format.genre Data Compression))
```

the surface names of the entities mentioned in a new target question will appear in the generated preliminary logical forms as shown in the demonstration. Through in-context learning, LLM is tasked with generating such friendly logical forms for a new question by following the demonstrations.

3.2 Knowledge Base Binder

The preliminary logical forms generated by the large language model provide us with a macroscopic view of the question from the perspective of semantics and structure relationships. Starting from the generated drafts, we separately perform the entity and relation binding over the KB.

Entity Binder To identify the exact MIDs of the entities mentioned in the questions, we directly extract their surface names from the generated drafts. If the extracted surface names consistently match the friendly names of some MIDs from the knowledge base, we retrieve all the MIDs corresponding to the matched friendly names and select the most popular n of them based on FACC1. If the surface names match no friendly name of any entity from the knowledge base, we then utilize BM25 to retrieve the most similar existing one in KB and exploit it as the anchor to extract the MID candidates. If we detect multiple surface names from the drafts, we bind their potential MIDs independently first. And all the permutations of their combinations will be considered in the final execution.

Relation Binder In spite of the fact that the generated preliminary relations in the drafts are very likely to not exist in the knowledge base, their format and semantic meaning are still supposed to be analogical to the real-existed ones, learning from the demonstration of the prompts. With this assumption, we utilize each of the related items together with the original question as the search query to retrieve the most similar ones with BM25 from the whole knowledge base relation collection. To enlarge the possibility of successful execution of the logical form, we only keep the top m among all the two-hop relation items starting from the MIDs of the current permutation and filter out the ones out of this constraint. For each combination of MIDs, we iterate all the m retrieved relations candidates accordingly.

Majority Vote Following the above workflow, a generated draft can be bound to hundreds of potential logical form candidates. And each of them can be converted to a SPARQL query to be ultimately executed on the KB. We record all the answerable logical form candidates and their corresponding answers. As self-consistency can improve the robustness of the predictions of large language model (Wang et al., 2022b), we repeat the paradigm for K times and adapt the majority vote strategy to decide the final consistent answer and its logical form. We name the model with self-consistency on the top K drafts as KB-BINDER(K).

Retrieved Exemplars To further boost the performance of our method in a training-free setting, we design another variant of KB-BINDER,

Dataset	Train	Dev	Test
GrialQA	44,337	6,763	13,231
WebQSP	3,098	–	1,639
GraphQA	2,381	–	2,395
MetaQA-1hop	96,106	9,992	9,947
MetaQA-2hop	118,980	14,872	14,872
MetaQA-3hop	114,196	14,274	14,274

Table 1: Dataset statistics.

named KB-BINDER(K)-R. Instead of selecting the exemplars from the training sets randomly, KB-BINDER(K)-R leverages BM25 to retrieve the most similar N questions with the target one as the demonstrations. So that the logical forms of the N questions are more likely to cover the schema items that are related or even exactly the same as the target one. This setting is supposed to be especially advantageous over questions of I.I.D. type.

4 Experiment

In this section, we briefly introduce the benchmarks used to evaluate the performance of our framework. And we demonstrate the detailed setting of KB-BINDER and its result on each of the datasets compared with the fully-trained baselines. Ultimately, we make an analysis of the variation of design choices and their corresponding potential causes.

4.1 Datasets

We evaluate KB-BINDER on four public KBQA datasets as follows:

GrailQA (Gu et al., 2020) is a diverse KBQA dataset built on Freebase, covering 32,585 entities, 3,720 relations across 86 domains. It is designed to test three levels of generalization of KBQA models: I.I.D., compositional, and zero-shot.

GraphQA (Su et al., 2016) is also a diverse dataset that covers a wide range of domains. It builds by sentence-level paraphrasing from graph queries and evaluating compositional generalization.

WebQSP (Yih et al., 2016) contains questions from WebQuestions that are answerable by Freebase. It tests i.i.d. generalization on simple questions.

MetaQA (Zhang et al., 2017) consists of a movie ontology derived from the WikiMovies Dataset and three sets of question-answer pairs written in different levels of difficulty. It evaluates the effectiveness in a specific domain.

Table 1 shows the detail of train/dev/test splits of the datasets. We evaluate our pipeline on all the test sets and conduct ablation studies on a subset of the dev set from GrailQA with 500 randomly sampled examples.

4.2 Baselines

We compare our method with all the systems that have a publication on the official leaderboard of each dataset and record their results from the paper directly with the same evaluation matrix. Notice that all the competitive baseline methods utilized the entire set of training data as supervision.

4.3 Implementation Details

In the draft generation step, we leverage `code-davinci-002` from OpenAI API[1] to obtain the top K drafts for each question, we test the cases with $K = 1$ and $K = 6$, and refer to them as KB-BINDER (1) and KB-BINDER (6). Specifically, we randomly sample $N = 100$ exemplary questions from the training sets of WebQSP and GraphQA respectively. For GrailQA, we sample $N = 40$ exemplars for testing due to the long inference time on more than ten thousands of testing data. For MetaQA, we only sample 5 questions for demonstration, as the KB is relatively small in this benchmark. We run each of the experiment for three times and averaged the performances as reported.

In the binding step, we set $n = 15$ for all the questions in the entity binder. We deploy BM25 and Contriever (Izacard et al., 2021) provided by Pyserini[2] as a hybrid searcher to retrieve the originally unmatched friendly names and the top relation items. After obtaining the globally ranked relations, we focus on the relations bound by 2-hop relations from the detected entities. We traverse the top 10 (i.e., $m = 10$) relation candidates within the 2-hop constraint for GrailQA, WebQSP and GraphQA, and the top 1 (i.e., $m = 1$) for MetaQA. After the drafts are bound to the potential candidates, they will be translated to SPARQL and executed on the Virtuoso server following the instructions[3].

4.4 Main Result

We demonstrate the model performance on the test sets of four public datasets in Table 2, 3, 4 and 5 for GrailQA, WebQSP, GraphQA and MetaQA respectively. KB-BINDER (1) refers to our method in

[1] https://openai.com/blog/openai-codex/

[2] https://github.com/castorini/pyserini

[3] https://github.com/dki-lab/Freebase-Setup

Method	Overall	
	EM	F1
GloVe + Transduction (Gu et al., 2020)	17.6	18.4
QGG (Lan and Jiang, 2020)	-	36.7
BERT + Transduction (Gu et al., 2020)	33.3	36.8
GloVe + Ranking (Gu et al., 2020)	39.5	45.1
BERT + Ranking (Gu et al., 2020)	50.6	58.0
ReTraCk (Chen et al., 2021b)	58.1	65.3
S^2QL (Zan et al., 2022)	57.5	66.2
ArcaneQA (Gu and Su, 2022)	63.8	73.7
RnG-KBQA (Ye et al., 2021)	68.8	74.4
DecAF (Yu et al., 2022)	68.4	78.7
TIARA (Shu et al., 2022)	**73.0**	**78.5**
Few-shot in-context		
KB-BINDER (1)	47.0	51.6
KB-BINDER (6)	50.6	56.0
KB-BINDER (6)-R	53.2	58.5

Table 2: 40-shot Results of KB-BINDER/KB-BINDER-R and baselines on GrailQA.

Method	F1
ReTraCk (Chen et al., 2021b)	71.0
QGG (Lan and Jiang, 2020)	74.0
ArcaneQA (Gu and Su, 2022)	75.6
PullNet (Sun et al., 2019)	62.8
RnG-KBQA (Ye et al., 2021)	75.6
TIARA (Shu et al., 2022)	76.7
DecAF (Yu et al., 2022)	**78.8**
Few-shot in-context	
KB-BINDER (1)	52.5
KB-BINDER (6)	53.2
KB-BINDER (6)-R	74.4

Table 3: 100-shot Results of KB-BINDER/KB-BINDER-R and baselines on WebQSP.

default-setting with top 1 draft, and KB-BINDER (6) involves mass voting to achieve self-consistency with top 6 drafts, while KB-BINDER (6)-R refers to KB-BINDER (6) using retrieved exemplars 3.2. In general, all the variations of KB-BINDER have strong performance on all the selected datasets. According to the results from the tables, KB-BINDER (6) can generally outperform KB-BINDER (1) in line with our expectations, while KB-BINDER (6)-R can further boost the performance in most of the cases. And we observe that our few-shot method can achieve on par and even better performances compared to the fully supervised SOTAs on WebQSP, GraphQA and MetaQA, and it shows competitive performance with the BERT-ranking baseline on GrailQA.

KB-BINDER Results Specifically, we show KB-BINDER (K) few-shot result on GrailQA and compare it with a series of fully-trained baselines in

Method	F1
AUDEPLAMBDA (Reddy et al., 2017)	17.7
SPARQA (Sun et al., 2020)	21.5
BERT + Ranking (Gu et al., 2020)	25.0
ArcaneQA (Gu and Su, 2022)	31.8
Few-shot in-context	
KB-BINDER (1)	39.3
KB-BINDER (6)	**39.5**
KB-BINDER (6)-R	38.7

Table 4: 100-shot Results of KB-BINDER/KB-BINDER-R and baselines on GraphQA.

Method	1-hop	2-hop	3-hop
KV-Mem (Miller et al., 2016)	96.2	82.7	48.9
VRN (Zhang et al., 2017)	**97.5**	89.9	62.5
GraftNet (Sun et al., 2018)	97.0	94.8	77.7
PullNet (Sun et al., 2019)	97.0	**99.9**	91.4
Emb (Saxena et al., 2020)	**97.5**	98.8	94.8
NSM (He et al., 2021)	97.1	**99.9**	98.9
Few-shot in-context			
KB-BINDER (1)	93.5	99.6	96.4
KB-BINDER (1)-R	92.9	**99.9**	**99.5**

Table 5: 5-shot Results of KB-BINDER/KB-BINDER-R and baselines on MetaQA.

Table 2. With merely 40 examples, KB-BINDER (6) achieves 50.6 EM score, which is the same as BERT + Ranking setting, finetuned on the whole training sets with around 45k annotations. Although the overall scores of the two systems are on par, we notice from Table 6 that our pipeline has better generalization performance on compositional and zero-shot questions, where the specific logical form is unseen in the training data. The EM scores of KB-BINDER (6) for compositional and zero-shot questions are 5.1 and 1.3 points higher than BERT+Ranking Table 6. We notice there is a gap between our method and the state-of-the-art supervised methods on GrailQA, however, it is still exciting to see few-shot methods is at the level of supervised methods.

As shown in Table 4, KB-BINDER (1) and KB-BINDER (6) achieve 39.3 and 39.5 F1 score on GraphQA dataset, surpassing the previous sate-of-the-art models 7.7 in F1 score. In Table 5, KB-BINDER (1) achieves 99.6 % and 96.4 % Hits@1 scores on 2-hop and 3-hop MataQA dataset correspondingly, which are on par with the state-of-the-art models. These competitive performances show the advantage of KB-BINDER on some special scenarios. For the case of GraphQA, it has a relatively small scale of training examples (*i.e.*, 2,381 in to-

tal), however, all the questions in the test set are of compositional type. Therefore, it is hard for the fine-tuned models to become generally adapted to the novel composition of schema items, but relatively easier for LLM to generalize on this situation (Brown et al., 2020b; Kumar et al., 2022). For the case of MetaQA, the scale of the knowledge base (*i.e.*, WikiMovies) involved in the dataset is relatively small with only dozens of unique relations under the same domain. In this case, the context and topic of the demonstration match exactly the target one, so five demonstrations are enough for LLM to generate highly accurate preliminary relation candidates.

In addition, the corresponding variances for KB-BINDER(1) and KB-BINDER(6) on three runs are $47.0(\pm 3.8)$ and $50.6(\pm 4.5)$ for GrailQA, $52.5(\pm 4.8)$ and $53.2(\pm 4.5)$ for WebQSP, $39.3(\pm 1.7)$ and $39.5(\pm 0.6)$ for GraphQA.

KB-BINDER-R Results As recorded in Table 5, KB-BINDER (1)-R sets new SoTA Hits@1 score on 3-hop MetaQA as 99.5 %, and it achieves exactly the same performance with the previous fully-trained SoTA on 2-hop MetaQA as 99.9 %. From the recordings on all the tables, we observe that KB-BINDER (K)-R has a generally better performance than KB-BINDER (K). Nevertheless, it is worth noting that the improvement on GrailQA is only 2.6 points, while the performance is even slightly weakened on GraphQA by 0.8 points. But KB-BINDER (K)-R dramatically increases the F1 score from 53.2 to 74.4 on WebQSP. It can be rationally explained by the inherent characteristics of the datasets that GrailQA is largely composed of compositional and zero-shot questions and GraphQA only contains compositional questions, while all the questions of the test set on WebQSP are of I.I.D type, which makes the unseen questions more similar to the retrieved exemplars.

In a nutshell, according to the presented experiment results, few-shot approaches with LLMs, such as KB-BINDER (K) can at least achieve performance on par with previous fully-trained SoTAs on KBQA tasks in the following two situations: 1) There is no large-scale annotated training data, but the inference requires high generalizability of the model (i.e., GraphQA); 2) The knowledge base and the corresponding questions are very specific to one domain, so that the search space of the schema items is relatively small, but the inference requires multi-hop reasoning (i.e., MetaQA). And when it comes to a totally I.I.D setting (i.e., WebQSP), KB-BINDER (K)-R can boost the performance to be on par with the supervised models. However, for the case of a large amount of training data with a high requirement for generalizability during inference (i.e., GrailQA), the previous models may have advantages over KB-BINDER due to the fact that the coverage of logical form structures and schema items is restricted in our method.

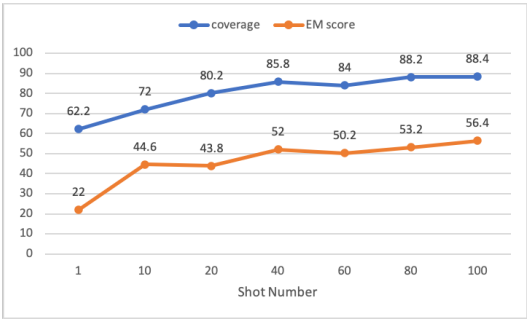

Figure 3: KB-BINDER coverage and EM scores trend with shot number.

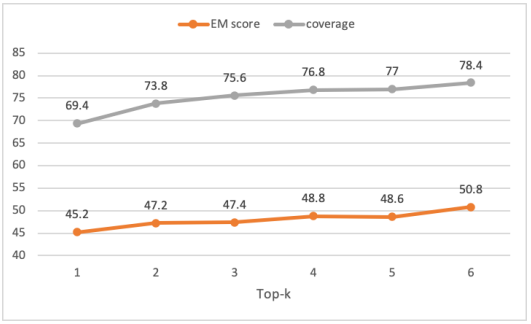

Figure 4: KB-BINDER coverage and EM scores trend with top K self-consistency.

4.5 Ablation Study

We conduct ablation studies to understand the influence of the number of examples demonstrated during drafts generation on the final EM score. Due to the long inference time to complete all the testing questions, we evaluate the performance on 500 randomly sampled questions from the dev set of GrailQA. We set the number of few shot exemplars from 1 to 100, and test the coverage and EM score on each choice. The coverage here refers to the number of questions that can be grounded to at least one executable logical form over the total number of questions in the sampled set. As shown in Figure 3, there is an apparent trend that both the coverage and the EM score will increase with a larger number of examples.

Similarly, we also test KB-BINDER (K) perfor-

Method	IID		Compositional		Zero-shot	
	EM	F1	EM	F1	EM	F1
GloVe + Transduction (Gu et al., 2020)	50.5	51.6	16.4	18.5	3.0	3.1
BERT + Ranking (Gu et al., 2020)	59.9	67.0	45.5	53.9	48.6	55.7
RnG-KBQA (Ye et al., 2021)	86.2	89.0	63.8	71.2	63.0	69.2
TIARA (Shu et al., 2022)	**87.8**	**90.6**	69.2	76.5	**68.0**	**73.9**
Few-shot in-context						
KB-BINDER (6)	51.9	57.4	50.6	56.6	49.9	55.1
KB-BINDER (6)-R	72.5	77.4	51.8	58.3	45.0	49.9

Table 6: Results of KB-BINDER/KB-BINDER-R and baselines on different question types of GrailQA.

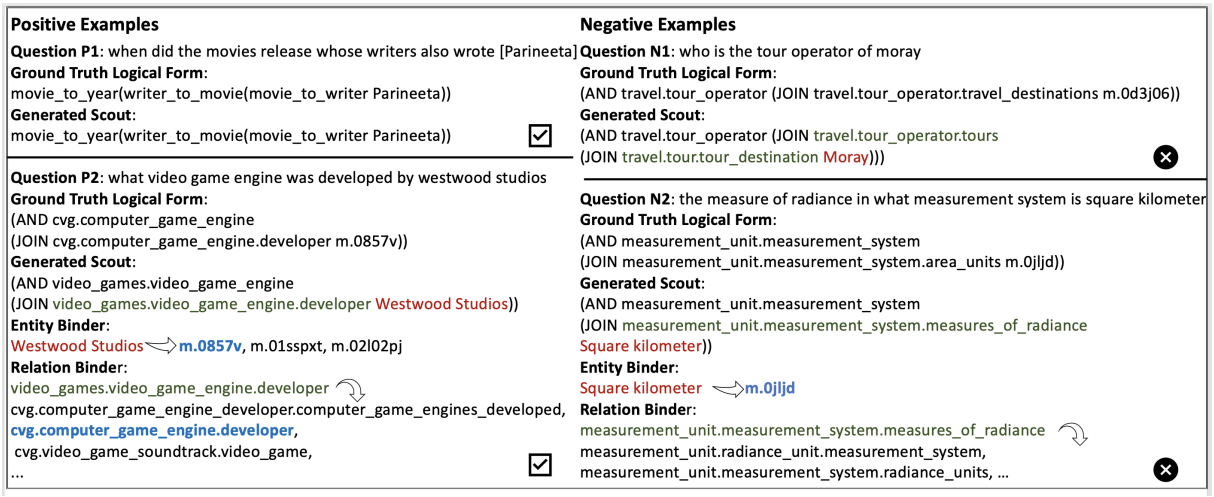

Figure 5: Positive and negative examples generated by KB-BINDER.

mance with respect to the different numbers of top drafts generated by Codex to perform the majority voting. With 40 exemplars, the result is plotted as Figure 4. Generally, increasing the number of drafts from 1 to 6 can contribute to an improvement of coverage by 19% and EM score by 5.6%. As if there are more drafts, more logical form structures and more formats of preliminary schema items can be covered in the first place.

However, it is also worth noting that increasing the number of shots and the number of generated drafts can also increase the inference time and cost for KB-BINDER to find the answer. Taking this reason into account, we only report the results of 40 exemplars with the top 6 drafts on GrailQA, as there is always a trade-off between accuracy and the cost of time. And it also implies that there is still space for improvement for KB-BINDER if we increase both of the parameters.

Moreover, we also observe from Table 6 that for all the supervised baselines, there is a relatively big gap between I.I.D. typed questions and the other two types (*i.e.*, the decreased EM score ranging from 10 to 47.5 points). But with KB-BINDER, the performances are stable among all the types. This is due to the fact that all the questions may not come from I.I.D type for few-shot setting, so there is rarely bias among the three types.

4.6 Case Study

In Figure 5, we show representative correct and error cases in the KB-BINDER pipeline. For Question P1, the generated logical form could exactly match the target one. While for Question P2, it generates the draft in correct logic but the hallucinated entity names and relations need an extra binding step to locate the executable logical form. Question N1 is an error case where the draft does not generate correct logic. On the other hand, Question N2 gets draft logic generated correctly but grounded into wrong entities or relations.

Error Analysis We analysed the performance of each component as the recall of correct MIDs and relations before and after the effect of Entity Binder and Relation Binder, together with the logical path frame generated in draft. On 500 randomly sampled GrailQA dev set, with shot number as 40,

KB-BINDER (1) can achieve 0.9 and 0.78 recall for entity and relation binding respectively, and the recall of the logic frame is 0.66 for the top 1 draft, which account for most error cases. We compare the results with the ones before passing to the two proposed binders on the same split dev set and setting. The recall of correct MIDs and relations are 0.78 and 0.0 correspondingly. After the effect of our entity binder and relation binder, the recalls increase by 12% and 78%, which verifies the effectiveness of each of the proposed components.

In addition, we also conduct a head-to-head comparison for KB-BINDER with one of the baselines in few-shot setting as described in A.1.

5 Conclusion

KB-BINDER is the first framework that enables the challenging few-shot learning on KBQA with the reasoning capability of large language models. It first generates drafts with LLM as preliminary logical forms, and then binds the entities and schema items of the drafts to the target knowledge base iteratively until an executable one can be found. KB-BINDER (K) adopts majority voting, further enlarging the proportion of answerable questions with the help of more diverse formats of top K drafts. KB-BINDER (K)-R with retrieved exemplars is proved to be especially advantageous when applied to I.I.D questions. In general, KB-BINDER and its derivatives achieve strong performance on all the common-used KBQA datasets we select, and we hope it can set a strong baseline for future work on KBQA with a low-resource setting.

Limitations

As in-context learning with LLM heavily depends on the selected exemplars in the prompt, the performance of KB-BINDER might vary from different subsets of randomly sampled examples, especially in a low-shot setting. But KB-BINDER still shows strong performance on thousands of data points on each testing dataset with randomly sampled exemplars, which verifies the robustness of our method to a degree. In the meantime, the performance of KB-BINDER is restricted with the one-time generated drafts from the perspective of the imaginary frame and schema items of the preliminary logical forms, which can be further improved with interactively generation and retrieval. Moreover, we have not explored whether the performance can be further improved with explanation/instruction during the stage of draft generation. We will take these limitations into account and mitigate them in future work.

References

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

Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021a. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374.

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021b. Retrack: A flexible and efficient framework for knowledge base question answering. In *Annual Meeting of the Association for Computational Linguistics*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *ArXiv*, abs/2211.12588.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, R.K. Nadkarni, Yushi Hu, Caiming Xiong, Dragomir R. Radev, Marilyn Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2022. Binding language models in symbolic languages. *ArXiv*, abs/2210.02875.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43.

Alexander Dunn, John Dagdelen, Nicholas Walker, Sanghoon Lee, Andrew S. Rosen, Gerbrand Ceder, Kristin Persson, and Anubhav Jain. 2022. Structured information extraction from complex scientific text with fine-tuned large language models.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *ArXiv*, abs/2211.10435.

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments.

Yu Gu, Sue E. Kase, Michelle T. Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2020. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. *Proceedings of the Web Conference 2021*.

Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. 2022. Knowledge base question answering: A semantic parsing perspective. In *4th Conference on Automated Knowledge Base Construction*.

Yu Gu and Yu Su. 2022. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. In *International Conference on Computational Linguistics*.

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. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.

Yuncheng Hua, Yuan-Fang Li, Gholamreza Haffari, Guilin Qi, and Tongtong Wu. 2020. Few-shot complex knowledge base question answering via meta reinforcement learning. *ArXiv*, abs/2010.15877.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. *ArXiv*, abs/2202.10054.

Andrew K. Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L. McClelland, Jane X. Wang, and Felix Hill. 2022. Can language models learn from explanations in context?

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *International Joint Conference on Artificial Intelligence*.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Annual Meeting of the Association for Computational Linguistics*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *CoRR*, abs/2101.06804.

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

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario

Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Conference on Empirical Methods in Natural Language Processing*.

Apoorv Saxena, Aditay Tripathi, and Partha Pratim Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Annual Meeting of the Association for Computational Linguistics*.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje F. Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. Tiara: Multi-grained retrieval for robust question answering over large knowledge bases. *ArXiv*, abs/2210.12925.

Yu Su, Huan Sun, Brian M. Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Conference on Empirical Methods in Natural Language Processing*.

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

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Conference on Empirical Methods in Natural Language Processing*.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. Sparqa: Skeleton-based semantic parsing for complex questions over knowledge bases. In *AAAI Conference on Artificial Intelligence*.

Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Huai hsin Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *ArXiv*, abs/2210.09261.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *North American Chapter of the Association for Computational Linguistics*.

Xingyao Wang, Sha Li, and Heng Ji. 2022a. Code4struct: Code generation for few-shot structured prediction from natural language. *ArXiv*, abs/2210.12810.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022a. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Peiyun Wu, Xiaowang Zhang, and Zhiyong Feng. 2019. A survey of question answering over knowledge base. In *China Conference on Knowledge Graph and Semantic Computing*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

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

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

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

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

Daoguang Zan, Sirui Wang, Hongzhi Zhang, Yuanmeng Yan, Wei Wu, Bei Guan, and Yongji Wang. 2022. S2ql: Retrieval augmented zero-shot question answering over knowledge graph. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alex Smola, and Le Song. 2017. Variational reasoning for question answering with knowledge graph. In *AAAI Conference on Artificial Intelligence*.

Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Huai hsin Chi. 2022a. Least-to-most prompting enables complex reasoning in large language models. *ArXiv*, abs/2205.10625.

Hattie Zhou, Azade Nova, H. Larochelle, Aaron C. Courville, Behnam Neyshabur, and Hanie Sedghi. 2022b. Teaching algorithmic reasoning via in-context learning. *ArXiv*, abs/2211.09066.

A Appendix

A.1 Few-shot Comparison

To realize a head-to-head comparison with baseline on few-shot setting, we select ArcaneQA (Gu and Su, 2022) as one of the representative baselines to conduct few-shot experiment on GrailQA. We evaluate it on the same 500 dev set we sampled for ablation study of KB-BINDER, and we obtain its EM score as 16.5, 35.2, and 41.9 under 1-shot, 10-shot and 100-shot respectively. Compared to ArcaneQA under same few-shot setting, KB-BINDER (Figure 3) outperforms it by 5.5%, 9.4% and 14.5% for shot number of 1, 10 and 100 correspondingly.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
4.4, 4.5, Limitation

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract; Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 4

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.