Few-shot Logical Form Generation for KGQA via Pseudo Label Pre-training and Knowledge Distillation

Anonymous ACL submission

Abstract

With the emergence of neural language models, extensive research has been conducted on question-answering systems. Knowledge Graph Question Answering (KGQA) remains a hot topic because it returns answers from reliable knowledge graphs, while language models sometimes suffer from hallucinations and produce unfaithful answers. An intuitive and explicable solution for KGQA involves generating logical forms (such as SPARQL, s-expression, Cypher, etc.) that can be executed against the KG. However, due to the heterogeneous nature of KG schemas across different KGs, distinct logical forms are required, thereby necessitating various models. The training process for such models to adapt to diverse KG schema settings is resource-intensive or data-hungry when built upon large language models or smaller models, respectively. In this work, we propose a novel pseudo logical form pre-training strategy to separate the learning process into a pre-training stage and a fine-tuning stage. In the pre-training stage, the model learns to generate KG items according to its understanding of the question. In the fine-tuning stage, the model may focus on learning logical form grammar with limited labeled data. Besides, the proposed strategy can be combined with knowledge distillation to further boost the model's performance. Experimental evaluations conducted on MetaQA and KQA Pro show that our model outperforms several strong baselines, thus substantiating the efficacy of our proposed techniques.

1 Introduction

Knowledge graphs question answering (KGQA) aims at answering natural language questions according to fact triples stored in the knowledge graph (KG). It plays a fundamental role in many industrial applications like search engines and AI assistants. Previous methods involve information extraction and retrieval, where they first identify the topic entity from the input question and then ranking the retrieved entities to select final answer. Besides, constructing a Query Graph and converting it into executable logical forms is a more explainable method (Yih et al., 2015; Luo et al., 2018; Lan and Jiang, 2020; Qiu et al., 2020; Qin et al., 2021). With developement of pre-trained generative models, directly generate logical forms conditioned on the input question in an end-to-end manner is becoming popular (Cao et al., 2022). This is similar to the recent advances in Text-to-SQL, where SQL queries are generated conditioned on the inputs consists of questions and database schema items (Zhong et al., 2017; Xu et al., 2018, 2022; Qi et al., 2022; Qin et al., 2022). However, the quantity of a KG's schema items is typically much greater than that of a relational database, resulting more challenges for end-to-end generation of logical forms.

Concretely, there are three challenges to generate correct logical forms. First, the model needs a full understanding of the intent of the input question. Second, the model needs to align the semantics of the question to KG items, including entity identifiers and relationships. Third, the model

Question: what are the languages spoken in the films directed by Joel Zwick?

GraphQ_IR
```
What is the attribute <A> language </A> of <ES> ones that <R> directed by
</R> backward to <E> Joel Zwick </E> </ES>
```
KoPL
```
Find(Joel Zwick).Relate(directed by,backward).QueryAttr(language)
```
SPARQL
```
SELECT DISTINCT ?pv WHERE { ?e <directed_by> ?e_1 . ?e_1 <pred:name> "Joel
Zwick" . ?e <language> ?pv . }
```
Cypher (s1: *language* as a relationship)
```
MATCH (n1{name:"Joel Zwick"})<-[:DIRECTED_BY]-(n2)-[:language]->(n3)
RETURN DISTINCT n3.name
```
Cypher (s2: *language* as a property)
```
MATCH (n1)<-[:DIRECTED_BY]-(n2) WHERE n1.name = "Joel Zwick"
RETURN DISTINCT n2.language
```
Pseudo Label
```
<mask>directed_by<mask>name<mask>"Joel Zwick"<mask>language<mask>
```

Figure 1: An example from MetaQA dataset. GraphQ_IR, KoPL, SPARQL and Cypher are logical forms that will return answer for the given question when executed against the KG. When the KG is hosted using Neo4j, the corresponding logical form will be Cypher. According to the schema setting we use, *language* can be a relationship between nodes (s1) or a property name of a node (s2).

should be aware of the grammar of target logical forms to generate executable sequences. With the power of pre-training on massive text corpus, the transformer-based language models are proved to have a good language understanding ability and recent researches mainly focus on the second and the third challenges. For the second challenge, although some work have focused on injecting knowledge from a KG and gained certain improvements on some specific tasks (Agarwal et al., 2021; Thorne et al., 2021; Moiseev et al., 2022), the hallucination issue still exists in current models (Maynez et al., 2020; Sun et al., 2023). Providing linked KG schema items in the input sequence helps alleviate but fails to prevent the model from generating semantic similar but unfaithful results. For the third challenge, there are already various methods to help the model to generate syntactically correct logical forms. Most commonly used method is grammar-guided decoding (Krishnamurthy et al., 2017; Yin and Neubig, 2018; Guo et al., 2019), where an abstract syntax tree (AST) is firstly generated. Scholak et al. (2021) propose a pluggable method called PICARD to refuse incorrect results at each autoregressive generation step. However, all of these methods requires large amounts of labeled data for training because the model has to learn to generate KG items from either natural questions or retrieved results and acquire the grammar of logical forms at the same time.

With the rising of large language models (LLMs), researchers have explored their abilities in few-shot and zero-shot settings and results have proved their superiority in intent understanding and instruction following. Gu et al. (2023) proposed to employ LLM's discriminative ability to guide agents to iteratively search over the possible sequence space. In this way, agents generate faithful and syntactically correct logical form candidates and the model only faces the challenge of intent understanding, i.e., the model only needs to decided which candiate matches the question intent preicisely. Similarly, Li et al. (2023) employes LLMs to generate *drafts* of logical forms and binds *surface names* of entities and relationships using retrieval tools. In their proposed KB-BINDER, the challenge of aligning from question semantics to KG items is assigned to external retrieval module, the grammar of logical forms is obtained via in-context learning ability of the LLM, and the final answer is decided by self-consistancy (Wang et al., 2023) and majority voting. Although these methods demonstrate astonishing efficacy without the need for a training phase, their utilization of LLM still necessitates substantial computational resource expenditure and raises concerns regarding data security.

Inspired by the pre-training tasks of BART (Lewis et al., 2020), we propose a method of constructing pseudo logical forms for pre-training, where the pseudo logical forms only consist of KG items and $[mask]$ tokens. In the fine-tuning stage, the $[mask]$ tokens are trained to recover to tokens about syntactics of the target logical form. This approach allows us to separate the learning of entity alignment and grammar of logical forms into the pre-training and fine-tuning stages, respectively. As answering questions only necessitates a limited subset of the entire grammatical structure, it becomes markedly simpler for the model to acquire this knowledge. Our experimental results on the KQA and MetaQA datasets demonstrate the effectiveness of our approach, as even with a small number of annotated samples for fine-tuning, satisfactory performance can be achieved. Moreover, this method can be combined with knowledge distillation, thereby leveraging the benefits from a well-pre-trained teacher model.

To conclude, our contributions are:

- We propose pseudo logical form pre-training, separating the KG item alignment and logical form grammar acquisition into pre-training and fine-tuning stage respectively.
- The proposed pseudo logical form pre-trainig is compatible with knowledge distillation and therefore could further leverage benefits from a well-pre-trained teacher model.
- Experiments on two KGQA datasets have demonstrated the effectiveness on generating various types of logical forms using few labeled data.

2 Related Work

2.1 KGQA

With the proliferation of comprehensive knowledge graphs such as DBPedia (Auer et al., 2007), Freebase(Bollacker et al., 2008), and Wikidata(Vrandečić and Krötzsch, 2014), research into KGQA has been rapidly advancing. Broadly, information retrieval (IR)-based methods and semantic parsing (SP)-based methods are two main

streams in this research area (Lan et al., 2021). SP-based methods, which are gaining popularity due to their superior interpretability compared to IR-based methods, frame KGQA as the challenge of transforming natural language queries into executable logical forms. In early researches, Query Graphs are built and converted into logical forms. Bornea et al. (2021) and Kapanipathi et al. (2021) utilize intermediate forms like AMR to help the generation of target logical forms. With the advance of PTMs like BART (Lewis et al., 2020), Cao et al. (2022) directly generates logical forms from natural questions .

To address the challenges of generating logical forms, retrieval is commonly used to provide faithful context for generation models (Ye et al., 2022; Gu and Su, 2022; Hu et al., 2022). Shu et al. (2022) employ multi-grained retrieval to provide semantic and syntactic context to ease the generation. They also constrain the decoding process by constructing trie (prefix tree) of KG items. Yu et al. (2022) combine the generation of logical forms and direct generation of the answer, achieving impressive results on several datasets. Nie et al. (2022) proposes a new intermediate representation which is more close to natural language than traditional logical forms. With the rising of LLMs, Gu et al. (2023) propose to use LLMs to discriminate candiate logical forms generated by interacting with real-world environment and achieve promising results on several KGQA datasets. KB-Binder works in a few-shot in-context learning paradigm to generate logical forms with the power of LLMs (Li et al., 2023). However, it involves a complex pipeline to generate hunderds of candiates thus results in high computational costs. In this work, we aim at developing an end-to-end logical form generation model with less computational resources.

2.2 Knowledge Distillation

Since when knowledge distillation is proposed, it has been extensively studied to perform model compression or domain migration via effective knowledge transfer (Hinton et al., 2015). Previous works have studied the potential of distilling BERT (Devlin et al., 2019) for text generation (Chen et al., 2020) . Many neural machine translation researches towards low-resource languages also have explored the utilization of knowledge distillation (Ansell et al., 2023; Wang et al., 2020). There are two major categories of knowledge distillation methods used for text generation: word-level and sequence-level (Kim and Rush, 2016). In this work, we designed an optional knowledge distillation method that can be integrated with proposed pseudo logical form pre-training to further augment model's ability in question understanding and KG item alignment. Through knowledge distillation, intricate features and representations learned by the comprehensive teacher model are effectively transferred to the student model. In doing so, the student model inherits the teacher's capability to understand natural language questions, leveraging the dark knowledge encapsulated in soft labels to refine its own acknowledgment of the KG contents.

3 Task Definition

3.1 Knowledge Graph

Knowledge graph is a special knowledge base that stores knowledge in graph structure. A typical knowledge base consists of an ontology $\mathcal{O}$ and a model $\mathcal{M}$ (Gu et al., 2022) . $\mathcal{M}$ is data model representing facts and $\mathcal{O}$ is the ontology containing schema configurations. Schema configurations includes property, relationships and their interconnectivity between different types of nodes. For example, different schema settings results in different Cypher queries as shown in Figure 1.

For RDF data model, $\mathcal{M} \subseteq (\mathcal{E} \cup \mathcal{R}) \times \mathcal{R} \times (\mathcal{C} \cup \mathcal{E} \cup \mathcal{V})$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of binary relations, $\mathcal{C}$ is a set of classes, and $\mathcal{V}$ is a set of literal values. For Neo4j's labeled property graph model , $\mathcal{M} \subseteq \mathcal{N} \times \mathcal{R} \times \mathcal{P} \times \mathcal{L}$, where $\mathcal{N}$ is a set of nodes, $\mathcal{R}$ is a set of directed relations, $\mathcal{P}$ is a set of properties, and $\mathcal{L}$ is a set of labels. Both nodes and relationships possess distinctive identifiers and can store properties represented as key–value pairs. Nodes can be labeled to be grouped. The edges in LPG representing the relationships always have a start node and an end node, making the graph a directed graph.

3.2 Task Formulation

Given $\mathcal{G} = \{\mathcal{O}, \mathcal{M}\}$ and a natural language question $\boldsymbol{q} = \{x_1, x_2, ..., x_n\}$, the task is to generate a logical form $\boldsymbol{l} = \{y_1, y_2, ..., y_\ell\}$ that is executable against $\mathcal{G}$ and returns an answer $\boldsymbol{a}$ to the question. This procedure is formulated as:

$$P(\boldsymbol{l}) = \prod_{i=1}^{\ell} P(y_i | \boldsymbol{q}, y_1, y_2, ..., y_{i-1}) \quad (1)$$

$$\boldsymbol{a} = \phi(\mathcal{G}, \boldsymbol{c}) \quad (2)$$

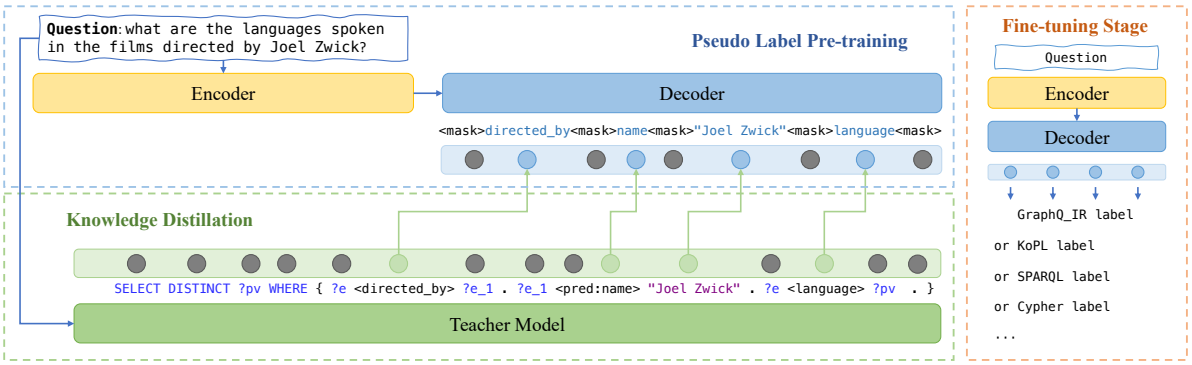

Figure 2: The overview of the proposed method. We take BART as the base model. The teacher model is a proficient logical form generation model whose input is natural questions.There is no restriction to the architecture of the teacher model, both seq2seq models and causal models will fit.

where n is the question length , ℓ is the target logical form length, and ϕ is the execution function against $\mathcal{G}$. The answer $\boldsymbol{a}$ can be a node, a relationship, a number or a boolean value.

3.3 Logical Forms

In this work, we investigate four kinds of logical forms: SPARQL , Cypher, GraphQ_IR and KoPL.

SPARQL SPARQL is the query language for RDF data (Consortium et al., 2014). A target graph pattern is usually delineated in a *where* clause, wherein variables (which start with a question mark), relationships, and literal values combine to form triple sequences. SPARQL is the most commonly used logical form in KGQA tasks because RDF is the most popular data format of KG triple storage (Gu et al., 2022).

Cypher Cypher is the graph query language of Neo4j and a logical form for KGQA. A typical CQL statement commences with the keyword *MATCH*, which is then succeeded by a graph pattern expression. In the graph pattern expression, parentheses are used to represent a node, and square brackets are used to represent an edge. Take the Cypher query in Figure 1 for example, `(n1{name:"Joel Zwick"})<-[:DIRECTED_BY]-(n2)` represents node $n2$ has a relationship `:DIRECTED_BY` directed to node $n1$. The arrow near the node $n1$ indicates the direction of the relationship.

KoPL KoPL is first introduced in Cao et al. (2022). It is designed to describe the explicit reasoning processes for complex questions over knowledge bases. There are 27 functions in the KoPL library, providing an explicit modeling of the reasoning steps.

GraphQ IR To mitigate the discrepancy between traditional logical forms and natural language utterances, Nie et al. (2022) have designed GraphQ IR (graph query intermediate representation) which has natural-language-like expression and formally defined syntax. It cannot be executed directly but can be transpiled into logical forms such as KoPL and SPARQL[1].

4 Methodology

4.1 Pseudo Logical Form

To generate correct logical forms end-to-end, the model needs two abilities: generate KG items conditioned on input question and generate grammatically correct logical forms. The first one requires awareness of KG contents and mapping knowledge from question semantics to KG items, where KG items (i.e., entity names and relationships) are different from entity mentions in the question or are implicitly embeded in the question. The second one is commonly learned during traing process and extra grammar module will help in the decoding process. Previous methods, whether aided with extra retrieval modules, train the model to acquire such abilities at the same time, therefore increaing the demand for labeled data. We notice that KG items in the logical forms are invariant when the question and the KG are fixed, where other tokens in the logical forms vary conditioned on question intents and grammar of logical forms. For example, entity *"Joel Zwick"* and relationships *directed_by, name* and *language* will always appear in logical forms as shown in Figure 1.

Inspired by the pre-train task of BART, where $[mask]$ tokens are trained to reconstruct text spans,

[1]The transpile process involves rule-based transpilers, which are schema-specific, and implementing such transpilers requires expertise in professional tools.

we propose to build pseudo logical forms that only consists of KG items and $[mask]$ tokens to be the pre-training labels. Formally, the pseudo logical form $\boldsymbol{p}$ for question $\boldsymbol{q}$ is defined as:

$$\boldsymbol{p} = \{p_1, p_2, ..., p_m\}, p_m \in \{[mask] \cup \boldsymbol{y}^{inv}\} \quad (3)$$

$$\boldsymbol{y}^{inv} = \mathcal{M} \cap \boldsymbol{l} \quad (4)$$

where m is the length of the pseudo logical form and each token of the invariant KG items, noted as $y_i \in \boldsymbol{y}^{inv}$, is surrounded by $[mask]$ tokens. We construct pseudo logical forms as target labels for every question in the training set and pre-train the model to maximize $P(\boldsymbol{p}|\boldsymbol{q})$. Through pseudo pre-training, the model acquires the knowledge of mapping between natural question semantics and KG items. In the subsequent fine-tuning stage, the model can only focus on generating tokens that are associated with logical form grammar by filling $[mask]$ tokens.

4.2 Knowledge Distillation

Assuming the presence of a proficient question-to-logical-form model $\boldsymbol{\theta_t}$, the suggested approach of incorporating pseudo logical form pre-training could potentially yield advantages when coupled with a knowledge distillation strategy utilizing $\boldsymbol{\theta_t}$ as the teacher model.

Similar to previous formulation, the teacher model $\boldsymbol{\theta_t}$ generates logical form $\boldsymbol{l}^t = \{y_1, y_2, ..., y_t\}$ conditioned on input question $\boldsymbol{q}$. Since the teacher model is trained on complete logical forms, its knowledge about KG items is more comprehensive compared to the student model trained using pseudo-labels. Therefore, we combine the pre-training of the student model with knowledge distillation from the teacher model by pushing the token logits of $\boldsymbol{y}^{inv}$ to the teacher model's outputs. The loss functions for pre-training and knowledge distillation are as follows:

$$\mathcal{L}_{LM} = -\frac{1}{s}\sum_{i=1}^{s} log P_{\theta_c}(p_i) \quad (5)$$

$$\mathcal{L}_{KD} = \frac{1}{k}\sum_{p_i \in \boldsymbol{y}^{inv}}^{k} KL(P_{\boldsymbol{\theta_s}}(p_i|\boldsymbol{q}) || P_{\boldsymbol{\theta_t}}(p_i|\boldsymbol{q})) \quad (6)$$

$$\mathcal{L} = \mathcal{L}_{LM} + \mathcal{L}_{KD} \quad (7)$$

where p_i are tokens from $\boldsymbol{p}$, k is the number of KG related tokens, $\boldsymbol{\theta_s}$ is the student model and $\boldsymbol{\theta_s}$ is the teacher model.

5 Experiments

5.1 Datasets

MetaQA MetaQA (Zhang et al., 2018) is derived from the WikiMovies and contains over 400k annotated question answer pairs in different levels of difficulty. It is a domain specific dataset, where number of entities and relationships is limited as shown in Table 8. In this paper, we use the version provided by Nie et al. (2022), where GraphQ_IR, KoPL, SPARQL and Cypher annoations are available. For the few-shot setting, we also following Nie et al. (2022) to sample $k \in 1, 3, 5$ examples for each question type as shown in Table 7 .

KQA Pro KQA Pro is currently the largest KGQA dataset for complex questions, providing more than 100k question-answer pairs with KoPL and SPARQL annoations. It is based on a delicate database which is a Wikidata subset customized with FB15K-237 (Toutanova et al., 2015). According to the KoPL annotation in KQA Pro, there are 12 types of questions. As the annotation for the test set of KQA Pro is not publicly available, we randomly selected $3,000$ samples from the validation set while keeping the distribution of the 12 types of questions. The remaining 8,797 samples in the validation set are treated as the test set, see Table 7. To simulate practical low-resource scenario, we randomly sample 50, 100, and 200 training samples from the train set for the experiments.

5.2 Baselines

Seq2seq Models Following the original KQA Pro paper (Cao et al., 2022), we take BART (Lewis et al., 2020) as our seq2seq baseline. T5 is also a popular seq2seq model and has similar pre-training tasks to BART. However, the curly brackets ({ and }) which is a key element in logical forms like SPARQL and Cypher are not included in the vocabulary of the T5 series models. We skip T5 series because it is not a key point to compare between pre-trained models [2]. Transfer Learning (TL) is a classic method for low-resource translation, which is similar to our task. We implement BART+TL by initializing from a model[3] trained on NL-SPARQL pairs and fine-tuning on NL-Cypher pairs. SKILL (Moiseev et al., 2022) infuses KG

[2]Our method can be applied T5 by modifying the vocabulary or replace curly brackets in logical forms using other tokens like $< lb >$ and $< rb >$.

[3]Take from Cao et al. (2022), also used as the teacher model for KD

Model	#Samples 50			#Samples 100			#Samples 200		
	BLEU-4	ER	Acc	BLEU-4	ER	Acc	BLEU-4	ER	Acc
KvMemNet (Miller et al., 2016)	-	-	1.64	-	-	6.25	-	-	4.22
RGCN (Schlichtkrull et al., 2018)	-	-	4.23	-	-	9.71	-	-	10.56
ChatGPT$_{Rand}$	56.35	94.90	14.50	54.64	93.60	13.90	54.28	94.30	14.50
ChatGPT$_{BM25}$	72.78	96.60	34.10	75.26	96.60	37.70	78.46	97.70	41.40
GPT2 large (Radford et al., 2019)	48.08	**90.38**	5.08	53.17	**97.45**	6.74	55.08	92.37	11.44
LLaMA 7B (Touvron et al., 2023a)	55.28	89.16	7.37	63.38	91.35	11.67	74.89	**93.04**	31.70
LLaMA2 7B (Touvron et al., 2023b)	65.08	80.00	17.54	68.59	85.12	19.85	73.38	86.19	25.59
BART base (Cao et al., 2022)	63.66	72.25	10.19	66.77	77.74	14.82	78.47	90.91	23.51
BART+SKILL* (Moiseev et al., 2022)	59.29	71.96	9.95	67.23	81.37	15.12	74.86	88.41	26.69
BART+GraphQ IR (Nie et al., 2022)	-	-	21.00	-	-	24.07	-	-	31.98
BART+TL (Zoph et al., 2016)	70.96	49.31	9.31	78.63	70.93	32.92	86.94	81.36	50.20
Ours	**72.87**	76.66	**23.98**	**84.27**	83.97	**40.68**	**90.16**	89.54	**52.12**

Table 1: The performances on KQA Pro trained with different numbers of data samples. BART-TL stands for transfer training. For ChatGPT results, we use a five-shot in-context learning setting, where ChatGPT$_{Rand}$ randomly samples five exemplars and ChatGPT$_{BM25}$ uses the top five exemplars returned from a BM25 retriever.

knowledge into language models via direct pre-training on serialized KG triples. We implement BART+SKILL* by convert triples into Cypher patterns, e.g., <House, has_genre,Horror> is formatted as `(:Entity{name:"House"})-[:R{name:"has_genre"}]-(:Entity{name:"Horror"})`.

Causal Models For decoder-only causal language models, we have selected GPT2 (Radford et al.) and the recently released LLaMA (Touvron et al., 2023a) to compare in the fine-tuning setting. We further explored the capabilities of ChatGPT[4] to generate Cypher via an in-context learning paradigm. To eliminate any randomness introduced by sampling, we set the temperature parameter T to zero in the API request.

Others KvMemNet (Miller et al., 2016) and RGCN (Schlichtkrull et al., 2018) are two baselines for KQA Pro, we use the implementation of Cao et al. (2022) and fine-tuning under our few-shot setting.

5.3 Metrics

The primary goal of KGQA is to obtain the correct response to the input question. Following Cao et al. (2022), we adopt Answer Accuracy (**Acc**) as the principal evaluation metric on KQA Pro. Besides, we take the widely-used text-generation metric, **BLEU-4**, to measure the similarity between the generated logical forms and golden labels. To measure models' acquisition of the grammar, we define the metric Executable Rate (**ER**) as the percentage of whether the generated logical form can be parsed into a valid abstract syntax tree. Exact Match (**EM**) of the prediction and reference is a substitution of answer accuracy. When the execution endpoint is absence, we use the exact match score to replace answer accuracy because the answer is assuredly correct if the generated logical form matches the label.

[4]The API used is gpt-3.5-turbo.

	BLEU-4	ER	EM
LLaMA2 7B (Touvron et al., 2023b)	72.75	97.80	13.14
BART base (Cao et al., 2022)	89.81	98.29	62.55
BART+SKILL* (Moiseev et al., 2022)	89.36	99.74	59.15
BART+GraphQ IR (Nie et al., 2022)	-	-	67.46
Ours w/o KD	-	-	**71.13**

Table 2: Results on MetaQA under 1-shot setting.

5.4 Implementaion Detail

We use BART base as the base model and fine-tune in half precision mode. In the pre-training stage, we use AdamW optimizer with a learning rate of 5e-5, and train 10 epochs on the trainset where input is the question and target label is the pseudo logical form. For knowledge distillation, we use temperature $T = 5$. In the fine-tuning stage, we train the model for 10,000 steps and select checkpoint according to the loss value on validation set. All experiments are conducted on an RTX 3090.

5.5 Main Results

In Table 1, the models' performances are compared across three training sample sizes: 50, 100, and 200. The results show a general trend of improved performance with an increase in the number of training samples. This is evident across all models, with notable improvements in BLEU-4, ER, and Acc. Our model consistently achieves the highest scores in both BLEU-4 and answer accuracy across all training sample sizes, setting it apart as the best-

	BLEU-4	ER	Acc
Ours	84.27	83.97	40.68
w/o PT	81.62 (-2.65)	74.79 (-9.18)	34.64 (-6.04)
w/o KD	73.01 (-11.26)	82.00 (-1.97)	18.81 (-21.79)

Table 3: Ablation study on KQA Pro trained with 100 samples. PT and KD stands for pseudo label pre-training and knowledge distillation, respectively.

Logical Form	1 shot	3 shot	5 shot
Fine-tune from BART base			
SPARQL	71.43	90.10	91.52
Cypher	62.55	84.23	92.71
KoPL	78.76	79.97	94.00
GraphQ_IR	55.52	90.45	94.85
Fine-tune from pseudo pre-trained			
SPARQL	75.06 (+3.61)	91.44 (+1.33)	93.31 (+1.79)
Cypher	61.00 (+1.55)	87.62 (+3.39)	92.56 (-0.15)
KoPL	78.69 (-0.07)	93.13 (+13.16)	93.90 (-0.10)
GraphQ_IR	71.13 (+15.61)	93.26 (+2.81)	96.50 (+1.65)

Table 4: The EM score of logical forms on MetaQA.

Logical Form	#samples 50	#samples 100	#samples 200
Fine-tune from BART base			
SPARQL	4.69	9.4	15.52
Cypher	5.22	10.04	18.7
KoPL	8.12	8.55	20.43
GraphQ_IR	11.42	15.41	25.84
Fine-tune from pseudo pre-trained			
SPARQL	8.53 (+3.84)	11.77 (+2.37)	21.44 (+5.92)
Cypher	8.20 (+2.98)	11.38 (+1.34)	21.06 (+2.36)
KoPL	9.86 (+1.74)	13.89 (+5.34)	24.84 (+4.41)
GraphQ_IR	11.9 (+0.48)	17.89 (+2.48)	26.12 (+0.28)

Table 5: The EM score of logical forms on KQA Pro.

performing model. Notably, GPT2 performs well in ER with the worst BLEU-4 score. The reason behind this phenomenon is that GPT2 tends to generate simple and short sequences, thus achieving higher ER score. LLaMA2 shows superiority with 50 and 100 training samples, while LLaMA performs better with 200 training samples. LLaMA models are comparable to BART+SKILL*, which is infused with KG knowledge, highlighting the power of pre-training and large parameter sizes. For ChatGPT methods, BM25 (Robertson et al., 2009) selects similar question and enable ChatGPT to learn from corresponding logical forms that are similar to the target logical form, resulting in relatively high performance. However, ChatGPT fails to get good results when provided with randomly selected context. More discussion about ChatGPT is in Appendix B.5. BART+SKILL* is slightly better than BART base when the number of training samples increases. GraphQ IR generates intermediate representations (IR) similar to natural language, and the IRs are transpiled into other logical forms to get the answer. It yields the best performance model without aid from a teacher model. Although BART+TL showed superiority in BLEU-4 and Acc, it was the worst-performing model in terms of executable rate because it sometimes generates sequences with mixed logical form syntax(E.g., half SPARQL and half Cypher). Our model absorbs knowledge from pseudo labels and a well-trained teacher model and is free from the teacher's stereotype of SPARQL syntax; thus, it learns well in both syntax and semantics and achieves the highest answer accuracy.

Resutls for MetaQA are shown in Table 2. We have no execution endpoint so the EM score is used to replace Acc. Compared with directly fine-tuning with GraphQ IR, our pseudo label pre-trained model achieves certain improvements[5]. The KD method is not applied on this dataset because we have not found a suitable teacher model that is publicly available.

5.6 Ablation Study

As shown in Table 3, both PT and KD contribute to our model. When KD is removed, the BLEU-4 and Acc drop more, indicating the knowledge transferred from the teacher model helps the student to learn the question intent and relationships between entities. When PT is removed, the ER drops a lot, indicating that PT is positive or KD is negative to the learning of logical form grammar. Compared to the BART base, whose ER score is 77.74, we can conclude that both influences exist as PT increases the ER score by 4.26 points and KD decreases by 2.95 points. That means the teacher's knowledge about SPARQL grammar is transferred through the distillation of invariant tokens and harms the student's adaptation to Cypher syntax.

6 Discussion

6.1 Generalization across Logical Forms

We have conducted the pre-train-then-finetune experiments across four logical forms on both MetaQA and KQA Pro datasets.

As expected, both starting points (BART base and pseudo logical form pre-trained) demonstrate

[5]The BLEU-4 and ER score is not reported because the transpiler of GraphQ IRs handles errors and always generates syntax-correct SPARQL queries.

Model	#Samples 50			#Samples 100			#Samples 200		
	Easy	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
GPT2 large	4.87	6.28	1.57	7.77	7.09	4.05	19.13	8.81	8.10
LLaMA 7B	5.12	8.37	7.58	10.76	12.90	9.14	53.42	26.24	16.07
LLaMa2 7B	34.67	12.57	7.32	33.01	17.24	8.10	40.31	23.33	10.32
BART base	24.42	5.61	3.14	28.18	10.99	6.73	39.50	19.74	11.17
BART+SKILL*	19.34	6.95	5.23	25.92	12.55	6.86	45.00	21.51	15.35
BART+GraphQ IR	39.67	15.92	8.82	46.97	17.22	11.04	57.56	25.12	14.89
BART+TL	26.17	7.86	3.52	35.14	11.50	5.55	41.67	20.76	10.45
Ours	**47.14**	**16.51**	**12.61**	**61.66**	**37.19**	**19.86**	**71.69**	**48.56**	**33.64**

Table 6: Answer accuracy on different question complexity levels. The proportion of *easy*/*medium*/*hard* questions in the test set is $26.6\%/56\%/17.4\%$.

an improvement in performance with an increase in the number of shots. This reaffirms the common understanding that more training data typically yields better model performance. However, the crux of the analysis revolves around the difference between the BART base and the pseudo pre-trained models.

On MetaQA, as shown in Table 4, the pseudo logical form pre-training strategy, on the whole, outperforms the BART base, especially in 1-shot GraphQ_IR and 3-shot KoPL tasks, with improvements of $+15.61\%$ and $+13.16\%$ respectively. These significant boosts suggest that this strategy has a pronounced effect on these specific logical forms. On the other hand, while there are instances where the pseudo pre-trained model marginally underperforms compared to the BART base, these deviations are minimal (around 0.1%).

Different from MetaQA, KQA Pro is an open-domain dataset with complex questions that require multiple reasoning steps to answer. As shown in Table 5, the proposed pseudo label training strategy is also effective on this dataset, resulting in 0.28% to 5.92% certain improvements on various settings.

6.2 Question Complexity

We define three question complexity levels according to the length of annotated KoPL provided in the original KQA Pro. Questions that can be solved in fewer than four function steps are defined as *easy* level, questions that require more than six function steps to solve are defined as *hard* level, while the remaining questions are grouped as *medium* level. As shown in Table 6, all of the models perform badly when trained with only 50 samples. It is noteworthy that the LLaMA series and GraphQ IR show strong performance on easy questions and have a generalization ability to longer questions compared to other models, revealing the power of unsupervised pre-training on the massive corpus. Our model achieves the best performance due to its explicit acquisition of relationships and entity knowledge and implicit forward-looking knowledge from the teacher model regarding the generation of complete logical forms. It is particularly evident when facing hard questions that have many more reasoning steps and involve more entities and relationships.

7 Conclusion

In this work, we introduced a novel pseudo-logical form pre-training approach that splits the learning process of logical form generation models into a pre-training and a fine-tuning phase. During pre-training, the model gains the capability to generate KG items based on its interpretation of a given question. Fine-tuning then emphasizes logical form grammar learning, where the model learns to fill mask tokens with minimal training data. We experiment with golden pseudo labels and noisy ones that are automatically generated via in-context learning of LLMs, and the results have demonstrated the effectiveness of the proposed method to boost few-shot performances in generating multiple kinds of logical forms. Besides, we also propose a knowledge distillation method that can further improve model performances when proficient teacher models are available. Our model outperforms several strong baselines in various experimental settings, demonstrating the effectiveness of the proposed methods. The proposed methods facilitate seamless transitions of KGQA systems across different schema settings, such as moving from an RDF database to a Neo4j backend and switching from SPARQL to Cypher. In the future, we will explore the utilization of LLM capabilities to provide guidance for smaller models due to the scarcity of proficient teachers in most scenarios.

8 Limitation

Our method is inspired by the pre-training task "corrupt text span" of seq2seq language models like T5 and Bart. The $[mask]$ token in our pseudo label will be filled with an indeterminate number of tokens in the subsequent fine-tuning phase. However, based on some experimental results with causal models (GPT, Llama), the absence of the mask token significantly impacts the performance of our method. Nonetheless, according to our experiments, small seq2seq models like BART base can surpass the performance of causal models which is much larger in parameter size. It does not appear to be a substantial limitation as seq2seq model represents a superior option in terms of both performance and computational cost.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. 2023. Distilling efficient language-specific models for cross-lingual transfer.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, volume 4825, pages 722–735. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data - SIGMOD '08*, page 1247, Vancouver, Canada. ACM Press.

Mihaela Bornea, Ramon Fernandez Astudillo, Tahira Naseem, Nandana Mihindukulasooriya, Ibrahim Abdelaziz, Pavan Kapanipathi, Radu Florian, and Salim Roukos. 2021. Learning to Transpile AMR into SPARQL.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. 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)*, pages 6101–6119, Dublin, Ireland. Association for Computational Linguistics.

Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2020. Distilling Knowledge Learned in BERT for Text Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7893–7905, Online. Association for Computational Linguistics.

World Wide Web Consortium et al. 2014. Rdf 1.1 concepts and abstract syntax.

Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:4171–4186.

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

Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. 2022. Knowledge Base Question Answering: A Semantic Parsing Perspective.

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

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. pages 1–9.

Xixin Hu, Xuan Wu, Yiheng Shu, and Yuzhong Qu. 2022. Logical form generation via multi-task learning for complex question answering over knowledge

bases. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1687–1696, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging Abstract Meaning Representation for Knowledge Base Question Answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894, Online. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

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 *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4483–4491, Montreal, Canada. International Joint Conferences on Artificial Intelligence Organization.

Yunshi Lan and Jing Jiang. 2020. Query Graph Generation for Answering Multi-hop Complex Questions from Knowledge Bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980, Toronto, Canada. Association for Computational Linguistics.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. 2018. Knowledge Base Question Answering via Encoding of Complex Query Graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2185–2194, Brussels, Belgium. Association for Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. 2022. Skill: Structured knowledge infusion for large language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1581–1588, Seattle, United States. Association for Computational Linguistics.

Lunyiu Nie, Shulin Cao, Jiaxin Shi, Qi Tian, Lei Hou, Juanzi Li, and Jidong Zhai. 2022. GraphQ IR: Unifying Semantic Parsing of Graph Query Language with Intermediate Representation.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3215–3229, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. 2022. A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions.

Kechen Qin, Cheng Li, Virgil Pavlu, and Javed Aslam. 2021. Improving Query Graph Generation for Complex Question Answering over Knowledge Base. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4201–4207, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yunqi Qiu, Kun Zhang, Yuanzhuo Wang, Xiaolong Jin, Long Bai, Saiping Guan, and Xueqi Cheng. 2020.

Hierarchical Query Graph Generation for Complex Question Answering over Knowledge Graph. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1285–1294, Virtual Event Ireland. ACM.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. page 24.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 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*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Bin Sun, Yitong Li, Fei Mi, Fanhu Bie, Yiwei Li, and Kan Li. 2023. Towards fewer hallucinations in knowledge-grounded dialogue generation via augmentative and contrastive knowledge-dialogue. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1741–1750, Toronto, Canada. Association for Computational Linguistics.

James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. 2021. Database reasoning over text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3091–3104, Online. Association for Computational Linguistics.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

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

Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. 2023. Scott: Self-consistent chain-of-thought distillation.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Fei Huang, and Kewei Tu. 2020. Structure-Level Knowledge Distillation For Multilingual Sequence Labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3317–3330, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kuan Xu, Yongbo Wang, Yongliang Wang, Zihao Wang, Zujie Wen, and Yang Dong. 2022. SeaD: End-to-end Text-to-SQL Generation with Schema-aware Denoising. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1845–1853, Seattle, United States. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. SQL-to-Text Generation with Graph-to-Sequence Model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 931–936, Brussels, Belgium. Association for Computational Linguistics.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation Augmented Iterative Ranking for Knowledge Base Question Answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

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 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A Transition-based Neural Abstract Syntax Parser for Semantic Parsing and Code Generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Brussels, Belgium. Association for Computational Linguistics.

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

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018. Variational Reasoning for Question Answering With Knowledge Graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

A Experiment Data

A.1 Dataset split

	Train	Validation	Test
MetaQA 1-shot	49	4,900	39,093
MetaQA 3-shot	147	4,900	39,093
MetaQA 5-shot	240	4,800	4,800
KQA Pro #50	50	3,000	8,797
KQA Pro #100	100	3,000	8,797
KQA Pro #200	200	3,000	8,797

Table 7: Data splits for the experiments.

B Experiments Details

B.1 Full Ablation Study

We have also conducted ablation study with 50 and 200 training samples as shown in Table 9 .

Dataset	#Entity	#Relationship	Max Hops
MetaQA	43,234	9	3
KQA Pro	829,351	1,101	14

Table 8: Statistics of experiment datasets.

B.2 Pre-training on Noisy Pseudo Label

In our experimental settings, we construct pseudo logical forms from annotated logical forms. However, the golden annotation for questions is expensive in practical situations. We attempt to leverage the in-context learning (ICL) capability of LLMs to aid the construction of pseudo logical forms. Due to the limitation of computational resources, we only conduct noisy pseudo logical form construction on the MetaQA dataset, where logical forms are much shorter than that of KQA Pro. We set up an LLM [6] locally for this process. For the ICL instruction, we describe the task and list all nine relationships. Followed by the instruction are several manually annotated questions and pseudo logical form pairs. During the inference, we only change the question appended to the prompt. Token <e> is used to inform the model when to stop, and we discard generation results that do not end with <e>. It takes over 50 hours to obtain noisy pseudo logical forms for questions of the MetaQA train set. The prompt that we use for the construction of noisy pseudo logical forms is shown in Figure 3.

After pre-training on the obtained noisy pseudo labels, the fine-tuned results are listed in Table 13. It is impressive to observe that the proposed pre-training strategy can still effectively enhance the performance of the model in a few-shot setting, even when trained with data containing significant amounts of noise. However, there is one exception, which is the 5-shot Cypher generation. Upon examining the prediction results, we discovered that the majority of errors stem from superfluous whitespace appearing before the generated literal values, for instance, `n2.name = " claude berri"` instead of correct version `n2.name = "claude berri"`. This error may have been caused by the overfitting of certain tokens when the noisy sequences are tokenized.

The prompt we use to construct noisy pseudo logical forms consists of several manually annotated (question, pseudo label) pairs. Since MetaQA involves only nine relationships, we are able to put

[6] https://huggingface.co/baichuan-inc/Baichuan-13B-Chat

Model	#Samples 50			#Samples 100			#Samples 200		
	BLEU-4	ER	Acc	BLEU-4	ER	Acc	BLEU-4	ER	Acc
Ours	**75.48**	71.59	**23.47**	**85.06**	**86.22**	**41.38**	**90.25**	**90.45**	**52.60**
w/o KD	68.40	**77.10**	14.64	73.01	82.00	18.81	75.84	89.03	22.30
w/o PT	71.92	53.36	14.77	81.62	74.79	34.64	89.10	82.90	49.93

Table 9: Full ablation study

```
Extract entities and relations from the sentence, where entity is part of the sentence and relation are restricted to the
following relations: 'pred:name', 'has_imdb_rating', 'directed_by', 'in_language', 'has_genre', 'written_by', 'release_year',
'has_tags', 'has_imdb_votes', 'starred_actors'.
which films are about jacques tati?
['has_tags', 'pred:name', '"jacques tati"'] <e>
the films acted by Sharon Tate were released in which years?
['starred_actors', 'pred:name', '"Sharon Tate"', 'release_year'] <e>
who acted in the movies directed by the director of Some Mother's Son?
['starred_actors', 'directed_by', 'directed_by', 'pred:name', '"Some Mother's Son"'] <e>
what are the languages spoken in the films whose directors also directed Police?
['directed_by', 'directed_by', 'pred:name', '"Police"', 'language'] <e>
$question_to_predict
```

Figure 3: The prompt we have used to construct noisy pseudo logical forms via in-context learning.

Logical Form	1 shot	3 shot	5 shot
SPARQL	73.64 (+2.21)	93.39 (+3.29)	93.42 (+1.9)
Cypher	65.68 (+3.33)	88.15 (+3.92)	87.73 (-4.98)
KoPL	79.64 (+0.88)	93.15 (+13.18)	94.42 (+0.42)
GraphQ_IR	65.24 (+9.72)	93.66 (+3.21)	94.85 (+0.0)

Table 10: Results of noisy pseudo label pre-training on MetaQA. Improvements compared to BART base are in brackets.

Logical Form	#samples 50	#samples 100	#samples 200
Fine-tune from BART base			
SPARQL	60.80	70.76	76.78
Cypher	54.65	60.65	73.32
KoPL	62.43	62.21	76.55
GraphQ_IR	67.60	71.28	79.23
Fine-tune from pseudo pre-trained			
SPARQL	67.46 (+6.66)	74.26 (+3.50)	81.66 (+4.88)
Cypher	61.93 (+7.28)	66.35 (+5.70)	75.87 (+2.55)
KoPL	63.17 (+0.74)	71.02 (+8.81)	79.57 (+3.02)
GraphQ_IR	71.81 (+4.21)	17.89 (+5.36)	79.64 (+0.34)

Table 11: The BLEU-4 scores on KQA Pro.

Logical Form	1 shot	3 shot	5 shot
Fine-tune from BART base			
SPARQL	94.39	98.34	98.88
Cypher	89.81	96.23	98.66
KoPL	95.23	96.30	98.55
GraphQ_IR	88.43	97.05	98.62
Fine-tune from pseudo pre-trained			
SPARQL	95.84 (+1.45)	98.85 (+0.51)	99.15 (+0.27)
Cypher	90.20 (+0.39)	97.78 (+1.55)	98.89 (+0.23)
KoPL	94.83 (-0.40)	98.80 (+2.50)	98.89 (+0.34)
GraphQ_IR	92.11 (+3.68)	98.23 (+1.18)	99.13 (+0.51)

Table 12: The BLEU-4 scores on MetaQA.

them in the prompt. For KGs that contain more relationships, a retrieval module can be employed to retrieve relevant relationships dynamically.

B.3 BLEU Scores across Logical Forms

Here we provide the BLEU-4 scores in addition to EM scores for experiments on MetaQA and KQA Pro. As shown in Table 11, 12 and 13, the BLEU-4 scores have certain improvements across all logical forms on both datasets. Significantly, the progress made on the KQA Pro dataset stands out more prominently because the dataset encompasses a wider range of entities and relationships. Regarding the comparatively basic MetaQA dataset, employing an LLM with 13 billion parameters to autonomously generate pseudo-labels infused with noise still leads to performance enhancements. This demonstrates the ease of applying this methodology, as it requires minimal effort to achieve positive results.

B.4 Efficiency and Privacy

We have evaluated the generation efficiency of our model compared to other large language models, such as LLaMA. Our base version of BART, which has 216 million parameters, demonstrates that even with significantly fewer parameters—over 30 times less than LLaMA 7B's—it can still deliver promising results. On identical hardware equipped with an RTX 3090, our model's inference speed is 36 times faster, taking only 2.5 minutes as opposed to approximately 90 minutes to predict our test set from KQA Pro. Hence, in scenarios requiring the

Logical Form	1 shot	3 shot	5 shot
SPARQL	96.71 (+2.32)	99.10 (+0.8)	99.03 (+0.15)
Cypher	93.80 (+3.99)	98.34 (+2.11)	98.27 (-0.39)
KoPL	95.08 (-0.15)	98.81 (+2.51)	98.93 (+0.38)
GraphQ_IR	90.39 (+1.96)	98.65 (+1.60)	98.21 (-0.41)

Table 13: BLEU-4 score results of noisy pseudo label pre-training on MetaQA. Improvements compared to BART base are in brackets.

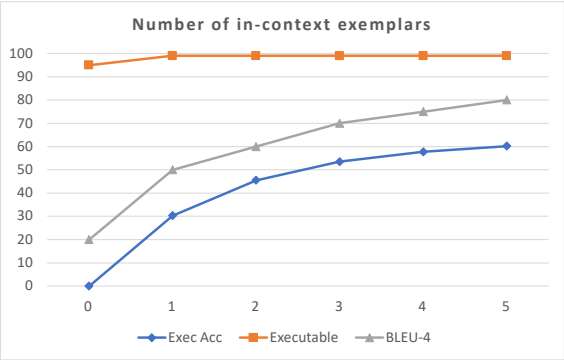

Figure 4: The ICL results of ChatGPT$_{BM25}$ with different exemplars.

generation of logical forms with limited resources, smaller models maintain high performance and efficiency, making them viable technical candidates. Furthermore, while more powerful LLMs like ChatGPT can achieve impressive performance through training-free in-context learning methods, this approach raises data privacy concerns due to the need for processing data in the cloud. The method we propose serves as a good precursor for building low-resource KGQA systems locally.

B.5 More ChatGPT Results

The quantity of exemplars provided in the context is an intuitive factor that affects the final performance for in-context learning. In our research, we retrieved the K most similar samples to the question using the BM25 (Robertson et al., 2009) to form context, where K ranges from 1 to 5. The experimental results are illustrated in Figure 4. As observed, ChatGPT achieved a high executable rate in generating CQL with reference to contextual demonstrations. However, there was a significant gap between the accuracy rate of answers and the executable rate, especially in the zero-shot setting. When no exemplar provided, ChatGPT generates syntax correct but unfaithful CQL, indicating that ChatGPT is familiar with CQL grammar but has little knowledge about the schema of specific KG. Moreover, we note that result in Figure 4 retrieves relevant samples from 24,000 annotated logical forms. Compared to results in Table 1, where only hundreds of data were used to retrieve relevant samples, we can see a significant performance gap (i.e., the decrease of answer accuracy ranging from 19.0 to 26.3 points).