Improving Complex Knowledge Base Question Answering via Question-to-Action and Question-to-Question Alignment

Yechun Tang, Xiaoxia Cheng, Weiming Lu[*]
College of Computer Science and Technology , Zhejiang University
{tangyechun, zjucxx, luwm}@zju.edu.cn

Abstract

Complex knowledge base question answering can be achieved by converting questions into sequences of predefined actions. However, there is a significant semantic and structural gap between natural language and action sequences, which makes this conversion difficult. In this paper, we introduce an alignment-enhanced complex question answering framework, called ALCQA, which mitigates this gap through question-to-action alignment and question-to-question alignment. We train a question rewriting model to align the question and each action, and utilize a pretrained language model to implicitly align the question and KG artifacts. Moreover, considering that similar questions correspond to similar action sequences, we retrieve top-k similar question-answer pairs at the inference stage through question-to-question alignment and propose a novel reward-guided action sequence selection strategy to select from candidate action sequences. We conduct experiments on CQA and WQSP datasets, and the results show that our approach outperforms state-of-the-art methods and obtains a 9.88% improvements in the F1 metric on CQA dataset. Our source code is available at https://github.com/TTTTTTTTy/ALCQA.

1 Introduction

Complex knowledge base question answering (CQA) aims to answer various natural language questions with a large-scale knowledge graph. Compared to simple questions with single or multi-hop of relations, complex questions have more kinds of answer types such as *numeric* or *boolean* types and require more kinds of aggregation operations like *min/max* or *intersection/union* to yield answers. Semantic parsing approaches typically map questions to intermediate logical forms such as query graphs (Yih et al., 2015; Bao et al., 2016; Bhutani et al., 2019; Maheshwari et al., 2019; Lan and Jiang, 2020; Qin et al., 2021), and further transform them into queries like SPARQL query language. Recently, many works (Liang et al., 2017; Saha et al., 2019; Ansari et al., 2019; Hua et al., 2020a,b,c) predefine a collection of functions with constrained argument types and represent the intermediate logical form as a sequence of actions that can be generated using a seq2seq model. Sequence-based methods are natural to accomplish more complex operations by simply expanding the function set, thus making some logically complex questions answerable while they're difficult to answer using query graphs.

The seq2seq model has been widely used and achieved good results on many text generation tasks, such as machine translation, text summarization and style transfer. In these tasks, the source and the target sequence are both natural language texts and thus share some low-level features. However, semantic parsing aims to transform unstructured texts into structured logical forms, which requires a difficult alignment between them. This problem becomes more serious when the complexity of the question rises. Some works propose to solve this problem by modelling the hierarchical structure of logical forms. Dong and Lapata (2016) introduces a sequence-to-tree model with an attention mechanism. Dong and Lapata (2018) proposes to decode a sketch of the logical forms which contain a set of functions at first and then decode low-level details like arguments. Guo et al. (2021) iteratively segments a span from the question by a segmentation model and parses it using a base parser until the whole query is parsed. Li et al. (2021) uses a shift-reduce algorithm to obtain token sequences instead of predicting the start and end positions of the span. However, most of these works require intermediate logical forms or sub-questions to train models, which are usually difficult to obtain. Guo et al. (2021) and Li et al. (2021) propose to pretrain a base parser firstly, and then search good segments

[*] Corresponding author.

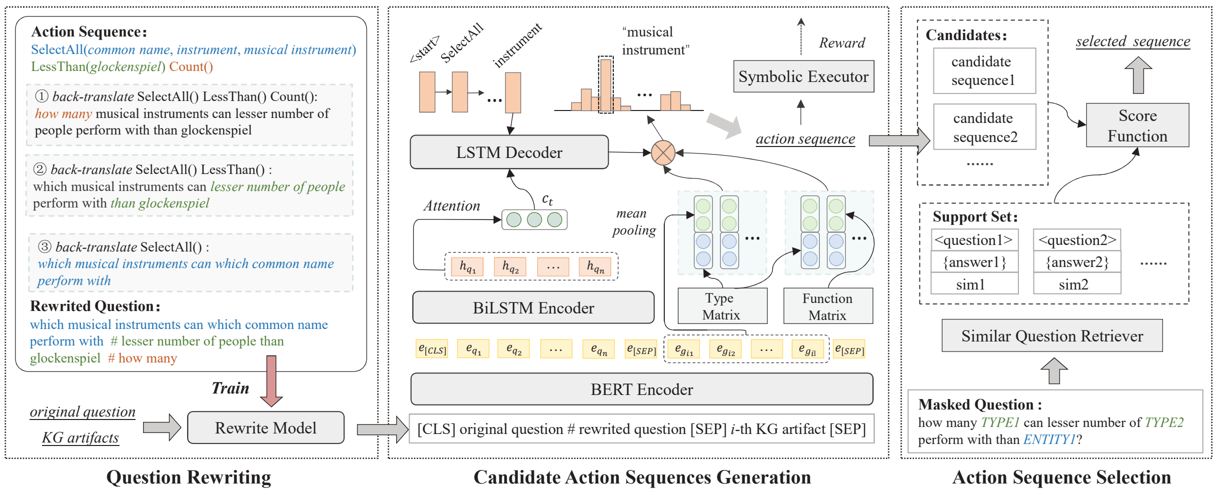

Action Sequence:
SelectAll(common name, instrument, musical instrument)
LessThan(glockenspiel) Count()
① back-translate SelectAll() LessThan() Count(): how many musical instruments can lesser number of people perform with than glockenspiel
② back-translate SelectAll() LessThan() : which musical instruments can lesser number of people perform with than glockenspiel
③ back-translate SelectAll() : which musical instruments can which common name perform with
Rewrited Question:
which musical instruments can which common name perform with # lesser number of people than glockenspiel # how many
Train
original question
KG artifacts
Rewrite Model
Question Rewriting
<start>
SelectAll
instrument
"musical instrument"
Reward
Symbolic Executor
action sequence
LSTM Decoder
Attention
c_t
mean pooling
h_{q_1} h_{q_2} … h_{q_n}
BiLSTM Encoder
Type Matrix
Function Matrix
$e_{[CLS]}$ e_{q_1} e_{q_2} … e_{q_n} $e_{[SEP]}$ $e_{g_{i1}}$ $e_{g_{i2}}$ … $e_{g_{il}}$ $e_{[SEP]}$
BERT Encoder
[CLS] original question # rewrited question [SEP] i-th KG artifact [SEP]
Candidate Action Sequences Generation
Candidates:
candidate sequence1
candidate sequence2
……
selected_sequence
Score Function
Support Set:
<question1> {answer1} sim1
<question2> {answer2} sim2
……
Similar Question Retriever
Masked Question :
how many TYPE1 can lesser number of TYPE2 perform with than ENTITY1?
Action Sequence Selection

Figure 1: An overview of the proposed approach. The question is first converted into a more structured form, then multiple candidate action sequences are generated by the seq2seq model, and finally the candidate action sequences are scored based on similar question-answer pairs.

that predicted sub logical forms are part of or can be composed into the golden meaning representation. They don't necessarily require training pairs but have the limitation that decomposed utterances are continuous segments of the original question.

In this paper, we propose a novel framework to boost the alignment between unstructured text and structured logical forms. We decompose the semantic parsing task into three stages: question rewriting, candidate action sequences generation and action sequence selection. In the question rewriting stage, we utilize a question rewriting model to explicitly transform a query into a set of utterances, each corresponding to a single action, thus reducing the complexity of the question. We propose a two-phase training method to train the rewriting model on the lack of training pairs. In the candidates generation stage, we build a seq2seq model to generate logical forms with beam search algorithm and consider KG artifacts like entities as candidate vocabularies in the decoding stage. To further align the question and action sequence, we concatenate a question and a KG artifact as input and encode it using a pretrained language model (PLM) like BERT (Devlin et al., 2018). The cross attention mechanism of PLM can effectively align between the question and KG artifacts implicitly, which makes decoding easier. Moreover, we innovatively propose to improve complex knowledge base question answering via question-to-question alignment. Motivated by the phenomenon that the more similar two questions are, the more similar their corresponding action sequences will be, we build a memory consisting of question-answer pairs and retrieve a set of question-answer pairs as the support set based on the similarity with the current question during action sequence selection phase. We then propose a reward-guided selection strategy that scores each candidate action sequence according to the support set.

Our main contributions are as follow:

- We propose a novel framework that mitigates the gap between natural language questions and structural logical forms through question-to-action alignment and question-to-question alignment.
- We propose a novel question rewriting mechanism that rewrites a question into a more structured form without requiring a dataset or adding any constraints, and employ a reward-guided action sequence selection strategy that utilizes similar question-answer pairs to score candidate action sequences.
- We conduct experiments on several datasets, and experimental results show that our approach is comparable to the state-of-the-art on WQSP dataset and obtains a 9.88% improvements in the F1 metric on CQA dataset.

2 Methodology

2.1 Overview

In this task, with training set $\mathcal{T} = \{(q_1, a_1), ..., (q_s, a_s)\}$, where (q_i, a_i) is a question-answer pair, the objective is to transform complex questions into logical forms, which can be further derived into KG queries to find answers. We define the logical form as a sequence of actions involving a function and multiple arguments. Following NS-CQA (Hua et al., 2020c), we design 16 functions with arguments comprised of numerical values and KG artifacts including entities, relations, and entity types. We recognize these arguments in the preprocessing step. Denote the input question as q, the set of predefined functions as function set $\mathcal{F}$, question related numerical values and KG artifacts as argument set $\mathcal{G}$, parameters of model as $\boldsymbol{\theta}$, our goal can be normalized as maximizing the probability $P(\mathcal{L} \mid \boldsymbol{q}\,; \boldsymbol{\theta})$, where $\mathcal{L}$ is the action sequence that produces correct answers and each word in $\mathcal{L}$ belongs to $\mathcal{F}$ or $\mathcal{G}$.

As shown in Figure 1, our framework consists of three stages: question rewriting, candidate action sequences generation, and action sequence selection. In the first stage, we rewrite a complex query into a more structured form by a seq2seq model, the details of training the model will be described in 2.2. The rewritten query then can be combined with the original question as input, and a newly seq2seq model is used to generate multiple candidate action sequences sequentially. And finally, We retrieve k question-answer pairs that are most similar to the current question from a pre-constructed memory. The candidates are then modified according to the KG artifacts in these k questions and scored based on the comparison results between the execution results and respective answers, separately.

2.2 Question Rewriting

An action sequence consists of multiple consecutive actions, and it is difficult for the seq2seq model to decide which part of the question to focus on when generating each action. We train a question rewriting model that transform a query into a set of utterances which are concatenated by the symbol "#" and each utterance corresponds to a single action. With the rewritten question, the model can focus on a certain part of the question when generating action in the sequence, thus reducing the difficulty of decoding.

To train the rewriting model, we require an adequate training corpus which is difficult to obtain. On the lack of golden datasets, we propose a two-phase approach to convert queries into rewritten questions and use them for training of the rewriting model as shown in Module 1. In the first phase (line 1-2), we employ a breadth-first search algorithm to find pseudo action sequences for some questions, and then train a seq2seq model that translates an action sequence into a query. In the second phase (line 3-13), we construct a training corpus for question rewriting based on searched question-logical form pairs and the model trained in the previous stage. Specifically, given an action sequence $\mathcal{L} = \{f_1; f_2; ...f_k\}$, we delete the last action f_k, back-translate the shorter action sequence into a new query, and compare it with the original question. We can determine that the tokens which appear in the original question but not in the current generated question are the ones we should most focus on when generating the deleted action. For example, the left part of Figure 1 illustrates the process of decomposing the question "*how many musical instruments can lesser number of people perform with than glockenspiel*". We firstly delete the last action "*Count*()" and then the seq2seq model translates the newly formed sequence "*SelectAll*(...)*LessThan*(...)" into query "*which musical instruments can lesser number of people perform with than glockenspiel*". The words "*how many*" should be paid more attention because they do not appear in the generated question. We

Module 1: Question Rewriting Training

```
Input: $\mathcal{T} = \{(q_1, a_1), ..., (q_n, a_n)\}$
Output: $M_r$, which is the trained model for
        rewriting questions
1  Search pseudo action sequences and obtain
     $\mathcal{T}' = \{(q_1, a_1, \mathcal{L}_1), ..., (q_n, a_n, \mathcal{L}_n)\}$,
     where $\mathcal{L}_i = \{f_1; f_2; ...f_k\}$ is the pseudo
     action sequence of $(q_i, a_i)$;
2  Train $M_q$ which transforms action
     sequences into questions using $\mathcal{T}'$;
3  $\mathcal{Q} \leftarrow \{\}$;
4  for $(q_i, \mathcal{L}_i)$ in $\mathcal{T}'$ do
5      $q_{ori} \leftarrow q_i$ ;
6      for $j \in [k, 1]$ do
7          $\mathcal{L}' \leftarrow \{f_1; f_2; ...; f_{j-1}\}$;
8          $q_{del} \leftarrow \mathrm{Translate}(\mathcal{L}', M_q)$;
9          $q'_{ij} \leftarrow \mathrm{Compare}(q_{ori}, q_{del})$;
10         $q_{ori} \leftarrow q_{del}$ ;
11     end
12     $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{q_i, \{q'_{i1}; q'_{i2}; ...; q'_{ik}\}\}$ ;
13 end
14 Train $M_r$ using $\mathcal{Q}$;
```

iteratively perform *delete*, *back-translate* and *compare* operations until the action sequence is empty and concatenate the compare results of each step using symbol "#".

Thus, we can construct the question rewriting dataset $\mathcal{Q}$ and train a question rewriting model M_r. To make the rewriting model learn to output KG artifacts in the rewritten query, we concatenate the original question and KG artifacts as input, and wrap KG artifacts with symbols like $\langle\text{entity}\rangle$ and $\langle/\text{entity}\rangle$. We initialize models in both phases using BART (Lewis et al., 2020), an outstanding pretrained seq2seq model that demonstrates high performance on a wide range of generation tasks, and finetune them by constructed datasets.

2.3 Encoder-decoder Architecture

We use BERT and BiLSTM (Hochreiter and Schmidhuber, 1997) to construct the encoder. Given a question q with n tokens and the argument set $\mathcal{G} = \{g_1, ..., g_m\}$, where m is the size of argument set with respect to q and $g_i = \{g_{i1}, ..., g_{il}\}$ is a KG artifact or numerical value with l tokens, we concatenate the question and each argument separately using [SEP] as the delimiter to construct BERT input sequences. In this case, we obtain question embedding $E_q \in \mathbb{R}^{n\times d_e}$, and argument embedding $\vec{g}_i \in \mathbb{R}^{d_e}$ by mean pooling over E_{g_i}. We then stack embeddings of arguments to construct a matrix $E_{\mathcal{G}} \in \mathbb{R}^{m\times d_e}$ and feed E_q into a BiLSTM encoder to obtain the final question representation $H \in \mathbb{R}^{n\times d_h}$.

$$\begin{aligned} E &= \text{BERT}(\{[\text{CLS}], q, [\text{SEP}], g_i, [\text{SEP}]\}) \\ H &= \text{BiLSTM}(E_q) \\ \vec{g}_i &= \text{MeanPooling}(E_{g_i}) \end{aligned} \tag{1}$$

Decoding is implemented using LSTM, and at each time step, the current hidden state $s_t \in \mathbb{R}^{d_s}$ is updated based on the hidden state and output of the previous time step as follows:

$$\begin{aligned} s_t &= \text{LSTM}([o_{t-1}; \tau_{t-1}; c_t], s_{t-1}) \\ c_t &= \sum_i \alpha_{ti} h_i \\ \alpha_t &= \text{Softmax}(e_t) \\ e_t &= s_{t-1} W_a H^T \end{aligned} \tag{2}$$

where [;] denotes vector concatenation. o_{t-1} is the embedding of output in the last step which obtains from learnable embedding matrix W_{func} if the output is a function or from $E_{\mathcal{G}}$ if the output is an argument. τ_{t-1} is a vector that obtains from learnable embedding matrix W_{type} according to the type of last output. c_t is the context vector resulting from the weighted summation of h_i, the *i*-th row of the question embedding H, based on the attention mechanism. $W_a \in \mathbb{R}^{d_s\times d_h}$ is a projection matrix.

We then calculate the vocabulary distribution based on hidden state s_t. Our vocabulary consists of two parts, a fixed vocabulary containing a collection of predefined functions and a dynamic vocabulary consisting of arguments, i.e., numerical values and KG artifacts related to the question. We feed s_t through one linear layer W_o and a softmax function to compute the probability of each word in the fixed vocabulary. To obtain the probabilities of the words in the dynamic vocabulary, we project the hidden vectors s_t to the same dimension through the projection matrix $W_p \in \mathbb{R}^{d_s\times d_e}$ and then compute the similarities with each word by taking the dot product.

$$\begin{aligned} P_{fix} &= \text{Softmax}(W_o s_t) \\ P_{dyn} &= \text{Softmax}(s_t W_p E_{\mathcal{G}}^T) \end{aligned} \tag{3}$$

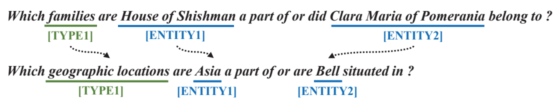
Next, we calculate the probability P_t that generate from the fixed vocabulary at the current time step through a linear layer followed by the activation function, and combine the two vocabulary distributions based on P_t. Note that if w is a word in fixed vocabulary, then $P_{dyn}(w)$ is zero; similarly $P_{fix}(w)$ is zero when w is in dynamic vocabulary.

$$\begin{aligned} P(w) &= P_t P_{fix}(w) + (1 - P_t) P_{dyn}(w) \\ P_t &= \sigma(W_f c_t) \end{aligned} \tag{4}$$

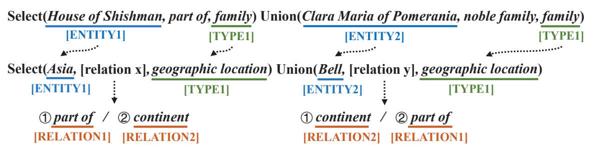
2.4 Reward-guided Action Sequence Selection Strategy

To improve accuracy, we generate multiple candidate action sequences with beam search algorithm and design a reward-guided action sequence selection strategy. In general, the more similar the structure and semantics of the two questions are, the more similar their corresponding action sequences will be. Therefore, we propose that similar questions can be used to help the selection of correct action sequence. Specifically, we build a memory consisting of question-answer pairs in the training set. Note that we don't require golden logical forms of these questions.

To retrieve similar questions with answers from memory, we use edit distance to calculate the similarity between two questions. To improve the generalization of the questions, we replace the entity mentions, type mentions and numerical values in the questions with the symbol [ENTITY], [TYPE] and [CONSTANT], respectively. We don't mask relations because it is always hard to recognize relation mentions. In addition, the presence of some antonyms including *atmost* and *atleast*, *less* and *greater*, can lead to the exact opposite semantics of questions with similar contexts. Therefore, we construct a set of antonym pairs and set the similarity to 0 when there is an antonym pair in the two questions. We retrieve k question-answer pairs with the highest similarity to form the support set $S = \{\{q_1, a_1, d_1\}, ..., \{q_k, a_k, d_k\}\}$, where d_i is the similarity computed by edit distance.


Which families are House of Shishman a part of or did Clara Maria of Pomerania belong to ?
[TYPE1]
[ENTITY1]
[ENTITY2]
Which geographic locations are Asia a part of or are Bell situated in ?
[TYPE1]
[ENTITY1]
[ENTITY2]

(a) question-to-question alignment


Select(House of Shishman, part of, family) Union(Clara Maria of Pomerania, noble family, family)
[ENTITY1]
[TYPE1]
[ENTITY2]
[TYPE1]
Select(Asia, [relation x], geographic location) Union(Bell, [relation y], geographic location)
[ENTITY1]
[TYPE1]
[ENTITY2]
[TYPE1]
① part of / ② continent
[RELATION1] [RELATION2]
① continent / ② part of
[RELATION2] [RELATION1]

(b) action-to-action alignment

Figure 2: An example of adjusting candidate action sequences. The upper and lower parts of (a) are the original question and a question in the support set, respectively. We first obtain a relation-masked action sequence (the second line of (b)) based on the alignment results of entities and types between two questions as shown in (a), and then output multiple action sequences according to all possible combinations of relations.

We then propose a reward-guided action sequence selection strategy that scores each candidate action sequence according to its fitness to the retrieved support set. Specifically, given a candidate $\mathcal{A}_i$ and an item $\{q_j, a_j, d_j\}$ in the support set, we adjust the arguments in $\mathcal{A}_i$ to arguments of q_j according to their positions in the text as Figure 2, and then score it by compute F1 scores between a_j and execution results of modified sequences on the lack of golden action sequences. Due to the positions of the relations being unknown, we obtain all possible orders of relations and generate multiple modified action sequences. We then take the highest F1 as the score of $\{q_j, a_j, d_j\}$ to $\mathcal{A}_i$ and denote it as r_i^j. The overall score of $\mathcal{A}_i$ then can be calculated as follows:

$$s_i = \frac{\sum_{j=1}^{k} d_j r_i^j}{\sum_{j=1}^{k} d_j} \tag{5}$$

where $\sum_{j=1}^{k} d_j$ is a normalized term. We take the candidate action sequence with the highest score as the output sequence in the inference stage.

2.5 Training

We use REINFORCE (Williams, 1992) algorithm to train our model. We view F1 scores of the answers generated by predicted action sequence with respect to ground-truth answers as original rewards. To improve the stability of training, we use the adaptive reward function (Hua et al., 2020c) to adjust rewards. Moreover, we use a breadth-first search algorithm on a subset of data to obtain pseudo-action sequences and pretrain the model to prevent the cold start problem.

3 Experiments

3.1 Experimental Setup

Our method aims to solve various complex questions, and we mainly evaluate it on ComplexQuestionAnswering (CQA) (Saha et al., 2018) dataset which is a large-scale KBQA dataset containing seven types of complex questions, as shown in Table 1. We show the details and some examples of this dataset in Appendix A. We also conduct experiments on WebQuestionsSP (WQSP) (Yih et al., 2015) which contains 4737 simple questions. The results show that our method also works well on simple datasets.

We employ standard F1-measure between predicted entity set and ground truth answers as evaluation metrics. For some categories whose answers are boolean values or numbers on CQA dataset, we view answers as single-value sets and compute the corresponding F1 scores. The training details and model parameters can be found in Appendix B

3.2 Baselines

We compare our framework with seq2seq based methods. KVmem (Saha et al., 2018) presents a model consisting of a hierarchical encoder and a key value memory network. CIPITR (Saha et al., 2019) proposes to mitigate reward sparsity with auxiliary rewards and restricts the program space to semantically correct programs. CIPITR proposes two training ways, one training a single model for

Question Category	**KVmem**	**CIP-All**	**CIP-Sep**	**NSM**	**MRL-CQA**	**MARL**	**NS-CQA**	**Ours**
Simple Question	41.40%	41.62%	**94.89%**	88.33%	88.37%	88.06%	<u>88.83%</u>	88.73%
Logical Reasoning	37.56%	21.31%	<u>85.33%</u>	81.20%	80.27%	79.43%	81.23%	**88.73%**
Quantitative Reasoning	0.89%	5.65%	33.27%	41.89%	45.06%	49.93%	<u>56.28%</u>	**76.30%**
Comparative Reasoning	1.63%	1.67%	9.60%	64.06%	62.09%	64.10%	<u>65.87%</u>	**83.09%**
Verification (Boolean)	27.28%	30.86%	61.39%	60.38%	85.62%	<u>85.83%</u>	84.66%	**88.18%**
Quantitative (Count)	17.80%	37.23%	48.40%	61.84%	62.00%	60.89%	<u>76.96%</u>	**80.41%**
Comparative (Count)	9.60%	0.36%	0.99%	39.00%	40.33%	40.50%	<u>43.25%</u>	**60.80%**
Overall macro F1	19.45%	19.82%	47.70%	62.39%	66.25%	66.96%	<u>71.01%</u>	**80.89%**
Overall micro F1	31.18%	31.52%	73.31%	76.01%	77.71%	77.71%	<u>80.80%</u>	**85.31%**

Table 1: The overall performances on CQA dataset. Best results are bolded for each category and second-best results are underlined.

all question categories, denoted by CIP-ALL, and the other training a separate model for each category, denoted by CIP-SEP. NSM (Liang et al., 2017) utilizes a key-variable memory to handle compositionality and helps find good programs by pruning the search space. MRL-CQA (Hua et al., 2020a) and MARL (Hua et al., 2020b) propose meta-reinforcement learning approaches that effectively adapts the meta-learned programmer to new questions to tackle potential distributional biases, where the former uses an unsupervised retrieval model and the latter learns it alternately with the programmer from weak supervision. NS-CQA (Hua et al., 2020c) presents a memory buffer that stores high-reward programs and proposes an adaptive reward function to improve training performance. SSRP (Ansari et al., 2019) presents a noise-resilient model that is distant-supervised by the final answer. CBR-KBQA (Das et al., 2021) generates complex logical forms conditioned on similar retrieved questions and their logical forms to generalize to unseen relations.

We also compare our method with graph-based methods on WQSP dataset. STAGG (Yih et al., 2015) proposes a staged query graph generation framework and leverages the knowledge base in an early stage to prune the search space. TEXTRAY (Bhutani et al., 2019) answers complex questions using a novel decompose-execute-join approach. QGG (Lan and Jiang, 2020) modifies STAGG with more flexible ways to handle constraints and multi-hop relations. OQGG (Qin et al., 2021) starts with the entire knowledge base and gradually shrinks it to the desired query graph.

3.3 Overall Performances

The overall performances of our proposed framework against KBQA baselines are shown in Table 1 and 2. Our framework significantly outperforms the state-of-the-art model on CQA dataset while staying competitive on WQSP dataset. On CQA dataset, our method achieves the best overall performance of 80.89% and 85.31% in macro and micro F1 with 9.88% and 4.51% improvement, respectively. Moreover, it can be observed that our method achieves the best result on six of seven question categories. On *Logical Reasoning* and *Verification (Boolean)*, which are relatively simpler, our model obtain a 3.40% and 2.35% improvement in macro F1, respectively. On *Quantitative Reasoning*, *Comparative Reasoning*, *Quantitative (Count)* and *Comparative (Count)*, whose questions are complex and hard to parse, out model obtain a considerable improvement. To be specific, the macro F1 scores increase by 20.02%, 17.22%, 3.45% and 17.55%, respectively. Our proposed method doesn't outperform CIP-Sep on *Simple Question* which trains a separate model on this category but still achieves a comparable result with the second-best baseline. On WQSP dataset, our method outperforms all the sequence-based methods and stay competitive with the graph-based method which having the best results. Our method doesn't gain a lot because most questions in this dataset are one hop and simple enough while our frameword aims to deal with various question categories. We don't compare with graph-based methods on CQA dataset because they always start from a topic entity and interact with KG to add relations into query graphs step by step, which can not solve most question types like *Quantitative Reasoning* and *Comparative Reasoning* in this dataset.

The experimental results demonstrate the ability of our method to parse complex questions and generate correct action sequences. The main improvement of the proposed method comes from two as-

Method	F1
NSM	69.0%
SSRP	72.6%
NS-CQA	72.0%
CBR-KBQA†	72.8%
STAGG	66.8%
TEXTRAY	60.3%
QGG	**74.0%**
OQGG	66.0%
Ours	73.6%

Table 2: The overall performances on WQSP dataset. † denotes supervised training.

pects. On the one hand, we employ a rewrite model to decompose a complex question into several utterances, allowing the decoder to focus on a shorter part when decoding each action. On the other hand, we make full use of existing question-answer pairs and determine the structure of action sequences indirectly through the alignment between question-question pairs.

3.4 Ablation Studies

We conduct a series of ablation studies on CQA dataset to demonstrate the effectiveness of the main modules in our framework. To explore the impact of question rewriting module, we remove it and only use the original question as input of the seq2seq model. The performance drops by 1.79% in macro F1 as shown in Table 3. To prove the effectiveness of the action sequence selection module, we generate candidate action sequences using beam search mechanism and directly use the action sequence with the highest probability as the output instead of selecting by action sequence selection module. The macro F1 drops by 2.49% after removing this module. To verify that the cross-attention mechanism in BERT can lead to alignment between question and KG artifacts and further improve the generation result, we encode question and KG artifacts separately and find the performance drops by 0.97%. Experimental results show that every main module in our framework has an important role in performance improvement.

Settings	macro F1	micro F1
Full Model	80.89%	85.31%
w/o question rewriting	79.10%	84.15%
w/o candidates selection	78.40%	83.55%
w/o cross-attention	79.92%	84.63%

Table 3: Ablation studies on main components.

To explore the impact of employing different underlying embeddings, we conduct experiments on two settings, initializing an embedding matrix randomly and encoding with BERT. We finetune the embedding matrix during the training stage in the first setting while freezing the parameters of the BERT model. As shown in Table 4, BERT embedding achieves the best result and improves by 4.40% compared to random embedding. It is reasonable because BERT is pretrained with a large corpus to represent rich semantics and uses a cross-attention mechanism to align the question and KG artifacts better. Note that our proposed method still outperforms state-of-the-art methods without using BERT.

Settings	macro F1	micro F1
Random Embedding	76.49%	81.63%
BERT Embedding	80.89%	85.31%

Table 4: Ablation studies for different underlying embeddings.

To investigate the effect of the number of candidate action sequences and the size of the support set on the selection of action sequences, we conduct experiments and plot the results in Figure 3. It can be observed that the macro F1 score increases with the size of the support set at the beginning, whatever the number of candidates. This trend slows down gradually and the macro F1 score peaks when the size is about 6. Then, as the size of the support set continues to increase, the macro F1 score decreases slightly. It's mainly caused by the simple and rough method we use to calculate the question similarity, which leads to the assumption that similar questions have similar action sequence structures not always hold. In contrast, a certain number of similar questions can alleviate this problem and improve performance. However, when the number reaches a certain level, the newly added questions become less similar to the original questions and introduce noise instead. In addition, the increase in the number of generated candidates also improves performance. If the number is too high, this boost becomes less apparent or even negative because of the lower quality of the newly added candidates.

3.5 Case study

We show some examples to illustrate the ability of our modules. Table 5 shows a complex question of category *Quantitative (Count)*. We can observe that

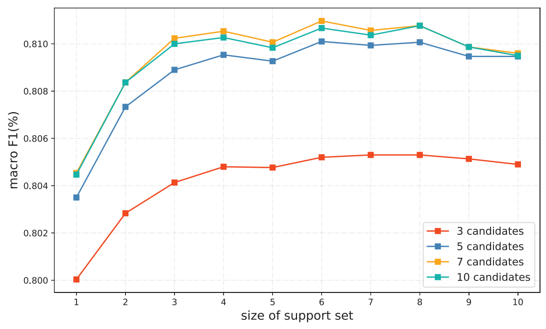


Figure 3: Trends of macro F1 when the size of support set increases.

Question	how many works of art feature approximately 5 fictional taxons or people
Rewritten Question	which works of art contain which fictional taxon # and which common name # approximately 5 people # how many
w/o Module	SelectAll(*fictional taxon*, *present in work*, *work of art*) SelectAll(*common name*, *present in work*, *work of art*) AtLeast(*5*) Count()
w/ Module	SelectAll(*fictional taxon*, *present in work*, *work of art*) SelectAll(*common name*, *present in work*, *work of art*) Around(*5*) Count()

Table 5: Test case on quesiton rewriting module

Question	Does Janko Kroner have location of birth at Peraia, Pella and Povazska Bystrica ?
w/o Module	Select(*Janko Kroner*, *place of birth*, *administrative territorial entity*) # Bool(*Povazska Bystrica*) # Bool(*Povazska Bystrica*) **Prob:** 0.6085 **Selection Score:** 0.7333
w/ Module	Select(*Janko Kroner*, *place of birth*, *administrative territorial entity*) # Bool(*Peraia, Pella*) # Bool(*Povazska Bystrica*) **Prob:** 0.3519 **Selection Score:** 1.0000

Table 6: Test case on action sequence selection module

the model wrongly predicts the third action in the absence of rewriting module but makes a correct generation with the help of rewritten utterances. It's reasonable because the seq2seq model learns to focus on "approximately 5 people" when predicting the third action. Table 6 shows a query of category *Verification (Boolean)*. It's confusing for the model to decide which entity to output, and the correct action sequence is given a lower probability. However, it's much easier to choose through action sequence selection module. The wrong logical form produces an incorrect result in the majority of cases and thus receives a lower selection score, as shown.

4 Related Work

Semantic parsing is the task of translating natural language utterances into executable meaning representations. Recent semantic parsing based KBQA methods can be categorized as graph-based (Yih et al., 2015; Bao et al., 2016; Bhutani et al., 2019; Lan and Jiang, 2020; Qin et al., 2021) and sequence-based (Liang et al., 2017; Saha et al., 2019; Ansari et al., 2019; Hua et al., 2020a,b,c; Das et al., 2021). Graph-based methods build a query graph which is a graph-like logical form proposed by (Yih et al., 2015). (Bao et al., 2016) proposed multi-constraint query graph to improve performance. Ding et al. (2019) and Bhutani et al. (2019) decomposed complex query graph into a set of simple queries to overcome the long-tail problem. (Lan and Jiang, 2020) employed early incorporation of constraints to prune the search space. (Chen et al., 2021) leveraged the query structure to constrain the generation of the candidate queries. (Qin et al., 2021) generated query graph by shrinking the entire knowledge base. Sequence-based methods define a set of functions and utilize a seq2seq model to generate action sequences. Liang et al. (2017) augmented the standard seq2seq model with a key-variable memory to save and reuse intermediate execution results. Saha et al. (2019) mitigated reward sparsity with auxiliary rewards. Ansari et al. (2019) learned program induction with much noise in the query annotation. Hua et al. (2020a,b) employed meta-learning to adapt programmer to unseen questions quickly. Hua et al. (2020c) proposed a adaptive reward function to control the exploration-exploitation trade-off in reinforcement learning.

Compared to graph-based methods, sequence-based methods can generate logical forms directly using the seq2seq model, which is easier to implement and can handle more question categories by simply expanding the set of action functions. However, the semantic and structural gap between natural language utterances and action sequences leads to poor performance on translation.

5 Conclusion

In this paper, we propose an alignment-enhanced complex question answering framework, which reduces the semantic and structural gap between question and action sequence by question-to-action and question-to-question alignment. We train a question rewriting model to align question and sub-action sequence in the absence of training data and employ a pretrained language model to align the question and action arguments implicitly. Moreover, we utilize similar questions to help select the correct action sequence from multiple candidates. Experiments show that our framework achieves state-of-the-art on the CQA dataset and performs well on various complex question categories. In the future, how to better align questions with logical forms will be considered.

Limitations

In our method, we view KG artifacts as tokens and generate logical forms using a seq2seq model, which can handle more types of complex questions, i.e., superlative quesions without topic entities. However, for single and multi-hop questions, graph-based methods may gain better performance. The reason is that they start from a topic entity and interact with KG to add relations into query graphs step by step, which can prune the search space more effectively. Moreover, we control the vocabulary size through entity and relation recognition, which makes the preprocessing step more complex.

Acknowledgement

This work is supported by the National Key Research and Development Project of China (No. 2018AAA0101900), the Key Research and Development Program of Zhejiang Province, China (No. 2021C01013), CKCEST, and MOE Engineering Research Center of Digital Library.

References

Ghulam Ahmed Ansari, Amrita Saha, Vishwajeet Kumar, Mohan Bhambhani, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2019. Neural program induction for kbqa without gold programs or query annotations. In *IJCAI*, pages 4890–4896.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514.

Nikita Bhutani, Xinyi Zheng, and HV Jagadish. 2019. Learning to answer complex questions over knowledge bases with query composition. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 739–748.

Yongrui Chen, Huiying Li, Yuncheng Hua, and Guilin Qi. 2021. Formal query building with query structure prediction for complex question answering over knowledge base. *arXiv preprint arXiv:2109.03614*.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jiwei Ding, Wei Hu, Qixin Xu, and Yuzhong Qu. 2019. Leveraging frequent query substructures to generate formal queries for complex question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2614–2622.

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.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2021. Iterative utterance segmentation for neural semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12937–12945.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yuncheng Hua, Yuan-Fang Li, Gholamreza Haffari, Guilin Qi, and Tongtong Wu. 2020a. Few-shot complex knowledge base question answering via meta reinforcement learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5827–5837.

Yuncheng Hua, Yuan-Fang Li, Gholamreza Haffari, Guilin Qi, and Wei Wu. 2020b. Retrieve, program, repeat: Complex knowledge base question answering via alternate meta-learning. In *IJCAI*.

Yuncheng Hua, Yuan-Fang Li, Guilin Qi, Wei Wu, Jingyao Zhang, and Daiqing Qi. 2020c. Less is more: Data-efficient complex question answering over knowledge bases. *Journal of Web Semantics*, 65:100612.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. 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.

Yuntao Li, Bei Chen, Qian Liu, Yan Gao, Jian-Guang Lou, Yan Zhang, and Dongmei Zhang. 2021. Keep the structure: A latent shift-reduce parser for semantic parsing. In *IJCAI*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33.

Gaurav Maheshwari, Priyansh Trivedi, Denis Lukovnikov, Nilesh Chakraborty, Asja Fischer, and Jens Lehmann. 2019. Learning to rank query graphs for complex question answering over knowledge graphs. In *International semantic web conference*, pages 487–504. Springer.

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.

Amrita Saha, Ghulam Ahmed Ansari, Abhishek Laddha, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2019. Complex program induction for querying knowledge bases in the absence of gold programs. *Transactions of the Association for Computational Linguistics*, 7:185–200.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

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.

A CQA Dataset

Complex Question Answering(CQA) dataset contains the subset of the QA pairs from the Complex Sequential Question Answering(CSQA) dataset, where the questions are answerable without needing the previous dialog context. There are 944K, 100K and 156K question-answer pairs in the training, validating and test set, respectively. This dataset has seven types of complex questions, making it difficult for the model to answer correctly. We show some examples of each question category in Table 7. For simple questions, the corresponding action sequence contains only one action, and for some complex questions, the length of action sequence may up to 4.

B Training Details

To compare with previous works and reduce training time, we also randomly select two small subsets(about 1% each) from the training set to train models. We use BFS algorithm to search pseudo action sequences for the first subset to train the question rewriting model as introduced in 2.2 and pretrain the action sequence generation model. We use the second one for subsequent reinforcement learning of the action sequence generation model. We evaluate our trained model on the whole test set.

We initialize two models in the question rewriting stage with the base version of BART and fine-tune them using Adam Optimizer with a learning rate of 1e-5. For the action sequence generation model, we adopt the uncased base version of BERT for underlying embeddings and freeze the parameters to improve training stability. We set the dimension of type embedding to 100, the hidden sizes of one-layer BiLSTM Encoder and LSTM Decoder to 300. We train the model for 100 epochs and 50 epochs using Adam with learning rates of 1e-4 and 1e-5 in the pretraining and reinforcement learning stages, respectively, and finally choose the checkpoint with the highest reward in the development set. We generate 5 candidate action sequences with a beam size of 10, and retrieve 3 questions with a similarity greater than threshold 0.6 as the support set. If no similar question meets the condition, we

Question Category		Example Question
Simple Question (599K)	Direct	Where did the expiration of Brian Hetherston occur ?
Logical Reasoning (138K)	union	Which people were casted in Cab Number 13 or Hearts of Fire ?
	Intersection	Who have location of birth at Lourdes and the gender as male ?
	Difference	Which people are a native of Grenada but not United Kingdom ?
Verification (63K)	Boolean	Is United Kingdom headed by Jonas Spelveris and Georgius Sebastos ?
Quantitative Reasoning (118K)	Min/Max	Who had an influence on max number of bands and musical ensembles ?
	Atleast/Atmost	Which applications are manufactured by atleast 1 business organizations and business enterprises ?
	exactly/around *n*	Which films had their voice dubbing done by exactly 20 people ?
Comparative Reasoning (62K)	Less/More/Equal	Which positions preside the jurisdiction over more number of administrative territories and US administrative territories than Minister for Regional Development ?
Quantitative Reasoning (Count) (159K)	Direct	How many nucleic acid sequences encodes Dynein light chain 1, cytoplasmic ?
	Union	How many system software or operating systems are the computing platforms for which Street Fighter IV were specifically designed ?
	Intersection	How many people studied at Harvard University and Ecole nationale superieure des Beaux-Arts ?
	exactly/around *n*	How many musical instruments are played by atmost 7998 people ?
Comparative Reasoning (Count) (63K)	Less/More/Equal	How many administrative territories have less number of cities and mythological Greek characters as their toponym than Bagdad ?

Table 7: The examples of various question types on CQA dataset.

directly select the top one action sequence generated by beam search as output.