Knowledge-augmented Self-training of A Question Rewriter for Conversational Knowledge Base Question Answering

Xirui Ke[1, 2], Jing Zhang[1, 2]*, Xin Lv[3], Yiqi Xu[1, 2], Shulin Cao[3], Cuiping Li[1, 2], Hong Chen[1, 2], Juanzi Li[3]
[1]School of Information, Renmin University of China, Beijing, China
[2]Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education
[3]Department of Computer Science and Technology, Tsinghua University, Beijing, China
{kexirui, zhang-jing, xuyiqi, licuiping, chong}@ruc.edu.cn
{lv-x18, caosl19}@mails.tsinghua.edu.cn, {lijuanzi}@tsinghua.edu.cn

Abstract

The recent rise of conversational applications such as online customer service systems and intelligent personal assistants has promoted the development of conversational knowledge base question answering (ConvKBQA). Different from the traditional single-turn KBQA, ConvKBQA usually explores multi-turn questions around a topic, where ellipsis and coreference pose great challenges to the single-turn KBQA systems which require self-contained questions. In this paper, we propose a rewrite-and-reason framework to first produce a full-fledged rewritten question based on the conversation history and then reason the answer by existing single-turn KBQA models. To overcome the absence of the rewritten supervision signals, we introduce a knowledge-augmented self-training mechanism to transfer the question rewriter from another dataset to adapt to the current knowledge base. Our question rewriter is decoupled from the subsequent QA process, which makes it easy to be united with either retrieval-based or semantic parsing-based KBQA models. Experiment results demonstrate the effectiveness of our method and a new state-of-the-art result is achieved. The code and dataset are available online now[1].

1 Introduction

Knowledge Base Question Answering (KBQA) has been a recent surge of research interest, due to the appearance of large-scale knowledge bases (KBs), such as Wikidata (Vrandečić and Krötzsch, 2014), Freebase (Bollacker et al., 2008), as well as some domain KBs such as the Amazon product KB (Dong et al., 2020) and the academic KB (Tang et al., 2008). KBQA provides users an easier way to seek for factual knowledge in the natural language in spite of KBs' underlying structures. However, instead of the single-turn KBQA requirement, people tend to start from a topic and explore it with follow-up questions in a conversational manner. Especially, driven by the rise of practical conversational applications such as online customer service system and intelligent personal assistant, Conversational KBQA (ConvKBQA) has been attracting more attention.

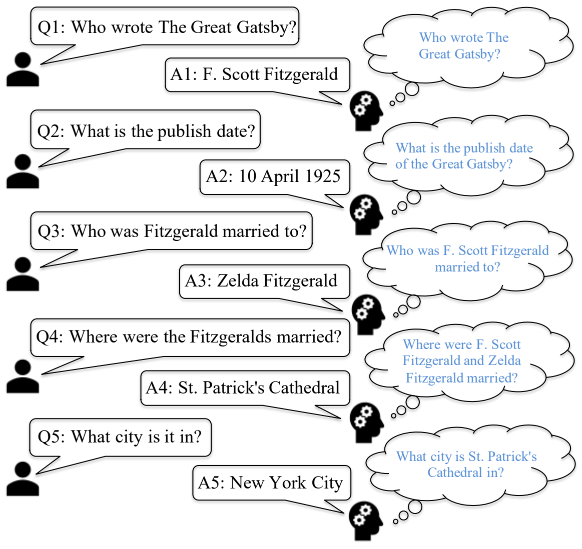

Q1: Who wrote The Great Gatsby?
Who wrote The Great Gatsby?
A1: F. Scott Fitzgerald
Q2: What is the publish date?
What is the publish date of the Great Gatsby?
A2: 10 April 1925
Q3: Who was Fitzgerald married to?
Who was F. Scott Fitzgerald married to?
A3: Zelda Fitzgerald
Q4: Where were the Fitzgeralds married?
Where were F. Scott Fitzgerald and Zelda Fitzgerald married?
A4: St. Patrick's Cathedral
Q5: What city is it in?
What city is St. Patrick's Cathedral in?
A5: New York City

Figure 1: An example of Conversational KBQA.

ConvKBQA poses great challenges for existing QA systems, which target to process a single self-contained question once a time. Because, in a conversational setting, the follow-up questions are usually incomplete with missing entities, which is referenced as ellipsis and coreference phenomenon.

Figure 1 shows an example of ConvKBQA, where the initial question is usually full-fledged mentioning clear topic entities, from which the conversation often drifts away. In this example, it revolves around three subjects alternately: the book, its author and the cathedral. Question Q2 that omits the title of the book is an instance of entity ellipsis. Q3-Q5 have entity coreference phenomenon

*Corresponding author.

[1]https://github.com/RUCKBReasoning/QuestionRewriter ConvKBQA

because "Fitzgerald" in Q3 and "Fitzgeralds" in Q4 refer to "F. Scott Fitzgerald" and the couple respectively. To facilitate such conversation, we can replace or supplement with the entities they refer to or leave out to make them self-contained. That is to say, the follow-up questions can be rewritten as shown on the right in Figure 1.

Existing works leverage the conversation history as well as the underlying KB to overcome the above ellipsis and coreference problem. Similar to the single-turn KBQA (Lan et al., 2021), the ConvKBQA methods can be categorized into the retrieval-based (Kaiser et al., 2021; Lan and Jiang, 2021) and semantic parsing (SP)-based ones (Kacupaj et al., 2021; Marion et al., 2021; Plepi et al., 2021). The former ones first identify the topic entities for each turn's question, and then expand a subgraph for answer reasoning. They usually maintain a gradually growing topic entity candidate set during the conversation, but take the risk of redundant and noisy candidates as well as error propagation caused by the identified candidates of previous turns. The latter ones parse each turn's question into an executable logic form, but suffer from the lack of the annotated logic forms and the incompleteness of the questions. They usually augment the input with conversation history and extra KB information, but the long textural input and its distribution difference with the logical output increase the difficulty of parsing.

This paper proposes rewrite-and-reason, a different pipeline framework by first rewriting the incomplete question into a self-contained one and then adopting any single-turn KBQA model. Compared with the existing retrieval-based methods, the rewriter in the first step of the framework does not reply on entity linking, and the rewritten questions of previous turns do not affect the next turn, which reduces error propagation. Besides, the rewriter can maintain the original input format (*i.e.*, the natural language question) to further enable SP-based single-turn KBQA models in the second step. Compared with the SP-based ConvKBQA models that directly parse an incomplete question, such decoupled rewrite-and-reason (rewrite-and-parse if the SP-based model is used for reasoning) can reduce the parsing difficulty.

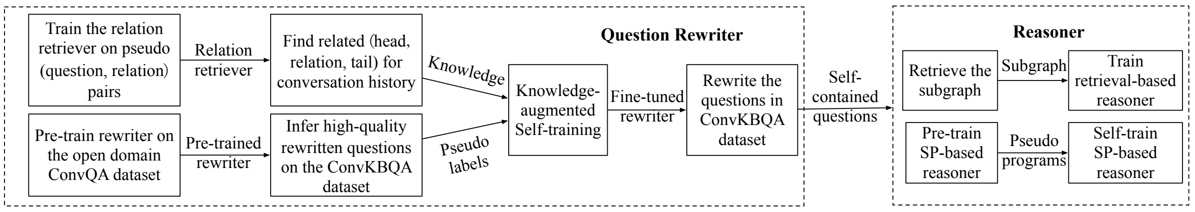
Since the annotations of the rewritten questions are unavailable in the target ConvKBQA dataset, we take advantage of existing annotations in other conversational QA (ConvQA) datasets as the inherent regular patterns of ellipsis and coreference are similar in conversation. However, the conversation styles are quite different that ConvQA is more chatty and verbose while ConvKBQA is more knowledge-centric and concise. Even worse, when facing the entities and relations not involved before, the rewriter might be at a loss. So, we further fine-tune it by knowledge-augmented self-training, where the augmented knowledge exposes the current underlying KB to the rewriter while the self-training process tries to adapt the rewriter to the current conversation style. The idea can also be applied to using other annotated datasets for training the subsequent SP-based KBQA models.

We conduct extensive experiments on ConvQuestions (Christmann et al., 2019), a ConvKBQA dataset grounded in Wikidata. The results reveal three major advantages: (1) The question rewriter, equipped with a retrieval-based reasoner NSM (He et al., 2021) or a SP-based reasoner KoPL (Cao et al., 2022) both achieves significant gains (+11.6% and +3.1% Hits@1 respectively) and creates new state-of-the-art results. (2) Compared with existing ConvKBQA systems, the rewritten questions can help identify the topic entities which results in a higher answer coverage rate (+12.9%). (3) Ablation studies demonstrate the knowledge-augmented self-training can indeed make the pre-trained rewriter/KoPL adapt to the concerned ConvKBQA task.

Contributions . (1) We propose a question rewriter decoupled from the subsequent KBQA model to enable a plug-and-play framework for ConvKBQA, which can adopt both the retrieval- and the SP-based single-turn KBQA models. (2) We devise a knowledge-augmented self-training strategy for adapting the pre-trained question rewriter to the concerned conversation style and the underlying KBs. (3) Both NSM and KoPL equipped with the rewriter achieve new SOTA performance on the well-adopted benchmark ConvQuestions.

2 Related Work

Single-turn KBQA falls into two mainstream methods, retrieval-based (Feng et al., 2021; He et al., 2021; Qiu et al., 2020; Sun et al., 2019) and SP-based (Cao et al., 2022; Gu et al., 2021; Ye et al., 2022). The former ones usually encode the entities and relations in KBs as well as the questions into a unified embedding space, based on which the answers are inferred. Recently, these



Train the relation retriever on pseudo (question, relation) pairs
Relation retriever
Find related (head, relation, tail) for conversation history
Knowledge
Pre-train rewriter on the open domain ConvQA dataset
Pre-trained rewriter
Infer high-quality rewritten questions on the ConvKBQA dataset
Pseudo labels
Question Rewriter
Knowledge-augmented Self-training
Fine-tuned rewriter
Rewrite the questions in ConvKBQA dataset
Self-contained questions
Reasoner
Retrieve the subgraph
Subgraph
Train retrieval-based reasoner
Pre-train SP-based reasoner
Pseudo programs
Self-train SP-based reasoner

Figure 2: Overview of the rewrite-and-reasoner framework. A question rewriter is pre-trained on the open domain ConvQA dataset and is fine-tuned on the concerned ConvKBQA dataset by knowledge-augmented self-training, where the knowledge is obtained by a relation retriever that is trained on pseudo (question, relation) pairs. Given the self-contained questions output by the rewriter, we train a retrieval-based reasoner on the retrieved subgraph, and also train a SP-based reasoner by the similar pre-training and fine-tuning paradigm.

methods often restrict the embedding space within a question-relevant subgraph expanding from the topic entities. Instead of representing KBs and questions, the latter ones often parse the questions into logical forms, which can be executed over the KB to get the answers. The type of intermediate logical forms can be various, such as the SPARQL used in (Das et al., 2021), the skeleton grammar proposed in (Sun et al., 2020), and the KoPL programming language designed in (Cao et al., 2022).

Conversational KBQA follows traditional single-turn KBQA systems and can also be categorized into retrieval- and SP-based methods. Different from the single-turn KBQA, the crux here is to deal with the ellipsis and coreference problem. The former ones usually identify the topic entities of each turn's question and then retrieve the subgraph for answer reasoning. The major differences lie in how to identify the topic entities. For example, Lan and Jiang (2021) builds an entity transition graph and applies a graph neural network to derive the topic entity distribution, while Kaiser et al. (2021) defines four heuristic measures to estimate such distribution. The SP-based methods (Kacupaj et al., 2021; Marion et al., 2021; Plepi et al., 2021) also parse from each turn's question to the logic form similar to the single-turn KBQA, but the difficulty is the incompleteness of the questions to be parsed. So they augment the model input with conversation history as well as extra KB information for parsing out the more correct logic form. But meanwhile, it inevitably introduces additional noises which might hamper the parsing performance.

Conversational QA performs conversational QA over unstructured text data instead of the structured KB. Thus different from ConvKBQA, ConvQA usually performs question rewriting, document retrieval, and reading comprehension on text data (Anantha et al., 2021; Elgohary et al., 2019). Besides, the conversations in ConvQA are more chatty than those in ConvKBQA. Such difference also hinders us from directly using the annotations in ConvQA to train ConvKBQA models.

3 Problem Formulation

A KB $\mathcal{K}$ is composed of a great number of (h, r, t) triplets, where h, r, and t represent a head entity, a relation, and a tail entity respectively. KBQA is to seek answers to a natural question from the given KB. ConvKBQA extends the single-turn QA into the multiple-turn QA conversations. To learn such a ConvKBQA system, we collect a dataset $\{C_1, C_2, ..., C_N\}$ including N conversations, where each conversation C_i starts with a seed entity s_i and lasts for K turns, denoted as $C_i = \{(q_1^i, A_1^i), (q_2^i, A_2^i), ..., (q_K^i, A_K^i)\}$ with (q_t^i, A_t^i) being the t-th turn conversation. ConvKBQA is to seek answers to each q_t^i from a given $\mathcal{K}$ based on the conversation history $H_t^i = \{s_i, (q_1^i, A_1^i), (q_2^i, A_2^i), ..., (q_{t-1}^i, A_{t-1}^i)\}$. Specially, for the initial question q_1^i, the history $H_0^i = \{s_i\}$.

4 The Rewrite-and-Reason Framework

In this section, we first introduce the proposed rewrite-and reason framework, and then elaborate on each part of our model, *i.e.*, the pre-training and fine-tuning of the question rewriter, as well as the downstream reasoners used in our work.

4.1 Overview

To address the challenges of ellipsis and coreference in the conversational setting, we propose a rewrite-and-reason framework that first rewrites the incomplete questions into the self-contained ones, and then adopts a single-turn KBQA model to solve the rewritten questions. The general workflow of the framework is illustrated in Figure 2.

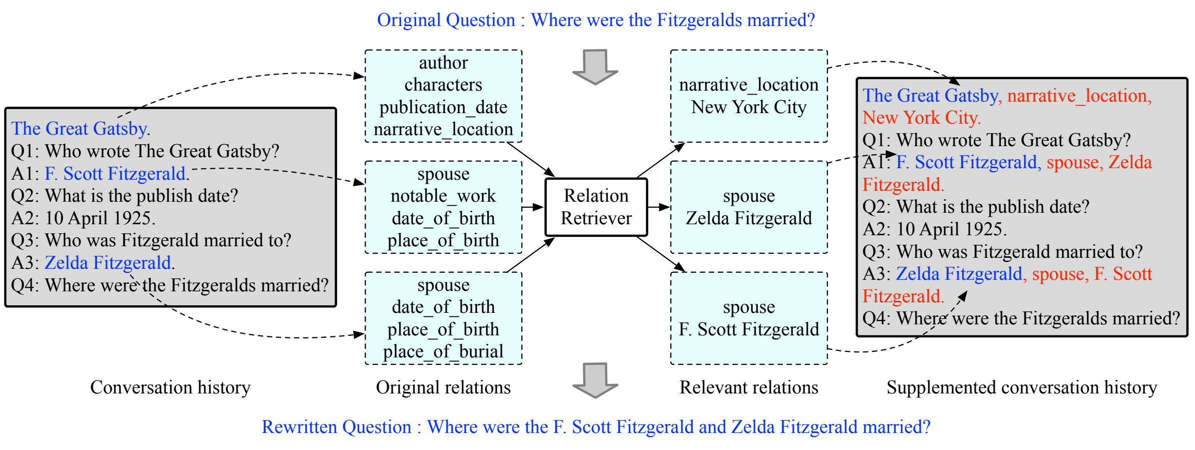

Original Question : Where were the Fitzgeralds married?
The Great Gatsby.
Q1: Who wrote The Great Gatsby?
A1: F. Scott Fitzgerald.
Q2: What is the publish date?
A2: 10 April 1925.
Q3: Who was Fitzgerald married to?
A3: Zelda Fitzgerald.
Q4: Where were the Fitzgeralds married?
author
characters
publication_date
narrative_location
spouse
notable_work
date_of_birth
place_of_birth
spouse
date_of_birth
place_of_birth
place_of_burial
Relation Retriever
narrative_location
New York City
spouse
Zelda Fitzgerald
spouse
F. Scott Fitzgerald
The Great Gatsby, narrative_location, New York City.
Q1: Who wrote The Great Gatsby?
A1: F. Scott Fitzgerald, spouse, Zelda Fitzgerald.
Q2: What is the publish date?
A2: 10 April 1925.
Q3: Who was Fitzgerald married to?
A3: Zelda Fitzgerald, spouse, F. Scott Fitzgerald.
Q4: Where were the Fitzgeralds married?
Conversation history
Original relations
Relevant relations
Supplemented conversation history
Rewritten Question : Where were the F. Scott Fitzgerald and Zelda Fitzgerald married?

Figure 3: An example of conversation history supplementation. The retrieved relevant relations to the current turn's question are supplemented to the seed entities and answer entities in the conversation.

For the rewriter, we propose a pre-training plus fine-tuning paradigm due to the absence of supervision signals in ConvKBQA. Specifically, the rewriter is first pre-trained on the open domain ConvQA dataset CANARD (Elgohary et al., 2019) with gold annotations and then fine-tuned on the ConvKBQA dataset ConvQuestions (Christmann et al., 2019) with knowledge-augmented self-training.

For the reasoner, we explore both the retrieval- and SP-based models. Specifically, based on the rewritten questions, we retrieve the subgraphs to enable the reasoning of the former ones, and adopt a similar pre-training and fine-tuning paradigm for learning the latter ones.

4.2 Question Rewriter

We explain how to (1) **pre-train** the question rewriter and (2) fine-tune it by **knowledge-augmented self-training**.

4.2.1 Pre-training

As all the conversation history is posed in the natural language, it's applicable to leverage the pre-trained sequence-to-sequence (Seq2Seq) language models (PLMs) such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) for question rewriting. A simple and effective way is to concatenate the historical question-answer pairs H_t^i as well as the current question q_t^i as the input $[H_t^i; q_t^i]$ of the PLMs and output the rewritten question $\hat{q}_t^i$, which is supposed to be a standalone question. Then, with the supervision of the gold rewritten question $\bar{q}_t^i$, we continue to pre-train the PLMs by maximizing the probabilities of generating all the K-turn gold rewritten questions, *i.e.*,

$$\mathcal{L}_{qr\_pt} = \max_{\theta} \sum_{i=1}^{N} \sum_{t=1}^{K} \log p_\theta(\bar{q}_t^i | [H_t^i; q_t^i]) \quad (1)$$

Because of the absence of gold annotations for rewritten questions in ConvKBQA, we can only pre-train the question rewriter on an existing ConvQA dataset with the annotated rewritten questions.

4.2.2 Knowledge-augmented Self-training

The regular patterns of ellipsis and coreference across different conversations are captured by the above pre-trained question rewriter. However, there is an inevitable gap between the distributions of more chatty open-domain conversations and more factual KBQA conversations. Thus, we leverage self-training, a well-adopted domain adaptation method, to overcome the lack of annotations on the target domain (Wei et al., 2021; Mukherjee and Awadallah, 2020; Xie et al., 2020; Zou et al., 2019).

Self-training is to train the model on the labeled data, and then apply it on the unlabeled data to generate pseudo labels. It is crucial to select confident pseudo labels on the target domain for adapting the pre-trained model to the target domain.

Different from the traditional self-training, we change the input $[H_t^i; q_t^i]$ of the original pre-trained model into the knowledge-injected one $[\mathcal{H}_t^i; q_t^i]$. Below we explain why and how we inject the knowledge and select the pseudo labels for self-training.

Knowledge Injection. Injecting the knowledge into the original conversation can help the rewriter distinguish the right entities from the ambiguous ones. For example, in Figure 3, "the Fitzgeralds" in

Q4 is successfully rewritten into “F. Scott Fitzgerald and Zelda Fitzgerald” rather than only one of them. Because when coming across Q4, both the entities “F. Scott Fitzgerald” and “Zelda Fitzgerald” are augmented with the relation “spouse”, which helps the rewriter correctly recognize the two topic entities in the question.

To enable the knowledge-augmented self-training, we train a relation retriever to identify the most relevant relations to the current turn’s question for knowledge injection. The relation retriever is instantiated by BERT (Kenton and Toutanova, 2019), which accepts the concatenation of a question q_m and a relation r_j as the input and takes the [CLS] token as the output embedding to compute the relevance score between q_m and r_j.

The ConvKBQA dataset only contains the (question, answer) pairs without the relations that can derive the answer, so we construct a pseudo dataset for training the above relation retriever. Since the seed entity of each conversation is given, and the follow-up questions are usually around the entities mentioned before, we can start from the already appearing entities to check whether they can arrive at the answers or not following their one-hop relations. The one-hop relation that can derive the answer and the current question compose a pseudo (question, relation) label.

Specifically, we automatically construct the pseudo (question, relation) labels from each conversation as follows. A topic entity set is initialized with the seed entity given in the first question. At each turn of the conversation, we enumerate each entity in the set and expand its one-hop relations, including both the outgoing and incoming relations. If a relation can derive the gold answer to this turn’s question, we make the relation and the question as a pseudo label. Then we augment the topic entity set with non-string answers of the last turn. We repeat the above operations until the final turn of the conversation. Figure 4 illustrates the construction process of the example in Figure 1. As a result, we construct a pseudo dataset consisting of about 32,000 (question, relation) pairs.

With the relation retriever, we can retrieve the most relevant relation to the current turn’s question for the seed entity and each answer entity in the conversation history H_t^i. Then we supplement these relations to the corresponding entities in H_t^i. In addition to injecting useful relation information, we also pad the tail entities expecting to provide

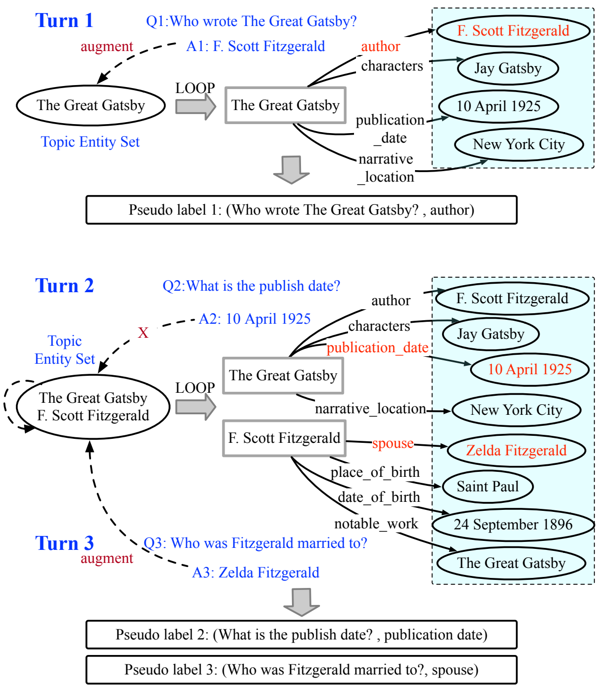


Figure 4: The automatic construction process of pseudo (question, relation) labels.

the rewriter with more choices for the topic entities. For example, given a conversation with the seed entity as “Harry Potter”, Q1 as “What is the first book of Harry Potter?” and Q2 as “Where is the author born?”, if we can inject “J. K. Rowling” as the tail entity of the relation “author” for “Harry Potter”, it gives the rewriter a chance to directly rewrite Q2 as “where is J. K. Rowling born?” rather than “where is the author of Harry Potter born?”, which is a more simple one-hop question to be answered.

Pseudo Label Selection. The pseudo labels are usually quite noisy which might take a negative effect on model fine-tuning. Thus, we need to carefully devise the pseudo label selection strategy.

We select the pseudo rewritten questions according to the relationship with the corresponding answers. Specifically, for the question q_t^i, we first generate the rewritten question $\hat{q}_t^i$ by the pre-trained rewriter, and then sift out those containing topic entities whose one-hop subgraphs cover the correct answers, because we conjecture that a rewritten question is more likely to be right if it can derive the answer by one-hop reasoning.

For identifying the topic entities, we use ELQ (Li et al., 2020), an entity linking tool for questions, to obtain the topic entity candidate set. Since it may contain some topic-irrelevant entities, we only keep those appearing in the conversation history or in the one-hop neighbors of the entities in the conversation history. If none of such entities can be recognized, we use the seed entity of the whole conversation as the topic entity of the current turn's question. Algorithm 1 in the attachment shows the details of topic entity identification.

The objective function of knowledge-augmented self-training is:

$$\mathcal{L}_{qr\_st} = \max_{\theta} \sum_{i=1}^{N'} \sum_{t=1}^{K'_i} \log p_{\theta}(\hat{q}_t^i | [\mathcal{H}_t^i; q_t^i]), \quad (2)$$

where $\hat{q}_t^i$ is the selected pseudo label of the original question q_t^i. Since we select partial labels on the whole dataset, the resultant conversation size N' and the turn number K'_i are less than the original sizes N and K.

4.3 Reasoner

With the advantage of the decoupled framework, the rewritten questions can flexibly adapt to different downstream reasoners. We explore both retrieval- and SP-based reasoners. For the former ones, an additional procedure of topic entity identification from the rewritten questions is required for the subsequent subgraph retrieval and reasoning. While for the latter ones, we need to overcome the lack of logic form labels.

4.3.1 Retrieval-based Reasoner

Retrieval-based methods usually represent the entities and questions as embeddings, based on which the relevance is calculated to rank the candidate answers. To improve the accuracy and efficiency, the answer candidates are usually restricted within the subgraph expanded from the topic entities.

We employ the state-of-the-art NSM (He et al., 2021) as the retrieval-based reasoner. For each turn's rewritten question $\hat{q}_t^i$, we also use Algorithm 1 to find the topic entities T_t^i. Then we retrieve a τ-hop subgraph starting from each topic entity and merge all of them to a unified subgraph S_t^i. We use the answers A_t^i as supervision signals to train NSM. The objective function is defined as:

$$\mathcal{L}_{nsm} = \max_{\phi} \sum_{i=1}^{N} \sum_{t=1}^{K} \log p_{\phi}(A_t^i | \hat{q}_t^i, T_t^i, S_t^i). \quad (3)$$

4.3.2 Semantic Parsing-based Reasoner

The SP-based methods usually parse the questions into logic forms, which can be directly executed on the KB to retrieve the answer, but they depend on the annotated logic forms. Here, we explore a similar pre-training and fine-tuning paradigm with knowledge-augmented self-training as the question rewriter to train such SP models.

We first pre-train a SP model on the KQA pro dataset (Cao et al., 2022) consisting of (question, KoPL) labels, where KoPL is of logic form. The pre-trained SP model can translate a question into its KoPL program. We apply it to generate the KoPL program $\hat{p}_t^i$ for each rewritten question $\hat{q}_t^i$ on the ConvKBQA dataset. Then we sift out the $(\hat{q}_t^i, \hat{p}_t^i)$ pairs in which $\hat{q}_t^i$ is filtered following the rewriter's pseudo label selection strategy and $\hat{p}_t^i$ is restricted to the program that contains the identified topic entity as well as the corresponding relation and can be executed to obtain the correct answers. We also use Algorithm 1 to identify topic entities.

To inject the knowledge when self-training, we supplement $\hat{q}_t^i$ with relevant triplets of the topic entities as $\hat{\mathcal{Q}}_t^i$, where the relation in the triplet is determined by the relation retriever. The objective function of the SP model self-training is:

$$\mathcal{L}_{kopl} = \max_{\varphi} \sum_{i=1}^{\hat{N}} \sum_{t=1}^{\hat{K}_i} \log p_{\varphi}(\hat{p}_t^i | \hat{\mathcal{Q}}_t^i), \quad (4)$$

where $\hat{N}$ and $\hat{K}_i$ represent the conversation size and turn number in the i-th conversation of the selected pseudo dataset. To further improve the accuracy, we also modify the inferred KoPL programs with the identified topic entities and corresponding relevant relations by Algorithm 2. We present more details about the training and inference process of KoPL in Appendix A.5.

5 Experiments

5.1 Experimental Settings

Dataset. We evaluate our method on ConvQuestions (Christmann et al., 2019)[2] , a ConvKBQA dataset created on Wikidata by crowdworkers on Amazon Mechanical Turk. ConvQuestions contains about 11,000 conversations from five domains:"Movies", "TV Series", "Music", "Books" and "Soccer", which are partitioned into 7,000/2,000/2,000 for training/validating/testing. Each conversation goes in a 5-turn dialog, only with the annotated ground truth answers.

[2]https://convex.mpi-inf.mpg.de/

Table 1: QA performance (H1/F1) and answer coverage rate (ACR) evaluated on ConvQuestions (%).

Model	Overall	Overall		Movies		TV Series		Music		Books		Soccer	
	ACR	H1	F1	H1	F1	H1	F1	H1	F1	H1	F1	H1	F1
CONVEX	-	18.2	18.1	21.6	21.6	17.5	17.5	10.2	10.2	24.6	24.6	17.2	17.0
OAT	-	25.0	-	31.3	-	31.8	-	18.1	-	20.9	-	22.8	-
CONQUER	-	29.6	29.6	32.1	32.1	27.1	27.1	25.7	25.7	37.2	37.2	25.7	25.7
Focal Entity	64.7	32.4	29.8	24.5	21.4	40.7	37.6	21.1	19.6	42.9	40.4	32.7	30.1
QR+RR	**77.6**	32.3	31.9	26.5	26.5	35.2	34.3	29.6	29.6	35.5	35.1	34.6	34.1
QR+KoPL	**77.6**	35.5	37.0	**48.3**	**50.3**	40.5	42.2	**34.9**	**35.6**	27.6	30.6	26.4	26.4
QR+NSM	**77.6**	**44.0**	**43.4**	45.0	42.2	**54.2**	**51.6**	30.4	33.3	**51.3**	**49.3**	**39.3**	**40.5**

We do not evaluate on another ConvKBQA dataset CSQA (Marion et al., 2021)[3], because CSQA is less challenging in the conversational setting. The turns in CSQA are simply linked together into a conversation if adjacent questions have overlapped entities or relations, which indicates that the topic entities of a question can be found in either current or last turn. Thus the challenge of CSQA mostly lies in the downstream reasoning against KB, rather than the ellipsis and coreference in the conversational setting.

Evaluation Metrics. To evaluate the question rewriter, since we do not have the ground truth, we use the intermediate answer coverage rate (ACR), *i.e.*, the percentage of rewritten questions from which we can extract topic entities that can reach the correct answers in their one-hop subgraphs.

For the QA performance, we use the top-1 hit ratio (H1) to evaluate whether the top-1 predicted entity is the correct answer, and also report the F1 score by viewing questions as multi-answer ones.

When evaluating any question, the answers to the last turns are predicted by models instead of the ground truth answers given in the dataset.

Baselines. We compare with CONVEX (Christmann et al., 2019), OAT (Marion et al., 2021), CONQUER (Kaiser et al., 2021), and Focal Entity (Lan and Jiang, 2021) as baselines. OAT is a SP-based method which incorporates conversation history and extra KB information besides the current question as the input to decode their defined logic forms. The other three are retrieval-based methods. CONQUER maintains a topic entity candidate set during the conversation for identifying the topic entities, which are scored by heuristic measures. While CONVEX and Focal Entity further build a transition graph on the candidate set, where CONVEX defines some heuristic rules and Focal Entity adopts the graph neural networks to identify topic entities. They all infer the answer based on the relevance between the current question and the paths derived from the topic entities.

[3]https://amritasaha1812.github.io/CSQA

5.2 Overall QA Evaluation

We compare our proposed rewrite-and-reason framework with existing state-of-the-art ConvKBQA methods, including both end-to-end(SP-based) and pipeline(retrieval-based) frameworks. The results are shown in Table 1. Compared with the baselines, we can observe an obvious performance improvement, *e.g.*, 11.6% H1 improved by the question rewriter (QR) combined with NSM, and 3.1% with KoPL. The results show that the question rewriter with knowledge-augmented self-training outperforms both end-to-end and pipeline methods and can improve the performance not only equipped with the retrieval-based methods but also the SP-based methods.

Moreover, to intuitively demonstrate the effectiveness of our question rewriter, we also report the intermediate answer coverage rate in the first column of Table 1, which is also the quantitative analysis of the noise introduced in the pipeline process. Since our rewriter outputs the rewritten questions instead of the topic entities, we leverage Algorithm 1 to identify the topic entities, from which we retrieve their one-hop related entities to compute the answer coverage rate. In terms of ACR, our proposed pipeline method rewriter-reasoner outperforms the SOTA pipeline method Focal Entity by 12.9%, which demonstrates the proposed question rewriter could result in more accurate topic entities and the noises are significantly reduced by our method in the pipeline process.

5.3 Ablation Experiments

To investigate the effectiveness of different parts in our framework, we respectively remove the injected knowledge in the self-training input as well as the whole self-training step in the question

Table 2: Ablation studies of knowledge injection and self-training for rewriter by answer coverage rate (%). All means evaluating on train, validation, and test sets.

Model	ConvQuestions			
	All^{1st}	All	Test^{1st}	Test
QR	80.1	74.2	83.6	75.5
QR (w/o k)	80.6	73.8	83.6	74.6
QR (w/o st)	79.6	71.1	82.9	73.1

Table 3: Ablation studies of knowledge injection and self-training for rewriter by QA performance (%).

Model	ConvQuestions	
	H1	F1
QR + NSM	44.0	43.4
QR (w/o k) + NSM	42.7	43.4
QR (w/o st) + NSM	39.6	43.4
QR + KoPL	35.5	37.0
QR (w/o k) + KoPL	35.1	36.7
QR (w/o st) + KoPL	34.5	36.0

rewriter one at a time to show the effects of them.

5.3.1 Effect of Knowledge Injection

We explore whether the knowledge incorporated in self-training, retrieved by the relation retriever, can improve the performance or not through three aspects: the direct retrieval performance on the pseudo (question, relation) dataset, the rewriter's answer coverage rate, and the final QA performance.

First, the retrieval results on the pseudo (question, relation) dataset in terms of H1, H3, and H5 are 83.0%, 91.5%, and 93.6% respectively. To further prove its validity, we also use it as the reasoner combined with the question rewriter. The overall H1 is 32.3% as shown in Table 1. Although it is much simpler than NSM (He et al., 2021), the QA performance can be comparable to the baseline Focal Entity which also uses a more complex reasoner. All the results show that the relation retriever can retrieve relatively accurate relations.

Second, we use the answer coverage rate to evaluate the quality of the rewriter. Here, for a fair comparison, we use gold answers instead of the predicted answers in the last turns to get rid of the impact of reasoning capability. Moreover, we strictly restrict the rewritten questions to containing the complete topic entity names to exclude the impact of the entity linking procedure. The answer coverage results with and without the injected knowledge (*i.e.*, w/o k) are reported in Table 2. We report the answer coverage rates on all the dataset

Table 4: Ablation studies of knowledge injection and self-training for KoPL by QA performance (%).

Model	ConvQuestions	
	H1	F1
QR+KoPL	31.5	32.6
QR+KoPL (w/o k)	30.3	31.4
QR+KoPL (w/o st)	25.0	26.8

and only the test set respectively, and also distinguish the performance on the first-turn questions from all the turns. The results of different model variants on the first questions are similar because they are self-contained, and do not need the rewritten effort. On the questions of all turns, we could observe that removing the injected knowledge reduces 0.4% coverage rate on All and 0.9% on Test. The results indicate the effectiveness of injecting knowledge in self-training.

Finally, We evaluate the effect of the injected knowledge by the QA performance. The results in Table 3 also indicate that when removing the knowledge from the rewriter's self-training process, the QA performance of the subsequent reasoner including both NSM and KoPL drops.

5.3.2 Effect of Self-Training

To demonstrate the effectiveness of our proposed self-training mechanism, we remove the whole self-training process and directly use the pre-trained rewriter for question rewriting (*i.e.*, w/o st). The answer coverage rate and the QA performance are given in Table 2 and Table 3 respectively. We can see that without self-training, the answer coverage rate reduces 3.1% on All and 2.4% on Test, H1 reduces 4.4% equipped with NSM and 1.0% with KoPL. These results demonstrate the effectiveness of self-training for the rewriter.

5.4 Evaluation of SP-based Methods

When exploring KoPL as the reasoner, we adopt the similar knowledge-augmented self-training to transfer it from another dataset to the concerned dataset. Thus, we also conduct the ablation study to observe whether the knowledge injection and the self-training are useful for KoPL. As is shown in Table 4, both the injected knowledge and self-training promote the QA performance on ConvQuestions, which are consistent with their effects on the proposed question rewriter. Since we study the transfer capability of the KoPL reasoner here, we do not report the performance of the modified KoPL programs as in Table 1 and Table 3, but only report the performance of the originally inferred programs by the KoPL model.

5.5 Discussions

Performance on Multiple Coreferences and Ellipsis. How is the capability of the question rewriter when facing more complicated circumstances, such as multiple coreferences or multiple ellipses of entities or relations?

For multiple coreferences, since the gold rewritten questions are unavailable, we manually select and evaluate 100 samples with multiple coreferences from the ConvQuestions (Christmann et al., 2019) dataset. The accuracy is about 93%. We list three original questions and correct rewritten questions by our model as follows: (1) "Is she younger than him?" => "Is Leslie Stefanson younger than James Spader?" (2) "Were the two living at the same period?" => "Were Mikhail Bulgakov and Mikhail Lermontov living at the same period?" (3) "Was that their last album?" => "Was The White Album The Beatles last album?"

For multiple ellipses, since such samples cannot be observed in the ConvQuestions dataset, we manually create and evaluate 5 samples like "Birthplace and birthtime?" and " When and where?". The rewritten questions are "Birthplace and birthtime of soccer player Zinedine Zidane?" and "When and where did Haruki Murakami win the Franz Kafka Prize?". The 5 samples are all correctly rewritten because it is easy for the rewriter to generalize from the single ellipsis such as "when" or "where" to the multiple ellipses "when and where".

Performance on Self-Contained Questions. Whether the question rewriter has negative effect on the rewriting process if the question is already self-contained? Since the questions in ConvQuestions are almost incomplete, we randomly sample 100 self-contained questions from CSQA (Marion et al., 2021) and rewrite them by our model. We observe that the rewriter prefers to remain the original question as much as possible. About 95% of the rewriting questions are exactly the same as the original questions, and the remaining 5% are averagely 0.978 similar (#common words at the same position / #words in the longer question) to the original questions. The results indicate that the proposed question rewriter does not influence the orginal self-contained questions.

6 Conclusion

This paper proposes a rewrite-and-reason framework to address the ellipsis and coreference for ConvKBQA. To overcome the lack of annotations, we introduce a knowledge-augmented self-training mechanism for training the question rewriter. Thanks to the decoupled design, we can unite the rewriter with both the retrieval- and SP-based single-turn KBQA models. The experiment results show that our method can outperform existing ConvKBQA systems and achieve new state-of-the-art.

Acknowledgments

This work is supported by National Natural Science Foundation of China (62076245, 62072460, 62172424); National Key Research & Develop Plan(2018YFB1004401); Beijing Natural Science Foundation (4212022).

Limitations

Despite the strong capability of the question rewriter, it still has some limitations and deficiencies. The question rewriter suffers from the missing relations. For example, given a conversation with seed entity as "Harry Potter", Q1 as "What is the first book of Harry Potter?", Q2 as "Who is the author?", Q3 as "Where is the author born?" and Q4 as "And when?", it is difficult for the question rewriter to produce "And when was J. K. Rowling born?" for Q4 based on the Q1-Q3 conversation history. And in practice, as the conversation goes on, the conversation history becomes longer and the irrelevant knowledge increases in the rewriter's input. In that case, the effectiveness and the efficiency of the rewritten process might be impacted.

References

Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2021. Open-domain question answering goes conversational via question rewriting. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 520–534.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

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.

Philipp Christmann, Rishiraj Saha Roy, Abdalghani Abujabal, Jyotsna Singh, and Gerhard Weikum. 2019. Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 729–738.

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.

Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, et al. 2020. Autoknow: Self-driving knowledge collection for products of thousands of types. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2724–2734.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. Can you unpack that? learning to rewrite questions-in-context. 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 5918–5924.

Yu Feng, Jing Zhang, Gaole He, Wayne Xin Zhao, Lemao Liu, Quan Liu, Cuiping Li, and Hong Chen. 2021. A pretraining numerical reasoning model for ordinal constrained question answering on knowledge base. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1852–1861, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 553–561.

Endri Kacupaj, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova. 2021. Conversational question answering over knowledge graphs with transformer and graph attention networks. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 850–862.

Magdalena Kaiser, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Reinforcement learning from reformulations in conversational question answering over knowledge graphs. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 459–469.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

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, IJCAI-21*, pages 4483–4491. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Yunshi Lan and Jing Jiang. 2021. Modeling transitions of focal entities for conversational knowledge base question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3288–3297.

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.

Belinda Z Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6433–6441.

Pierre Marion, Pawel Nowak, and Francesco Piccinno. 2021. Structured context and high-coverage grammar for conversational question answering over knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8813–8829.

Subhabrata Mukherjee and Ahmed Awadallah. 2020. Uncertainty-aware self-training for few-shot text classification. In *Advances in Neural Information Processing Systems*, pages 21199–21212.

Joan Plepi, Endri Kacupaj, Kuldeep Singh, Harsh Thakkar, and Jens Lehmann. 2021. Context transformer with stacked pointer networks for conversational question answering over knowledge graphs. In *European Semantic Web Conference*, pages 356–371. Springer.

Yunqi Qiu, Yuanzhuo Wang, Xiaolong Jin, and Kun Zhang. 2020. Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 474–482.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. Sparqa: skeleton-based semantic parsing for complex questions over knowledge bases. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8952–8959.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998.

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

Chen Wei, Kihyuk Sohn, Clayton Mellina, Alan Yuille, and Fan Yang. 2021. Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10857–10866.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698.

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.

Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. 2019. Confidence regularized self-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5982–5991.

A Implementation Details

A.1 Setup

KB. We download and adopt the KB cache[4] collected by Focal Entity (Lan and Jiang, 2021) in our experiments. For the KB facts they do not explore, we query the latest WikiData dump[5] for use.

Entity Linking. We use ELQ (Li et al., 2020) as the entity linking tool to link entities in questions. We clone the BLINK repo[6] and follow the ELQ setup steps[7] to build the entity linking environment.

Computing Infrastructure. We implement our method by PyTorch and run experiments on the server with one RTX A6000(48G) GPU and 256G memory.

A.2 Baselines.

For OAT (Marion et al., 2021), its codes and datasets are unavailable, so we directly report the results from their papers. For CONVEX (Christmann et al., 2019) and Focal Entity (Lan and Jiang, 2021), we re-train and use our metrics to evaluate their models. For CONQUER (Kaiser et al., 2021), they manually augment the dataset with reformulated questions, so we evaluate it on ConvQuestions without reformulations by running the code[8] they provide for a fair comparison.

A.3 Relation Retriever

We use BERT-base (Kenton and Toutanova, 2019)[9] as the backbone of the relation retriever and finetune it on our constructed pseudo dataset, which contains 31,520 (question,relation) pairs with 18,848/5,696/6,976 splits for training/validating/testing. A dropout layer is set before the last MLP with the dropout probability as 0.1. We set the initial learning rate as 1e-4 with the warm-up steps as 100 and the weight decay as 3e-5.

[4]github.com/lanyunshi/ConversationalKBQA#download-the-pre-trained-model-and-the-kb-cache-optional

[5]query.wikidata.org

[6]github.com/facebookresearch/BLINK.git

[7]github.com/facebookresearch/BLINK/tree/main/elq

[8]https://github.com/magkai/CONQUER

[9]huggingface.co/bert-base-uncased

The model is trained for 5 epochs with the batch size as 10, where for each of the 10 questions, we retrieve the positive relations following Figure 4 and complete 64 relations in total by negative sampling. We save the checkpoint obtaining the best H1 on the validation set. A training epoch takes about 15-20 minutes and it takes about 1-2 hours for the model to converge.

A.4 Question Rewriter

Pre-training. We use T5-base[10] as the backbone of the question rewriter and finetune it with the Transformer Trainer[11] on CANARD (Elgohary et al., 2019). We set the learning rate as 1e-4, the gradient accumulation steps as 4, the batch size as 32, and the epochs as 5. A training epoch takes about 8-12 minutes and it takes about 1 hour for the model to converge.

Pseudo Label Generation. We adopt the pre-trained question rewriter to generate pseudo labels on our dataset ConvQuestions for fine-tuning. The whole generation process takes about 30-60 minutes.

Once we get the pseudo labels, *i.e.*, the rewritten questions, we perform ELQ (Li et al., 2020) to obtain topic entities. However, since the tool may produce some irrelevant entities, we only keep the entities appearing in the conversation history or in the one-hop neighbors of the entities in the conversation history. If none of such entities can be recognized, we use the seed entity of the whole conversation as the topic entity of the current turn's question. Algorithm 1 presents the details. The process of topic entity identification on the whole ConvQuestions dataset takes about 3-4 hours.

We select rewritten questions that contain entities whose one-hop subgraphs can cover the correct answers. That is to say, on one hand, the identified topic entities from the rewritten question can reach the correct answers, and on the other hand, the textual names of the topic entities should be completely appearing in the rewritten question. In this way, we exclude the impact of the entity linking procedure and acquire high-quality rewritten questions.

Finally, we obtain 23,909/7,065 high-quality pseudo (original question, rewritten question) pairs for rewriter fine-tuning/validating.

[10]huggingface.co/t5-base

[11]huggingface.co/docs/transformers/v4.20.0/en/main_classes/trainer

Algorithm 1: Topic Entity Identification

```
Input: $K$ turns, seed entity $s_i$, conversation
    $\mathcal{C}_i = \{(q_1^i, A_1^i), (q_2^i, A_2^i), ..., (q_K^i, A_K^i)\}$.
Output: Topic entities $\mathbb{T}_i = \{T_1^i, T_2^i, \cdots, T_K^i\}$.
1: Get 1-hop neighbors $N_{s_i}$ of seed entity $s_i$.
2: Initialize neighbor set $N = \{s_i, N_{s_i}\}$.
3: Initialize topic entities $\mathbb{T}_i = \{\}$.
4: for $t = 1$ to $K$ do
5:   while True do
6:     $T_t^i = \text{ELQ}(q_t^i)$;
7:     Subtract entities not in $N$ from $T_t^i$;
8:     if $T_t^i = \varnothing$ and $s_i$ not in $q_t^i$ then
9:       $q_t^i = s_i +$“,”$+ q_t^i$;
10:      continue;
11:    else if $T_t^i = \varnothing$ and $s_i$ in $q_t^i$ then
12:      $T_t^i = \{s_i\}$;
13:      break;
14:    else
15:      break;
16:    end if
17:  end while
18:  Update topic entities $\mathbb{T}_i$ by adding $T_t^i$;
19:  Get 1-hop neighbors $N_{A_t^i}$ of answers $A_t^i$;
20:  Update neighbor set $N = N \cup A_t^i \cup N_{A_t^i}$;
21: end for
22: Return topic entities $\mathbb{T}_i$;
```

Fine-tuning. We continue to use the Transformer Trainer with the same training arguments as the pre-training process to fine-tune the question rewriter, but replace the T5-base model with the pre-trained question rewriter and replace the CANARD dataset with the pseudo rewritten question dataset. The fine-tuning and generating process take nearly as much time as the pre-training of the question rewriter.

A.5 Reasoner

NSM. We set up the running environment of NSM following its settings[12].

We use the original questions rather than the rewritten questions to train NSM because most of the questions only miss the topic entities and the remaining part can already reflect the reasoning relation. For subgraph retrieval, we set $\tau = 1$, which means we only take the one-hop subgraph into consideration. Because after being rewritten, most of the questions can be answered only replying on

[12]https://github.com/RichardHGL/WSDM2021_NSM

the one-hop subgraph. We also try to retrieve additional two-hop outgoing relations, but the answer coverage gain is quite small, which means in the uncovered parts, the small subgraph scale is not the pain point. We filter the triplets with a number of the heads or tails more than 500 and keep the triplets with the head and tail types as entities, datetime, numerical values, and the string values in English.

For training NSM on ConvQuestions, We follow the training arguments on MetaQA given in NSM[13]. A training epoch takes about 3-5 minutes and it takes about 1.5-2.5 hours for NSM to converge. When performing the question answering in the conversational setting, the question rewriter combined with NSM takes about 4-6 hours (6.5-9.5 seconds for one conversation and 1.3-1.9 seconds for one question).

KoPL. For the pre-trained KoPL model, we download the checkpoint[14] provided by Cao et al. (2022). We use the pre-trained model to generate the KoPL programs and then execute them to get the answers in our ConvQuestions dataset. The whole generation and execution process take about 4-5 hours.

From all the generated programs, we select the (rewritten question, KoPL program) pairs in which the rewritten question is filtered following the rewriter's pseudo label selection strategy and the KoPL program can be executed to obtain the correct answers. Moreover, we further restrict the topic entity and its relevant relation to be in the KoPL program. Because sometimes even if the KoPL program could retrieve the correct answers, the relation encoded in the program might not match the relation retrieved by our relation retriever. For example, in the KQA pro dataset, the relation "author" is often described by "after a work by", which leads to a mismatch between the KQA pro dataset and our ConvQuestions dataset. Under all the above restrictions, it is easier for the model to learn the mapping from the rewritten question to the correct KoPL program in our dataset.

Finally, we obtain 5105/1513 high-quality pseudo (rewritten question, KoPL program) pairs for training/validating. The KoPL model is built on Bart-base[15] and we use the Transformer Trainer to fine-tune it. We set the Trainer with the selected pseudo (rewritten question, KoPL program) dataset, the pre-trained KoPL model, and the training arguments as follows: the learning rate as 3e-5, the warm-up ratio as 0.1, the weight decay as 1e-5, the Adam epsilon as 1e-8, the gradient accumulation steps as 1, the batch size as 64, and the epochs as 10. A training epoch takes about 15-20 seconds and it takes about 2-4 minutes in total to converge.

When inferring on the test set, to further improve the program's accuracy, we modify and reconstruct KoPL programs according to the identified topic entities and the retrieved relevant relations as shown in Algorithm 2. When performing the question answering in the conversational setting, the question rewriter combined with KoPL takes about 3-5 hours (5-8 seconds for one conversation and 1-1.6 seconds for one question).

Algorithm 2: KoPL Program Modification

```
Input: Question $q_i$, topic entities $T_i$, relevant
    relations $R_i$ of $T_i$.
Output: Answers $A_i$.
 1: Operation list and Value list
    $F_i, I_i$ = Question2KoPL($q_i$);
 2: $A_i$ = KoPLExecutor($F_i, I_i$);
 3: if $A_i = \varnothing$ then
 4:     Replace the relations in $I_i$ with $R_i$;
 5:     $A_i$ = KoPLExecutor($F_i, I_i$);
 6:     if $A_i = \varnothing$ then
 7:         $F_i$ = ["Find", "Relate", "What"];
 8:         $I_i$ = [$T_i$, [$R_i$, "forward"], []];
 9:         $A_i$ = KoPLExecutor($F_i, I_i$);
10:         if $A_i = \varnothing$ then
11:             $F_i$ = ["Find", "QueryAttr"];
12:             $I_i$ = [$T_i$, $R_i$];
13:             $A_i$ = KoPLExecutor($F_i, I_i$);
14:         end if
15:     end if
16: end if
17: Return Answers $A_i$;
```

[13]github.com/RichardHGL/WSDM2021_NSM/blob/main/run_metaqa.sh

[14]github.com/shijx12/KQAPro_Baselines/tree/master/Bart_Program#checkpoints

[15]huggingface.co/facebook/bart-base