CycleKQR: Unsupervised Bidirectional Keyword-Question Rewriting

Andrea Iovine[1], **Anjie Fang**[2], **Besnik Fetahu**[2], **Jie Zhao**[2], **Oleg Rokhlenko**[2], **Shervin Malmasi**[2]
[1]University of Bari Aldo Moro, Italy
[2]Amazon.com, Inc., Seattle, WA, USA
andrea.iovine@uniba.it
{njfn,besnikf,olegro,malmasi}@amazon.com

Abstract

Users expect their queries to be answered by search systems, regardless of the query's surface form, which include *keyword queries* and *natural questions*. Natural Language Understanding (NLU) components of Search and QA systems may fail to correctly interpret semantically equivalent inputs if this deviates from how the system was trained, leading to suboptimal understanding capabilities. We propose the *keyword-question rewriting* task to improve query understanding capabilities of NLU systems for all surface forms. To achieve this, we present CycleKQR, an *unsupervised approach*, enabling effective rewriting between keyword and question queries using non-parallel data.

Empirically we show the impact on QA performance of unfamiliar query forms for open domain and Knowledge Base QA systems (trained on either keywords or natural language questions). We demonstrate how CycleKQR significantly improves QA performance by rewriting queries into the appropriate form, while at the same time retaining the original semantic meaning of input queries, allowing CycleKQR to improve performance by up to 3% over supervised baselines. Finally, we release a dataset of 66k keyword-question pairs.[1]

1 Introduction

Information Retrieval and Question Answering (QA) systems aim to fulfill user requests, from traditional Web search to voice assistants. Users can issue queries in two forms (White et al., 2015): as *keyword* queries ("`iphone 14 price`") or as fully-formed natural language *questions* ("`how much is an iPhone 14?`"). Natural Language Understanding (NLU) components are expected to correctly interpret semantically equivalent queries, regardless of the form. Figure 1 (a) shows examples of queries with the same intent, but in different forms.

[1]https://github.com/amzn/kqr

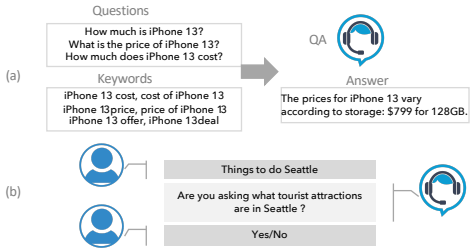

Figure 1: (a) Questions & keywords with the same intent. (b) Keyword-question rewriting for disambiguation.

Work on question paraphrases has grown recently, with current approaches utilizing encoder-decoder architectures for generating questions (Hosking and Lapata, 2021). However, NLU systems that handle different input forms must support two important functionalities beyond questions:

- **Understanding:** Correctly interpret different query forms (cf. Figure 1 (a)), and create equivalent semantic representations, regardless of the query form, style, or other lexical variations.
- **Expression:** Rewrite queries into different forms (e.g. keywords into questions). This increases transparency by relaying the query interpretation back to the users, or to ask clarification questions (Kiesel et al., 2018) to ensure that their requests are correctly understood (cf. Figure 1 (b)).

To address these, we introduce the *keyword-question rewriting* task, which aims to rewrite questions into semantically equivalent keyword queries, and *vice versa*. This task enables training models for the above desiderata, allowing NLU systems to handle either keywords or questions. Further, this capability enables asking clarification questions about user's input query (Ding and Balog, 2018), (cf. Figure 1 (b)), by rewriting it into a question.

While previous work has tackled the problem of rewriting keywords into questions (Zhao et al., 2011; Bhagat and Hovy, 2013), it focuses on unidirectional (keyword $\mapsto$ question) generation using supervised datasets. We propose a bidirectional and unsupervised approach in this paper.

Specifically, we propose CycleKQR, an approach based on cycle-consistent training (Lample et al., 2017; Iovine et al., 2022), which is an unsupervised approach that effectively maps two non-parallel sets of keywords and questions via two cycles, keywords-to-question (K2Q) and question-to-keywords (Q2K). These two cycles learn interactively from each other, and K2Q & Q2K are trained using reconstruction losses. Although unsupervised, CycleKQR is more robust since the its rewriting models are trained simultaneously, while a supervised approach has to train both models in two rounds. As there are no existing datasets for keyword-question rewriting we also construct and release a dataset of 66k keyword-question pairs.

We focus on (i) *intrinsic* and (ii) *extrinsic* evaluation strategies. For intrinsic evaluation, we measure the *bidirectional rewriting quality* of competing approaches using human studies and automated metrics such as ROUGE and BLEU.

Our extrinsic evaluation focuses on QA systems, where the ability to correctly interpret the user query is critical, and if necessary, query rewriting approaches are applied to ensure answer accuracy (Vakulenko et al., 2021; Elgohary et al., 2019). More concretely, we measure the downstream impact on *QA performance* using the rewritten keyword or question queries.[2] We assess the impact on two types of QA systems: *open domain* and *Knowledge Base QA* systems.

Experiments show that QA systems perform poorly on query forms that they are not trained on. We show that CycleKQR can effectively rewrite between keyword and question queries without annotations, and at the same time improve QA performance. CycleKQR obtains QA performance improvements of up to 3% over a supervised baseline using the same neural architecture. This is due to the ability to cope with noisy input and target query pairs, given that CycleKQR computes a reconstruction loss in each cycle, thus allowing it more flexibility in the rewriting step, and avoiding overfitting on noisy pairs.

In sum, we make the following contributions:

- Introduce a keyword-question rewriting task for improved query understanding;
- Propose CycleKQR, an unsupervised approach for keyword-question rewriting;
- Release a publicly available dataset with 66k pairs for keyword-question rewriting.

[2]Given a keyword query, we rewrite it into its question query equivalent, and use that to assess the performance of a QA system trained on natural language questions.

2 Related Work

Text Rewriting. Keyword-question rewriting shares similarities with *paraphrasing* and *style transfer*. Paraphrasing aims to generate alternative surface forms of an input text, while keeping its semantic meaning (Madnani and Dorr, 2010). Style transfer focuses more on transforming the input into a specific syntactic style, e.g. Shakespeare or Twitter (Bhagat and Hovy, 2013). Both tasks have been employed in applications such as data augmentation (Iyyer et al., 2018), text simplification (Xu et al., 2015), stylometry (Gröndahl and Asokan, 2020), and translation (Sellam et al., 2020). Our task focuses on bidirectional keyword-question rewriting, with the aim of improved query understanding, while retaining their semantic meaning.

Two techniques have been proposed for paraphrasing and style transfer: rule-based and generative approaches. Khosmood (2012) compare rule-based paraphrasing approaches including phrase replacement (e.g. using WordNet), translation and tense change. Gröndahl and Asokan (2020) employ simple rules (e.g. changing the tense of the sentences) to generate multiple candidates, and select the best one using a ranker. Hosking and Lapata (2021) propose a neural encoder-decoder model to generate and control the output questions using question templates. Krishna et al. (2020) first normalize a given sentence by removing its style and then apply GPT2 to transfer it to a desired style. Rules and templates used in existing works are domain and dataset specific, and obtaining them for any target domain and dataset is time-consuming, hence, we focus on encoder-decoder architectures.

Question-Keyword Rewriting. The first attempts to rewrite keyword-based queries into questions focused on using templates (Zhao et al., 2011; Zheng et al., 2011; Dror et al., 2013). Question templates are typically obtained from user-clicked questions, and for a given a keyword query, the most relevant template is used to generate a question. Ding and Balog (2018) propose a statistical model to generate a synthetic set of question-keyword query pairs. Given a question, they generate a keyword query by randomly sampling based on its length and the included terms. The pairs are then used to train a neural model (Gu et al., 2016). The resulting keyword queries are highly noisy and require additional filtering before being used. Moreover, it does not guarantee that the output will retain the meaning of the question.

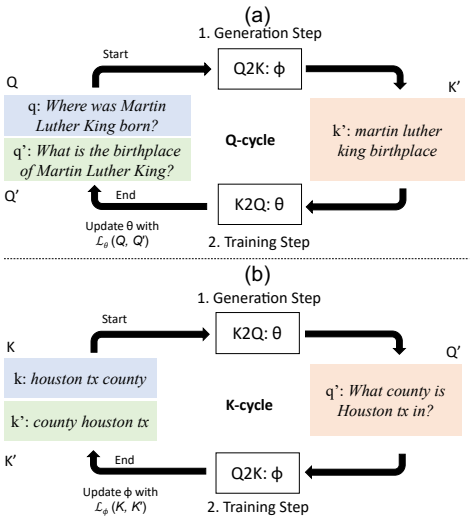

Figure 2: The training framework of CycleKQR.

Cycle-consistent Training. Cycle-consistency is the concept of enforcing transitivity in training: models learn a transformation from the input to output, and *vice versa*. This can result in in models that generalize better to unseen data (Meng et al., 2018). It has been successfully applied in various NLP applications (Lample et al., 2017; Mohiuddin and Joty, 2019). In neural machine translation, it learns a mapping between two languages using two non-parallel sets of text in the two languages. It has been later used in graph captioning (Guo et al., 2020), and Named Entity Recognition (Iovine et al., 2022). Cycle-consistent training is suitable for our task, where we aim to learn bidirectional mappings between two different query forms.

3 Task Definition

We define the task of rewriting keywords into question queries and vice-versa through two rewriting functions: (i) *keywords-to-question* (K2Q), and (ii) *question-to-keywords* (Q2K). Let Q be a set of *natural language questions*, and K a set of keyword queries. K2Q rewrites a keyword query $k \in K$ into a question $q' \in Q$, i.e. $K \mapsto Q$. Similarly, its inverse function, Q2K, rewrites a question into its corresponding keyword query, $Q \mapsto K$.

For example, given the keyword query "`houston tx county`", K2Q will output the question "`What county is Houston TX in?`". Conversely, given the question "`Where was Martin Luther King born?`", Q2K will output "`martin luther king birthplace`". Both K2Q and Q2K must preserve the semantic meaning of the original input into the generated output, i.e. both the input and output must convey the same information need.

4 CycleKQR: Cycle-Consistent Training for Keyword-Question Rewriting

Figure 2 shows an overview of our approach. We now describe how we use cycle-consistency to train K2Q and Q2K for keyword-question rewriting.

Overview: For the K2Q and Q2K, CycleKQR uses two encoder-decoder models (Lewis et al., 2020) with parameters denoted as θ and ϕ. To train them, we follow Iterative Back-translation (Hoang et al., 2018), in which the output of each model is used to generate intermediate training data for the other. The core idea behind cycle-consistent training is to jointly train the K2Q and Q2K functions, allowing CycleKQR to align the latent spaces of keyword and question queries. In the following, we describe the two training cycles.

Q-Cycle: The input data is the set of training questions $q \in Q$. For a question q, in the first step, called *generation step* (cf. Figure 2 (a)) we generate an intermediate keyword query by passing q as input to the Q2K model, generating $k' = Q2K(q)$. The resulting output, $\langle q, k' \rangle$, is used to train K2Q, allowing us to back-translate k' into $q' = K2Q(k')$, where q' is supposed to be semantically similar to q. This completes the Q-cycle, where the training is guided by the *reconstruction loss* as the average cross-entropy between q and q', that is back-propagated to the K2Q model:

$$\mathcal{L}_\theta(Q, Q') = -\frac{1}{|Q|} \sum_{q \in Q} \frac{\sum_{i<|q|} p(q_i) \log g(q'_i)}{|q|} \quad (1)$$

where, p and g represent the real and predicted token probabilities, $|q|$ is the question length, $|Q|$ is the total number of training questions, and q_i and q'_i are the i-th token of q and q', respectively.

K-Cycle: K-cycle represents the opposite direction of Q-cycle. For an input keyword query k, K2Q outputs the intermediate question $q' = K2Q(k)$. Similarly, the $\langle k, q' \rangle$ pair is used to train Q2K, which generates the back-translated keyword query $k' = Q2K(q')$. Here too, the reconstruction loss is used to train Q2K model:

$$\mathcal{L}_\phi(K, K') = -\frac{1}{|K|} \sum_{k \in K} \frac{\sum_{i<|k|} p(k_i) \log g(k'_i)}{|k|} \quad (2)$$

where, $|k|$ is the keyword query length, $|K|$ is the number of keyword queries, and k_i and k'_i represent the i-th token in the original and back-translated keyword queries, respectively.

Training: The two cycles are repeated iteratively. First, we run one iteration of Q-cycle with one batch of training questions, followed by an iteration of K-cycle using one batch of training keyword queries. This alternating process in CycleKQR is repeated until both models converge. Both cycles are instrumental in the learning process: Q-cycle trains K2Q to reconstruct a natural question q from a given keyword query k', while K-cycle trains Q2K to generate a keyword query k from a question q'. Preliminary experiments confirm that this training process effectively trains K2Q and Q2K to generate the correct keyword and question queries even on the first iterations. As training continues, the distribution of the intermediate data generated by K2Q and Q2K becomes closer to that of the real keyword and question queries.

5 Keyword-Question Rewriting Dataset

Next, we describe our method for extracting the keyword-question rewriting dataset, namely how we construct the non-parallel sets of keyword and question queries used to train CycleKQR as well as the supervised pairs used for training the QA models and testing all competing approaches.

5.1 Non-Parallel Data Collection

Our approach relies on non-parallel keyword and question queries for training, which we extract from the MS-MARCO (Bajaj et al., 2018) and ORCAS (Craswell et al., 2020) datasets, consisting of natural language questions (1M instances) and keyword queries (10M instances), respectively.

We apply several filtering criteria to generate the training corpus. First, questions from MS-MARCO are filtered using a strategy proposed in (Dror et al., 2013), selecting questions that are syntactically correct, typically starting with a *wh*–word or an auxiliary verb, and contain at least five tokens. Next, keyword queries from the ORCAS dataset are filtered such that they do not start with a *wh*–word or auxiliary verb, and contain less than three words.

To ensure that questions and keyword queries share a similar domain, we compute the cosine similarity across all pairs, and remove those with a similarity less than 0.6. Finally, we randomly sample two separate sets of 100k questions and 100k keyword queries, thus ensuring they are non-parallel. We call this the *non-parallel* corpus (cf. Table 1), and we use it to train CycleKQR.

5.2 Parallel Data Collection

As we are the first to propose the bidirectional rewriting of keyword and question queries, we collect a corpus of aligned keyword and question pairs, used to *evaluate* all the competing rewriting approaches. The parallel corpora come from two main domains: (i) Open-domain QA, and (ii) Knowledge Base QA. Additionally, these two domains represent our extrinsic evaluation strategy, which measures the impact of query rewriting on the respective QA systems.

5.2.1 Open-Domain QA

As in §5.1, we leverage MS-MARCO and ORCAS to construct a parallel corpus of keyword and question pairs. While questions in MS-MARCO are associated with an answer, keyword queries in ORCAS are not. The answer is key in establishing an alignment between the keyword queries and questions that point to the same answer, hence, sharing the same information need.

To align the keyword and question pairs with high accuracy, we propose the following steps.

Filtering We apply the same filtering process described in §5.1, ensuring that questions are syntactically in the correct natural language question form, and similarly keywords do not contain natural language questions. The result is a set of 632k questions and 7M keyword queries.

Name	Source	Train	Dev	Test
Non-parallel	MS-MARCO & ORCAS	100k	10k	-
MS-pairs	MS-MARCO & ORCAS	27k	4.5k	4.5k
QSP-pairs	WebQuestionsSP	2k	1k	1.6k

Table 1: Statistics of our Non-parallel and parallel keyword-question rewriting datasets.

Question to Keyword Matching We next compute the similarity between each question and all keyword queries (Cer et al., 2018). Then, we associate a question to its closest matching keyword query. Due to the low semantic overlap between MS-MARCO questions and ORCAS queries, the extracted pairs are noisy. There is no guarantee that for each question, a keyword query with the same answer is always available. To avoid adding noisy matching pairs into the dataset, we filter pairs by setting a minimum semantic similarity threshold of 0.8, discarding all pairs with lower similarity. Since a natural language question usually expresses

a clear request, matching from question to keyword query also helps remove ambiguous keyword queries. We obtain 62k keyword-question pairs, 38k of which have an answer in MS-MARCO's passage collection.

Manual Alignment To validate and further improve the quality of the generated parallel data, we conduct a human evaluation. For 11k question and keyword pairs, we asked annotators whether the question and keyword query express the same information need. Results showed that 87% (9.1k) of the pairs were deemed to have the same information need. We filtered out the remaining 13% pairs, and the high-quality ones were then divided[3] into two equal sets for *testing* and *validation*. The remaining 27k pairs are used for training. We call this set *MS-pairs* as shown in Table 1.

5.2.2 Knowledge Base QA

For our final dataset, we rely on Web**Q**uestions**SP** (Yih et al., 2016), a popular dataset for Knowledge Base QA, containing 4.7k question-answer pairs. Given its small size, we manually extract the keyword queries from the questions. Keyword extraction was done by proficient annotators with domain expertise, ensuring high data quality. For a given question, the annotators were asked to generate a keyword query expressing the same information need. The annotated data is split into 2k, 1k and 1.6k for training, validation, and test, denoted as *QSP-pairs* in Table 1. Note that all *QSP-pairs* are associated with answers.

6 Experimental Setup

6.1 Evaluation Strategies and Metrics

We evaluate the different competing approaches based on two evaluation strategies.

Intrinsic: Approaches are directly evaluated in terms of correctness and similarity of the rewritten keyword or question query compared to the ground-truth target, as measured through automated metrics such as ROUGE and BLEU, following a similar strategy to Ding and Balog (2018). Additionally, we carry out a small scale analysis using human evaluation to assess the preference of annotators on the rewritten queries for the competing approaches.

[3]Note that the MS-MARCO test set answers are not publicly available. Among these 9107 triples, 5648 questions are from the current MS-MARCO training set and 3459 are from the current development set.

Extrinsic: We measure the downstream impact of rewriting keyword and question queries into the appropriate form for a given QA system. We distinguish between two cases: (1) Question-based QA (QuA), for natural questions; and (2) Keyword-based QA (KeA) for keyword queries. The two cases represent common applications in real-world, e.g., Google and Alexa voice assistants answer natural questions, while many retrieval systems are keyword-based. Both QuA and KeA are treated as *black boxes* (the QA systems cannot be modified).

This evaluation shows the utility of rewriting keyword and question queries into the appropriate form for a target QA, when compared to issuing keyword queries to QuA or questions to KeA. In terms of evaluation metrics, for the open-domain QA scenario, we report MRR and recall metrics, which are the standard evaluation metrics employed in the MS-MARCO Passage Ranking challenge (Choi et al., 2021), whereas for Knowledge Base QA, we report the F1 metric (Ye et al., 2021).

6.2 QA Setup

For the extrinsic evaluation, we choose two state-of-the-art QA systems as the basis for QuA and KeA. In the open-domain setting, we used the BM25 + BERT-base model (Liu et al., 2021), whereas for the Knowledge Base setting, we used RnG-KBQA (Ye et al., 2021), composed of a BERT-based ranker and a T5-based generation model.

We train four QA systems: (1) open-domain QuA, trained using the complete question-answer pairs from MS-MARCO; (2) open-domain KeA, trained using 27k keyword-answer pairs from *MS-pairs*; (3) Knowledge Base QuA and (4) KeA, trained using 2k question-answer and keyword-answer pairs from *QSP-pairs*.

6.3 Models and Baselines

Our approach – `OURS`: We train two versions of CycleKQR, both using the T5 encoder-decoder model (Raffel et al., 2019) as the basis for the K2Q and Q2K functions. For open-domain QA, CycleKQR is trained on the *Non-parallel* dataset from Table 1. For Knowledge Base QA, we distinguish between two configurations: (1) A clean configuration $\texttt{OURS}_C$, where CycleKQR is trained using only *QSP-pairs*;[4] and (2) combining *non-parallel* and *QSP-pairs* (a more realistic scenario), resulting in a level of noise that is comparable to the open-

[4]Unpaired questions and keyword queries for training.

domain setting (OURS$_R$). The two configurations allow us to assess whether our approach benefits from additional training data despite higher noise.

Supervised Baseline – SB: Here we train K2Q and Q2K without cycle-consistent training. This corresponds to a standard supervised approach, where each input is paired with an output. For this purpose, we use the questions and keyword queries from *Non-parallel*, which we automatically pair using the semantic similarity strategies described in §5.2. We follow the same strategy as in OURS and train three separate models: SB for open-domain QA, SB$_C$ and SB$_R$ for Knowledge Base QA.

Semantic Similarity – SIM: It replaces questions with keyword queries and vice versa, following the same strategy we used to match questions to keyword queries (see §5.2). Since the *MS-pairs* test set was generated by pairing together high-similarity questions and keyword queries, it is biased in favor of this baseline: for each question, it will simply retrieve the paired keyword query, and vice versa. To avoid this bias, we created an auxiliary test set specifically for evaluating SIM. For QuA we randomly sampled 112 keyword queries from ORCAS, and asked experts to manually assign an answer from the MS-MARCO passage corpus. Each query is matched to its most similar question from MS-MARCO. For KeA, we randomly sampled 7k different questions from MS-MARCO, and matched them to the keyword queries in ORCAS. Differently from the data collection in §5, the pairs were not filtered in any way. The results obtained using this auxiliary set are labeled with the A subscript, e.g. OURS$_A$ and SIM$_A$. Note that no auxiliary test set is needed for the Knowledge Base QA setting, since the *QSP-pairs* test set is generated using a different approach.

Stop Word Removal – STWR: This unsupervised baseline generates a keyword query from a question by removing stop words.[5]

Upper bound – UPPER: The QA system receives ground-truth queries in their correct form (ideal scenario), e.g., questions to QuA, keyword-based queries to KeA.

Lower bound – LOWER: Contrary to UPPER, the QA system receives the query in an unfamiliar form, keyword queries to QuA, questions to KeA.

[5]A subset of the NLTK stop word corpus was used. We kept wh-words, as they are important for query understanding.

6.4 Research Questions

RQ1: Can cycle-consistent training be applied to train K2Q and Q2K without annotations?

RQ2: Can the QA performance be increased through question-keyword rewriting?

7 Evaluation Results

(a) Open-Domain QA metrics

	Q2K			K2Q		
	ROUGE-1	**ROUGE-L**	**BLEU**	**ROUGE-1**	**ROUGE-L**	**BLEU**
STWR	0.669	0.630	0.359	-	-	-
SB	0.863	0.832	0.744	0.741	0.718	0.600
OURS	0.769	0.715	0.588	0.717	0.687	0.557

(b) Knowledge Base QA

	Q2K			K2Q		
	ROUGE-1	**ROUGE-L**	**BLEU**	**ROUGE-1**	**ROUGE-L**	**BLEU**
STWR	0.663	0.579	0.367	-	-	-
SB$_C$	0.813	0.771	0.652	0.744	0.720	0.562
OURS$_C$	0.781	0.733	0.598	0.732	0.702	0.542
SB$_R$	0.787	0.749	0.614	0.708	0.684	0.519
OURS$_R$	0.764	0.716	0.566	0.706	0.680	0.512

Table 2: Intrinsic evaluation results (automatic metrics).

	Open Domain		Knowledge Base	
	K2Q	Q2K	K2Q	Q2K
OURS$_R$	12	11	7	14
SB$_R$	7	7	6	5
Tie	31	32	37	30

Table 3: Human preference for rewrites.

7.1 Intrinsic Evaluation

Table 2 reports ROUGE and BLEU scores for the competing approaches calculated on the test sets of *MS-pairs* and *QSP-pairs*. Results show that K2Q is harder than Q2K. Both OURS and SB have the highest scores, with SB obtaining higher scores.

The higher scores of SB are intuitive, given that the approach is trained in a supervised manner, and it is optimized to generate the target output query. Note that SB can potentially overfit on its training data, and can be affected by noisy pairs generated by the automatic matching process. This is not necessarily the case for OURS, due to its cycle-consistent training. However, automated natural language generation metrics such as ROUGE and BLEU, cannot capture possible lexical variability in rewriting, which may lead to lower scores.

To verify this, through a user study we evaluated with domain experts the rewriting quality. Given an input query, the annotator is asked to choose the better rewrite in terms of semantic preservation and syntactic correctness among OURS and SB.[6] Table 3 shows the annotator preferences for the

[6]Rewrite order is shuffled, and annotators are unaware which model produced the rewrite.

rewrites on 100 questions (50 each extracted from *MS-pairs* and *QSP-pairs*) and 100 keyword queries (same as above). On both QA domains, rewrites from OURS are significantly[7] preferred over SB.

Results from this section validate RQ1. Table 2 showed that OURS achieves high automated metric scores, whereas Table 3 shows that our rewrites are significantly preferred by human annotators. Examples from our models are shown in Appendix A.

7.2 Extrinsic Evaluation

7.2.1 Open-domain QA

Table 4 reports the downstream impact of the different rewriting approaches on QA performance in open domain for QuA and KeA.

K2Q for QuA. Directly issuing keyword queries to QuA results in noticeably lower MRR compared to using natural questions (0.184 vs. 0.229 MRR@10).[8] This shows that QuA is negatively affected by unfamiliar query forms. By rewriting keyword queries into questions, OURS improves QuA performance for both MRR and recall (MRR@10=0.189, R@10=0.415, MRR@100=0.206, R@100=0.842) over LOWER and SB. OURS obtained up to 4% relative improvement in MRR over SB. This confirms our hypothesis that rewriting keyword queries into questions improves QuA.

Cycle-consistent training allows our K2Q model to keep the keyword query intent, while the weakly-supervised K2Q may change its meaning, due to the noisy training data.

Q2K for KeA. Unlike for K2Q, here UPPER obtains lower scores than LOWER for both MRR and recall. Although OURS significantly improves QA performance over SB (increase of +10.4% for MRR@10), it does not show improvement over LOWER. Theoretically, the performance of UPPER should be always better than that of LOWER, since KeA is trained to fit keyword queries in both recall and MRR. Two factors can be attributed to this outcome: (1) Our KeA is trained using BERT, which interprets better questions even after being fine-tuned on keywords;[9] and (2) the test data may contain noise that prevents us from obtaining accurate results. Despite this, Table 4 shows that OURS outperforms all other baselines and LOWER in recall.

We also compare our approach against a simple strategy based on semantic similarity (SIM), shown as OURS_A and SIM_A in Table 4. OURS outperforms SIM since SIM heavily relies on having the correct question/keyword for each input. When such a question/keyword query is not available, SIM will inevitably fail. The poor performance of SIM also confirms the low overlap between MS-MARCO questions and ORCAS keyword queries discussed in §5.2: despite the size of MS-MARCO and ORCAS, and the effectiveness of USE embeddings, a semantically-equivalent question/keyword query is often missing. This also justifies the filtering and human annotation steps applied to ensure the quality of the *MS-pairs* test set.

Open-domain QA Rewriting Examples. Given the keyword query "student loans without a cosigner" shown in Table 5 (a), OURS can reformulate it into "how to get student loans without a cosigner?". The rewritten question improves the ranking of the correct passage from the 14th to 7th rank, matching UPPER. SB on the other hand, generates the question "how do you get a cosigner on student loans?", which lowers performance as the meaning of the request has been changed.

For the question "how to eject disc from hp laptop?" in Table 5 (b), OURS generates "eject disc from hp laptop". This simple transformation improves the ranking of the correct answer from 17th to 6th position, while SB removed important information (brand of the laptop).

7.2.2 Knowledge Base QA

Here, we show the QA performance on two settings, which vary on the data used to train OURS and SB: (1) clean setting, where the models (OURS_C and SB_C) are trained using 2k training *QSP-pairs*, and (2) realistic scenario, which additionally adds 100k keyword and question queries from the open domain QA for training OURS_R and SB_R. Note that this realistic setting contains noise.

K2Q for QuA. We again observe that UPPER has much higher performance, with F1=0.762 compared to LOWER with F1=0.698. It further confirms that an unfamiliar query form causes low QA performance. All OURS models

[7] Measured per Wilcoxon signed-rank test with $p = 0.034$

[8] Our MRR scores are lower than those in MS-MARCO leaderboard. For comparison, the pre-trained OpenMatch model (state-of-the-art) obtains MRR@10=0.235 on *MS-pairs* test set. We conclude that the performance difference is due to the use of a different test set.

[9] We tried to verify this by training an LSTM model, however, it did not obtain reasonable rewriting performance.

	QuA(K2Q Task)				KeA(Q2K Task)			
	MRR@10	MRR@100	R@10	R@100	MRR@10	MRR@100	R@10	R@100
UPPER	0.229	0.245	0.492	0.9	0.189	0.205	0.411	0.829
LOWER	0.184*	0.201*	0.411	0.839	**0.205**	**0.221**	0.428	0.842
STWR	-	-	-	-	0.199	0.216	0.425	0.839*
SB	0.182*	0.198	0.401	0.829*	0.183*	0.200*	0.397*	0.824*
OURS	**0.189** (3.8%↑)	**0.206** (4.0%↑)	**0.415** (3.5%↑)	**0.842** (1.6%↑)	0.202	0.218	**0.431** (8.6%↑)	**0.849** (3.0%↑)
SIM_A	0.142*	0.154*	0.146*	0.537*	0.145*	0.159*	0.322*	0.712*
OURS_A	**0.402**	**0.411**	**0.389**	**0.837**	**0.268**	**0.282**	**0.524**	**0.869**

Table 4: Open-domain QA results. Best results are highlighted in bold. * means that OURS is significantly better than this baseline.

	Approach	Query	Rank
(a)	Original Question	how to get student loans without a cosigner?	7
	Original Keywords	student loans without a cosigner	14
	SB	how do you get a cosigner on student loans?	19
	OURS	how to get student loans without a cosigner?	7
(b)	Original Keywords	eject cd from hp laptop	4
	Original Question	how to eject disc from hp laptop?	17
	STWR	how eject disc hp laptop	37
	SB	eject disc from laptop	155
	OURS	eject disc from hp laptop	6

Table 5: Rewriting examples for Open-domain QuA (a) and KeA (b). More examples are shown in Appendix A.

	QuA F1	KeA F1
UPPER	0.762	0.757
LOWER	0.698*	0.710*
STWR	-	0.705*
SIM	0.429*	0.532*
SB_C	0.739*	**0.736**
OURS_C	**0.746**	0.731
SB_R	0.731*	0.685*
OURS_R	**0.751** (2.7%↑)	0.713

Table 6: Knowledge Base QA results. The best results are highlighted in bold. * means that at least one of our approaches scores significantly higher than this baseline.

perform close to UPPER and better than SIM and LOWER. In both clean and realistic settings, we observe that OURS performs better than SB, with a 2.7% relative improvement in the latter.

Q2K for KeA. Similarly, we obtain positive results in KeA. In the clean setting, OURS_C obtains F1=0.732, and outperforms all baselines, except for SB_C, which obtains F1=0.736 (insignificant difference). This is intuitive, given that SB_C has access to accurate paired training data. On the other hand, OURS_R performs better than the supervised baseline with F1=0.716 compared to SB_R, which obtains F1=0.685, a 7.5% relative improvement. This indicates that our approach is more resilient to noise compared to the supervised one. This is because the objective of cycle-consistency training is to learn the semantic representations of questions and keyword queries by reconstructing them from noisy data, rather than comparing against a specific target output.

Rewriting in Knowledge Base QA. Table 7 (a) shows an example query, "emily osment highschool", and its rewrites. OURS infers the question "where did emily osment go to highschool?", matching the performance of SB_C. On the other hand, the noisy training data caused SB_R to change the intent of the input, while SIM failed to find a matching question from MS-MARCO. Table 7 (b) shows the example question "who owns chrysler corporation 2011?", which is correctly understood by OURS, rewriting it into the keywords "chrysler corporation owner 2011". This example confirms that Q2K is a non-trivial task, differing from simple keyword extraction: the rewritten keywords are not simply a subset of the input, but a coherent query in which linguistic expressions must also be converted into appropriate keywords.

Finally, the results in §7.2.2 and §7.2.1 validate RQ2, showing that rewriting queries into their appropriate form has a significant impact on QA performance. Furthermore, rewritten queries from OURS trained based on cycle-consistent training outperforms strong baselines such as SB.

8 Conclusion

We introduced the bidirectional keyword-question rewriting task, which improves QA performance by rewriting queries into the desired form for a given QA system. Furthermore, we presented CycleKQR, which learns the K2Q and Q2K functions simultaneously through cycle-consistent training in an unsupervised manner. CycleKQR allows for the two rewriting functions to be less susceptible to noise coming from unsupervised data, and can be easily adapted to different QA systems and scenarios, for which non-parallel data is available.

	Approach	Query	F1
(a)	Original Question	what highschool did emily osment go to?	1.0
	Original Keywords	emily osment highschool	0.67
	SIM	was emily kinney in harry potter?	0
	SB_C	where did emily osment go to high school?	1.0
	$OURS_C$	what highschool did emily osment go to?	1.0
	SB_R	how old is emily osment?	0
	$OURS_R$	where did emily osment go to highschool?	1.0
(b)	Original Keywords	chrysler corporation owner 2011	0.89
	Original Question	who owns chrysler corporation 2011?	0
	STWR	who owns chrysler corporation 2011	0
	SIM	chrysler corporation headquarters	0
	SB_C	chrysler corporation owner 2011	0.89
	$OURS_C$	chrysler corporation owner 2011	0.89
	SB_R	chrysler corporation 2011	0
	$OURS_R$	chrysler corporation owner 2011	0.89

Table 7: Rewriting examples for Knowledge Base QuA (a) and KeA (b).

We carried out a detailed evaluation of different competing rewriting approaches for two QA scenarios (Open Domain and Knowledge Base), assessing their performance through an intrinsic and extrinsic evaluation. Experimental results show that rewriting queries into the correct form improves QA performance, and that CycleKQR is able to provide highly accurate rewrites, which retain the original query intent and improve the ranking of the correct answer for the underlying QA system.

Finally, we contributed a keyword-question rewriting dataset, consisting of a total of 66k keyword-question pairs which can be used to facilitate research on keyword-question rewriting.

9 Limitations

A notable limitation of this work is that we do not explicitly investigate how to handle ambiguous keywords (vague or broad queries that could be mapped to multiple different questions) and paraphrases (Q-K pairs with equivalent meaning but different surface forms). Both of these are active research areas in the community. We addressed this with the simplifying assumption introduced in §5.1 that sets a minimum semantic similarity threshold between our Q and K data, as determined by an existing sentence encoder. We believe that this is a key future direction in this line of research. The ambiguity issues requires the detection of such keyword queries, and specific mechanisms to generate diverse question interpretations. The paraphrasing problem can be addressed by diversifying the non-parallel data selection process.

Furthermore, our extrinsic evaluation focused on K2Q for QA under the assumption that bidirectional cycle training leads to improved K2Q performance, and we did not include a specific extrinsic application of Q2K. One example of such an application is query expansion for questions, where the original question can be augmented with additional keywords coming from a Q2K model that uses diversity-based or lexically-constrained sampling to generate keywords that are not in the question itself. Due to space limitations, we leave the exploration of this direction for future work.

In the current work we did not consider a supervised cycle-training setting. While the unsupervised setting allows us to leverage non-parallel data, we believe that supervised fine-tuning (even in a few-shot setting) could deliver further improvements. We leave this for future work.

Finally, for deploying our proposed approach in real-world scenarios (e.g. in a search engine or voice assistant), we must consider the task of how to decide when to apply either Q2K or K2Q to an input. This is relevant to our bidirectional approach, as well as previous work on unidirectional transformations. We did not discuss this component in the paper for reasons of space, but we do not believe that this is a significant challenge. The most straightforward approach is to employ a question classifier to determine if an input is a fully-formed question, and use this information to decide whether to apply Q2K or K2Q. Such an approach is common in practical settings, and other heuristics such as input length are also commonly used. We plan to further explore this area, including empirical evaluation of binary classification of K vs. Q, as part of future work.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268 [cs]*. ArXiv: 1611.09268.

Rahul Bhagat and Eduard Hovy. 2013. What Is a Paraphrase? *Computational Linguistics*, 39(3):463–472.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, and Chris Tar. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Donghyun Choi, Myeongcheol Shin, Eunggyun Kim, and Dong Ryeol Shin. 2021. Adaptive Batch Scheduling for Open-Domain Question Answering. *IEEE Access*, 9:112097–112103. Conference Name: IEEE Access.

Nick Craswell, Daniel Campos, Bhaskar Mitra, Emine Yilmaz, and Bodo Billerbeck. 2020. ORCAS: 18 Million Clicked Query-Document Pairs for Analyzing Search. *arXiv:2006.05324 [cs]*. ArXiv: 2006.05324.

Heng Ding and Krisztian Balog. 2018. Generating Synthetic Data for Neural Keyword-to-Question Models. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 51–58, Tianjin China. ACM.

Gideon Dror, Yoelle Maarek, Avihai Mejer, and Idan Szpektor. 2013. From query to question in one click: suggesting synthetic questions to searchers. In *Proceedings of the 22nd international conference on World Wide Web*, pages 391–402.

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, Hong Kong, China. Association for Computational Linguistics.

Tommi Gröndahl and N. Asokan. 2020. Effective writing style transfer via combinatorial paraphrasing. *Proc. Priv. Enhancing Technol.*, 2020(4):175–195. Number: 4.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training. *arXiv preprint arXiv:2006.04702*.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24.

Tom Hosking and Mirella Lapata. 2021. Factorising Meaning and Form for Intent-Preserving Paraphrasing. 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 1405–1418, Online. Association for Computational Linguistics.

Andrea Iovine, Anjie Fang, Besnik Fetahu, Oleg Rokhlenko, and Shervin Malmasi. 2022. CycleNER: An Unsupervised Training Approach for Named Entity Recognition. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*, page 9, Virtual Event, Lyon, France.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Foaad Khosmood. 2012. Comparison of sentence-level paraphrasing approaches for statistical style transformation. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer ....

Johannes Kiesel, Arefeh Bahrami, Benno Stein, Avishek Anand, and Matthias Hagen. 2018. Toward voice query clarification. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1257–1260. ACM.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating Unsupervised Style Transfer as Paraphrase Generation. *arXiv:2010.05700 [cs]*. ArXiv: 2010.05700.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

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.

Zhenghao Liu, Kaitao Zhang, Chenyan Xiong, Zhiyuan Liu, and Maosong Sun. 2021. OpenMatch: An Open Source Library for Neu-IR Research. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2531–2535.

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Zhong Meng, Jinyu Li, Yifan Gong, and Biing-Hwang (Fred) Juang. 2018. Cycle-Consistent Speech Enhancement. In *Proc. Interspeech 2018*, pages 1165–1169.

Tasnim Mohiuddin and Shafiq Joty. 2019. Revisiting adversarial autoencoder for unsupervised word translation with cycle consistency and improved training. *arXiv preprint arXiv:1904.04116*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. 2021. Question rewriting for conversational question answering. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 355–363. ACM.

Ryen W. White, Matthew Richardson, and Wen-tau Yih. 2015. Questions vs. Queries in Informational Search Tasks. In *Proceedings of the 24th International Conference on World Wide Web*, pages 135–136, Florence Italy. ACM.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2021. RnG-KBQA: Generation Augmented Iterative Ranking for Knowledge Base Question Answering. *arXiv:2109.08678 [cs]*. ArXiv: 2109.08678.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The Value of Semantic Parse Labeling for Knowledge Base Question Answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Shiqi Zhao, Haifeng Wang, Chao Li, Ting Liu, and Yi Guan. 2011. Automatically generating questions from queries for community-based question answering. In *Proceedings of 5th international joint conference on natural language processing*, pages 929–937.

Zhicheng Zheng, Xiance Si, Edward Y. Chang, and Xiaoyan Zhu. 2011. K2Q: generating natural language questions from keywords with user refinements. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*, pages 947–955. The Association for Computer Linguistics.

Appendix

A Examples of CycleKQR Output

Tables 8 and 9 show example outputs of the K2Q and Q2K models trained with CycleKQR for the open-domain setting. Tables 10 and 11 show example outputs of the K2Q and Q2K models trained with CycleKQR for the knowledge-based setting.

Input Keywords	Output Question
disney epcot attractions	what attractions are at disney epcot?
kennel cough duration	how long does kennel cough last?
honda civic transmission fluid change	how to change transmission fluid on honda civic?
lipids blood test	what blood test measures lipids?
foods that cause kidney stones	what foods cause kidney stones?
filing taxes for previous years	how to file taxes for previous years?
golgi apparatus function	what does the golgi apparatus do?
led tv power consumption watts	how many watts does a led tv consume?
convert liters to grams	how to convert liters to grams?
castor oil to grow eyelashes	can castor oil be used to grow eyelashes?

Table 8: Output examples of OURS K2Q on the open domain setting.

Input Question	Output Keywords
how many calories to cut to lose 1 pound	calories to cut to lose 1 pound
how to solve a scrambled rubik's cube	solving a scrambled rubik's cube
what does the name cora mean	name cora meaning
what foods should i eat to help lower cholesterol	foods to help lower cholesterol
what is the max dosage of ibuprofen	max dosage of ibuprofen
what time does chase bank open today	chase bank opening time today
who is shaun white's girlfriend	shaun white girlfriend
how effective has the flu vaccine been	flu vaccine effectiveness
what hours of the day can you forex trade	forex trading hours of the day
what causes hot flashes	hot flashes causes

Table 9: Output examples of OURS Q2K on the open domain setting.

Input Keywords	OURS$_C$	OURS$_R$
niall ferguson wife	who is niall ferguson's wife?	who is niall ferguson's wife?
smokey robinson songs	what songs did smokey robinson sing?	what songs did smokey robinson sing?
beethoven music period	what period did beethoven play music?	what music period was beethoven in?
ravens last super bowl win	what was the ravens last super bowl win?	when did the ravens last win a super bowl?
phillies spring training stadium location	where is the phillies spring training stadium?	where is the phillies spring training stadium located?
prom night kellan lutz character	what character did kellan lutz play in prom night?	what character did kellan lutz play in prom night?
shaq first year in the nba	what was shaq first year in the nba?	when was shaq first in the nba?
john edwards crime	what crime did john edwards commit?	what crime did john edwards commit?
jackie robinson first team	who did jackie robinson play for first?	what team did jackie robinson play for first?
religions in malaysia	what religions do malaysia practice?	what religions do they practice in malaysia?

Table 10: Output examples of OURS K2Q on the knowledge-based setting.

Input Question	OURS$_C$	OURS$_R$
where is jamarcus russell from?	jamarcus russell hometown	jamarcus russell hometown
what type of music did john lennon sing?	john lennon music type	john lennon music type
what was the book written by charles darwin?	charles darwin book	book written by charles darwin
what states make up the midwest us?	states that make up the midwest us	midwest us states
where did alexander graham bell die?	alexander graham bell death place	alexander graham bell place of death
what is the sacred text of daoism?	sacred text of daoism	sacred text of daoism
what is rihanna mum called?	rihanna mum name	rihanna mum called
who does owen schmitt play for?	owen schmitt team	owen schmitt play for
what county is frederick md in?	frederick md county	frederick md county
what type of government does france use?	type of government in france	france type of government

Table 11: Output examples of OURS Q2K on the knowledge-based setting.