Published as a conference paper at ICDE 2021

Query Rewriting via Cycle-Consistent Translation for E-Commerce Search

Yiming Qiu[1†], Kang Zhang[1†], Han Zhang[1], Songlin Wang[1], Sulong Xu[1], Yun Xiao[2], Bo Long[1], Wen-Yun Yang[2*]
[1]JD.com, Beijing, China
[2]JD.com Silicon Valley Research Center, Mountain View, CA, United States

***Abstract*—Nowadays e-commerce search has become an integral part of many people's shopping routines. One critical challenge in today's e-commerce search is the semantic matching problem where the relevant items may not contain the exact terms in the user query. In this paper, we propose a novel deep neural network based approach to query rewriting, in order to tackle this problem. Specifically, we formulate query rewriting into a cyclic machine translation problem to leverage abundant click log data. Then we introduce a novel cyclic consistent training algorithm in conjunction with state-of-the-art machine translation models to achieve the optimal performance in terms of query rewriting accuracy. In order to make it practical in industrial scenarios, we optimize the syntax tree construction to reduce computational cost and online serving latency. Offline experiments show that the proposed method is able to rewrite hard user queries into more standard queries that are more appropriate for the inverted index to retrieve. Comparing with human curated rule-based method, the proposed model significantly improves query rewriting diversity while maintaining good relevancy. Online A/B experiments show that it improves core e-commerce business metrics significantly. Since the summer of 2020, the proposed model has been launched into our search engine production, serving hundreds of millions of users.**

***Index Terms*—query rewriting, neural networks, neural machine translation, e-commerce search**

I. Introduction

Over recent decades, online shopping platforms (e.g., eBay, Walmart, Amazon, Tmall, Taobao and JD) have become increasingly popular in people's daily life. E-commerce search, which helps users to find what they need from billions of products, is an essential part of those platforms, contributing to the largest percentage of transactions among all channels. For instance, the top e-commerce platforms in China, *e.g.*, Tmall, Taobao and JD, serve hundreds of million active users with gross merchandise volume of hundreds of billion US dollars.

However, a lot of e-commerce search queries do not have satisfactory results from traditional search engines. This is due to the nature of e-commerce search: a) item titles are often short, thus hard for the inverted index to retrieve, b) significant numbers of new internet users tend to make natural language alike search queries, *e.g.*, "cellphone for grandpa", "gift for girlfriend", c) polysemous queries are more common in e-commerce search, *e.g.*, "apple" could mean Apple company's products or the fruit apple. Based on our internal analysis of real log data, these three cases causes most of the unsatisfactory search results in our company's search engine, one of the largest e-commerce search platform in the world.

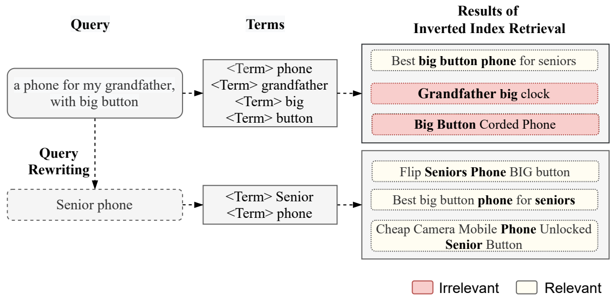

Query
Terms
Results of Inverted Index Retrieval
a phone for my grandfather, with big button
<Term> phone
<Term> grandfather
<Term> big
<Term> button
Best big button phone for seniors
Grandfather big clock
Big Button Corded Phone
Query Rewriting
Senior phone
<Term> Senior
<Term> phone
Flip Seniors Phone BIG button
Best big button phone for seniors
Cheap Camera Mobile Phone Unlocked Senior Button
Irrelevant
Relevant

Fig. 1. Illustration of query rewriting process that retrieves more relevant results.

The traditional search engine typically performs the candidate retrieval stage using an inverted index, which is built to efficiently retrieve candidate items based on term matching. This stage greatly reduces the number of candidates from billions to thousands. It is a core step in the search engine. However, due to the term mismatch between query and item titles, the candidate retrieval stage contributes most to the failure cases in our production.

In academia, *semantic matching* refers to this kind of term mismatch problems, *i.e.*, semantically relevant items cannot be retrieved by inverted indexes, since the item title does not contain the exact terms of a query. This is particularly common in e-commerce search because item titles are often short. We notice that a lot of long-tail queries or natural language alike queries are failed in this way. As an example, it is almost impossible to retrieve items titled "senior mobile phones" for a query "cellphone for grandpa".

Traditional web search technology employs the rule-based query rewrite, which transforms the original query to a similar but more standard query, to solve the above hard queries. Figure 1 illustrates an example of how this query rewriting works in practice. Those query rewriting rules are normally from a few sources: human compilation, data aggregation and so on. However, these rule-based approaches need lots of human efforts that are costly and time-consuming, and they can not cover more subtle cases and long-tail cases. Therefore, there is a significant need for an advanced and powerful system

[†] Both authors contributed equally
[*] Corresponding author at wenyun.yang@jd.com

that can solve this problem in a more scalable fashion.

Recently, there is another trend of learning embedding representation to solve this term mismatch problem [1] and recommendation problems [2], [3]. The basic idea is to mapping queries or users, and item titles to a semantic embedding space, where the queries are close to the relevant items. Therefore, items that do not contain the exact terms but semantically relevant to the query, can be retrieved by the nearest neighbor search in the embedding space. However, in practice, we find this approach suffers from other drawbacks: 1) it is hard to balance the semantic matching ability and too much generalization, which could retrieve irrelevant items. For example, a search query with a very specific intention of a certain model or style of a necklace could retrieve other models or styles of the necklace. 2) It is hard to decide how many items to retrieve from the nearest neighbor search. For some long tail queries with a very specific intention, the number of available and relevant items could be much less than a hyperparameter value, the number of nearest neighbors to retrieve. Thus, the extra retrieved items could cause burdens for the later relevant scoring stages.

In this paper, we will develop a novel approach to the *semantic matching* problem from another perspective of automated and scalable query rewriting. We formulate the query rewriting problem into a cyclic machine translation problem, that first translates query to item titles, and then translates back to queries. To guide the cyclic translation optimally adapt to query rewriting task, we also introduce a novel optimization term to encourage the cyclic translation "translates back" to the original query. Our model is flexible enough to leverage most state-of-the-art neural machine translation (NMT) models, *e.g.*, attention-based NMT [4] and transformer-based NMT [5], both of which are based on an encoder-decoder architecture. In practice, we choose the transformer model structure as a skeleton of our query rewriting model, since the transformer has shown superior translation quality and the ability to leverage GPU parallel operations.

The main contributions of this paper can be summarized as follow

- In Section III, we present our methods, including models, training and inference algorithms, and system optimizations. Specifically, we develop a novel deep neural network model to query rewriting, which is composed of a cyclic translation formulation of query rewriting in Section III-B, a cyclic consistency likelihood in Section III-C, an approximated and efficient training algorithm in Section III-D, an optimal inference algorithm in Section III-E, a set of sequence decoding techniques in Section III-F, a few tradeoffs for online serving in Section III-G and a few system optimizations in Section III-H.
- In Section IV, we conduct extensive experiments including, ablation studies in Section IV-B to clearly illustrate how the models work to generate high quality rewritten queries, offline experiments Section IV-C to show that cyclic consistency helps improving the performance in, and online A/B experiments in Section IV-D to show that the proposed method is able to improve users' experience significantly.
- In Section V, we conclude our contributions and discuss a few challenging aspects that we have also explored in this query rewriting problem. We will take those directions as our future work, and we also look forward to inspiring more researchers to work on those problems collectively for great practical impacts.

II. Related works

Our work leverages state-of-the-art neural machine translation models in a novel manner to solve the long existing query rewriting problem, as a counterpart of the embedding based retrieval approach. Thus, we review recent progress in neural machine translation and embedding retrieval. Also, we review classic query rewriting approaches for references.

A. Neural Machine Translation

Neural Machine Translation (NMT) based on a neural network encoder-decoder architecture recently surpasses its precedent statistical machine translation (SMT) as state-of-the-art for machine translation [6]–[8]. Specifically, the encoder learns a fixed-length embedding as a representation for any token sequences in the source language, which is then used by the decoder to output tokens in the target language. A few years ago, NMT models are usually based on complex recurrent neural network (RNN), long short-term memory (LSTM) [9] and gated recurrent unit (GRU) [10] to capture the sequential information. Later, attention mechanism [4] between encoder and decoder are introduced to further improve the performance. More recently, researchers propose to use a pure attention mechanism without any RNN structure, *i.e.*, transformer, to achieve state-of-the-art performance for machine translation [5]. The attention mechanism is able to capture global dependencies within the source and the target, and interactions between them. In addition, extra position embedding is added to token embeddings to carry the order information of each token. So far, the transformer has been widely adopted in not only machine translation, but also almost all areas of natural language processing. Several state-of-the-art natural language processing (NLP) models, GPT [11], GPT2 [12], GPT3 [13] and BERT [14], are all based on transformer structure.

Beyond the standard encoder-decoder structure, Tu et al. [15] add a novel reconstructor block to ensure the target sentence contains "all information" of the input. Specifically, the reconstructor is trained to translate the target sentence back to the source. The concept of the back-translation is similar to our idea of cyclic translation. This mechanism helps the decoder keep all useful information from the source sentence since otherwise, the reconstructor has an insufficient message to translate back to the source.

B. Embedding Retrieval in Search Engine

Recently, embedding retrieval technologies have been widely adopted in the modern recommendation and advertising systems [2], [3], [16], while have not been widely used in search engine yet. We find a few works about retrieval problems in search engine [17], [18], while they have not been applied to the industrial production system. DPSR [1] is one of the first practical explorations in this direction of applying embedding retrieval in the industrial search engine system. Moreover, the well known DSSM [19] and its following work CDSSM [20] have pioneered the work of using deep neural networks for relevance scoring, which is a very different task from embedding retrieval though.

C. Classic Approaches to Query Rewriting

Query expansion, as part of query rewriting, is generally used to formulate a raw query into another one with explicit meaning by adding extra information. In the literature, Bhogal and his colleagues [21] proposed an ontology-based approach to capture keywords semantics to improve query representation. As shown in another study [22], ontologies may improve the searching performance as an annotated corpus. Moreover, thesaurus-based query rewriting is another competitive approach. The company eBay deploys an adaptive approach to generate promising synonym queries, which relies on external resources such as WordNet [23]. These methods heavily rely on the quality of external resources, which makes them not salable to more general circumstances.

The Simrank algorithm [24] proposes to construct an object graph to compute similarities. Each edge represents a preference relationship between two objects. The underlying assumption is that similar objects should have common preferences. Thus, similar objects can be identified by the number of shared preferences. As for query rewriting, Antonellis proposed Simrank++ [25] to generate similar queries using user click logs, by assigning weight to each edge according to its total number of clicks. However, this method is not scalable to the current industrial scale of data, which prevents it from widely adopted in modern e-commerce search engine.

More recently, He et al. [26] propose a "learning to rewrite framework" for the industrial scenario. They take advantage of multiple query rewriters to generate query rewriting candidates, then rank these queries to get the final results. Thus, they do not actually propose a model to generate the rewritten query by itself.

Our approach differs from the above methods in that we train state-of-the-art neural machine translation models, which are capable of leveraging large industrial scale click log data. Apart from the algorithm part, the system of query rewriting is also important.

III. Method

In this section, we first introduce an overview of the query rewriting problem and its underlying motivations in Section III-A. Then, we develop the proposed method step by step as follows: in Section III-B we formulate query rewriting as a cyclic machine translation problem that could already generate reasonable query rewriting results, in Section III-C we introduce a novel cyclic consistency loss that can improve the query rewriting performance, in Section III-D we present an efficient training algorithm that can optimize the originally intractable loss function. Next, we introduce the model inference method as follows: in Section III-E we formulate the inference workflow, which, together with an inference trick presented in Section III-F, can generate more diverse query rewriting results. Finally, we discuss some practical issues that we find particularly important to deploy the proposed method into an industrial system: in Section III-G we talk about some tradeoffs that one has to make between inference speed and accuracy to fully deploy the model, and in Section III-H we introduce another system optimization technique for downstream inverted index retrieval fed with query rewriting results.

A. Overview

Query rewriting, which aims at rewriting a given query to another query that can retrieve more relevant items, is a critical task in modern e-commerce search engines. The major reason is that item descriptions and user queries are created by different sets of people, who may use different vocabularies and distinct language styles. Consequently, even when the queries can perfectly match users' information needs, the e-commerce search engines may be still unable to locate relevant items. For example, users who are unfamiliar with Apple's new products, might just type "newest iPhone" in an e-commerce website's search input box. However, those relevant items are actually indexed with terms such as "iPhone 12" or "iPhone 12 Pro" at this moment. Ideally, we would like to learn a model which can automatically do the following query rewriting.

newest iPhone	iPhone 12
↑	↑
model input	model output

The straightforward way to learn a query-to-query translation model from a given data set of enough query rewriting logs. As one of the largest e-commerce platforms, we still can not collect a sufficient number of high-quality query rewriting logs, since the raw query rewriting logs are not of guaranteed quality. Thus, human labeling or rule-based methods are necessary to extract a small number of high-quality data, which becomes too expensive and impractical to train deep learning models. Thus, to overcome this obstacle, we have to resort to some other approaches to query rewriting.

One can easily find out that the availability of a tremendous amount of click log data could be a rescue to the problem, though the click logs are actually query-to-title data. How to utilize this type of data to learn the query-to-query translation model would be an interesting and potentially promising direction to explore. Basically, as illustrated in the left part of Figure 2, the intuition is that we can train two translation models, the forward (query-to-title) model and the backward (title-to-query) model, to accomplish this task. However, two separately trained translation models are not going to be optimal in terms of generating the best rewritten queries. In Section III-C, we will talk about how to jointly train the two translation models to get the optimal rewritten queries.

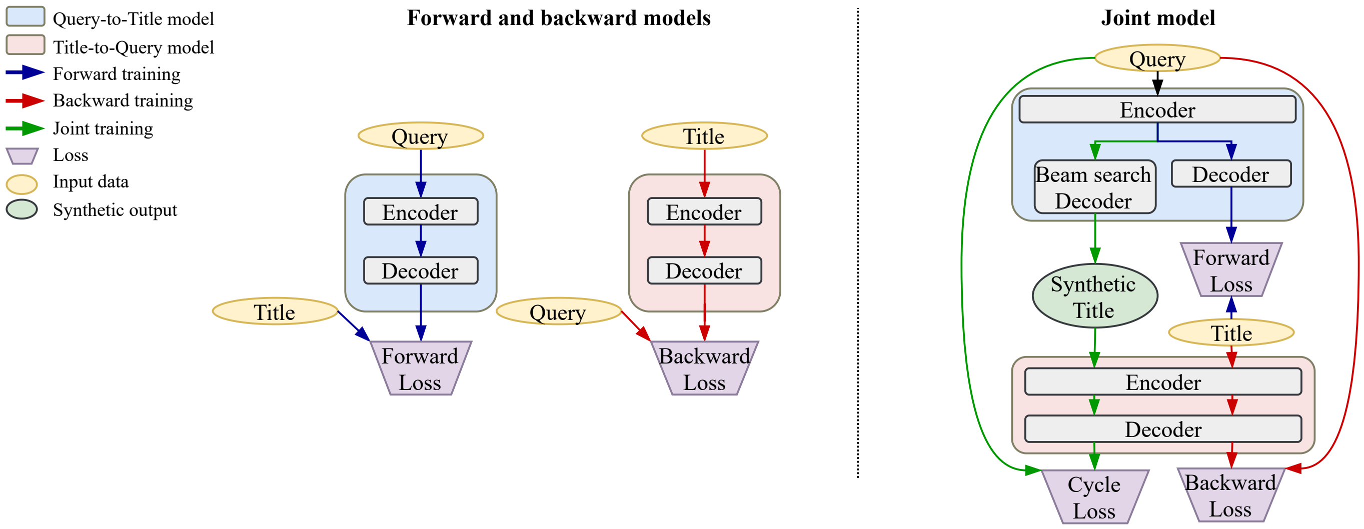


Fig. 2. Illustration of the separately trained forward (query-to-title) and backward (title-to-query) models and the jointly trained model.

B. Query Rewriting As a Cyclic Translation

Given a click log data $\mathcal{D} = \{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$, where $\mathbf{x}$ denotes query, $\mathbf{y}$ denotes item title, and N denotes the number of training samples, the standard training objective in most translation models is to maximize the log likelihood of the training data

$$L_f(\boldsymbol{\theta}_f) = \sum_{n=1}^{N} \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f), \quad (1)$$

$$L_b(\boldsymbol{\theta}_b) = \sum_{n=1}^{N} \log P(\mathbf{x}^{(n)}|\mathbf{y}^{(n)}; \boldsymbol{\theta}_b), \quad (2)$$

where $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_f)$ and $P(\mathbf{x}|\mathbf{y};\boldsymbol{\theta}_b)$ are query-to-title (forward) and title-to-query (backward) neural translation models, parameterized by $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_b$, respectively. Note that the subscripts f and b are shorthands for forward and backward, respectively. Two objective functions L_f and L_b are independent on each other. Thus, the model can be trained separately without loss of accuracy.

In practice, we find that the query-to-title model requires more memorization capability in order to generate good enough item titles, potentially due to that target item titles are normally much longer than source queries and potentially item title space is much larger than query space. Thus, the translation model has to "memorize" the item titles to generate corresponding titles for a given query. On the other hand, we find that the title-to-query model is more like a text summarization model. Thus, it does not require a large model size to memorize.

Our query rewriting model is general enough to leverage most neural machine translation (NMT) models for both forward and backward directions. We have experimented with attention-based model [4] and transformer-based model [5], both of which work well in our scenario but the latter shows slightly better performance (see Experiments in Section IV-C1). Thus, we choose a 4-layers transformer for the query-to-title model, and 1-layer transformer for the title-to-query model. The detailed model setup can be found in Section IV-A.

After the models are trained, we can simply run the two models sequentially to generate a rewritten query, via intermediate synthetic item titles, as illustrated in Figure 3. We will present a more disciplined approach in Section III-E.

C. Cyclic Consistency

The above two separately trained models work reasonable well in practice (see Table III, Table VII). However, as one can imagine, it is sub-optimal in terms of generating the best query rewriting, since the learning algorithm does not specifically take the task, query rewriting, into account.

Our idea is to leverage a *cycle consistency* in learning the two models. For our mission, we have two translation models, query-to-title and title-to-query. To get a better query rewriting model, the intuition is to encourage the two translation models can collaboratively "translate back" to the original query. Thus, model parameters should be learned to maximize the likelihood of "translating back" the original query.

Formally, we introduce a cycle consistent likelihood $L_c(\boldsymbol{\theta}_f, \boldsymbol{\theta}_b)$ to encourage two models collaboratively "translate back" the original query as follows

$$\begin{aligned} L_c(\boldsymbol{\theta}_f, \boldsymbol{\theta}_b) &= \sum_{n=1}^{N} \log P(\mathbf{x}^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f, \boldsymbol{\theta}_b) \\ &= \sum_{n=1}^{N} \sum_{\mathbf{y}\in\mathcal{Y}} \log P(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f) P(\mathbf{x}^{(n)}|\mathbf{y}; \boldsymbol{\theta}_b), \end{aligned} \quad (3)$$

where for each training query $\mathbf{x}^{(n)}$, it computes the "translating back" probability $P(\mathbf{x}^{(n)}|\mathbf{x}^{(n)})$ by marginalize over all possible item titles $\mathbf{y}$.

The final likelihood function is a linear combination of forward, backward and cyclic consistency likelihoods in Equations (1), (2) and (3) as follows

$$L(\boldsymbol{\theta}_f, \boldsymbol{\theta}_b) = L_f(\boldsymbol{\theta}_f) + L_b(\boldsymbol{\theta}_b) + \lambda L_c(\boldsymbol{\theta}_f, \boldsymbol{\theta}_b),$$

where the hyper-parameter λ controls the tradeoff between the bi-directional translation likelihood and the cycle consistent likelihood.

Thus, the optimal model parameters, $\boldsymbol{\theta}_f^*$ and $\boldsymbol{\theta}_b^*$, are learned by

$$\begin{aligned}\boldsymbol{\theta}_f^* = \arg\max \Bigg\{ &\sum_{n=1}^{N} \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f) + \\ &\lambda \sum_{n=1}^{N} \log P(\mathbf{x}^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f, \boldsymbol{\theta}_b) \Bigg\},\end{aligned}$$

$$\begin{aligned}\boldsymbol{\theta}_b^* = \arg\max \Bigg\{ &\sum_{n=1}^{N} \log P(\mathbf{x}^{(n)}|\mathbf{y}^{(n)}; \boldsymbol{\theta}_b) + \\ &\lambda \sum_{n=1}^{N} \log P(\mathbf{x}^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f, \boldsymbol{\theta}_b) \Bigg\}.\end{aligned}$$

We can see that the query-to-title and the title-to-query models are connected via the cyclic consistency and they can hopefully benefit each other in joint training.

D. Training

The partial derivative of $L(\boldsymbol{\theta}_f, \boldsymbol{\theta}_b)$ with respect to the forward model parameter $\boldsymbol{\theta}_f$ can be written as follows.

$$\begin{aligned}\frac{\partial L(\boldsymbol{\theta}_f, \boldsymbol{\theta}_b)}{\partial \boldsymbol{\theta}_f} &= \sum_{n=1}^{N} \frac{\partial \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f)}{\partial \boldsymbol{\theta}_f} \\ &+ \lambda \sum_{n=1}^{N} \frac{\partial \log \sum_{\mathbf{y}_i \in \mathcal{Y}} P(\mathbf{y}_i|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f) P(\mathbf{x}^{(n)}|\mathbf{y}_i; \boldsymbol{\theta}_b)}{\partial \boldsymbol{\theta}_f}.\end{aligned} \tag{4}$$

We skip the partial derivative for the backward model for brevity, which can be derived similarly.

However in practice, it is prohibitively expensive to compute the sums in Equation (4) due to the exponential search space of $\mathcal{Y}$. Alternatively, we propose to use a subset of the full space $\tilde{\mathcal{Y}} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_k\} \subset \mathcal{Y}$ to approximate the second term in Equation (4) as follows

$$\begin{aligned}&\sum_{n=1}^{N} \frac{\partial \log \sum_{\mathbf{y}_i \in \tilde{\mathcal{Y}}} P(\mathbf{y}_i|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f) P(\mathbf{x}^{(n)}|\mathbf{y}_i; \boldsymbol{\theta}_b)}{\partial \boldsymbol{\theta}_f} \\ =&\sum_{n=1}^{N} \frac{\sum_{\mathbf{y}_i \in \tilde{\mathcal{Y}}} P(\mathbf{x}^{(n)}|\mathbf{y}_i; \boldsymbol{\theta}_b) \cdot \frac{\partial P(\mathbf{y}_i|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f)}{\partial \boldsymbol{\theta}_f}}{\sum_{\mathbf{y}_i \in \tilde{\mathcal{Y}}} P(\mathbf{y}_i|\mathbf{x}^{(n)}; \boldsymbol{\theta}_f) P(\mathbf{x}^{(n)}|\mathbf{y}_i; \boldsymbol{\theta}_b)}.\end{aligned} \tag{5}$$

In practice, we use the top-k set of item titles $\mathbf{y}_i$ translated by the forward model for the given query $\mathbf{x}^{(n)}$. As $k \ll |\mathcal{Y}|$, it is feasible to calculate Equation (5) efficiently by enumerating all $\mathbf{y}_i$ in $\tilde{\mathcal{Y}}$.

The subset $\tilde{\mathcal{Y}}$ can be obtained by any sequence decoding methods (see Section III-F) with the forward model $P(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}_f)$. In practice, we find this step is much more time consuming than other steps since we have to run the decoder network for several steps according to the item title length. Also, consider that the cyclic consistency only makes sense when the two models are well trained. Thus, we only perform the cyclic consistency term in Equation (5) after a certain number of warmup steps. We use Adam optimizer [27] for the training steps. The detailed algorithm is shown in Algorithm 1. We also illustrate the training process visually in Figure 2.

Algorithm 1 Cyclic Consistent Training Algorithm

```
**input**: Dataset $D$=$\{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$, batch size $B$, max steps $T$, beam width $k$, warmup training steps $G$.
Random initialize model parameters as $\boldsymbol{\theta}_f^{(0)}$ and $\boldsymbol{\theta}_b^{(0)}$.
**for** $t = 1 \ldots T$ **do**
    Sample a batch of $b$ examples $\mathcal{B} = \{\mathbf{x}^{(b)}, \mathbf{y}^{(b)}\}_{b=1}^{B} \subseteq \mathcal{D}$.
    **if** $t \leq G$ **then**
        Compute two model gradients $\frac{\partial L_f}{\partial \boldsymbol{\theta}_f^{(t-1)}}$ and $\frac{\partial L_b}{\partial \boldsymbol{\theta}_b^{(t-1)}}$.
        Update model parameters $\boldsymbol{\theta}_f^{(t)}$ and $\boldsymbol{\theta}_b^{(t)}$ by Adam optimization step.
    **else**
        Perform top-$n$ sampling with forward model $P(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}_f^{(t)})$ to generate $k$ synthetic titles for each query.
        Collect synthetic data set $\left\{\{\mathbf{x}^{(b)}, \mathbf{y}_i^{(b)}\}_{i=1}^{k}\right\}_{b=1}^{B}$.
        Compute two model gradients $\frac{\partial L}{\partial \boldsymbol{\theta}_f^{(t-1)}}$ and $\frac{\partial L}{\partial \boldsymbol{\theta}_b^{(t-1)}}$.
        Update model parameters $\boldsymbol{\theta}_f^{(t)}$ and $\boldsymbol{\theta}_b^{(t)}$ by Adam optimization step.
    **end if**
**end for**
```

E. Inference

After the forward and backward models are trained, we obtain the optimal model parameters $\boldsymbol{\theta}_f^*$ and $\boldsymbol{\theta}_b^*$. Then the inference problem can be formulated as follows. Given a query $\mathbf{x}$, we would like to find another query $\mathbf{x}^* \neq \mathbf{x}$ that maximizes the following probability

$$\begin{aligned}x^* &= \arg\max_{x'} P(\mathbf{x}'|\mathbf{x}; \boldsymbol{\theta}_f^*, \boldsymbol{\theta}_b^*) \\ &= \arg\max_{x'} \sum_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}_f^*) P(\mathbf{x}'|\mathbf{y}; \boldsymbol{\theta}_b^*).\end{aligned}$$

However, it is infeasible to enumerate all possible item title $\mathcal{Y}$ for the sum. Therefore, again, we have to resort to a subset $\tilde{\mathcal{Y}} \subset \mathcal{Y}$ to approximate the inference.

The full workflow of inference is shown in Figure 3. Specifically, we perform top-n sampling decoding method (detail is described in III-F), where n means the number of candidate token ids, for a given query $\mathbf{x}$ with the forward model to generate k synthetic item titles in $\tilde{\mathcal{Y}} = \{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$, according to translation probabilities $P(\mathbf{y}_i|\mathbf{x}; \boldsymbol{\theta}_f^*)$. Then, we perform top-$n$ sampling for each item title $\mathbf{y}_i$ with the backward model to generate k synthetic queries each, denoted as

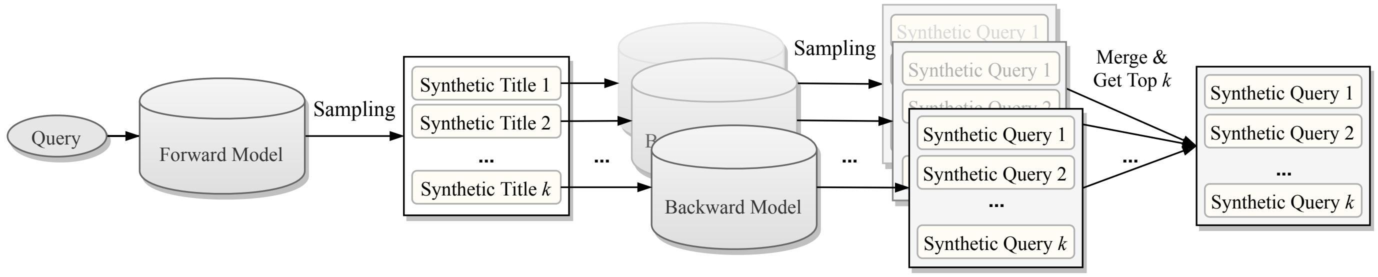


Fig. 3. Inference process of generating top synonymous queries for one user input query.

$\mathbf{x}_{ij}$ where $1 \le j \le k$, according to translation probabilities $P(\mathbf{x}_{ij}|\mathbf{y}_i; \boldsymbol{\theta}_b^*)$. Finally, we get the most likely k synthetic queries from the candidate set $\{\mathbf{x}_{ij}|i,j\}$ of size k^2, according to the probability

$$P(\mathbf{x}_{ij}|\mathbf{x}) = \sum_{1 \le t \le k} P(\mathbf{y}_t|\mathbf{x}; \boldsymbol{\theta}_f^*) P(\mathbf{x}_{ij}|\mathbf{y}_t; \boldsymbol{\theta}_b^*).$$

Note that in practice, all the above computations are performed in log probability space, and we have to carefully apply a few tricks to avoid numerical issues [28].

F. Diverse Sequence Decoding

As we mentioned before in Equation (5), the top-k item titles in set $\tilde{\mathcal{Y}}$ are obtained by any sequence decoding algorithm. Generally speaking, the optimal sequence, *i.e.*, the most likely sequence, can only be obtained by exhaustive search, which is infeasible for most cases. In practice, typically, greedy search and beam search are the two most widely used approaches. In detail, the greedy search selects the most likely token at each step of the decoding process. Thus, it is not guaranteed to find the optimal sequence, as the global most likely sequence might not locally take the most likely token at each step. Beam search is an improved algorithm based on the greedy search. Instead of picking the most likely token at each step, the beam search maintains a number (*i.e.*, beam size) of most likely sequences during the decoding process. More details about these two sequence decoding methods can be found in [29].

In practice, we found the above two widely used methods are not well applicable to our problem. Greedy search outputs only one sequence, which does not fit into our algorithm. Beam search, however, outputs very similar sequences that lack diversity. For example, we found some synthetic item titles only differ in a blank space, or a single token. Thus, those almost identical item titles lead to very similar query rewriting results, which is not what we want in practice.

Therefore, we develop a novel sequence decoding technique, namely top-n sampling decoder, to generate more diverse sequences. As shown in Figure 4, we start by maintaining k candidate sequences. At the first step, we pick the most likely k tokens to ensure all the candidate sequences begin differently. This is a key step to increase the result's diversity. At the following steps, we perform top-n sampling to obtain the tokens for each candidate sequence. Specifically, we do sampling among the top n (n = 40 in our experiments) most likely tokens, according to their conditional generative probability (*i.e.*, the softmax layer output in the neural network). Thus, this top-n sampling could well balance the overall likelihood and diversity in the sequence output.

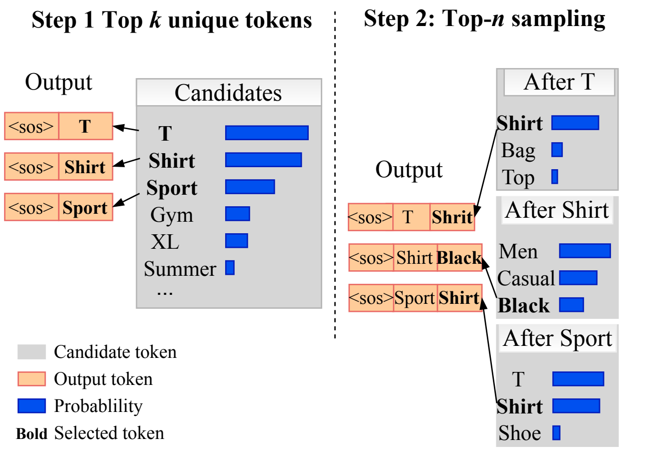


Fig. 4. Illustration of the top-n decoding method. At the first step, the most likely $k = 3$ unique tokens ("T", "Shirt", "Sport") are selected. In the following steps, for each candidate sequence, the top $n = 3$ most likely tokens are selected as candidate set, then we sample one token from them according to their probability.

G. Online Serving

Even though the proposed model is capable to generate reasonably good query rewriting, in practice we found it is still very challenging to deploy the model inference online. The typical total model inference latency for the query-to-title and title-to-query translations is more than 100 milliseconds, even in a modern GPU machine. Thus, it is significantly beyond the typical industrial backend system latency requirement, which is normally 50 milliseconds.

At the first step, we run the above proposed model offline to generate query rewriting for top 8 million popular queries, which are then fed into an online key-value store for fast online retrieval (less than 5 milliseconds). Those queries cover more than 80% of our search engine traffic. To cover all queries, especially for long tail queries that need query rewriting more than top queries, we have to speed up the proposed approach by trading off some accuracy potentially.

In practice, we make two modifications to speed up the model inference. First, we notice that the most time-consuming steps are the two sequence decodings, from query to title and from title to query. Thus, we simplify it by performing a query to query translation directly. The training data is prepared by collecting queries that share more than a certain number of clicks to the same items. We take these query pairs as synonymous queries. Then we train a single translation model on these synonymous query pairs, in order to reduce the sequence decoding time. Second, we notice that the transformer layer is more powerful but more time consuming than the traditional Recurrent Neural Network (RNN) layer. Especially, the transformer decoder is the bottleneck for the whole model inference, since the multi-head self attention [5] needs to be performed for all target tokens at each decoding step. Its counterpart RNN decoder, however, is much cheaper, since it only takes constant computing time at each decoding step. For the encoder, we still keep the transformer encoder for better accuracy (see Figure 9). Eventually, we are able to reduce the model inference time to about 30 milliseconds on a typical industrial CPU machine with 32 physical cores, which can be deployed online to handle all query rewritings.

H. System Optimization by Merging Syntax Trees

The query rewriting generates a few more queries in addition to the original query, which incurs extra burdens and challenges to the retrieval system, especially to the inverted index retrieval where the majority of computation happens during retrieval. In this section, we will explain how we optimize the inverted index retrieval to make the proposed method feasible in practice.

Given a user input query, our search engine first constructs a syntax tree by text tokenization and syntactic analysis, which is then used to extract document lists from the inverted index. The most straightforward way to implement the proposed query rewriting system is to construct as many syntax trees as the number of rewritten queries. This looks reasonable in theory. However, in practice, we find this straightforward approach is unfortunately inefficient, in terms of much more CPU usage and much longer system latency than the previous one-query-retrieval. To overcome this technical challenge in the system, we optimize the syntax tree construction to still keep only one tree by merging all the rewritten queries and the original query. The merged tree is much smaller than the sum of all syntax trees built separately for each query, since there are many common tokens between rewritten queries and the original query. As a result, we are able to keep the merged tree for multiple queries slightly larger than the previous tree for only the original query. This significantly reduces the retrieval system computation cost, in order to make the proposed model practical. Figure 5 shows an example of how we can merge two generated queries and an original query into one syntax tree. Typically, we use a *or* operation (denoted as "|" node) to merge all possible tokens if they diverge at the position.

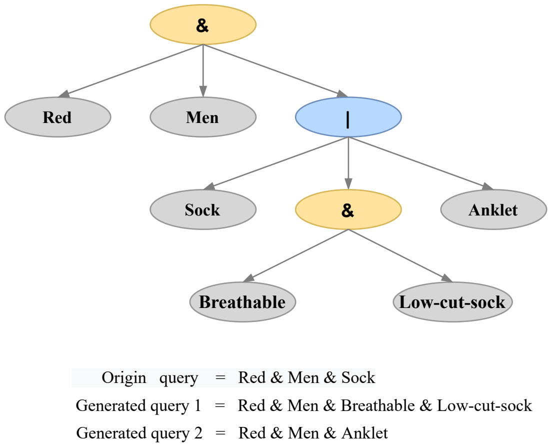

&
Red
Men
|
Sock
&
Anklet
Breathable
Low-cut-sock
Origin query = Red & Men & Sock
Generated query 1 = Red & Men & Breathable & Low-cut-sock
Generated query 2 = Red & Men & Anklet

Fig. 5. Merged syntax tree for two generated queries and one original query. The node "&" stands for logical "and" operation, and the node "|" stands for "or" operation for inverted index retrievals.

IV. Experiments

A. Setup

We use 60 days user click logs as our training data set, and only keep those samples with more than one click. Since only one click over a period of two months is very likely to be an accidental and unintentional click, which could pollute the quality of the dataset. The statistics of the dataset is shown in Table I.

TABLE I
Statistics of Data Set

	Query	Item Title
# Query Item Pairs	300 million	
# Search Sessions	5.6 billion	
Vocab Size	9744	
# Average Words	6.12	49.96

We tried a few model structures to balance the training speed and model performance. The final model is composed of two different transformers, whose detail structures are shown in Table II. Moreover, we set hyper-parameter $\lambda = 0.1$, beam width $k = 3$, and applied Adam optimizer [27] with learning rate=0.05, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Noam scheduler [5] that varies learning rate according to training steps is adopted in the optimization.

TABLE II
Model Hyperparameters

	Query-to-title	Title-to-query
# Transformer Layer	4	1
# Head	8	
Hidden Units of Feed-forward	1024	
Embedding Dimensionality	512	
Dropout Rate	0.1	

B. Ablation Study

1) Examples: In Table III and IV, we show some examples of the query rewriting results. As we can see from the table,

TABLE III
GOOD CASES FROM SEPARATELY TRAINED MODELS

Original Query	Synthetic Item Title	Rewritten Query
给爷爷的手机 (cellphone for grandpa)	Second hand Apple 8plus mobile phone apple iphone8plus apple 8p golden 64g full Netcom	苹果8plus (iphone 8plus)
老人奶粉 (milk powder for seniors)	Brand ’Yili’ milk powder golden collar crown series infant formula milk powder 3 segments 900g*3 cans	金领冠3段 (kido level-3)
自营猪年纪念币 (commemorative coins for Year of the Boar)	Chinese Gold Coin 2020 Year of the Rat Commemorative Coin 2010 Year of the Rat Commemorative Coin 2010	鼠年纪念币 (commemorative coins for Year of the Rat)
男士去皱 (wrinkle removal for men)	Nivea Men’s Eau De Toilette 50ml+Men’s Perfume 50ml+ Men’s Eau De Toilette 50ml+Men’s Eau De Toilette 50ml	男士香水 (men’s perfume)

TABLE IV
GOOD CASES FROM JOINTLY TRAINED MODEL

Original Query	Synthetic Item Title	Rewritten Query
给爷爷的手机 (cellphone for grandpa)	Little Pepper Mobile Phone Full Netcom 4g Dual SIM Dual Standby Mobile Phone for the Elderly Mobile Unicom 2g Dual SIM Mobile Phone for the Elderly, Student Standby Black	老人手机 (senior phone)
老人奶粉 (milk powder for seniors)	Imported from New Zealand Anjia anchor whole milk powder adult skimmed milk powder 1kg bag	奶粉成人 (adult milk powder)
自营猪年纪念币 (commemorative coins for Year of the Boar)	2019 Year of the Pig Zodiac Commemorative Coin Second Round of Zodiac Circulation Commemorative Coin 10 Yuan Face Value	猪年纪念币 (commemorative coins of for Year the Boar)
男士去皱 (wrinkle removal for men)	L’Oreal loreal men’s sharp anti-wrinkle firming and diminishing fine lines moisturizing facial skin care cosmetics set authentic five-piece set	男士护肤品套装 (men’s skin care set)

the proposed models are able to work surprisingly well for some hard queries, such as “cellphone for grandpa” to “senior cellphone”, “milk powder for seniors” to “milk powder for adults”. Also, we can observe that the jointly trained model performs better than the separately trained model (without the cyclic likelihood). These examples have already shown the difference.

2) Attention Visualization: To have a better idea of how the attention-based translation works for our models, we illustrate in Figure 6 a heat map of the attention weights of the two translation steps, from query to synthetic titles, and from synthetic titles to rewritten query. The example rewrites “Ah Di comfortable men’s shoe”, that is composed of a shorthand “Ah Di” for Adidas in Chinese and a vague descriptive word “comfortable”, to “Adidas men’s shoe” which is more standard for the retrieval task. From the attention weights, we can see that the query-to-title model is able to let “Ah Di”, a shorthand for Adidas in Chinese attend the English brand name “adidas”, to let “male student’s shoe” attend “men’s shoe” and to skip the vague descriptive word “comfortable”. Then, the title-to-query model is able to leverage the other full Chinese brand name for Adidas in title to be translated into a rewritten query. This is just an example to illustrate how the proposed models work in practice to generate more clean and standard queries for a vague user input query.

3) Training Convergence: We compared the model from jointly training and separately training. The below metrics are used for evaluations.

- Perplexity is a widely used metric in natural language processing, which measures how well a probability model predicts the training label. The value of perplexity can be computed by an exponential of cross entropy loss. Thus, the smaller perplexity is, the better the model performance is.
- Log probability stands for the log of the probability of “translate back” the original query, by marginalizing over a fixed number of intermediate synthetic titles, that are sampled using top-n sampling method. The higher log probability is, the better the model performance is.
- Accuracy is a similar metric to log probability. Instead of computing the probability of “translate back” the original query, we compute the accuracy of predicting the same token as the original query at each position.

Figure 7 shows the comparison between separately trained query-to-title (q2t) and title-to-query (t2q) models, and jointed trained ones. We can see that there is a significant jump on all metrics after $40,000$ warm-up steps, when the joint training starts adding the cyclic likelihood.

Also, we notice that the joint training does not affect the quality of title-to-query translation, since all the metrics keep the same. The query-to-title translation is slightly affected, presumably as a tradeoff for better query-to-query accuracy. The comparison demonstrates that the joint training with the cyclic likelihood could significantly boost the performance of query-to-query translation, as well as the query rewriting quality.

C. Offline Experiments

In this section, we talk about a few offline experiments that support our choice of translation models, human evaluation of query rewriting relevancy, and comparison with baseline methods.

1) Choices of Translation Models: Our proposed model is flexible in using different machine translation algorithm. We

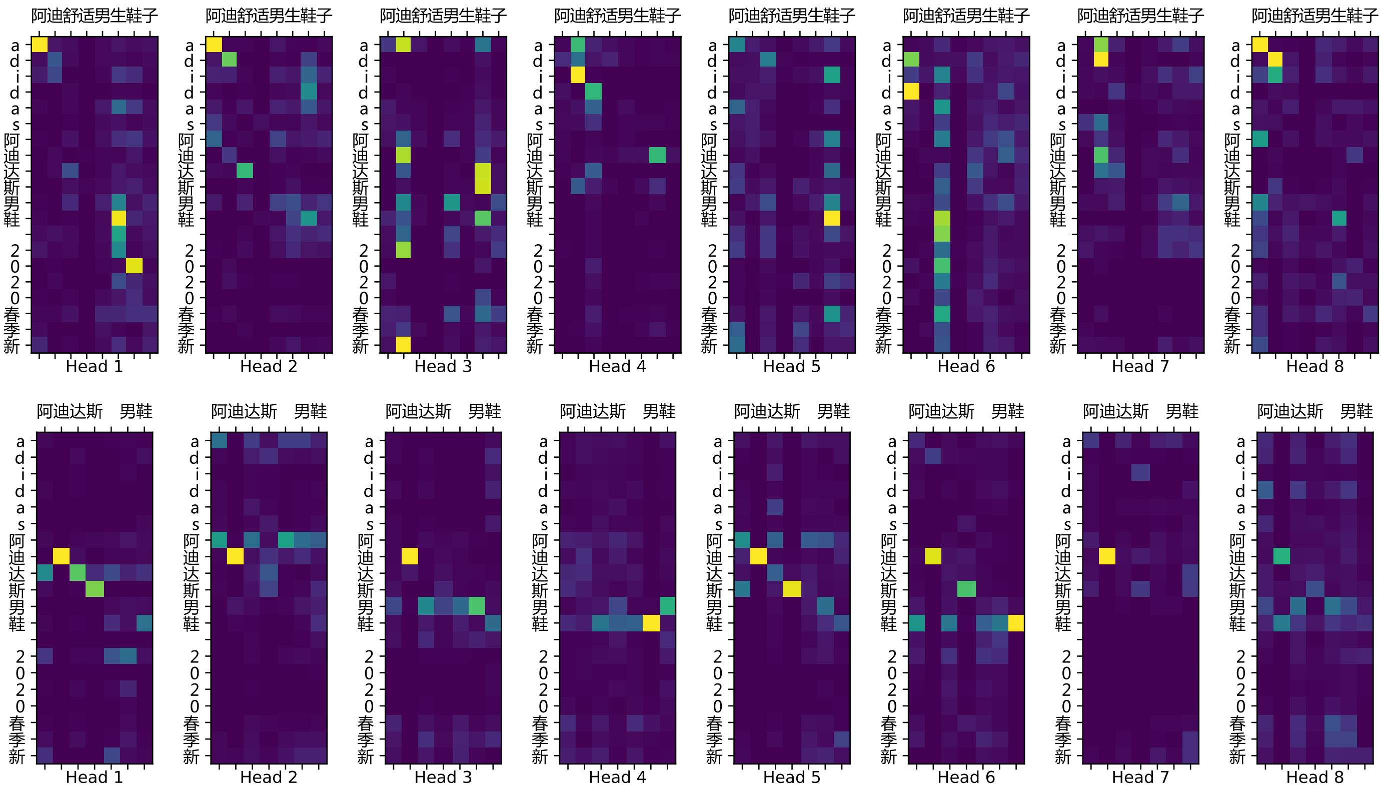


Fig. 6. Heat map of attention weights between query and synthetic titles (above), and between synthetic titles and rewritten query (below). The x axis corresponds to query and the y axis corresponds to synthetic title. The brightness of block represents the attention weights. Chinese (English) translations are 阿迪达斯(Adidas), 舒适(comfort), 男生(men's), 鞋子(shoe), 男鞋(men's shoe)，春季(spring), 新(new).

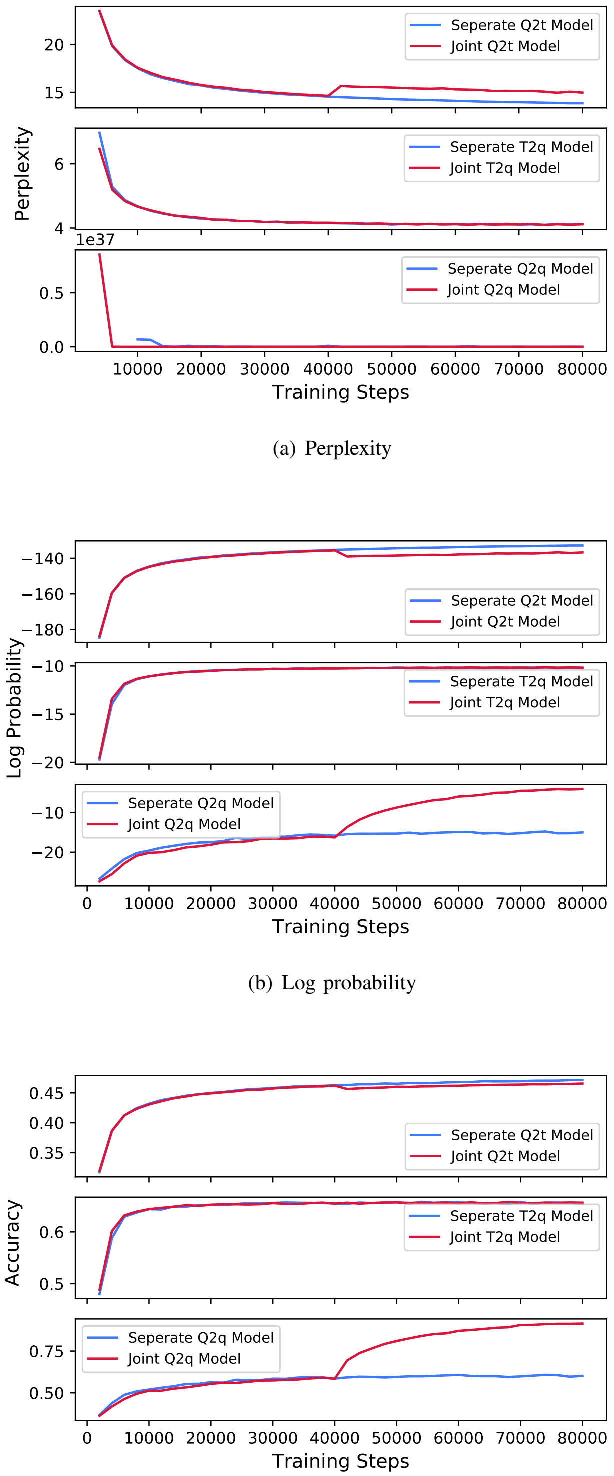
compare two most widely used model structures, transformer-based [5], and attention-based [4] ones, in our scenario. Figure 8 shows the comparison results. It is clear that the transformer-based model provides significantly better results than the attention-based model on all three metrics. However, we also notice that the transformer model requires more computations. Thus, we use a transformer-based machine translation model exclusively for all our experiments, if not specified otherwise.

As discussed in Section III-G, we have to simplify the model to deploy it online. We test the latency performance on different types of encoders and decoders on CPU with the same parameters: beam width as 3, the layer of encoder and decoder as 1, vocabulary size as 3000. and the maximum decode step as 15. As shown in Table V, transformer encoder and RNN decoder get the best results. Therefore, we consider two types of simplification: a pure RNN based model that uses RNN for both encoder and decoder, and a hybrid RNN model that uses RNN for decoder only. Figure 9 shows a comparison between these two types of models. The hybrid RNN model shows significantly better results than the pure RNN model, which indicates that the transformer encoder is still necessary for a balance between query rewriting quality and online serving latency.

TABLE V
LATENCY IN MILLISECONDS OF DIFFERENT TRANSLATION MODELS

	RNN	GRU	Transformer
Encoder	6	9	3.5
Decoder	30	35	67.5

2) *Human Evaluation of Relevancy:* Since rewritten query relevancy is hard to evaluate in an automatic way, we resort to human labeling for the evaluation. We first randomly select $1,000$ queries as our evaluation set which also have rule-based synonyms. Then we generate three rewritten queries for each query in the evaluation set using a separately trained model and jointly trained model respectively. As Table VI shows, human labelers are asked to evaluate twice on two comparisons, joint model versus separate model, and joint versus rule-based method. The result demonstrates that the jointly trained model generates more relevant rewritten queries than the separately trained model, which indicates the effectiveness of our joint training algorithm. We also compare the jointly trained model with the rule-based method for the same data set. It is expected that the rule-based method is more reliable on relevance since it often only replaces a single word in the query by lookup to a human curated dictionary. Surprisingly, the jointly trained model still wins in some cases of disambiguation



(a) Perplexity

(b) Log probability

(c) Accuracy

Fig. 7. Comparison of training convergence curve between separately trained models and jointly trained models.

of polysemous words. For example, the word “cherry” may be replaced with its synonym by a rule-based dictionary. However, the synonym describes, in fact, the “cherry” brand. In contrast, the joint model is able to correctly rewrite the query by leveraging the query context.

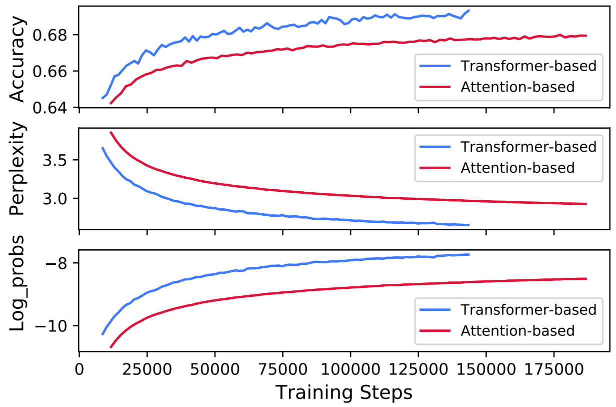


Fig. 8. Comparison between transformer-based and attention-based methods in our scenario.

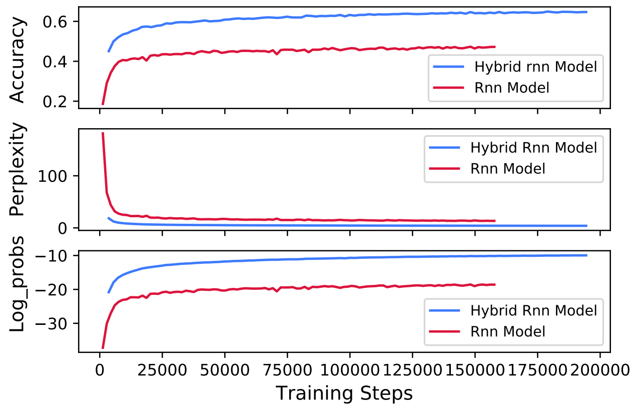


Fig. 9. Comparison between RNN and Hybrid RNN on direct query-to-query training.

3) Comparison with Baseline Methods: We are struggling to find a state-of-the-art query rewriting method with open-source code to compare with. Eventually, the only reasonable baseline we can compare with is a rule-based method as follows.

- Rule-based: A baseline method that has been widely used in our system, one of the largest e-commerce search engines in the world. The method starts from a human-curated synonym phrase dictionary. For a given query, it simply replaces the phrase in the query with its synonym phrase from the dictionary, to generate the rewritten query.

We use the following evaluation metrics.

- *F1 score* is calculated using the prediction precision (p) and recall (r) rate by the standard equation $2pr/(p + r)$. The queries, including the rewritten query and the original

TABLE VI
HUMAN EVALUATION RESULTS FOR QUERY REWRITING RELEVANCY

	Lose	**Tie**	**Win**
Joint vs Separate	22%	49%	29%
Joint vs Rule-based	29%	60%	11%

TABLE VII
COMPARISON BETWEEN BASELINE METHODS WITH OUR PROPOSED METHODS

	F1-score ↑	**Edit Distance ↓**	**Cosine Similarity ↑**
Rule-based	0.676	1.767	0.711
Separate	0.193	5.340	0.660
Joint	**0.254**	**4.821**	**0.668**

query, are both represented by a set of all its unigrams and bigrams. Then we are able to calculate the prediction precision, as the number of overlapping n-grams divided by the number of n-grams in rewritten query, and the prediction recall, as the number of overlapping n-grams divided by the number of n-grams in source query. This F1 score measures the n-gram similarity between the rewritten query and the original query. A higher F1 score indicates a more similar rewritten query to the original query.

- *Edit distance* stands for the Levenshtein distance [30] between the rewritten query and the original one. Smaller edit distance indicates more similar rewritten query to the original query.
- *Cosine similarity* calculates the embedding cosine similarity between the rewritten query and the original query. The embeddings are computed from an embedding retrieval model [1] in our production. Higher cosine similarity stands for more semantic relevancy in embedding space.

To sum up, the F1 score and edit distance measure the lexical similarity between the rewritten query and origin query. The cosine similarity measures the semantic similarity between them. In our query rewriting task, the ultimate goal is to retrieval more relevant items. Thus, we are actually looking for two paradoxical directions in metrics: lexical diverse but semantically relevant rewritten queries.

From the comparison results in Table VII, We can make the following observations: 1) the rule-based method achieves the highest F1 score and lowest edit distance, which indicates very high lexical similarity between the rewritten query and the original query. Though the highest cosine similarity indicates the good semantic relevancy, it is still not optimal from the perspective of query rewriting. Since those lexical similar rewritten queries won't contribute much to retrieve more relevant items. 2) The separately trained model and the jointly trained model have similar performance on all metrics, and the latter shows slightly better cosine similarity, which indicates slightly better semantically relevant rewritten queries, though the latter shows much better relevancy by human evaluation. But both models show much more diverse queries are generated, *i.e.*, much lower F1 score and much higher edit distance, while still maintaining good semantic relevancy, *i.e.*, slightly lower cosine similarity, comparing with the rule-based one.

D. Online Experiments

We would like to focus on the overall improvement of a search system using the proposed model as an additional retrieval method. We conducted live experiments on 10% of the entire site traffic during a period of 10 days using a standard A/B testing configuration.

In the control setup (baseline), it includes all the candidates available in our current production system, which are retrieved by inverted-index based methods with a standard query rewriting system. In the variation experiment setup, it generates at most 3 rewritten queries, each of which retrieves at most 1,000 candidates in addition to those in the baseline. For both settings, all the candidates go through the same ranking component and business logic. The ranking component applies a state-of-the-art deep learning method [31]. Here, we emphasize that our production system is a strong baseline to be compared with, as it has been tuned by hundreds of engineers and scientists for years, and has applied state-of-the-art query rewriting and document processing methods to optimize candidate generation.

Table VIII shows the A/B test results of the jointly trained model. To protect confidential business information, only relative improvements are reported. As we can see, the core e-commerce business metrics, including user conversion rate (UCVR), and gross merchandise value (GMV), as well as query rewrite rate (QRR), are all significantly improved. This A/B test results demonstrate the effectiveness of the proposed method. We hopefully will do more analysis to get insights on how the model improves online results, after we have a longer period for A/B test.

TABLE VIII
10-DAYS ONLINE A/B TEST IMPROVEMENTS

	UCVR	**GMV**	**QRR**
Joint	+0.5219%	+1.1054%	−0.0397%

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a novel deep neural network model to perform query rewriting in an industry scale e-commerce search engine. Specifically, 1) we formulated the long existing query rewriting into a novel cyclic machine translation problem, in order to leverage abundant click log data to train state-of-the-art machine translation models for this task. 2) We improved the bare-bones algorithm of separately training two translation models by introducing a cyclic consistent likelihood, which encourages a given query can "translate back" to itself in this cyclic translation process. 3) We proposed a system optimization by merging syntax trees into one, which makes this proposed method feasible in practice, *i.e.*, an industrial e-commerce search system. 4) We demonstrated in the ablation study that the proposed method can effectively generate semantically relevant but more standard queries, especially for vague long tail queries, in offline experiments that the cyclic consistent training could boost the model performance in terms of “translate back” accuracy, log probability and perplexity, and in online experiments that the proposed method could significantly improve all business core metrics in one of the world’s largest e-commerce search engine.

Apart from the proposed model, we have also explored another promising approach, by leveraging the pre-trained GPT2 model [12], which is a deep transformer-based language model trained on very large data. In the query rewriting scenario, we can add a special token between the query and title, *i.e.*, “query <sep1> title <sep2> query2”, and treat the whole sequence as a “special” language. Thus, the GPT2 model could be fine-tuned to learn the language model for this “special” language, which hopefully could generate a synthetic title for a given query, then generate a synthetic query from the title. This approach looks promising since the GPT2 model could benefit from pretraining on very large data. In practice, we, however, have not found it performs better than our jointly trained machine translation models yet. This is one of our future work, and we look forward to more inspirations in this direction.

Our future work also includes explorations on more novel decoding methods, to generate more diverse and likely sequences. For example, diverse beam search [32] is another way to increase diversity by optimizing a diversity-augmented objective directly.

Furthermore, developing more reasonable offline metrics to guide our offline model improvement is another important direction. We have found that neither the lexical similarity (F1 score and edit distance) nor the semantic similarity (cosine similarity) aligns well with the query rewriting objective, which is essential to generate diverse and semantically relevant queries. We believe such a metric could greatly benefit our model development.

References

[1] H. Zhang, S. Wang, K. Zhang, Z. Tang, Y. Jiang, Y. Xiao, W. Yan, and W.-Y. Yang, “Towards personalized and semantic retrieval: An end-to-endsolution for e-commerce search via embedding learning,” in *SIGIR*, 2020, p. 2407–2416.

[2] C. Li, Z. Liu, M. Wu, Y. Xu, H. Zhao, P. Huang, G. Kang, Q. Chen, W. Li, and D. L. Lee, “Multi-interest network with dynamic routing for recommendation at tmall,” in *CIKM*, 2019, p. 2615–2623.

[3] H. Zhu, X. Li, P. Zhang, G. Li, J. He, H. Li, and K. Gai, “Learning tree-based deep model for recommender systems,” in *SIGKDD*, 2018, p. 1079–1088.

[4] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *NIPS*, 2014, pp. 3104–3112.

[7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[8] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.

[9] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[11] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[13] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[15] Z. Tu, Y. Liu, L. Shang, X. Liu, and H. Li, “Neural machine translation with reconstruction,” *arXiv preprint arXiv:1611.01874*, 2016.

[16] P. Covington, J. Adams, and E. Sargin, “Deep neural networks for youtube recommendations,” in *RecSys*, 2016, pp. 191–198.

[17] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, “Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 4, pp. 694–707, 2016.

[18] T. Vu, D. Q. Nguyen, M. Johnson, D. Song, and A. Willis, “Search personalization with embeddings,” in *ECIR*. Springer, 2017, pp. 598–604.

[19] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, “Learning deep structured semantic models for web search using clickthrough data,” in *CIKM*, 2013, pp. 2333–2338.

[20] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, “Learning semantic representations using convolutional neural networks for web search,” in *WWW*, 2014, pp. 373–374.

[21] J. Bhogal, A. MacFarlane, and P. Smith, “A review of ontology based query expansion,” *Information processing & management*, vol. 43, no. 4, pp. 866–886, 2007.

[22] J. Wu, I. Ilyas, and G. Weddell, “A study of ontology-based query expansion,” in *Technical report CS-2011–04*, 2011.

[23] A. Mandal, I. K. Khan, and P. S. Kumar, “Query rewriting using automatic synonym extraction for e-commerce search.” in *eCOM@SIGIR*, 2019.

[24] G. Jeh and J. Widom, “Simrank: a measure of structural-context similarity,” in *SIGKDD*, 2002, pp. 538–543.

[25] I. Antonellis, H. Garcia-Molina, and C.-C. Chang, “Simrank++ query rewriting through link analysis of the clickgraph (poster),” in *WWW*, 2008, pp. 1177–1178.

[26] Y. He, J. Tang, H. Ouyang, C. Kang, D. Yin, and Y. Chang, “Learning to rewrite queries,” in *CIKM*, 2016, pp. 1443–1452.

[27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[28] F. Nielsen and K. Sun, “Guaranteed bounds on the kullback-leibler divergence of univariate mixtures using piecewise log-sum-exp inequalities,” *arXiv preprint arXiv:1606.05850*, 2016.

[29] R. Bisiani, “Encyclopedia of artificial intelligence,” *Beam search New York, NY: Wiley*, 1987.

[30] V. Levenshtein, “Levenshtein distance,” 1965.

[31] R. Li, Y. Jiang, W. Yang, G. Tang, S. Wang, C. Ma, W. He, X. Xiong, Y. Xiao, and E. Y. Zhao, “From semantic retrieval to pairwise ranking: Applying deep learning in e-commerce search,” in *SIGIR*, 2019, p. 1383–1384.

[32] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crandall, and D. Batra, “Diverse beam search: Decoding diverse solutions from neural sequence models,” *arXiv preprint arXiv:1610.02424*, 2016.