Weakly Supervised Co-Training of Query Rewriting and Semantic Matching for e-Commerce

Rong Xiao
Alibaba Group
Hangzhou, China
xiaorong.xr@alibaba-inc.com

Jianhui Ji, Baoliang Cui
Alibaba Group
Hangzhou, China
jianhui.jjh@alibaba-inc.com,
moqing.cbl@alibaba-inc.com

Haihong Tang, Wenwu Ou
Alibaba Group
Hangzhou, China
piaoxue@alibaba-inc.com,
santong.oww@alibaba-inc.com

Yanghua Xiao
Fudan University
Shanghai, China
shawyh@fudan.edu.cn

Jiwei Tan, Xuan Ju
Alibaba Group
Hangzhou, China
jiwei.tjw@alibaba-inc.com,
juxuan8@gmail.com

ABSTRACT

Relevance is the core problem of a search engine, and one of the main challenges is the vocabulary gap between user queries and documents. This problem is more serious in e-commerce, because language in product titles is more professional. Query rewriting and semantic matching are two key techniques to bridge the semantic gap between them to improve relevance. Recently, deep neural networks have been successfully applied to the two tasks and enhanced the relevance performance. However, such approaches suffer from the sparseness of training data in e-commerce scenario. In this study, we investigate the instinctive connection between query rewriting and semantic matching tasks, and propose a co-training framework to address the data sparseness problem when training deep neural networks. We first build a huge unlabeled dataset from search logs, on which the two tasks can be considered as two different views of the relevance problem. Then we iteratively co-train them via labeled data generated from this unlabeled set to boost their performance simultaneously.

We conduct a series of offline and online experiments on a real-world e-commerce search engine, and the results demonstrate that the proposed method improves relevance significantly.

CCS CONCEPTS

• **Information systems** → **Content ranking**; **Query reformulation**; **Similarity measures**;

WSDM '19, February 11–15, 2019, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5940-5/19/02…$15.00
https://doi.org/10.1145/3289600.3291039

KEYWORDS

Query Rewriting; Semantic Matching; Neural Networks; Co-Training

ACM Reference Format:
Rong Xiao, Jianhui Ji, Baoliang Cui, Haihong Tang, Wenwu Ou, Yanghua Xiao, and Jiwei Tan, Xuan Ju. 2019. Weakly Supervised Co-Training of Query Rewriting and Semantic Matching for e-Commerce. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3291039

1 INTRODUCTION

Retrieving the most relevant documents from a huge corpus for a query is the core problem of a search engine. One of the main challenges is that there exists a semantic gap between queries and documents, because they are usually described by different language styles and vocabularies [21]. In an e-commerce search engine, this gap is more significant, as sellers usually describe commodities by professional language in many different business domains, while a user expresses his demand by daily vocabulary. For example, when a user issues a query "`Nike basketball shoes`", a relevant product's title can be "`Kobe 11 elite`", which can hardly be retrieved via classical text matching models. How to understand queries and documents semantically and address the language discrepancy between them are crucial to improve relevance of the search results pages (SRP).

A typical search engine deals with this challenge by two techniques: *query rewriting (QR)* and *semantic matching (SM)*. A search engine retrieves relevant documents for a query with two major phases: *recall* and *ranking*. The *recall* phase first finds documents which contain all query terms. These query-document pairs are further fed into a *ranking* phase which sorts documents by relevance scores. In the *recall* phase, *QR* model enhances the query quality by leveraging different kinds of information, which has the potential to explore additional relevant documents. In the *ranking* phase, *SM* model maps the query and documents to a common semantic space and calculates their similarity as an important

feature of relevance ranking. The two components work together to improve the relevance of a search engine.

Recently, deep learning has made great progress in these two tasks, because it has the capability to learn semantic representation from raw text of query and document. However, training deep neural network models requires a large amount of labeled data, which is usually expensive to obtain. Therefore, it is popular to use clickthrough data as *implicit* feedback for the two tasks. Unfortunately, this supervision signal is *biased* and *noisy* especially in an e-commerce search engine. User's click behavior is influenced not only by query-document relevance but also by the image, price and user preference greatly. For example, a female user searches "`round collar shirt women`", and may click an irrelevant product whose title is "`Women Casual Sleeveless Printed Vest Dresses`", because its pictures are more attractive than a plain shirt's.

We find that the *QR* and *SM* tasks are highly correlated with each other. A *good* rewriting query always retrieves *relevant* documents for original query, while documents retrieved by a *bad* rewriting query are mostly *irrelevant*. For example in Figure 1, suppose we have learned a *QR* model which generates two rewriting candidates for query $q =$"`VR helmet`". The candidate $r_1 =$"`virtual reality headset`" is *good* and $r_2 =$"`VR games`" is *bad*. It is obvious that most of the documents retrieved by r_1 are *relevant* to the original query q, while most of the documents retrieved by r_2 are *irrelevant* to q.

On the other hand, the relevance between documents retrieved by a rewriting candidate r and original query q decides whether r is a *good* rewriting for q. Suppose that we have learned a *SM* model which evaluates whether a query-document pair is *relevant*. For $q =$"`VR helmet`" and another unknown candidate $r_3 =$"`virtual reality glasses`", because most of documents retrieved by r_3 are *relevant* to q, it should be a *good* candidate with a high probability. For candidate $r_4 =$"`3d glasses`", because some of the documents retrieved by r_4 are *irrelevant* to q, it is probably *bad*.

Co-training [20] is a semi-supervised learning paradigm which trains two individual models respectively from different views and let the two models label some unlabeled examples for each other. It can alleviates the problem of lacking large amount of labeled data especially when training deep networks. Inspired by this idea, we considered two tasks as two different views of the same relevance problem. A *SM* model directly discriminates a query-document relevance in *ranking* phase while a *QR* model indirectly judges relevance between the original query and a set of documents retrieved by a rewriting candidate. Indeed, they share a common objective of retrieving more relevant documents for a search engine in different phases from different views.

In this paper, we propose a co-training framework for deep neural network models built for *QR* and *SM*, where each of the two tasks only has a small amount of human labeled data. We first make use of clickthrough data to train two weak initial models. Then we build an unlabeled dataset from search logs and design a method to generate labeled data

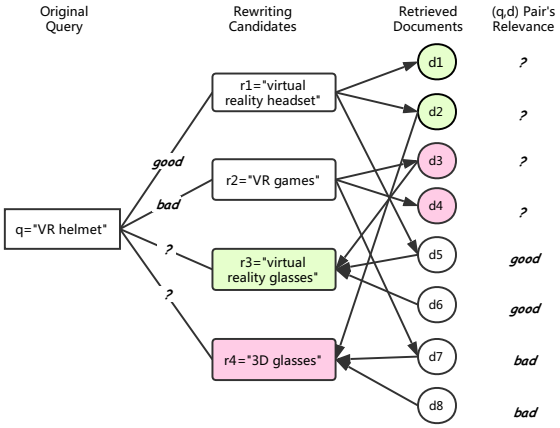

Figure 1: An example of relationship between *QR* and *SM*. We can infer that d_1 and d_2 are *relevant* with q because r_1 is a *good* rewriting of q, while d_3 and d_4 are *irrelevant* with q because r_2 is a *bad* rewriting of q. On the contrary, we can also label (q, r_3) as *good* because 2/3 of documents retrieved by r_3 are *relevant* with q, while label (q, r_4) as bad because 2/3 of documents retrieved by r_4 are *irrelevant* with q.

for each other. The two models are iteratively co-trained by the weakly labeled data generated for each other. This framework can make use of a large amount of unlabeled data to boost the two models' performance simultaneously. Our contribution can be summarized as follows:

(1) We propose a weakly supervised co-training framework for *query rewriting* and *semantic matching* to solve the problem of sparsely labeled data;
(2) The two tasks only have a small amount of quality training data. We make use of a large amount of *biased* and *noisy* search logs to improve the performance of the two tasks simultaneously;
(3) We conduct extensive offline and online experiments on an e-commerce search engine. The results demonstrate the effectiveness of our co-training framework.

In the rest of the paper, we first review related work in Section 2. Then we elaborate our weakly supervised co-training solution in Section 3 and details of models for the two tasks in Section 4. In Section 5, extensive experiments are presented to demonstrate the effectiveness of our method. Finally, we conclude the paper in Section 6.

2 RELATED WORK

In this section we will survey the related work for query rewriting, semantic relevance and co-training, respectively.

Query expansion and rewriting are important research topics in IR and have been studied for a long time. Pseudo-relevance feedback [21], which makes use of top-most documents retrieved by the original query for expansion, has been widely used. Expansion terms can be selected by unsupervised models [18] or supervised models [1]. Recently, several

attempts have been made to study deep learning in query rewriting. Grbovic et al. [5] proposed a query embedding method and expanded a query via simple K-nearest neighbor search. Zamani and Croft [22] developed a relevance-based word embedding model for query expansion. Further, Zamani et al. [23] proposed to use multiple weak supervision signal to train a neural query performance predictor, which could be used to the select best query reformulations. He et al. [7] proposed a *learning-to-rewrite* framework that consists of unsupervised *candidate generation* and supervised *candidate ranking*. In this paper, we attempt to deal with the query rewriting problem on weak supervision signals from a semantic relevance model.

There have been much work to study deep learning in relevance ranking. Most of the work concentrated on designing network architectures. These models can be generally divided into two groups: *representation-focused* models and *interaction-focused* models [6, 17]. The former models focus on learning representations for query and document to calculate similarity, such as DSSM [8] , C-DSSM [15, 16] and LSTM-DSSM [12]. The *interaction-focused* models focus on using deep neural networks to learn interaction patterns for matching, such as MatchPyramid [13] and DeepRank [14]. All of them are trained on human labeled dataset or on *implicit* feedback from clickthrough data. Most recently, Dehghani et al. [4] attempted to exploit an unsupervised model (BM25) to provide training data for a neural ranking model. Further, they proposed to leverage a small amount of human labeled data to train a *confidence network*, which can boost neural IR models' performance trained on weak labels [3]. In this paper, we study how to use a learned QR model to provide weakly labeled data for training a SM model.

Co-training is a semi-supervised learning paradigm which uses unlabeled data along with a few labeled examples to boost performance [20]. Co-training has been successfully applied to many NLP tasks [10, 19]. In this paper, we build an unlabeled dataset from search logs, on which the QR and SM tasks are considered as two different views of relevance problem.

3 WEAKLY SUPERVISED CO-TRAINING FRAMEWORK

This paper employs the correlation between the *query rewriting* and *semantic matching* tasks for co-training deep neural networks to boost their performance simultaneously. We will formally define these two tasks first, and then elaborate our co-training framework and the details of the models for these two tasks.

3.1 Problem Formulation

Query Rewriting. We formulate the QR task as two sub problems: *candidate generation* and *candidate ranking* [7]. Given an original query q, the rewriting candidate set $R = \{r_i|1{\leq}i{\leq}n\}$ are generated by a set of candidate generators $G = \{g_i|1{\leq}i{\leq}m\}$. Then, the candidate set R is ranked based on a score function: $f(q, r; \theta) \rightarrow \mathbb{R}$, where θ is the set of parameters. We adopt a *point-wise* function to optimize the parameter θ on a labeled dataset $D_1 = \{(q, r, y)|y \in \{0, 1\}\}$:

$$\theta^* = \arg\min_{\theta} \sum_{(q,r,y)\in D_1} -[ylog(f(q, r; \theta)) + (1 - y)log(1 - f(q, r; \theta))] \quad (1)$$

where the value of y indicates if q can be rewritten as r.

In previous work He et al. [7], the ranking function $f(q, r)$ is trained on the learning targets generated from clickthrough data with hand-crafted features. In this paper, we propose to apply deep neural network to obtain semantic representation of queries from raw text to improve the performance. Although the number of training examples obtained from the clickthrough data is enough for training a neural network, there are much *bias* and *noise* as described in Section 1, which will degrade the performance. We aim to improve the quality of training data for $f(q, r)$ by the co-training framework.

Semantic Matching. For a given query q and its corresponding candidate documents $\{d_i\}$, the goal of SM is to learn a scoring function: $s(q, d; \phi) \rightarrow \mathbb{R}$, that determines the semantic relevance for a (q, d_i) pair, where ϕ is the set of parameters. We use a point-wise loss function to optimise parameter ϕ on a labeled dataset $D_2 = \{(q, d, z)|z \in \{0, 1\}\}$:

$$\phi^* = \arg\min_{\phi} \sum_{(q,d,z)\in D_2} -[zlog(s(q, d; \phi)) + (1 - z)log(1 - s(q, d; \phi))] \quad (2)$$

where the value of z indicates whether d is *relevant* to q.

For this task, many neural network models have been proposed. However, we face the common problem of lacking large amount of labeled data for training a neural network model. This motivates us to propose a co-training framework to alleviate the sparseness problem of training data.

3.2 Co-Training Framework

We outline our co-training framework in Algorithm 1. At the beginning, we train two weak initial models on supervision signals from the clickthrough data. Then, in each iteration, the two models generate weakly labeled data for each other to boost their performance simultaneously. We also make full use of a few human labeled examples for fine-tuning them respectively in each iteration, which further improves their performance.

We are confronted with two specific problems when applying the generic co-training framework to our two tasks. First, we do not have enough human labeled data to train neural networks as initial models. We generate weak labels from biased and noisy clickthrough data to start the two models. Second, we do not explicitly have an unlabeled dataset for the two tasks, and the generation of labeled examples for each other is not straightforward. We need to build an unlabeled dataset, and then design a method to generate labeled data for each other in each iteration of co-training.

Algorithm 1 Co-training Framework for Query Rewriting and Semantic Matching

Input:
$f(q, r; \theta)$: QR candidate ranking function;
$s(q, d; \phi)$: Semantic relevance scoring function;
$D_1 = \{(q_i, r_i, y_i)\}$: weakly labeled dataset for f;
$D_1^* = \{(q_i, r_i, y_i)\}$: human labeled dataset for f;
$D_2 = \{(q_i, d_i, z_i)\}$: weakly labeled dataset for s;
$D_2^* = \{(q_i, d_i, z_i)\}$: human labeled dataset for s;
$U = \{(q, r, d)\}$: unlabeled triples dataset
Output: optimal θ^*, ϕ^*
Randomly initialize θ and ϕ;
Generate dataset D_1,D_2 from clickthrough data;
Pre-train $f(q, r; \theta)$ with D_1, and fine-tune with D_1^*;
Pre-train $s(q, d; \phi)$ with D_2, and fine-tune with D_2^*;
repeat
  Cleanse dataset D_1 and D_2;
  Generate weakly labeled data by s from U and add to D_1;
  Generate weakly labeled data by f from U and add to D_2;
  Pre-train $f(q, r; \theta)$ with D_1, and fine-tune with D_1^*;
  Pre-train $s(q, d; \phi)$ with D_2, and fine-tune with D_2^*;
until f and s converged

3.2.1 Initial Models. Although training examples from clickthrough data is biased and noisy, we can still use these data to train two weak initial models for $f(q, r)$ and $s(q, d)$.

For QR, we use the click numbers of the retrieved documents of a rewriting candidate to quantify its quality. For a (q, r) pair, the top-k documents retrieved by r is $D_r = \{d_r^j | 1 \leq j \leq k\}$ where d_r^j is the j-th ranked document. The click numbers between q and the documents set D_r are $C_r = \{c_{q,d_r^j} | 1 \leq j \leq k\}$ where c_{q,d_r^j} is the click numbers of document d_r^j. We use the *Logdiscounted log clicknum* [7] to assess the quality of r:

$$y'(q, r) = \sum_{j=1}^{k} \frac{log_2(c_{q,d_r^j})}{log(i+1)} \tag{3}$$

which smooths the click numbers and eliminates positional bias. The rewriting candidate set $R = \{r_i\}$ of q is ranked by the quality, and then the top-N is selected as positive examples and others are negative examples.

For SM, we train an initial weak model on the *implicit* feedbacks from clickthrough data as many previous work [8, 16] do. The clicked documents under query q are sampled as positive examples and non-clicked documents as negative examples.

3.2.2 Unlabeled Dataset. For the example shown in Figure 1, we combine the query q, rewriting candidates $\{r_1, r_2, r_3, r_4\}$ and documents $\{d_i | 1 \leq i \leq 8\}$ to form a set of triples $\{(q, r, d)\}$ which is shown in the bottom of Figure 2. Then the two models $f(q, r)$ and $s(q, d)$ are considered as two different views with respective subset features.

To generate enough labeled data for the two models, we need to build a large amount of unlabeled triples dataset. For this purpose, we first randomly select some queries from search logs, and then generate rewriting candidates for every query (candidate generation will be elaborated in Section 4.1.1). For every candidate, we select the top-k retrieved documents from logs. These queries, rewriting candidates and documents are combined to form a huge triple set $U = \{(q, r, d)\}$.

3.2.3 Labeled Data Generation. Figure 1 shows a simple example that two tasks infer same labels for each other on a small triples set. We introduce the details of inference here.

Labeled Data Generation for QR. We first use the score function $s(q, d)$ to predict the relevance between q and d for each triple of dataset $U = \{(q, r, d)\}$. For simplicity, if $s(q, d) > 0.5$ we consider q and d are *relevant*, otherwise they are *irrelevant*. For each unlabeled triple (r, q, d), we get a pseudo label by $sign(s(q, d) - 0.5)$.

Then we infer the label of a (q, r) pair by pseudo labels of all associated triples (r, q, d). The quality of rewriting query r can be measured by the ratio of *pseudo-relevant* documents:

$$y'(q, r) = \frac{1}{k} \sum_{i=1}^{k} sign(s(q, d_i) - 0.5) \tag{4}$$

The larger the ratio is, the more documents retrieved by the rewriting query r are *relevant* with original query q. Based on y', we select positive and negative examples for QR with a confidence threshold β:

$$y_{q,r} = \begin{cases} 1 & \text{if } y'(q, r) > \beta \\ 0 & \text{if } y'(q, r) \leq 1 - \beta \end{cases} \tag{5}$$

Labeled Data Generation for SM. In a similar way, we use the function f to generate prediction $f(q, r)$ for each triple of dataset $U = \{(q, r, d)\}$. Then, we also select the positive and negative examples for the SM model from the (q, r, d) triples with the same confidence threshold β as used in Eq 5:

$$z_{q,d} = \begin{cases} 1 & \text{if } f(q, r) > \beta \\ 0 & \text{if } f(q, r) \leq 1 - \beta \end{cases} \tag{6}$$

We iteratively co-train the two models by the weakly labeled data generated for each other as shown in Figure 2.

4 MODELS

4.1 Query Rewriting

For the QR task, we use the *learning-to-rewrite* framework that consists of *candidate generation* and *candidate ranking*. In the *candidate generation* phase, we exploit all available query information in search logs to build the candidate generators to recall candidates as diverse as possible. For *candidate ranking*, as our co-training framework generates a large amount of quality training data, the performance of the candidate ranking function can be significantly improved by utilizing the representation ability of a neural network.

4.1.1 Candidate Generators. There are many different types of query information in search logs, such as *clicked documents*, *text content* and surrounding queries in a *session*

Table 1: Query Representations for Rewriting Candidate Generators

Query Information	Representation	Details
Clicked Documents	$e_1 = (..., c_{q,d_i}, ...)$	q is represented as distribution of clicks over documents set, and c_{q,d_i} is the click number of d_i under query q.
Text Content	$e_2 = \frac{1}{\|q\|} \sum \frac{1}{p(w_t)} w_t$	q is represented as a weighted average of pre-trained word embeddings. w_t is an E-dimension embedding of the t-th word of q, which is trained on the corpus of all product titles by word2vec. $p(w_t)$ is the term frequency of w_t and $\|q\|$ is the term count of q.
Session	e_3 =Encoding q by Skip-thought	We consider every query as a "sentence" and a session as a "paragraph" composed of query sentences sorted by search timestamp i: $\langle ..., q^{i-1}, q^i, q^{i+1}, ... \rangle$. A Skip-thought [11] encoder-decoder model is trained to reconstruct the surrounding context of an encoded query.

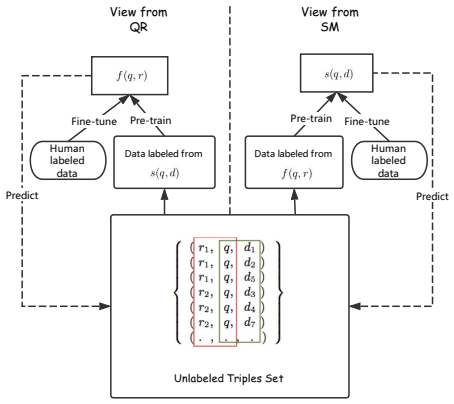

Figure 2: Flowchart of co-training the two tasks. $f(q,r)$ and $s(q,d)$ are considered as two different views on an unlabeled triple dataset $U = (q,r,d)$, and generate weakly labeled data for each other in each iteration of the training process.

context. We first build a generator for each type of the query information. Every generator g_i learns a representation e_i for each query as described in Table 1. Then for a given query q, g_i generates a candidates set R_i via finding the K-nearest neighbors in the vector space. The distance between two queries is measured by the cosine similarity of their vectors. The candidate set R for q is the union of the candidates from every generator: $R = \cup_{i=1}^{m} R_i$.

4.1.2 Candidate Ranking. We design a neural network for candidate ranking function $f(q,r)$ to make full use of the large amount of data generated by the co-training framework. We apply a Bi-LSTM and attention-based pooling to learn query representation, which can capture long term dependency and learn the weights of terms to improve the quality of representation. Then the representations of q and r are fused with hand-crafted features to score a (q,r) pair.

Query Representation. Specifically, for a given query $q = \langle w_1, ..., w_t, ..., w_l \rangle$ where w_t is an E-dimension embedding of the t-th word:

(1) The word embeddings of query q^i are fed into an Bi-LSTM. The hidden vectors of the t-th word from the forward and the backward LSTM are $h_t^{(f)}, h_t^{(b)}$;
(2) We represent the t-th word by concatenating the two hidden vectors: $h(t) = [h_t^{(f)}, h_t^{(b)}]$;
(3) q is represented as a weighted sum over the word vectors through attention-based pooling method:

$$h_q = \sum_{t=1}^{n} \alpha_t h_t \tag{7}$$

$$\alpha_t = \frac{exp(att(h_t; W_a))}{\sum_{i=1}^{n} exp(att(h_i; W_a))} \tag{8}$$

where att is an one-layer feedforward network that maps a word vector to a real valued score with parameter matrix W_a.

Feature Fusion. Given a pair (q,r), we treat their representations as h_q and h_r which are learned from the shared network. Next, the two semantic vectors h_q, h_r and the hand-crafted feature vector v are concatenated and followed by a fully connected layers with ReLU activation function:

$$f(q,r;\theta) = sigmoid(W_2(ReLU(W_1[h_q, h_r, v]))) \tag{9}$$

where θ is the set of network parameters. The hand-crafted feature vector is composed of features described in Table 2.

4.2 Semantic Matching

There have been many neural network models proposed for this task. In this paper, we use the same Bi-LSTM and attention-based pooling architecture as QR with different parameters to represent the semantic vectors of q and d as h_q and h_d. Then, the two vectors are concatenated and followed by a fully connected layers with ReLU activation function:

$$s(q,d;\phi) = sigmoid(W_4(ReLU(W_3[h_q, h_d]))) \tag{10}$$

where ϕ is the set of network parameters.

Table 2: Hand-crafted Features for a (q, r) Pair

Group	Feature
Pair Features	f_1 - difference of length between q and r
	f_2 - difference of word count
	f_3 - whether the candidate r contains q
	f_4 - jaccard similarity of words
	f_6 - cosine similarity of e_1
	f_7 - cosine similarity of e_2
	f_8 - cosine similarity of e_3
Candidate Features	f_9 - length of r
	f_{10} - word count of r
	f_{11} - number of searches of r within a week
	f_{12} - number of clicks of r within a week
	f_{13} - number of payments of r within a week
Query Features	f_{14} - length of q
	f_{15} - word count of q
	f_{16} - number of searches of q within a week
	f_{17} - number of clicks of q within a week
	f_{18} - number of payments of q within a week

5 EXPERIMENTS

5.1 Dataset

As introduced in Section 3.2, our co-training framework makes use of three types of data: *human labeled dataset* (D_1^* and D_2^*), *unlabeled triples set* (U) and *clickthrough data* for training the two models. Here we describe the details of building these datasets.

5.1.1 QR Human Labeled Dataset. We sample 19551 queries randomly from 1-year search logs of *www.taobao.com*. For every query, we select some rewriting candidates. Because the amount of *good* rewriting candidates is much less than *bad* candidates for a query, we can hardly get positive examples if randomly selecting from the candidate set. So the top 5 candidates from every generator will be selected first, and then 10 candidates are sampled from the rest of the candidate set. There are in total 254163 (q, r) pairs. Then five human editors label each pair as *good* or *bad* by the relevance between q and documents retrieved by r. For each pair, we adopt the label that most editors assign.

This human labeled dataset is divided into three subsets. We randomly select 80% of the queries and take their labeled candidates as training set for fine-tuning, and 10% as validation set for hyper-parameter tuning and 10% for evaluation.

5.1.2 SM Human Label Dataset. To build human labeled dataset for *SM*, we also randomly sample 41701 queries from 1-year search logs and retrieve the top 30 documents for each query. Five editors label each (q, d) pair as *relevant* or *irrelevant*. The pair will be labeled as *relevant* when the product's title satisfies the intent of the query, such as product name, brand, color, sex and so on. No matter whether the query terms are matched in the title. We also use the label that most editors assign. This dataset is also divided into three subsets, where 80% for fine-tuning, 10% for validation and 10% for evaluation.

5.1.3 Unlabeled Triples Set. We build the unlabeled triples dataset $U = \{(q, r, d)\}$ by first sampling 5% (about $3 * 10^7$) search queries from 1-year search logs, and then generating 20 candidates for every query from each of the three generators. For each (q, r) pair, the top 30 clicked documents of r are combined into (q, r) pair to form the triples. In total, we get an unlabeled dataset consisting of about $1.2 * 10^{10}$ (q, r, d) triples.

5.1.4 Training Data From Click Logs. At the beginning of co-training, we collect labeled data from click logs to train initial models for the two tasks.

For training $f(q, r)$, we generate about 15 million (q, r) pairs with learning targets by the approach described in Section 3.2.1 from 1-year click logs.

For training $s(q, d)$, we sample clicked (q, d) pairs as positive examples and non-clicked (q, d) pairs as negative examples. To avoid large gap between the amount of positive and negative examples, we keep the ratio of positive and negative examples in [0.5, 2] for every query, and any extra examples will be discarded. At last, we generate about 93.8 million (q, d) pairs with labels.

5.2 Models and Training details

We use the same Bi-LSTM and attention-based pooling architecture for the two models $f(q, r)$ and $s(q, d)$ with different parameters. The size of word vocabulary is about 1.01 million and the dimension of word embedding E is 100. All the word embeddings in this paper are pre-trained on the corpus of all documents' titles by word2vec. The hidden layer size of Bi-LSTM unit is 256. We truncate all input queries and documents such that the first 10 words for query and the first 30 words for document are kept. The size of attention-based pooling layer parameters W_a for the two models is 512*1.

For the *QR* candidate ranking network, the size of fully connected layer parameter W_1 is $(512+512+13)*1024$, and the size of W_2 is $1024 * 1$. For the *SM* network, the size of W_3 is $(256 + 256) * 1024$ and the size of W_4 is $1024 * 1$.

We implement our two models using Tensorflow and train them with Adam [9] optimizer on GPU. The hyper parameters of Adam optimizer for the two models are same: β_1, β_2 and ϵ are set to 0.95, 0.999, and 0.0001. The batch size of *QR* candidate ranking network is set to 256, and 512 for *SM*. The initial learning rate in each iteration for the two networks is selected from {0.00005,0.0001,0.0002,0.0006} based on the validation set respectively.

In our co-training framework, there is a hyper parameter: confidence-threshold β. We conduct experiments with 3 different values: 0.6, 0.7, 0.8 and select the optimal value based on the validation sets.

5.3 Baseline Methods

The two initial weak models are taken as baselines. Besides, we adopt other baseline methods for comparison, both unsupervised and supervised methods.

For *QR*, the baselines are

Table 3: Performance Comparison of *QR* Models

Method	AUC	P@3	NDCG@3	MAP
Clarity Score	-	0.5974	0.5762	0.6082
Clicked Documents	-	0.6076	0.5928	0.6240
Content	-	0.6170	0.6027	0.6485
Session	-	0.6198	0.6025	0.6616
Initial Model	0.5960	0.5743	0.5583	0.5923
FT Initial Model	0.6764	0.6194	0.6012	0.6341
GBDT	0.7226	0.6214	0.6320	0.6847
Co-training	**0.7775***	**0.6654***	**0.6409***	**0.7309***

(*Initial Model* is the initial weak model trained on data generated from clickthrough data. *FT Initial Model* is the *Initial Model* fine-tuned on a human labeled dataset. *Clicked Documents*, *Content* and *Session* are three unsupervised methods as describe in Section 4.1.1. Statistically significant improvements according to two-tailed student's t-test at $p < 0.05$ are marked with *.)

(1) *Clarity Score* [2] as a quality indicator between the original query and documents set retrieved by a candidate;
(2) Candidate generators described in Section 4.1.1 which are actually various unsupervised methods;
(3) A shallow supervised model (GBDT) which is learned on a few human labeled data and uses only hand-crafted features.

For *SM*, the baselines are

(1) BM25 score between the query and document;
(2) some unsupervised semantic models, such as LSA and PLSA;
(3) A recently proposed weakly supervised method which uses BM25 as weak supervision signals generator [4].[1]

5.4 Evaluation Metrics

To evaluate the two models' performance, we use three typical evaluation metrics for *learning-to-rank*: normalized discounted cumulative gain (nDCG), mean average precision (MAP) and precision of the top retrieved documents. We also consider the two models as *two-class classifiers* and use area under curve (AUC) for evaluation. We conduct a significance test using the two-tailed student's t-test.

5.5 Results and Discussion

To compare our approach with baselines, we select the experimental results of the 3rd iteration under confidence threshold $\beta = 0.7$. Table 3,4 show that the two initial models trained by the clickthrough data perform badly because of noise and bias in the training data. Their performance can be improved by fine-tuning with a few human labeled data respectively. Our co-training framework can significantly improve their performance for both of the two tasks.

For *QR*, the unsupervised methods including *Clarity Score* and three generators perform badly. The performance of supervised learning methods exceeds these unsupervised methods. Our neural network for the *QR* task $f(q, r)$ beats a shallow model (GBDT) on only hand-crafted features because it has the ability of learning semantic features from raw text.

For *SM*, neural network models trained with supervision data beat the unsupervised methods. Our approach also achieves better performance compared with the two models trained by weak supervision of BM25. That means our approach can generate training data which is of high quality and less noisy than BM25.

5.5.1 Convergency Analyse. Table 5 shows that the performance of the two models are improved in the co-training iteration process under different confidence thresholds. At the beginning, the training dataset D_1 and D_2 are noisy and biased for the two tasks, and the two models perform badly. The co-training process can generate higher quality dataset to train better models and they can converge quickly in no more than 4 iterations.

5.5.2 Influence of Confidence Threshold. The threshold β is an important hyper parameter in the co-training process. In our experiments, we test three thresholds: 0.6, 0.7, 0.8 and find that 0.7 is the optimal value. We find that β decides the quality and quantity of dataset D_1 and D_2. Table 6 shows the size of the generated training dataset in each iteration under different thresholds. When we set a larger threshold, the two models will generate more confident labeled data for each other, but the size of the dataset is smaller. For neural network models, both the quality and the quantity of training data are important.

5.5.3 Influence of Fine-tuning. Fine-tuning is usually effective method to improve performance of neural networks with

Table 4: Performance Comparison of Semantic Matching

Method	AUC	P@20	NDCG@20	MAP
BM25	-	0.7665	0.7290	0.1792
LSA	-	0.7428	0.7154	0.1773
PLSA	-	0.7465	0.7158	0.1778
Initial Model	0.6667	0.7346	0.6847	0.1787
FT Initial Model	0.6755	0.7703	0.7240	0.1866
BM25 Sup(*Score*)	0.6418	0.7809	0.7272	0.1730
BM25 Sup(*Rank*)	0.6830	0.7900	0.7410	0.1987
Co-training	**0.7815***	**0.8134***	**0.7639***	**0.2296***

(*BM25 Sup(Score)*, *BM25 Sup(Rank)* are the *Score* and *Rank* models trained on weak supervision signals generated by BM25. Statistically significant improvements according to two-tailed student's t-test at $p < 0.05$ are marked with *.)

[1]To prepare the training data, we sample 5.4 million queries from 1-year search logs. For every query, we take the top 1000 documents retrieved by BM25 from a collection of $7 * 10^8$ documents. In total, we have about 5 billion (q, d) pairs with BM25 score as weak labels. To compare with our method, we use the same network architecture and just modify the top activation function and loss function as described in [4]. Because the *RankProb model* is not practical in real-world application, we implement *Score model* and *Rank model* on this dataset. *Score model* is a point-wise ranking model that learns to predict retrieval scores for query-document pairs, while *Rank model* is a pair-wise ranking model.

Table 5: Performance of the Two Models under Confidence Threshold: $\beta = 0.6, 0.7, 0.8$

Confidence Threshold	Iteration	*QR*				*SM*			
		AUC	P@3	NDCG@3	MAP	AUC	P@20	NDCG@20	MAP
0.6	1	0.7126	0.6403	0.6251	0.7035	0.7370	0.7902	0.7422	0.1985
	2	0.7478	0.6618	0.6448	**0.7264**	**0.7568**	**0.8181**	**0.7670**	**0.2275**
	3	**0.7698**	**0.6620**	**0.6458**	0.7228	0.7566	0.8169	0.7640	0.2192
0.7	1	0.7523	0.6400	0.6230	0.6957	0.7227	0.7997	0.7538	0.1974
	2	0.7717	**0.6673**	**0.6421**	0.7149	0.7525	**0.8185**	**0.7703**	**0.2312**
	3	**0.7775**	0.6654	0.6409	**0.7309**	**0.7815**	0.8133	0.7639	0.2296
0.8	1	0.7326	0.6449	0.6288	0.7074	0.7413	0.7953	0.7428	0.1944
	2	0.7552	0.6566	0.6411	**0.7247**	**0.7603**	**0.8113**	**0.7607**	**0.2253**
	3	**0.7615**	**0.6634**	**0.6441**	0.7205	0.7526	0.8082	0.7547	0.2203

Table 6: Size of Generated Training Dataset for Different Confidence Thresholds

	QR			*SM*		
Iteration	0.6	0.7	0.8	0.6	0.7	0.8
1	$5.0 * 10^6$	$4.4 * 10^6$	$4.1 * 10^6$	$1.36 * 10^8$	$1.17 * 10^8$	$9.71 * 10^7$
2	$8.5 * 10^6$	$7.9 * 10^6$	$6.5 * 10^6$	$2.54 * 10^8$	$2.24 * 10^8$	$2.04 * 10^8$
3	$1.7 * 10^7$	$1.4 * 10^7$	$1.1 * 10^7$	$2.61 * 10^8$	$2.55 * 10^8$	$2.50 * 10^8$

Table 7: Performance of Pre-trained Models in Co-training of the Best Confidence Threshold: 0.7

QR				
Iteration	AUC	P@3	NDCG@3	MAP
1	0.6720	0.6311	0.6138	0.6585
2	0.7264	0.6449	0.6281	0.6824
3	0.7356	0.6450	0.6241	0.6885
SM				
Iteration	AUC	P@20	NDCG@20	MAP
1	0.7156	0.7604	0.7052	0.1902
2	0.7041	0.8011	0.7474	0.2205
3	0.7116	0.8003	0.7409	0.2181

a few human labeled data. As shown in Table 3 and Table 4, performances of the two initial weak models are improved significantly when they are fine tuned. However, compared with the improvement achieved by our approach, it is limited by the weakness of initial models and the quantity of human labeled data.

We also apply fine-tuning in every iteration of the co-training process. As the results shown in Table 7, compared with Table 5, fine-tuning the networks with a few human labeled data further improves their performance in each iteration. There are still a lot of noise in the generated training data, and a better model can improve the quality of generated data for the other in next iteration.

5.6 Online Evaluation

We conduct online evaluation in a real-world e-commerce search engine of *www.taobao.com* with standard A/B testing configuration. Top 5 candidates for every input query are selected to recall more relevant documents along with the original query. The score of *SM* model is deployed as an input feature of *learning-to-rank* for relevance ranking. We conduct three groups of experiments: deploying *QR* and *SM* separately, and deploying them simultaneously. For each test, 3% of the users are randomly selected as testing group. The effectiveness is evaluated by some business metrics: Gross Merchandise Volume (GMV), Conversion Rate (CVR) and Unit Price.

Overall Improvement. The results shown in Table 8 suggest that the two models can achieve overall improvement in terms of GMV and CVR, and they cooperate to achieve better improvement. The two models retrieve more relevant products for a user query in *recall* and *ranking* phase respectively. High quality products will have more opportunities to be retrieved and displayed, which increases the probability of user purchase. However, they have little impact on the price of purchased products, which is mainly influenced by user's consumption capacity and price preference.

Improvement for Different Queries. Increasing the amount of retrieved products has different degrees of influence on different types of queries. We divide the queries into three categories according to the search frequency: *top* query, *torso* query and *tail* query. Each type accounts for about 1/3 of the total search traffic.

The online evaluation results of different types of query are shown in Table 8. The *top* queries are usually short and can retrieve enough products. Hence the two models will have smaller impact on the performance of these queries. The *torso* and *tail* queries are usually long or contain rare words. Retrieving more relevant products is more important for them, and the improvements achieved on these queries are more significant. However, for *tail* queries, the two models increase the CVR markedly, but the unit price becomes much lower. This is because when users have more choices, they prefer to choose cost-effective products.

6 CONCLUSIONS

In this paper, we propose a co-training framework for *query rewriting* and *semantic matching*. The two tasks have a close relationship and we consider them as different views of the

Table 8: Performance Improvement for Overall and Different Type of Queries of Online A/B Testing

Query	Deployed Model	GVM	CVR	Unit Price
Overall	QR	+2.11%	+2.22%	-0.48%
	SM	+1.61%	+0.73%	+0.49%
	QR+SM	+2.87%	+2.63%	+0.3%
Top	QR	+0.63%	+0.56%	+0.3%
	SM	+0.28%	+0.87%	-0.59%
	QR+SM	+0.71%	+0.94%	-0.27%
Torso	QR	+5.2%	+4.94%	+0.27%
	SM	+2.62%	+2.32%	+0.36%
	QR+SM	+5.32%	+5.06%	+0.33%
Tail	QR	+2.13%	+9.51%	-5.38%
	SM	+4.35%	+3.17%	+0.56%
	QR+SM	+4.87%	+10.23%	-4.76%

same relevance problem. Our co-training framework let the two models provide weakly supervision signals for each other to address the sparseness problem of training data for the two tasks. We conduct a series of offline and online experiments on a real-world e-commerce search engine. The results show that our method is very effective to improve both the two models' performance.

Our future work includes two directions. First, we will try a *Shared-Private* multi-task model which learns the two tasks in one model. They share a task-invariant query embedding and each has a task-dependent query embedding. Second, we will investigate other networks to learn semantic representation in our framework.

ACKNOWLEDGMENTS

Yanghua Xiao was supported by National NSFC (No.61732004, No.61472085), Shanghai STCSMs R&D Program under Grant (No.16JC1420400)

REFERENCES

[1] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 243–250.

[2] Steve Cronen-Townsend, Yun Zhou, and W Bruce Croft. 2002. Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 299–306.

[3] Mostafa Dehghani, Aliaksei Severyn, Sascha Rothe, and Jaap Kamps. 2017. Avoiding Your Teacher's Mistakes: Training Neural Networks with Controlled Weak Supervision. *arXiv preprint arXiv:1711.00313* (2017).

[4] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *the 40th International ACM SIGIR Conference*. ACM Press, New York, New York, USA, 65–74.

[5] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context- and Content-aware Embeddings for Query Rewriting in Sponsored Search. In *the 38th International ACM SIGIR Conference*. ACM Press, New York, New York, USA, 383–392.

[6] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 55–64.

[7] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 1443–1452.

[8] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2333–2338.

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Svetlana Kiritchenko and Stan Matwin. 2001. Email classification with co-training. In *Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*. IBM press, 8.

[11] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. 3294–3302.

[12] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24, 4 (2016), 694–707.

[13] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition.. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. 2793–2799.

[14] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 257–266.

[15] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *the 38th International ACM SIGIR Conference*. ACM Press, New York, New York, USA, 373–382.

[16] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 101–110.

[17] Jiwei Tan, Xiaojun Wan, Hui Liu, and Jianguo Xiao. 2018. QuoteRec: Toward Quote Recommendation for Writing. *ACM Transactions on Information Systems (TOIS)* 36, 3 (2018), 34.

[18] Tao Tao and ChengXiang Zhai. 2006. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 162–169.

[19] Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-volume 1*. Association for Computational Linguistics, 235–243.

[20] Wei Wang and Zhi-Hua Zhou. 2010. A New Analysis of Co-Training. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. 1135–1142.

[21] Jinxi Xu and W Bruce Croft. 1996. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 4–11.

[22] Hamed Zamani and W Bruce Croft. 2017. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 505–514.

[23] Hamed Zamani, W. Bruce Croft, and J. Shane Culpepper. 2018. Neural Query Performance Prediction using Weak Supervision from Multiple Signals. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*. 105–114.