Advancing Query Rewriting in E-Commerce via Shopping Intent Learning

Mengxiao Zhang*
University of Southern California
Los Angeles, CA, USA

Yongning Wu
Amazon.Inc
Seattle, USA

Raif Rustamov
Amazon.Inc
Seattle, USA

Hongyu Zhu
Amazon.Inc
Seattle, USA

Haoran Shi
Amazon.Inc
Seattle, USA

Yuqi Wu
Amazon.Inc
Seattle, USA

Lei Tang
Amazon.Inc
Seattle, USA

Zuohua Zhang
Amazon.Inc
Seattle, USA

Chu Wang
Amazon.Inc
Seattle, USA

ABSTRACT

Query understanding plays a key role in the search process, and accurate understanding of search queries is the first step toward high-quality search results on e-commerce websites. While head queries with abundant historical data can be easier to interpret, tail queries pose a challenge to accurate understanding. To tackle the challenge, we focus on query rewriting to transform a tail query into a query with similar linguistic characteristics as head queries and preserve the shopping intent. In this work, we present a new training data construction process and extend the vanilla Seq2Seq model with multiple auxiliary tasks to achieve some desirable features for e-commerce applications. For the training data of query rewriting, we only rely on widely available search logs to generate (source, target) query pairs and additional shopping intent information. This additional information provides two auxiliary prediction tasks on product name and category into our model to fully capture the shopping intent. For the model, we propose a query matching loss based on a novel co-attention scheme to improve the source query representations, so that the overall model can be built and trained end-to-end with standard components and training protocols. The resulting model provides significant advantages over the vanilla Seq2Seq model in a range of experiments on rewriting quality. We also demonstrate the practical value of our query rewriting model with an application in sponsored search.

ACM Reference Format:
Mengxiao Zhang, Yongning Wu, Raif Rustamov, Hongyu Zhu, Haoran Shi, Yuqi Wu, Lei Tang, Zuohua Zhang, and Chu Wang. 2022. Advancing Query Rewriting in E-Commerce via Shopping Intent Learning. In *Spain '22: SIGIR, June 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 9 pages.

*The work is done during MZ's internship in Amazon.

SIGIR eCom'22, July 15, 2022, Madrid, Spain

1 INTRODUCTION

E-commerce platforms process billions of queries on a daily basis, and query understanding is the key to return high-quality search results. Popular queries are generally easier to interpret due to rich historical data available. However, it is far more challenging to understand tail queries and infer their shopping intents. Web query understanding researches [9, 20] and recent investigations in the e-commerce field [13] suggest that most tail queries express the same intents as related head queries, but with different linguistic characteristics like uncommon wording or typos. This opens the door to understanding tail queries via *query rewriting*, whereby a tail query is transformed into one that has the linguistic characteristics of a head query so that the search engine can generate high-quality recommendations.

For e-commerce applications, it is desirable to have query rewriting models which are able to 1) handle queries for new products, 2) correctly capture the shopping intents, and 3) require no additional input data beyond standard search log. Works based on mapping each tail query to a head query via semantic query matching [26] cannot handle queries relating to new products: for example, "iPhome 14" cannot be rewritten to "iPhone 14" as the latter has not been seen before. Despite the enormous success of deep learning in Natural Language Processing, including language translation [2], text summarization [10, 15], and text classification [8], building query rewriting systems that achieve all of the above features remains challenging. Approaches based on machine translation models such as Seq2Seq can handle new products, but will have difficulty correctly inferring the shopping intent unless a vast amount of data is available. For example, works such as [18, 24] modify vanilla Seq2Seq models, yet [24] requires extra query annotations as the model input.

In this work, we propose a new training data construction process and a novel transformer-based query rewriting Seq2Seq model that incorporates multiple improvements to achieve the above desired features. Specifically, we shift from noisy and limited users' query rewriting behavioral data to commonly available search logs and introduce a new process for constructing source and target query pairs. This process is designed to extract additional information about the shopping intent, which allows us to add two auxiliary prediction tasks into our model with the goal of learning query

representations that can capture the shopping intent. To further improve the representation quality and the generalizability of our model, we propose a semantic query matching loss using a novel co-attention mechanism. In contrast to the model proposed by [18] which has a complicated training protocol that is not desirable for production, our model is built and trained end-to-end using standard deep learning components and training protocols. It achieves significant practical advantages over the vanilla Seq2Seq model and its extensions as demonstrated on various metrics of query rewriting. We also show the practical value of our model with an application in sponsored search.

The paper is organized as follows. After reviewing the related work in Section 2, we provide an overview of our problem setting and model design in Section 3. We then go through the details of the data construction process and the proposed model in Section 4. We provide a range of experiments evaluating our model performance on rewriting quality and include a production experiment of sponsored search in Section 5.

2 RELATED WORK

2.1 Query Rewriting

A series of query rewriting works rely on a two-phase framework [6]. The first phase aims to generate a number of possible query candidates, and the second phase includes a ranking model which ranks the candidates from the first phase. Specifically, in [6], the first phase is trained using the product name corresponding to the source query as the target query label, and the second phase uses the discounted cumulative gain (DCG) as the label to rank the candidates. Many follow-up works improve this model in different ways. For the first phase, related works include the utilization of a search dataset [25], a synonym dictionary [14], part-of-speech prediction [22], and query-product N-gram pairs [11]. For the second phase, the improvements include using co-training [25], a random forest classifier with handcrafted features [14], and pretrained language models [11, 22].

Another type of query rewriting works only relies on a Seq2Seq model to generate rewritten queries [21], which does not involve a ranking phase. One advantage of this framework is that it can be trained in a fully end-to-end way with all the learning parameters differentiable, as opposed to the two-phase method where in the candidate generation phase, usually beam search or other candidate generation methods are applied. This framework has been widely used in many NLP applications, such as neural machine translation [2], text summarization [15], and visual question answering (VQA) [1]. To achieve these, many Seq2Seq structures are proposed. Related works involve recurrent neural networks like LSTM [7] and GRU [4], fully attention-based models like Transformers [23], as well as its variants BERT [5] and GPT [19].

Unlike classic neural machine translation tasks, e-commerce query rewriting involves many intrinsic difficulties, especially the lack of contextual information for the query. Therefore, Wang et al. [24] incorporate part-of-speech information for each word in the query and utilize the combined representation to decode the target popular entry. Qiu et al. [18] extract product name information for each query via search logs and use a cyclic training framework to learn two Seq2Seq models, namely from query to product name and from product name to query, simultaneously.

2.2 Semantic Matching among Queries

For better query information retrieval, another line of works focuses on query semantic matching, which aims to compute the similarity of two queries by learning latent representations. Yang et al. [26] use a BiLSTM model to learn representations for the source query and the rewritten query with handcrafted features as side information. In addition, to handle the issue of limited human-labeled data, they use uncertainty and novelty sampling schemes to augment their dataset in an active way. In fact, Yang et al. [26] also adopt the semantic matching model to generate rewritten queries by selecting the query of the highest similarity with the original product. However, this method can not generate unseen popular queries for new coming products, which is a disadvantage compared to methods discussed in Section 2.1.

Maji et al. [13] point out that using a single fixed vector to represent a query and further measure the query similarity may be problematic. A co-attention mechanism introduced in visual question answering (VQA) [12] is used to construct different embedding vectors for one query when comparing to different queries.

2.3 Query Product Relevance Model

Measuring the relevance between the source query and the product name is another way of query understanding which directly helps products or ads recommendation. Yao et al. [27] use a single encoder to learn embeddings for both queries and product names. Then, a multi-aspect attention module is applied to the two representations separately, and the relevance score is computed by applying a Multi-Layer Perceptron model to the output representations. [3] introduces a model which uses non-displayed products as unsupervised data and learns embeddings for queries and products with multiple loss components including the difference between the representation correlations among frequently displayed products and non-displayed products to improve the performance on tail products. Similar to query rewriting, context information can also be incorporated in query-product relevance models. Zhang et al. [28] use a single convolutional network to generate the representations for both the query and the product. Instead of directly computing the distance between the two representations, they combine an auxiliary query taxonomy and a product taxonomy classification with a progressively hierarchical structure to learn better representations for products and queries.

3 OVERVIEW: PROBLEM SETUP AND MODEL DESIGN

Given our emphasis on capturing the shopping intent, the vanilla Seq2Seq model is not sufficient. We intend to inject the shopping intent information via auxiliary tasks. The additional information is extracted by our data construction process, which can build an enriched training dataset. Our training data is derived from aggregated search logs and provides tuples of the form $Q = \{(q_i^o, q_i, s_i, y_i)\}_{i=1}^m$. Here, m is the number of data points, q_i^o denotes the i-th source query, and q_i is the i-th target query. Additionally, for every target query q_i, we are able to capture the name string s_i of the most

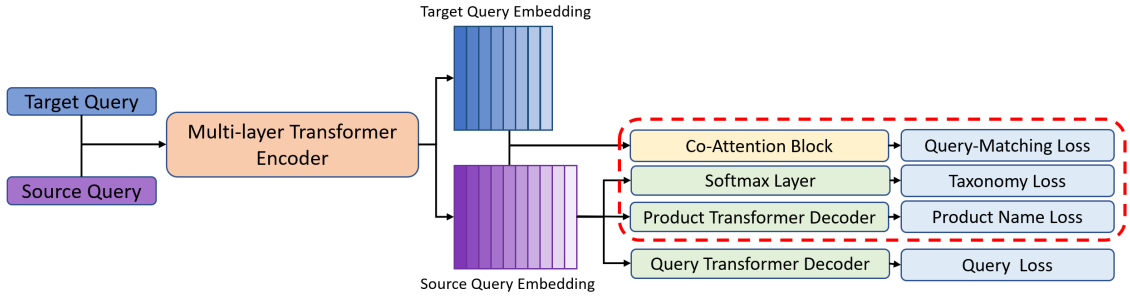

Figure 1: An overview of our model. Both the source query and the target query are fed into the model and are encoded by the same Transformer Encoder. Each column of the embedding represents one token in the query. The rest of the model includes four main modules which combine four different tasks. The main task is the query transformer decoder which generates the target query sequence. To improve this vanilla model, we introduce the three modules in the red dotted box. The co-attention block produces a query-matching loss with a distance function based on the source and target query embeddings. A Transformer decoder is used to decode the source query embedding into a product name sequence. A softmax layer is used to predict the category of the above-mentioned product. Finally, the four losses are combined together to train the model.

popular relevant product and this product's category label y_i. As an example, a training data point may look like this ("0 organik milk", "fat-free organic milk", "bouncy cow brand fat free organic milk", "grocery"). Of course, at inference time, only the source query is available and the model will output the target query.

Our approach to query rewriting is based on a Seq2Seq model with transformers [23] used for both encoder and decoder components. This Seq2Seq model with a parameter vector $\boldsymbol{\theta}$ takes $\{q_i^o\}_{i=1}^m$ as inputs and learns to output the corresponding queries $\{q_i\}_{i=1}^m$ by minimizing the negative log likelihood:

$$\mathcal{L}_{\text{query}}(\boldsymbol{\theta}) = -\sum_{i=1}^{m} \log \Pr(q_i \mid q_i^o, \boldsymbol{\theta}).$$

The encoder provides embeddings of queries in a high-dimensional space. We intuitively expect the embedding of a source query to capture some shopping intent and to map related queries nearby so that the decoder can output a query with the same intent that carries the linguistic characteristics of head queries. However, without additional loss terms, the Seq2Seq model's encoder has no incentive to infer the shopping intent. In addition, it is possible that the encoder can map related queries to disparate regions and rely on the powerful decoder to decode them into the same target query. We speculate that this can harm generalization, whereby a new related query gets mapped into a "hole" in the embedding space and the decoder ends up producing nonsensical output. To capture the shopping intent and favorably shape the embedding space, we take advantage of our enriched training dataset and modify the vanilla Seq2Seq model with several auxiliary tasks.

An overview of our model is shown in Figure 1. Specifically, we include three additional losses: two to capture the shopping intent and one to enforce contiguity for related queries. First, we add a product name prediction task with a product name loss $\mathcal{L}_{\text{product}}(\boldsymbol{\theta})$: namely the embedding of the source query q_i^o is used to generate the product name s_i with a transformer decoder. This provides extra supervision as there is usually a gap between queries and product names [13]. Second, a softmax layer is placed on top of the source query embedding to predict the category label y_i with a taxonomy loss $\mathcal{L}_{\text{cls}}(\boldsymbol{\theta})$. Third, for a query rewriting pair (q_i^o, q_i), as the queries q_i^o and q_i carry similar shopping intents, we expect the embeddings of both queries are close. Thus, we add a query-matching loss $\mathcal{L}_{\text{sim}}(\boldsymbol{\theta})$ with a novel co-attention scheme which generalizes the classic attention scheme in [23].

With all the above three losses, the loss function of our model is as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \mu_1 \mathcal{L}_{\text{query}}(\boldsymbol{\theta}) + \mu_2 \mathcal{L}_{\text{product}}(\boldsymbol{\theta}) + \mu_3 \mathcal{L}_{\text{cls}}(\boldsymbol{\theta}) + \mu_4 \mathcal{L}_{\text{sim}}(\boldsymbol{\theta}),$$

where $\{\mu_i\}_{i=1}^4$ are the hyper parameters. These parameters are chosen based on a grid search according to the performance on the validation set.

4 METHODOLOGY

In this section, we provide the details of data construction process and the proposed model. Section 4.1 introduces our dataset construction process from standard search logs. Section 4.2 describes two auxiliary tasks for product name and category predictions; Section 4.3 introduces the query-matching loss and our novel co-attention scheme.

4.1 Data Construction

Since one of key factors for a successful model is the training data, it is desirable to have a data construction process that relies on routinely collected logs in e-commerce, avoids manual effort to produce large amounts of data, controls the amount of noise, and captures the shopping intent. The first requirement is crucial in making the approach widely applicable to different e-commerce platforms, and the rest contributes to the improved model performance. Existing approaches include the use of search logs [22, 25, 27, 28], query and product name pairs [6], query autoencoding [18], synonym generation [11, 14], users' rewriting behaviours [24], or human active labeling [26]. These methods fail to satisfy all of the above requirements, and it leads us to propose a novel data construction approach from search logs.

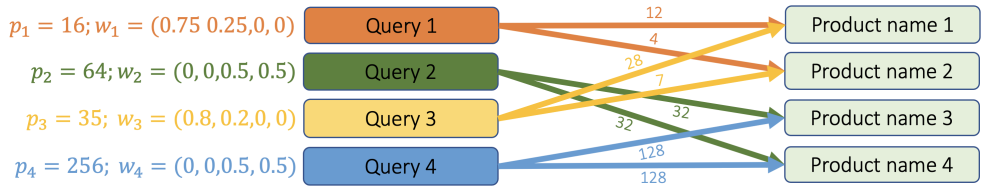

Figure 2: An overview of our dataset construction methodology based on a search log. The above sample graph is a bipartite graph between 4 queries and 4 product names, where an edge between a query q and a product i with weight $p_{q,i}$ means that users have used query q to have access to product i with popularity $p_{q,i}$. $p_q \triangleq \sum_{i=1}^{4} p_{q,i}$ denotes the total popularity of a query q with 4 products. Here, we use ℓ_1 distance as an example. Setting the popularity distance threshold $\sigma = 0.2$ and overall popularity threshold $\tau = 48$, we have $\text{dist}_{s,\ell_1}(w_1, w_3) = 0.1 \le \sigma$, $\text{dist}_{s,\ell_1}(w_1, w_2) = \text{dist}_{s,\ell_1}(w_1, w_4) = 1 > \sigma$. As $p_1 < p_3$, we will rewrite query 1 to query 3 and query 3 will be rewritten to itself. In addition, although $\text{dist}_{s,\ell_1}(w_2, w_4) = 0 \le \sigma$ and $p_4 > p_2$, query 2 and query 4 will both be rewritten to themselves as p_2 and p_4 are larger than τ, indicating that both queries are popular enough and should be well understood by the search engine.

A search log aggregates user behaviors to obtain the relationships between queries and products. It can be conceptualized as a bipartite graph between queries and products. An edge means the query has historically led to clicks or purchases of the product. The weighted sum of clicks and purchases is used to obtain edge weights. Additionally, based on a popularity threshold, we categorize queries into head and tail queries. We posit that two queries that have high overlap in relevant products should have the same shopping intent. Next, for every tail query, we find the most popular query with high enough overlap, and emit this query pair together with some extra shopping intent information as one training data point. Figure 2 illustrates the construction process with details provided below.

To be more precise, we want to rewrite a tail query to a popular one with similar shopping intent. Moreover, the popular queries should be rewritten to themselves since they can be already well understood by e-commerce systems. To this end, we first compute the edge weight $p_{q,i}$ of a query-product pair (q, i) as the weighted sum of clicks and purchases of product i by using query q. Second, we sort the products that are connected to q by edge weight and select the top t products $S_q = \{i_{q,1}, \ldots, i_{q,t}\}$ for each query q. This truncation is done because tail products may lead to noisy signals. Also it resolves the computational issues as the quantities below are computed via joins. Now, for each query, we build the sparse vector $\boldsymbol{w}_q \in \mathbb{R}^{n_{\text{product}}}$ where n_{product} is the total number of products, and the i-th entry of $\boldsymbol{w}_q$ is defined as the normalized edge weight

$$w_{q,i} = \begin{cases} 0 & \text{if } i \notin S_q, \\ \frac{p_{q,i}}{\sum_{i \in S_q} p_{q,i}} & \text{if } i \in S_q. \end{cases}$$

Next, we compute the similarity between two queries q_1 and q_2 by computing a distance $\text{dist}_s(\boldsymbol{w}_{q_1}, \boldsymbol{w}_{q_2})$ between $\boldsymbol{w}_{q_1}$ and $\boldsymbol{w}_{q_2}$. Here, one can pick any distance metric. In the main experiments, we use the cosine distance, but we have also experimented with ℓ_1 distance (see Appendix).

Finally, a distance threshold $\sigma > 0$ is set and we construct a source-target query pair (q^o, q) where q is the query of the highest overall popularity under the condition $\text{dist}_s(\boldsymbol{w}_{q^o}, \boldsymbol{w}_q) \le \sigma$. Note that using this method, a query q^o will be rewritten to itself if q^o itself is the query of the highest popularity under the weight distance constraint. One exception to the rule is that if q has total popularity (weighted degree) $p_q \triangleq \sum_{i=1}^{n_{\text{product}}} p_{q,i} \ge \tau$ (popularity threshold), the query q is rewritten to itself instead as this query has enough historic clicks/purchases and so is well understood.

An important aspect of our constructions is that it provides side information for the queries. First, for each pair (q^o, q), a product set S_q corresponding to q is available, which can be interpreted as the potential target products for the original query q^o as we assume q^o and q are close. Second, e-commerce platforms commonly have product attributes including product name and various taxonomies. We incorporate this extra information into our query rewriting dataset with the product name string s and the product category y of the product with highest weight in S_q. The final training data point is the tuple (q^o, q, s, y).

4.2 Capturing Shopping Intent

In this subsection, we introduce two auxiliary tasks for capturing the shopping intent in queries. These tasks utilize the additional information associated with the query: *product name* and *product category*. Intuitively, auxiliary tasks help the encoder to learn a better query representation by capturing the shopping intent in the embedding space.

Product Name Generation. For each training data point (q^o, q, s, y), besides the target query q, we have a product name s related to the target query in the search log. This indicates that the source query q^o should also be closely related to the product, which can serve as extra information to help query rewriting.

To utilize this information, we use a separate decoder to decode the product name based on the source query embedding (i.e. the output of the encoder). This helps the encoder to learn a better embedding of the source query as the encoded feature needs to be informative enough to decode the product name successfully. Formally, we introduce the product name prediction loss:

$$\mathcal{L}_{\text{product}}(\boldsymbol{\theta}) = -\sum_{i=1}^{m} \log \Pr\left(s_i | q_i^o, \boldsymbol{\theta}\right),$$

where m is the size of the dataset and s_i is the product name corresponding to the source query q_i^o.

A number of prior works also utilize the product name information in query rewriting. Among recent work is [18], which proposes an autoencoder model for queries with product name decoding as the bottleneck task. Compared to that model, our model has two main advantages. First, our model is trained in a fully end-to-end way. [18] requires a warm-up training stage because the initial model fails to generate meaningful product names and their bottleneck beam search process for product name generation is not differentiable. Second, [18] uses the autoencoder framework, which takes the source query as the target label query to train both the encoder and the decoder. In contrast, the target query is explicit in our dataset, which provides more direct supervision for the query rewriting task.

Category Prediction. Taxonomies are commonly available in different e-commerce platforms. Product taxonomies can help understand shopping intents and enrich search results [28]. In this work, we use the category label as an auxiliary task to help the encoder include more information about the source query. Specifically. we use our existing product types as the class of the input query, including hardlines (e.g. electronics, wireless), softlines (e.g. shoes), consumables (e.g. grocery). Some sampled classes are "mp3 player", "hardware plugs", "fashion sneakers", "soy milks". Given the embedding $\boldsymbol{V}_{q^o}$ of the source query q^o, we select the embedding vector $\boldsymbol{v}_{q^o,[sos]}$ of the first token "[sos]" and feed it directly into a softmax layer to obtain a distribution (in probability simplex) of the predicted category $u_{q^o,j} \in \Delta_r$, where r is the total number of classes. Formally, we construct the following cross-entropy loss function for the category label:

$$\mathcal{L}_{\text{cls}}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \sum_{j=1}^{r} -y_{i,j} \log u_{q_i^o,j},$$

where $y_i \in \Delta_r$ is the true category (or true category distribution in case of multiple labels) of the source query q_i^o.

4.3 Query Matching Loss via Co-Attention

The query matching loss is to achieve contiguity in the embedding space where related queries are mapped near each other. This is not straightforward as the encoder provides an embedding for each token in the query, and, therefore, the query embedding can be thought of as a matrix with a fixed number of rows (embedding dimension) but a varying number of columns (one per token). While equal-weighted aggregation over columns to obtain an overall representation of the query is an option, this can be sub-optimal as pointed out by [13]. Indeed, when comparing two queries, there are correspondences over subsets of words and some words should carry more weight than others depending on each individual *query pair*.

Before introducing our co-attention method, we first review the co-attention method in Maji et al. [13] which was originally proposed in the context of visual question answering in [12]. Given two queries q and q', the encoder computes their embeddings $\boldsymbol{V} = [\boldsymbol{v}_1; \ldots; \boldsymbol{v}_n] \in \mathbb{R}^{n \times d}$ and $\boldsymbol{V}' = [\boldsymbol{v}'_1; \ldots; \boldsymbol{v}'_{n'}] \in \mathbb{R}^{n' \times d}$, where n (resp. n') is the length of the query q (resp. q'), and d is the embedding dimension. To compute a semantically meaningful distance between $\boldsymbol{V}$ and $\boldsymbol{V}'$, Maji et al. [13] use a learnable attention matrix $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ and define the similarity matrix C as follows:

$$\boldsymbol{C} = \tanh\left(\boldsymbol{V}\boldsymbol{W}\boldsymbol{V}'^{\top}\right) \in \mathbb{R}^{n \times n'}. \tag{1}$$

The (i, j)-th entry of matrix $\boldsymbol{C}$ intuitively measures how the i-th token of query q is correlated to the j-th token of query q'. Then, the overall query embeddings for q and q' are computed as $\boldsymbol{h} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i \boldsymbol{v}_i$ and $\boldsymbol{h}' = \sum_{i=1}^{n'} \boldsymbol{\alpha}'_i \boldsymbol{v}'_i$, where

$$\boldsymbol{\alpha} = \text{softmax}(\max_j(C_{:,j})); \ \boldsymbol{\alpha}' = \text{softmax}(\max_i(C_{i,:})). \tag{2}$$

Now that we have a single d-dimensional vector per query, we can use any distance function between $\boldsymbol{h}$ and $\boldsymbol{h}'$ as a measure of (dis-)similarity between q and q'.

Although it seems reasonable at first glance, that model does not fit into the classic attention mechanism introduced by [23] as the co-attention is not *symmetric*. In our work, we introduce a new co-attention mechanism, which fits the idea of [23]. Formally, given a query-projection matrix $\boldsymbol{W}_Q \in \mathbb{R}^{d \times d_k}$ and a key-projection matrix $\boldsymbol{W}_K \in \mathbb{R}^{d \times d_k}$, we construct a similarity matrix $\widetilde{\boldsymbol{C}}$ for query q based on query q' and a matrix $\widetilde{\boldsymbol{C}}'$ for query q' based on query q as follows:

$$\widetilde{\boldsymbol{C}} = \tanh\left(\frac{\boldsymbol{V}\boldsymbol{W}_Q(\boldsymbol{V}'\boldsymbol{W}_K)^{\top}}{\sqrt{d_k}}\right), \tag{3}$$

$$\widetilde{\boldsymbol{C}}' = \tanh\left(\frac{\boldsymbol{V}'\boldsymbol{W}_Q(\boldsymbol{V}\boldsymbol{W}_K)^{\top}}{\sqrt{d_k}}\right). \tag{4}$$

Specifically, to know what tokens query q should focus on, we match the representation of each token in q with the key of each token in q' by doing a scaled inner product. Intuitively, the higher the (i, j)-th entry of C is, the more relevant the i-th token of query q is to the j-th token of query q'. C' can be interpreted in a similar way. Here, tanh operator is used to rescale the similarities in the range of $[-1, 1]$ as suggested in [12, 13].

Now, we compare Equation 3 and Equation 4 with Equation 1 to understand the difference. According to Equation 1 and Equation 2, the similarity matrix $\boldsymbol{C}'$ for query q' based on query q is

$$\boldsymbol{C}' = \boldsymbol{C}^{\top} = \tanh\left(\boldsymbol{V}'\boldsymbol{W}^{\top}\boldsymbol{V}^{\top}\right) \in \mathbb{R}^{n' \times n}. \tag{5}$$

We can find that Equation 1 and Equation 5 do not match Equation 3 and Equation 4 unless $\boldsymbol{W}$ is symmetric.

After obtaining the similarity matrices $\widetilde{\boldsymbol{C}}$ and $\widetilde{\boldsymbol{C}}'$, we compute the representations $\widetilde{\boldsymbol{h}}$ of q and $\widetilde{\boldsymbol{h}}'$ of q' as $\widetilde{\boldsymbol{h}} = \sum_{i=1}^{n} \widetilde{\boldsymbol{\alpha}}_i \boldsymbol{v}_i$ and $\widetilde{\boldsymbol{h}}' = \sum_{i=1}^{n'} \widetilde{\boldsymbol{\alpha}}'_i \boldsymbol{v}'_i$, where

$$\widetilde{\boldsymbol{\alpha}} = \text{softmax}(\max_j(\widetilde{\boldsymbol{C}}_{:,j})); \widetilde{\boldsymbol{\alpha}}' = \text{softmax}(\max_j(\widetilde{\boldsymbol{C}}'_{:,j})).$$

In this way, the similarity between q_i and q'_i can be computed with a classic distance function on $\widetilde{\boldsymbol{h}}_{q_i}$ and $\widetilde{\boldsymbol{h}}_{q'_i}$ and we can introduce the query matching loss as:

$$\mathcal{L}_{\text{sim}}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \text{dist}(\widetilde{\boldsymbol{h}}_{q_i}, \widetilde{\boldsymbol{h}}_{q'_i}).$$

In contrast to the attention scheme in [23], we do not use a value projection matrix on the output embeddings $\boldsymbol{V}$ and $\boldsymbol{V}'$. The reason is that we *do not* construct negative query pair samples, which is different from other query semantic matching works [3, 13, 27]. Specifically, we are computing the representations of the source

	$BLEU_4$	$Jaccard_1$	$Jaccard_2$	F_1 score	F_2 score
baseline	0.2691	0.3289	0.2165	0.4465	0.2999
+ query taxonomy	0.271	0.3312	0.2166	0.4489	0.2995
+ product name decode (sequentially)	0.2792	0.3384	0.2253	0.4573	0.3106
+ attn (fixed attn) (separately)	0.283	0.3415	0.2288	0.4605	0.3149
+ attn [13] (separately)	0.2883	0.3473	0.2363	0.4627	0.3187
+ attn (Section 4.3) (separately)	**0.307**	**0.3669**	**0.2537**	**0.4816**	**0.3368**

Table 1: Ablation study on the test dataset with cos distance. The first three rows are added sequentially, which means the third row shows the performance of the model with both query taxonomy loss and product name decoding loss. The last three rows show the results of the models with all the four losses but different attention mechanisms. The results show that the auxiliary tasks indeed improve the performance and our novel co-attention mechanism performs better than the existing co-attention mechanism.

query and the target query, which are always *positively* related. Suppose a value projection is applied; then we may encourage a trivial all-zero value projection matrix which makes all queries to have the same representation, reducing the query-matching loss to zero. Note that the lack of negative samples is not a problem in our setting because the source query embedding is used by the decoder to output the target query. This signal keeps the query embeddings separate without requiring negative query pairs.

5 EXPERIMENTS

To construct the dataset from the search log for model training, we apply the process introduced in Section 4.1 on a 180-day search log via two distance metrics: cos distance and ℓ_1 distance on the normalized popularity weight. To define the shopping intent for the query, for a rewriting pair (q^o, q), we assign the product name of the *top* popular product in S_q as the product name s for the source query q^o. We also assign the product category of the *top* popular product in S_q as the category label y of query q^o. All data used are aggregated, anonymized, and no identifiable customer information is used in the experiments. We defer the detailed configurations of the dataset and dataset samples to Appendix. We also defer the detailed model configurations to Appendix due to page limit.

To generate the rewritten sequences, we use standard beam search with beam size 4 and set the final top-1 sequence as the output. In the following subsections, we show the empirical results of our method. The evaluation of a query rewriting model is generally hard as there is no straightforward way to quantify the quality of the rewriting [18]. To that end, we first measure the performance of our model on the test dataset from the search log, and then we conduct real-world experiments in sponsored search to further demonstrate the advantage of the proposed method.

5.1 Query Rewriting Quality Evaluation

To measure the performance of our model on the test dataset constructed by the process introduced in Section 4.1, we use the following three standard metrics that are widely used in NLP: BLEU score, F_n score, and $Jaccard_n$ score (see Appendix for details).

5.1.1 Ablation Study. Table 1 shows the results of the three metrics of our model on the cos-distance-based datasets with an ablation study. The results of ℓ_1-distance-based datasets are listed in Table 7 in the Appendix. Generally, our model performs better in all of the three metrics with more shopping intent information combined during the training process. This confirms our intuition that both the query taxonomy task and the product name decoding task help the encoder to learn an embedding of the source query better. Moreover, we can observe from the table that, with a similarity loss between the source query and the target query, we have a significant boost in all of the metrics, which shows the advantage of learning better query embeddings.

In addition, to see whether the representation generated by co-attention actually improves the query rewriting quality, we compare the performance of the models with different similarity losses, where we use the embedding of the first token of the encoded query as the representation. We can see from Table 1 that the model with a co-attention scheme indeed outperforms the one with fixed representation, which confirms our intuition discussed in Section 4.3. To show the advantage of our co-attention scheme introduced in Section 4.3, we also implement the model with the co-attention scheme used in [13]. The empirical results in both datasets show that our model performs better.

5.1.2 Similarity of the query embeddings. Recall that the motivation of introducing the query matching task with a co-attention scheme is to help the encoder to generate an embedding nearby for a similar query. To show this, we use query embeddings to compare the similarities among queries. Specifically, we choose four queries: "apple", "apple vinegar", "apple airpods", "airpods case". For these queries, "apple" is the most ambiguous query, which may be interpreted in two different ways: a brand in electronics or a kind of fruit. Therefore, the query "apple" is related to all the other three queries but with different meanings. "apple vinegar" is a query which is related to the general query "apple" but unrelated to the electronic products "apple airpods" and "airpods case". In addition, when we compare "apple airpod" with the general query "apple", its representation should focus more on the brand "apple", while when compared with the specific query "airpods case", its representation should focus more on the product "airpods".

Figure 3 and Figure 4 demonstrate the query similarity matrices using the co-attention scheme in [13] and ours. We can observe

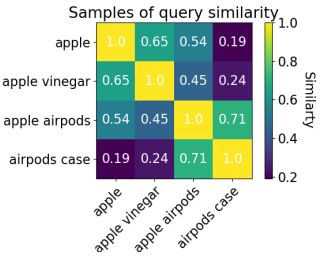

Figure 3: Similarity with attention scheme in [13]

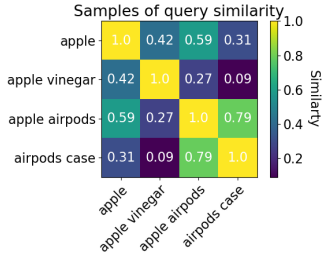

Figure 4: Similarity with our attention scheme

that our co-attention scheme measures the similarity more properly. Specifically, the query "airpods case" has higher similarity scores

Input query	Output query ([13] co-attention)	Output query (our co-attention scheme)
hiking boots men 9	hiking boots men 9	hiking boots men
diamantina comestible	edible gold	edible glitter
alouse	allolose	allulose syrup
fila 5jwoo250 - 262	fila boots	fila disruptor 2 women
everything but the gavel	everything but the elote boot	everything but the bagel seasoning

Table 2: Sample rewriting outputs. In the table, "hiking boots men 9" is too specific. "diamantina comestible" is non-English. "alouse" is an ambiguous query with typos. "fila 5jwoo250 - 262" and "everything but the gavel" are queries with typos.

with queries "apple" and "apple airpods" in the model with our co-attention scheme compared to the one with Maji et al. [13]'s scheme, growing from 0.19 to 0.31 and from 0.71 to 0.79. Moreover, for the irrelevant query pairs (apple vinegar, apple airpods) and (apple vinegar, airpods case), our model reduces the similarity scores from 0.45 to 0.27 and from 0.24 to 0.09. This makes "apple vinegar", as opposed to "apple" in the model with the co-attention scheme in [13], the least relevant query to "airpods case".

5.1.3 Case Study. Table 2 shows five rewriting examples of our model. We can see that our model is able to reasonably rewrite the non-popular queries of different types, including typos, ambiguities, wordy queries, and non-English queries. We can also see that the model with our co-attention scheme performs better than the one with the co-attention used in [13].

5.2 Application: Sponsored Search

Sponsored search aims at surfacing interesting and relevant sponsored contents like product or brand advertisements to customers based on their search queries. The number of ads in auction (auction density) and their quality are the main factors that affect customer experiences. Naturally, abundant ads are generally sourced for head queries while tail queries could benefit more from query rewriting to improve auction density.

To verify this conjecture, we collect a one-week search log, which contains all search queries and their auction densities. We uniformly sample 500000 queries, separate them into 4 equally-sized groups based on auction density, and name them the bottom, tail, torso, and head groups. For each group, we rewrite a source query and add the ads in the rewritten query's auction into the source query's auction. The auction density lift for each group is shown in Table 3. We can see that our model significantly boosts the auction density of queries originally with low auction density.

	Bottom	Tail	Torso	Head
Auction density lift	359.07%	69.23%	25.72%	5.16%

Table 3: Results on auction density boost for four groups of queries with different auction density bands.

To further evaluate the rewriting quality, we try to evaluate the relevance between the source query and the added ads from the rewritten query. We run an A/B test on production shopper traffic for a 7-day period of time. As indicated in Table 3, the bottom group includes queries with the lowest auction density, which means the most opportunity. Hence, we focus on the experiments only for that group. Intuitively, the original queries in the bottom group are difficult to understand, and hence, difficult to be matched with relevant ads. The rewritten query helps to bring in more relevant ads into auction so that there is a better chance that a good ad can win the auction. This has huge potential to improve customer experience on tail queries.

Directly evaluating the quality of query rewriting is difficult as we lack ground truth. Here, we rely on downstream business metrics to measure the improvement due to our query rewriting method. We look at the main business metrics including ad clicks, auction density, CTR (click through rate), and CPC (cost per click). The number of ad clicks shows overall customer engagement with the sponsored contents; CTR is considered as a proxy of customer satisfaction where a higher CTR indicates better customer experience; and CPC reflects the level of competition in auction. The

Clicks	Auction Density	CTR	CPC
+6.72%	+15.04%	+5.55%	+5.07%

Table 4: Production experiment results

improvement on all those metrics is included in Table 4. In particular, we observe auction density jumps most and it also drives up CPC because of more competitions in auction. Clicks and CTR both jump up because new ads added into auction via our query rewriting method are able to win auctions and outperform the previous auction winners, which eventually deliver better customer experiences. Our total traffic was $\approx$ 0.8 million ads impressions. Theoretically, a boost of $1/\sqrt{0.8M} \approx 0.11\%$ is a significant improvement under mild assumptions.

6 CONCLUSION AND FUTURE WORKS

In this work, we have introduced a novel seq2seq framework for the task of query rewriting in e-commerce applications. Our approach derives data from commonly available data sources in e-commerce platforms making it generally applicable. Our model has a number of enhancements to help capture the shopping intent and favorably shape the query representation space. Finally, the resulting overall model uses commonly available deep learning components and has an easy end-to-end training protocol making it suitable for production uses.

Specifically, we first design a dataset construction method which constructs reliable query rewriting pairs based on search logs. Second, we combine two sources of shopping intent, namely query taxonomy and item name sequence, in our model training. These shopping intents are directly available based on our dataset construction method and more importantly, they are only used as auxiliary tasks during *training* process but not inference process, which is more practical as we only need the source query as the input of the model. Third, we introduce a novel co-attention scheme and combine a query matching auxiliary loss between the representations of the source query and the target query. Finally, we combine all the above auxiliary tasks in a fully end-to-end model. Our ablation study shows the effectiveness of our proposed model.

There are many future directions of this work. First, for query rewriting, many previous works construct their dataset with their own method and in this work, one of our main contribution is that we propose a new method to construct a query rewriting dataset from search logs. A more systematic method of dataset construction may create a better dataset, which can improve the model performance.

Second, as mentioned in Section 2.1, a query may have different interpretations based on different contexts. Therefore, to better rewrite a query, it is reasonable to take the context information as input as well to see whether a query can be rewritten into queries with different intents based on different contents. In fact, we have already trained such model with a modification of our current model and it actually can do multi-direction decoding. However, as this requires context information as the input, it is not practical in real applications. An interesting direction is to design a model which can automatically rewrite the input queries into different intents.

Third, in this work, we use the metric of BLEU value, Jaccard similarity and F-score , which are standard in general machine translation tasks, to measure the performance. However, they may not be most optimal metrics for query rewriting. As pointed out in [18], a more reasonable metric in the field of query rewriting needs to be designed for offline dataset evaluation.

APPENDIX

DATASET CONFIGURATIONS AND EXAMPLES

In this section, we introduce the detailed configurations of our dataset construction. We set the size of product set related to each query to be $t = 20$. The popularity distance thresholds are chosen to be $\sigma_{\cos} = 0.3$ and $\sigma_{\ell_1} = 0.375$ based on manual checking. The overall popularity threshold is chosen to be $\tau = 625$ for both datasets. The sizes of the two datasets shown in Table 5. Table 6 gives some examples of the query pairs with their popularity and their distance of popularity weight vectors. We can see from the table that our popularity threshold and distance thresholds indeed construct meaningful query rewriting pairs.

MODEL PARAMETERS AND IMPLEMENTATION

To balance the model performance and the training speed, we compare different sets of parameters. We set the batch size to be 96 and use Adam optimizer with learning rate $\eta = 0.0001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For the choice of $\{\mu_i\}_{i=1}^4$, we fix $\mu_1 = 1$ and have a grid search for μ_2, μ_3 and μ_4. For our final model, we select $\mu_2 = 0.8$, $\mu_3 = 1.3$ and $\mu_4 = 0.7$ based on the performance on the validation set. When doing ablation study, we re-tune $\{\mu_i\}_{i=1}^4$ to fit different models. The distance metric for $\mathcal{L}_{\text{sim}}(\boldsymbol{\theta})$ is defined as ℓ_1 distance. We set the number of layers for encoding, query decoding and product name decoding to be 4; the dimension of embedding to be 512; the number of units in the fully-connected layer to be 2048; the number of attention head to be 8 and the dropout rate to be 0.1. The model is implemented using PyTorch [17] and is trained on a single NVIDIA Tesla V100 GPU.

Dataset	Train	Valid	Test	# of classes
cos distance	584816	73129	73001	1114
ℓ_1 distance	583971	73028	72915	1110

Table 5: Description of offline datasets constructed from users' search log

Input query	Output query	Product Name	Category
simply marvelous	simply marvelous rub	simply marvelous bbq cherry rub 3oz	72
hdmi cab	hdmi cable	CL3 rated high-speed hdmi cable	563

Table 6: Examples in the ℓ_1 distance dataset.

EVALUATION METRICS

For completeness, we explain the three standard metrics used in our experiments.

- BLEU score [16]: In our experiment, we use BLEU_4 score, which considers the 1, 2, 3, 4-grams overlaps.
- F_n score: The F-score is defined as the harmonic mean of the n-grams prediction precision and n-grams recall rate between the output sequence and the label sequence. The higher the F_n score is, the better the rewriting sequence matches the label sequence.
- Jaccard_n score: Jaccard_n calculates the ratio between the number of n-grams appear in both two sequences and the number of n-grams appear in either sequence. The higher the Jaccard_n score is, the more similar the sequences are.

RESULTS OF ℓ_1-DISTANCE-BASED DATASET

Table 7 shows the ablation study of our models on the ℓ_1-distance-based dataset. The trend is the same as the one shown in Table 1.

	BLEU_4	Jaccard_1	Jaccard_2	F_1 score	F_2 score
baseline	0.2528	0.3135	0.1949	0.4334	0.2773
+ query cls	0.2546	0.3153	0.1994	0.4357	0.2831
+ product name decode (sequentially)	0.2634	0.3225	0.205	0.4452	0.2912
+ attn (fixed attn) (separately)	0.2648	0.3236	0.2074	0.4449	0.2931
+ attn ([13]) (separately)	0.274	0.3332	0.2192	0.4522	0.3037
+ attn (Section 4.3) (separately)	**0.2843**	**0.3467**	**0.2271**	**0.4651**	**0.3105**

Table 7: Ablation study on the test dataset with ℓ_1 distance. Each row means the same model setup as shown in Table 1.

REFERENCES

[1] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*. 2425–2433.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Zhihong Chen, Rong Xiao, Chenliang Li, Gangfeng Ye, Haochuan Sun, and Hongbo Deng. 2020. Esam: Discriminative domain adaptation with non-displayed items to improve long-tail performance. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 579–588.

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

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

[6] 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*. 1443–1452.

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

[8] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (2018).

[9] Shuai Huo, Min Zhang, Yiqun Liu, and Shaoping Ma. 2014. Improving Tail Query Performance by Fusion Model *(CIKM '14)*. Association for Computing Machinery.

[10] Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345* (2019).

[11] Hanqing Lu, Yunwen Xu, Qingyu Yin, Tianyu Cao, Boris Aleksandrovsky, Yiwei Song, Xianlong Fan, and Bing Yin. 2021. Unsupervised Synonym Extraction for Document Enhancement in E-commerce Search. (2021).

[12] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. *Advances in neural information processing systems* 29 (2016), 289–297.

[13] Subhadeep Maji, Rohan Kumar, Manish Bansal, Kalyani Roy, Mohit Kumar, and Pawan Goyal. 2019. Addressing Vocabulary Gap in E-Commerce Search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 1073–1076. https://doi.org/10.1145/3331184.3331323

[14] Aritra Mandal, Ishita K Khan, and Prathyusha Senthil Kumar. 2019. Query Rewriting using Automatic Synonym Extraction for E-commerce Search.. In *eCOM@ SIGIR*.

[15] Mani Maybury. 1999. *Advances in automatic text summarization*. MIT press.

[16] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

[18] Yiming Qiu, Kang Zhang, Han Zhang, Songlin Wang, Sulong Xu, Yun Xiao, Bo Long, and Wen-Yun Yang. 2021. Query Rewriting via Cycle-Consistent Translation for E-Commerce Search. *arXiv preprint arXiv:2103.00800* (2021).

[19] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018).

[20] Yangqiu Song, Haixun Wang, Weizhu Chen, and Shusen Wang. 2014. Transfer Understanding from Head Queries to Tail Queries *(CIKM '14)*. Association for Computing Machinery.

[21] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215* (2014).

[22] Zehong Tan, Canran Xu, Mengjie Jiang, Hua Yang, and Xiaoyuan Wu. 2017. Query rewrite for null and low search results in eCommerce. In *eCOM@ SIGIR*.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[24] Yaxuan Wang, Hanqing Lu, Yunwen Xu, Rahul Goutam, Yiwei Song, and Bing Yin. 2021. QUEEN: Neural Query Rewriting in E-commerce. (2021).

[25] Rong Xiao, Jianhui Ji, Baoliang Cui, Haihong Tang, Wenwu Ou, Yanghua Xiao, Jiwei Tan, and Xuan Ju. 2019. Weakly supervised co-training of query rewriting andsemantic matching for e-commerce. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 402–410.

[26] Yatao Yang, Jun Tan, Hongbo Deng, Zibin Zheng, Yutong Lu, and Xiangke Liao. 2019. An Active and Deep Semantic Matching Framework for Query Rewrite in E-Commercial Search Engine. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 309–318.

[27] Shaowei Yao, Jiwei Tan, Xi Chen, Keping Yang, Rong Xiao, Hongbo Deng, and Xiaojun Wan. 2021. Learning a Product Relevance Model from Click-Through Data in E-Commerce. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 2890–2899. https://doi.org/10.1145/3442381.3450129

[28] Hongchun Zhang, Tianyi Wang, Xiaonan Meng, Yi Hu, and Hao Wang. 2019. Improving Semantic Matching via Multi-Task Learning in E-Commerce.. In *eCOM@ SIGIR*.