Query Rewriting in TaoBao Search

Sen Li*
Alibaba Group
Hangzhou, China
lisen.lisen@alibaba-inc.com

Fuyu Lv
Alibaba Group
Hangzhou, China
fuyu.lfy@alibaba-inc.com

Taiwei Jin
Guiyang Li
Alibaba Group
Hangzhou, China
taiwei.jtw@alibaba-inc.com

Yukun Zheng
Alibaba Group
Beijing, China
zyk265182@alibaba-inc.com

Tao Zhuang
Alibaba Group
Hangzhou, China
zhuangtao.zt@alibaba-inc.com

Qingwen Liu
Alibaba Group
Hangzhou, China
xiangsheng.lqw@alibaba-inc.com

Xiaoyi Zeng
Alibaba Group
Hangzhou, China
yuanhan@taobao.com

James Kwok
Hong Kong University of Science and Technology
Hong Kong, China
jamesk@cse.ust.hk

Qianli Ma
South China University of Technology
Guangzhou, China
qianlima@scut.edu.cn

ABSTRACT

In e-commerce search engines, query rewriting (*QR*) is a crucial technique that improves shopping experience by reducing the vocabulary gap between user queries and product catalog. Recent works have mainly adopted the generative paradigm. However, they hardly ensure high-quality generated rewrites and do not consider personalization, which leads to degraded search relevance. In this work, we present **C**ontrastive **L**earning **E**nhanced **Q**uery **R**ewriting (*CLE-QR*), the solution used in Taobao product search. It uses a novel contrastive learning enhanced architecture based on "query retrieval–semantic relevance ranking–online ranking". It finds the rewrites from hundreds of millions of historical queries while considering relevance and personalization. Specifically, we first alleviate the representation degeneration problem during the query retrieval stage by using an unsupervised contrastive loss, and then further propose an interaction-aware matching method to find the beneficial and incremental candidates, thus improving the quality and relevance of candidate queries. We then present a relevance-oriented contrastive pre-training paradigm on the noisy user feedback data to improve semantic ranking performance. Finally, we rank these candidates online with the user profile to model personalization for the retrieval of more relevant products. We evaluate *CLE-QR* on Taobao Product Search, one of the largest e-commerce platforms in China. Significant metrics gains are observed in online A/B tests. *CLE-QR* has been deployed to our large-scale commercial retrieval system and serviced hundreds of millions of users since December 2021. We also introduce its online deployment scheme, and share practical lessons and optimization tricks of our lexical match system.

CCS CONCEPTS

• **Information systems** → **Query reformulation**; • **Applied computing** → *Online shopping*.

KEYWORDS

Query Rewriting; Lexical Match; E-commerce Search

ACM Reference Format:
Sen Li, Fuyu Lv, Taiwei Jin, Guiyang Li, Yukun Zheng, Tao Zhuang, Qingwen Liu, Xiaoyi Zeng, James Kwok, and Qianli Ma. 2022. Query Rewriting in TaoBao Search. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557068

1 INTRODUCTION

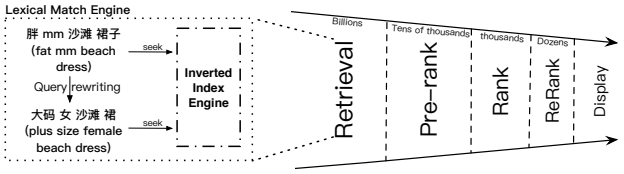


Figure 1: Overview of the query rewriting process and e-commerce search system in Taobao.

Online shopping has been increasingly popular in recent decades. Top shopping platforms (such as Amazon, eBay, Taobao, JD, and PDD) are visited by hundreds of millions of users every day, contributing to hundreds of billions of US dollars in terms of Gross

*Corresponding author.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557068

Merchandise Volume annually. Modern search engines allow users to discover relevant products from the massive merchandise. Figure 1 shows the multi-stage "retrieval–prerank–rank–rerank" architecture in the e-commerce search system of Taobao. Among the billions of candidate products, it can efficiently return a small group (dozens) of highly relevant products for display to the user.

A primary challenge in e-commerce search retrieval is the vocabulary gap between queries and products. For example, a user may be looking for "大码女沙滩裙 (plus size female beach dress)", and issues the query "胖mm沙滩裙子 (fat mm beach dress)". While the inverted index remains the bedrock of commercial search systems for its *stability* and *high search relevance*, the exact term matching engine based on inverted index does not understand the term "胖mm" and returns nothing. How to effectively bridge this gap while preserving stability and search relevance remains an open question.

Recent approaches [17, 22, 29, 47] usually adopt an embedding-based learning paradigm, in which user-product preference predictions are obtained by first embedding the users, queries, and products into a vector space. However, this embedding can dramatically hurt search relevance and lead to most of the online unfixable bad cases [22, 31]. Instead, query rewriting (*QR*) [40, 42] is more *controllable* and *interpretable*. As shown in Figure 1, QR converts an ambiguous query into a more well-expressed query. By performing term-based matching with both the original and rewritten queries, the inverted index engine can return more relevant products.

A number of query rewriting techniques have been used. In bidding/sponsored search, by restricting the target space to a fixed set of high-quality bid words, neural machine translation [3] is used to directly generate the rewritten query from the original query [21, 23, 28, 35]. In product search, by using the corpus of user click logs, JD [31] improved their thesaurus-based baseline which can suffer from query intent drift [2, 18]. Empirically, it has been observed that performing generative rewriting is harder than mining historical rewrites from user search logs [7] and degrades search relevance. Thus, in this paper, for a given query, we aim at finding the most relevant queries from hundreds of millions of historical search queries.

For scalability and flexibility, the proposed query rewriting system adopts a two-stage architecture ("query retrieval-semantic relevance ranking") [16]. First, relevant candidate queries are retrieved by two methods: (i) approximate nearest neighbor (ANN) search on *context-aware* (*context2vec*) and *content-aware* (*content2vec*) embeddings [15], and (ii) *collaborative filtering*. Then, for semantic relevance ranking, we finetune the *StructBERT* [39], which is pretrained with large-scale e-commerce domain data, with small-scale but precise human annotation data. It ranks the candidate rewrites generated by various methods based on some unified scores. However, *content2vec*, similar to *word2vec*, suffers from the query representation degeneration problem [13] and may reduce the candidate queries' quality and relevance. Besides, the discrepancy between *StructBERT*'s pre-training task and our finetuned relevance-oriented ranking task limits the relevance ranking performance.

Another problem is that relevance ranking without user-specific features is often insufficient in the e-commerce scenario. For example, consider the query "运动鞋 (sneakers)". Due to limited latency, the search engine cannot return all products containing "sneakers". Instead, the millions of sneakers-related products are first sorted (e.g., by using the TF-IDF) in the inverted index, and only the top-K results are returned to the user. However, men may only be interested in men's sneakers, while women may only be interested in women's sneakers. It is desirable to retrieve more personalized products in the top-K set by, for example, using the rewrite "男士运动鞋 (male sneakers)" for a male user.

This paper proposes a Contrastive Learning Enhanced Query Rewriting (*CLE-QR*) framework, investigates its effect on each stage of the search system, and introduces practical lessons on the lexical matching engine. To address the problems mentioned above, instead of using *content2vec* in query retrieval, we initialize a Sentence-BERT (*SBERT*) [32] with *StructBERT* and train it with an unsupervised contrastive loss to make the query representations more evenly distributed. This prevents representation collapse and thus alleviates the degeneration problem. Furthermore, to retrieve query candidates that bring additional benefits to search, we collect $(query, rewq)$ pairs (where *rewq* stands for the rewritten query), where clicked products are only retrieved by the rewritten query *rewq* from user search logs. We then train an *interaction-aware SBERT* with contrastive loss by taking these pairs as positive examples and the other in-batch examples as negative examples. For semantic ranking, we perform relevance-oriented pre-training on the noisy user feedback data with a contrastive objective to bridge the gap between pre-training and fine-tuning tasks. Finally, we use an online ranking module to possess query rewriting of personalization for retrieving more relevant products. To the best of our knowledge, *CLE-QR* is the first query rewrite solution for large-scale e-commerce that models both relevance and personalization. The effectiveness of *CLE-QR* is verified by offline qualitative and quantitative analysis and online A/B tests.

The main contributions of this work are summarized as follows:

- We propose a Contrastive Learning Enhanced Query Rewriting (*CLE-QR*) framework, consisting of "retrieval–semantic relevance ranking–online ranking", to find rewrites from hundreds of millions of historical search queries while modeling the relevance and personalization characteristics.
- We apply contrastive learning to solve the query representations degeneration, capture the interactive-aware rewrites, and bridge the gap between the pre-training and relevance ranking finetune tasks, boosting search performance.
- The offline qualitative and quantitative analysis, as well as online A/B tests of Taobao Product Search, demonstrate the effectiveness of *CLE-QR*.

2 RELATED WORK

Query rewriting (QR) is a fundamental topic in information retrieval [34]. Various techniques have been presented, which fall into three categories: (i) relevance/pseudo-relevance feedback, (ii) ontology/thesaurus-based, and (iii) leveraging user search logs.

Relevance feedback-based techniques [33] involve submitting the original query to retrieve initial results, then asking the user to select the relevant ones, and finally expanding the query with additional terms extracted from the relevant results and conducting the search with the modified query. Pseudo-relevance [4, 36, 43], a variant of relevance feedback, assumes that the top few results

returned from the original query are relevant [34]. These methods rely on the richness and complementarity of the content of initial results, and are not particularly suitable for product search whose title is verbose [26] and is much shorter than web documents [2]. Ontology/thesaurus-based techniques employ either externally available knowledge or create knowledge from the corpus to expand queries [26]. Wu et al. [41] suggest that using a well-built domain ontology can yield better retrieval performance. eBay [26] further develops a pipeline to automatically build domain-thesaurus, including synonym generation and candidate ranking. However, this may lead to query drift [2, 18] as they add terms to the query without considering the queries as a whole.

Techniques based on user search logs leverage the massive user logs to identify/generate queries rewrites. Downey et al. [11] and Manchanda et al. [25] train a contextual term-weighting model to retain the critical terms in user queries. However, it is hard to use with the relatively short top/torso search queries. Cui et al. [8], Fonseca et al. [12] and Antonellis et al. [2] expand query with terms and similar user queries from click bipartite graphs, respectively. Grbovic et al. [15] propose a query embedding algorithm and incorporate it with k-nearest neighbor search to find rewrites. Recently, some works [21, 23, 31, 35, 40] regard the click logs as a parallel corpus and apply neural machine translation. In bidding/sponsored search, by restricting the decoding space to a fixed set of submitted bid words, Microsoft [21], Baidu [23], and ByteDance [35] translate queries into commercial keywords. JD [31] and Amazon [40] introduce end-to-end solutions for generating rewrites in product search, and are more closely related to the proposed method. However, they cannot ensure good search relevance since the user click logs are noisy [7, 22, 42].

Another promising approach is proposed by Yahoo [16, 18], which consists of candidate generating and candidate ranking. The candidate generating phase can use most of the existing query rewriters. They then rank the candidates with click-through rate, which may retrieves irrelevant items. Yang et al. [45], Xiao et al. [42] further rank the candidates with human annotation data to improve search relevance. However, they do not consider the rewrite's search performance and personalization, which are essential in e-commerce [22, 47]. This paper will present a solution of query rewriting that takes into account both relevance and personalization for the first time.

3 PROPOSED METHOD

Figure 2 shows the overall structure of the proposed *CLE-QR*. We first define the problem and then introduce the query retrieval, semantic relevance ranking, and online ranking stages of *CLE-QR*.

3.1 Problem Definition

Let $\mathcal{U} = \{u_1, \ldots, u_N\}$ be a set of N users, $\mathcal{Q} = \{q_1, \ldots, q_L\}$ be the historical user search queries, and $\mathcal{I} = \{i_1, \ldots, i_M\}$ be a collection of items (products). Note that in the e-commerce scenario, L is often in the scale of hundreds of millions.

To ensure search relevance and good quality of rewrites, we rewrite the query by finding relevant ones from the existing historical user queries $\mathcal{Q}$. Specifically, at time t, a user $u \in \mathcal{U}$ submits a query q_u. We choose the most relevant query set $\mathcal{Q}' = \{q_1, \ldots, q_K\}$ from $\mathcal{Q}$. Each $q_r \in \mathcal{Q}'$ matches a subset of items $\{i | i \in \mathcal{I}\}$ satisfying the user's intent. This can be described as:

$$q_r = \mathcal{T}(q_u, u), \tag{1}$$

where $\mathcal{T}(\cdot)$ is a transform function and is instantiated with the "retrieval–semantic relevance ranking–online ranking" pipeline.

3.2 Query Retrieval

3.2.1 Collaborative Filtering. Based on the user behavior, we construct a query-item bipartite graph $\mathcal{G}$, in which a query is connected to an item that is interacted with the user (i.e., click/buy). $\mathcal{G}$ can be used to naturally describe query similarity, which is more complex than exact matching. Here, we use *Swing* [44] to $\mathcal{G}$ to exploit *collaborative signal* for the mining of similar queries.[1] In recommendation, *Swing* defines the similarity of product pairs by calculating the number of "user-item-user" networks (more details are referred to [44]). Similarly, we calculate "item-query-item" networks. Specifically, for queries q_a and q_b, the swing score $s(a, b)$ is defined as:

$$s(a, b) = \sum_{i \in I_a \cap I_b} \sum_{j \in I_a \cap I_b} \frac{1}{\alpha + \left| Q_i \cap Q_j \right|}, \tag{2}$$

where I_a (resp. I_b) is the set of clicked items using q_a (resp. q_b), and Q_i (resp. Q_j) is the set of queries that are connected with item i (resp. j), and α is a smoothing coefficient (which is set to 1).

3.2.2 Embedding-based Retrieval. Collaborative filtering performs well when the data is abundant. However, it can perform poorly on sparse data as it relies on user behaviors. Therefore, we further adapt the embedding-based approach to enrich the set of query candidates. These embedding techniques include *context-aware* and *semantic matching* (i.e., *content-* and *interaction-aware* matching). After obtaining the query embedding, we employ the *ANN* algorithm[2] to retrieve candidate queries from $\mathcal{Q}$ for the given query.

Context-aware Matching. By examining the user log, it is found that users often reformulate queries in a session[3] until they find an item that fits their interests. For example, a user submits an initial query "刀具置物架 (knife shelves)", but cannot find any desired item. He/She then reformulates the query to "刀具收纳 (knife holder)" and gets what he/she wants. By observing how users modify their queries to narrow the vocabulary gap, this provides query similarity information beyond literal matching. Given a set S of M search sessions, in which each session $s = (q_1, \ldots, q_n) \in S$ is a sequence of n queries. Using the negative sampler of the skip-gram model [27], we learn a context-aware query representation *context2vec* (which is similar to [15] but differs in motivation) by maximizing the following objective over S:

$$\mathcal{L} = \sum_{s \in \mathcal{S}} \sum_{q_i \in s} \sum_{-c \le j \le c, j \ne 0} \log(\sigma(q_i q_{i+j}) \prod_{k \in D_{neg}} \sigma(-q_i q_k)), \tag{3}$$

where c is the training context window size, D_{neg} is the set of negative samples, and $\sigma(\cdot)$ is the sigmoid function. In the experiments,

[1] Empirically, we find *Swing* [44] to be more robust to noisy clicks and can achieve better candidate queries than *Simrank++* [2].
[2] https://github.com/alibaba/proximabilin
[3] In the experiments, for an in-session query sequence, (i) queries must be issued by the same user within 15 minutes, and (ii) the query categories between two consecutive queries must intersect.

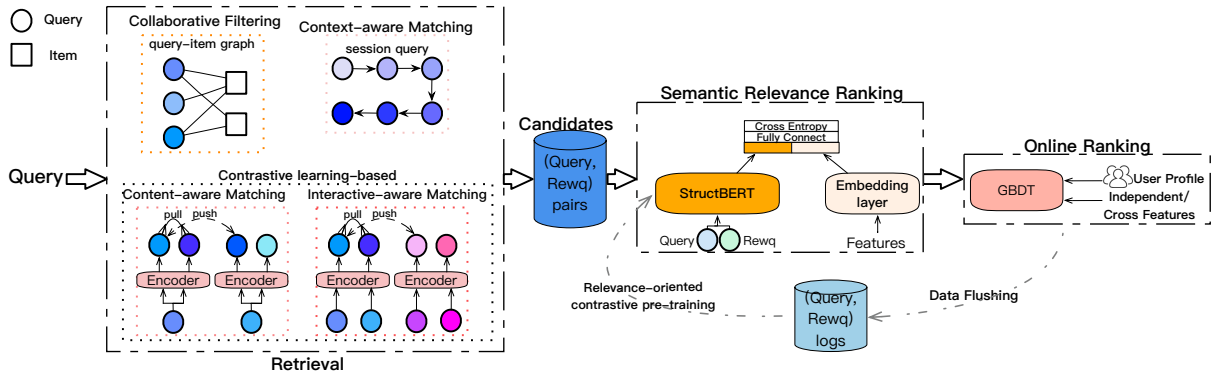


Figure 2: General structure of the proposed Contrastive Learning Enhanced Query Rewriting (*CLE-QR*).

we set $c = 5$ and use 128 negative samples. Note that q_i here refers to its representations.

Content-aware Matching. Besides context information, it is crucial to model the content for torso, low-frequency, and rare queries. To obtain the query's *content-aware* representations, we previously used *content2vec* [15]. However, as in *word2vec* [13], it suffers from the representation degeneration problem, which reduces the diversity and quality of the representations. To alleviate this problem, we follow [14] and finetune a Sentence-Bert (*SBERT*) [32] with the *InfoNCE* loss [6]. The query representations then become evenly distributed on a sphere [14], and the singular value distribution of the representation matrix is more dispersed and not concentrated on the first eigenvalue [13], thus improving the quality and discrimination of the representations.

Specifically, let h_i (resp. h_i^+) be the representation of query q_i (resp. positive sample q_i^+). For a batch with N pairs, the contrastive training objective for (q_i, q_i^+) is:

$$l_i = \log \frac{e^{\cos(h_i, h_i^+)/\tau}}{\sum_{j=1}^{N} e^{\cos(h_i, h_j^+)/\tau}}, \tag{4}$$

where $cos(\cdot)$ is the cosine function, τ is a temperature hyper-parameter (set to 0.05), and $N = 64$. We initialize the *SBERT* using *StructBERT* [39], a state-of-the-art language model pre-trained with the e-commerce corpus. We take the "[CLS]" token as the query representation. Inspired by unsupervised approaches [14], we feed each query $\{q_i \in Q\}$ to the model twice by applying different dropout masks to obtain positive-pair representations (h_i, h_i^+). We treat the other $N - 1$ dropped examples h_j^+ within a batch as negative examples.

Interaction-aware Matching. We use online user feedback logs as augmented data to provide *interaction-aware* matching. Specifically, for a search request, we examine the items retrieved by q_u, the original query issued by user u, and the rewriting queries $\{q_u, q_1, \ldots, q_K\}$. As the online system keeps track of which items are retrieved by which query, we propose to select the pair (q_u, q_r) as augmented training data if (i) q_u, q_r are in the same query list for a particular search request, and (ii) q_r retrieves items that q_u does not and that are interacted (clicked or purchased) by the user. These query pairs are beneficial since they reflect which rewritten query can retrieve clicked items that the original one cannot. Examples of these pairs include "(新生儿上衣 (newborn clothes), 保暖衣 婴儿 (warm clothes for babies))", "(咬胶婴儿 (chews for babies), 牙胶 (teether))", and "(碎发神器 (shredded hair artifact), 盘发神器 (curling hair))". As can be seen, most of the rewritten queries are semantically relevant. Statistically, 87.8% of the rewritten queries have high click-through rates and search frequencies.

By taking the interaction-aware pairs $\{(q_u, q_r)\}$ as positive pairs $\{(q_i, q_i^+)\}$ and the other $N - 1$ in-batch rewrites as negative examples, we finetune *SBERT* with contrastive learning. Specifically, given a data set $D = \{(q_{11}, q_{12}), \ldots, (q_{N1}, q_{N2})\}$ with N interaction-aware pairs, we use the contrastive loss:

$$\ell = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{\cos(h_{i1}, h_{i2})/\tau}}{\sum_{j=1}^{N} \mathbb{I}_{[j \neq i]} e^{\cos(h_{i1}, h_{j2})/\tau}}, \tag{5}$$

where h_{i1} (resp. h_{i2}) is the representation of q_{i1} (resp. q_{i2}), and $\mathbb{I}_{[j \neq i]} \in \{0, 1\}$ is an indicator function evaluating to 1 iff $j \neq i$. We set temperature $\tau = 0.05$ and $N = 64$. The latent space of *StructBERT* is optimized not only to model semantic relevance but also to push user queries closer to those that lead to good user interactions.

For a given query, the top-200 candidates for each retrieval method are used for semantic relevance ranking, ensuring high quality and enriching diversity of the query candidates.

3.3 Semantic Relevance Ranking

Ranking the candidate queries by click-through rate and the number of newly retrieved items inevitably reduces the search relevance since it is sensitive to click noise and will return irrelevant items [45]. To ensure high search relevance, given a pair $(query, rewq)$, we evaluate their semantic similarity manually. We label the pair "good" if the rewritten query *rewq* maintains the search intent of the original query; otherwise, it is labeled "bad". The whole semantic relevance task is then defined as a binary classification problem.

To reduce the amount of labeled data required, our semantic model uses both the pre-trained language model and the following set $\mathcal{F}$ of features: (i) f_{brand}: do $query, rewq$ have the same brands?, (ii) f_{model}: do $query, rewq$ have the same model type (extracted by query tagging)?, (iii) f_{cate_sim}: Jaccard similarity of the predicted

query categories of $(query, rewq)$, and (iv) f_{click}: difference of the clicked item sets between $(query, rewq)$, including the proportion of intersection and inconsistent proportion of the rewritten query compared with the original query.

Given the pair $(query, rewq)$, we feed the text "$[CLS]query[SEP]rewq[SEP]$" to *StructBERT* and take the output of $[CLS]$ as the semantic vector h_s. We then embed the feature set $\mathcal{F}$ into a feature vector h_f, and concatenate h_s, h_f to get the final representation h. Mathematically,

$$e_{f_i} = W_{f_i} \cdot f_i, \tag{6}$$

$$h_f = concat(\{e_{f_i} | f_i \in \mathcal{F}\}), \tag{7}$$

$$h_s = Encoder^{first}([CLS]Query[SEP]Rewq[SEP]), \tag{8}$$

$$h = W_h[h_f, h_s], \tag{9}$$

where W_{f_i} and W_h are the embedding matrices, and f_i is the feature in $\mathcal{F}$ (some are one-hot embedding and others are numeric value). $Encoder^{first}$ is the output of the first token (i.e., $[CLS]$ token) of *StructBERT*. Given a labeled data set $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of N samples, where $x_i = (query_i, rewq_i), y_i \in \{0, 1\}$, we finetune *StructBERT* with the following cross-entropy loss over D:

$$\ell = -\frac{1}{N}\sum_i [y_i \cdot \log(\sigma(h_i)) + (1 - y_i) \cdot \log(1 - \sigma(h_i))], \tag{10}$$

where σ is the sigmoid function and h_i is the representation of x_i.

Relevance-oriented Contrastive Pre-Training Paradigm. The pretrain-finetune paradigm alleviates the demand for labeled data. However, the pre-training objectives of *StructBERT* (masked language, word, and sentence structural) are not relevance-oriented and thus are not consistent with the semantic ranking task used in finetuning. To alleviate this problem, a naive approach is to exploit the post-click information of candidate queries, and use Eq. (10) to continually pre-train StructBERT, named *SL-StructBERT* (*SL* is short for Supervised Learning). Specifically, we examine the logs and see which candidate queries are associated with clicked items. The clicked ones are then labeled "good", and the others as "bad". However, the click data are noisy. For example, we observe that a user queries with "iphone12手机壳玻璃 (iphone12 mobile phone case glass)", but clicks an item retrieved by the rewritten query "iphone11手机壳玻璃钢化 (iphone 11 mobile phone case glass tempered)". This should indeed be a "bad" pair since the rewritten query has changed the user's intent. On the other hand, while these noisy pairs may not be strongly relevant, they do exhibit weakly relevant information. Specifically, the hidden representations of "iphone12手机壳玻璃 (iphone12 mobile phone case glass)" and "iphone11手机壳玻璃钢化 (iphone 11 mobile phone case glass tempered)" can be closer than the other non-mobile phone case glass queries.

To address these issues, we present a relevance-oriented contrastive pre-training paradigm, *CL-StructBERT* (*CL* is short for Contrastive Learning). We continually pre-train *StructBERT* with the contrastive loss in Eq. (5), using the pairs with pseudo-label "good" as positive examples and the other in-batch example pairs as negative examples. Since the query pairs are fed into *StructBERT* separately, this avoids the all-to-all interactions [19] between the original and rewritten queries while still retaining their interactions, thus preventing over-fitting. Moreover, with the contrastive loss, the latent space of *StructBERT* is optimized to be more robust to noisy data so that similar queries (such as "iphone12手机壳玻璃 (iphone12 mobile phone case glass)" and "iphone11手机壳玻璃钢化 (iphone 11 mobile phone case glass tempered)") have smaller distances, while dissimilar queries have larger distances. Additionally, we categorize the user feedback data into three groups based on their relevance-improving performance (see 5.4.4 for analysis). Finally, we finetune the continually pre-training *StructBERT* with Eq. (10) on manually labeled data.

3.4 Online Ranking

Semantic relevance ranking without user-specific features is insufficient in our e-commerce platform, and we further use a real-time inference ranking stage to model personalization for retrieving more relevant products.[4] Currently, the embedding-based product retrieval, pre-ranking, and ranking models (all of them are instanced by deep neural network) deployed in our search system occupy most computing resources. Thus, due to limited latency, we use a light-weight gradient boosting decision tree (*GBDT*) [5] with rich features for online ranking top-K ($K = 30$) rewrites from the semantic relevance ranking stage. The features used fall into three categories: (i) independent features: features of the original and rewritten queries, such as the number of characters and words, number of unique visitors, search frequency and predicted categories, click-through rate, and conversion rate, (ii) cross-features between the original and rewritten queries, e.g., the differences in lengths at character and word levels, Jaccard similarity of word-level and predicted categories, and the previous semantic relevance ranking score, and (iii) user profile, such as age, gender, and region.

We first expand the original query with 20 random rewrites whose semantic ranking score ≥ 0.7, and then use 1% of the online traffic to collect training data by displaying the retrieved products to the user (the rest of the product ranking system is unchanged). Therefore, given an original query, we consider rewrites in which the associated items have been clicked/purchased as positive samples and the others as negative samples. We then train a *GBDT* with the cross-entropy loss (Eq. (10)) with these feedback signals. Evaluated on a test set (which is a random sample with 1 million user logs on day $T + 1$), we obtain an AUC of 0.74. (when the user profile is not used, the AUC is 0.69).

4 SYSTEM ARCHITECTURE

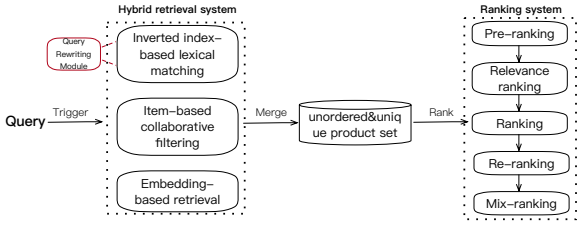


Figure 3: Overview of Taobao search engine.

[4]Incorporating personalization in the query retrieve stage is impractical as it requires additional inference time and cannot response quickly for online traffic.

Figure 3 shows a schematic diagram of the Taobao search engine. When a user issues a query, we first use a hybrid retrieval system to retrieve an unordered and unique candidate set $\mathcal{I}$. We then adopt a sophisticated ranking system (each stage uses a deep neural network) to screen $\mathcal{I}$ and finally display the top products. The proposed query rewriting module (*CLE-QR*) improves the inverted index-based lexical matching engine.

4.1 Online Serving

To meet the latency requirement, we first run the query retrieval and semantic relevance ranking stages of *CLE-QR* offline to generate potential candidate rewrites for unique user query within 15 days, and then store them and their features in an online key-value graph engine (*igraph*) for quick response. When a query arrives, we apply a well-trained *GBDT* to rank its rewrite candidates with rich features and user profile, and then expand the original query with top-K ($K = 5$ in our system) rewrites by operation "OR". Finally, *CLE-QR* covers about 89.2% of the online traffic. The rest of the traffic involve daily new queries and some queries that cannot find rewrites from the massive historical queries that ensure search relevance, which are handled by the existing deployed embedding-based product retrieval system [22] and other lexical match-improving methods (see Section 5.3.1). The query rewriting module increases the latency of the online hybrid retrieval system from 50ms to 60ms.

4.2 Optimization Tricks

The word boundary of Chinese is ambiguous. We thus provide three granular word segmentation results: (i) max semantic token, (ii) min semantic token, and (iii) retrieve token. For example, 液晶显示器 can be segmented into 液晶显示器, or 液晶#显示器, or 液晶#显示#器. Our lexical engine system performs retrieve token-based matching to maximize the recall. However, due to the unordered term-matching process, the user query terms can hit the item title in any order and disregard the word order of the query itself. It will result in unexpected hit situations and thus potentially return irrelevant products. To alleviate this problem, we adopt a multi-granularity matching strategy. The final query is expressed as:

```
(retrieve token) OR (min semantic token) OR (rewrites).
```

Since the min semantic token allows high precision of the matching process, we manually increase its retrieved items' relevance ranking scores of ranking system (see Figure 3) for more exposure.

5 EXPERIMENTS

5.1 Datasets

5.1.1 Industrial Offline DataSet. After filtering the spam users, we collect search logs from online **Mobile Taobao App** to train each stage of *CLE-QR*. Specifically, the "query-item" bipartite graph $\mathcal{G}$ is built from one week of query-item records. Edges that are clicked fewer than three times are removed, resulting in a total of 217 million records. The training data of session queries are collected from two weeks of exposure logs. After removing records with only 1 query in the session, we have a total of 435 million sessions. We use 70 million queries query within two weeks (the number of unique visitors is ≥ 3) to train the content-aware matching module. The number of training pairs for interaction-aware matching method and relevance-oriented contrastive pre-training is 20 million. Also, there are 350 thousand manually labeled relevance data records for finetuning semantic relevance ranking. For online ranking, there are about 80 million training pairs.

For evaluation, we randomly sample 1 million search records of $T + 1$ day, including (i) 0.5 million search click records; (ii) 0.5 million search purchase records. The offline inverted index engine is consistent with the online environment, which contains about 100 million candidate item.

5.1.2 Online Dataset. We deploy a well-trained *CLE-QR* in the Taobao search production environment serving hundreds of millions of user query requests (see Section 4.1).

5.2 Evaluation

5.2.1 Offline Evaluation. We use the metric of Recall@K to evaluate the offline performance since it has positive correlation with the online Gross Merchandise Volume (GMV) metric [22]. Specifically, given a query q_u, the items clicked or purchased by the user u are regarded as the target set $T = \{t_1, ..., t_N\}$, and the top-K items returned by the query rewriting module are regarded as the retrieval set $I = \{i_1, ..., i_K\}$. We set K to $6,000$, which is consistent with the online system. *Recall@K* is defined as:

$$Recall@K = \frac{\sum_{i=1}^{K} i_i \in T}{N}. \tag{11}$$

Besides Recall@K, we also conduct a qualitative analysis for query retrieval stage, including representations visualization, numerical analysis, and case study. For semantic ranking stage, we provide its performance of AUC on manually annotated data.

5.2.2 Online Evaluation. Gross Merchandise Volume (denoted *GMV*) is the most important online metric. It is defined as

$$GMV = \text{\#pay amount}. \tag{12}$$

Besides *GMV*, we also consider user search experience by reporting the proportion of products ($\mathbf{P_{good}}$) with good relevance on the item set $I = \{i_1, \ldots, i_K\}$ displayed to the users:

$$P_{good} = \frac{\sum_{i=1}^{K} \mathbb{I}(i_i)}{K}, \tag{13}$$

where $\mathbb{I}(\cdot)$ is an indicator function evaluating to 1 if item i is rated as good by the user. Moreover, we use the following metrics to measure the impact of query rewriting at each stage of the search system:

$\boldsymbol{Num_{rec}}$: The number of items recalled by the inverted index engine, averaged over N queries. The larger the value, the more products are returned.

$\boldsymbol{Num_{rank}}$: Given a retrieval set $I = \{i_1, \ldots, i_K\}$, Num_{rank} is the number of items entering the ranking stage. This reflects the importance of query rewriting in the search system.

$\boldsymbol{P_{r\_good}}$: The proportion of products with good relevance in the retrieval set $I = \{i_1, \ldots, i_K\}$ entering the ranking stage. It is rated by the online well-trained "query-item" relevance model [46] (with an AUC of **0.92**) with Eq. (13).

The above metrics are evaluated by replacing only the multi-channel retrieval system's query rewriting process. The other parts in the system are unchanged.

5.3 Experimental Setup

5.3.1 Compared Methods. Hundreds of engineers and scientists have tuned our search engine for many years, and it has become a complicated system. Hence, it is worth introducing an online experimental environment. For lexical matching, we have already deployed methods such as query/product tagging matching based on manual efforts and knowledge graphs [24], and BERT-based term weighting techniques to drop query terms (a BERT variant of state-of-the-art contextual term-weighting (*CTW*) [25]) for tail or rare queries, synonymous word/phrase expansion, and learning-to-rewrite-query (*LTRQ*). Because of system stability and commercial benefits, we cannot simply remove specific methods from the existing online system for comparisons. Hence, we only compare *CLE-QR* with the strong baseline *LTRQ*.

LTRQ: The previous generation of our query rewriting technique adopts the two-stage architecture of "query retrieval-semantic relevance ranking" as in [16, 42]. *LTRQ* uses *collaborative filtering*, *context2vec* and *content2vec* [15] with an *ANN* module to retrieve relevant candidate queries. It then uses manual annotation and statistical features finetuned *StructBERT* to rank rewrites (Section 3.3).

5.3.2 Implementation Details. For evaluating semantic relevance ranking, we divide the 350 thousand manually labeled data into training, validation, and test sets at a ratio of 7/1/2. Using the checkpoint with the best performance on the validation set, we report its AUC on the test set as offline semantic ranking performance. The BERT model uses the base setup [10] (12 layers, 12 self-attention heads, and 768 hidden dimensions). We initialize it with StructBERT [39], which is pre-trained with the Chinese e-commerce corpus. The hidden size of the query representation is 128 for all methods. The Adam optimizer [20], with an initial learning rate of $1 \times e^{-5}$, is used during finetuning. The experiments are run on our algorithm platform called Platform of AI (PAI),[5] which provides service of tensorflow [1] and most machine learning algorithms.

5.4 Offline Experimental Results

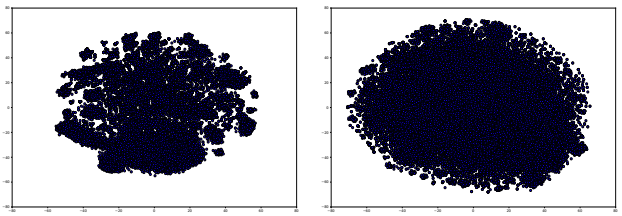
5.4.1 Comparison with the Strong Baseline. As mentioned in Section 5.2.1, we report the metrics of *Recall@K* on 1 million search click/purchase records. Since the purchase signal is important in e-commerce, we further report the metrics of *Recall@K* on the 0.5 million search purchase records, denoted as $Recall_p@K$. We here also study the effectiveness of each component of *CLE-QR*. Specifically, we replace the components in the strong baseline *LTRQ* one at a time. This leads to three ablation models: (i) *LTRQ-RE*, which replaces the **RE**trieval methods of *LTRQ* with *CLE-QR*'s, (ii) *LTRQ-SR*, which replaces the **S**emantic **R**anking model of *LTRQ* with *CLE-QR*'s, and (iii) *LTRQ-OR*, which adds an **O**nline **R**anking module to *LTRQ*.

As shown in Table 1, all the three ablation modules outperform strong baseline*LTRQ* in terms of higher metrics of *Recall*@6000 and $Recall_p$@6000, indicating the effectiveness of the proposed retrieval methods (*content/interactive-aware* matching), relevance-oriented contrastive pre-training, and online personalization ranking. Comparing *CLE-QR* and *LTRQ*, we can conclude that the proposed method can retrieve more products that satisfy users.

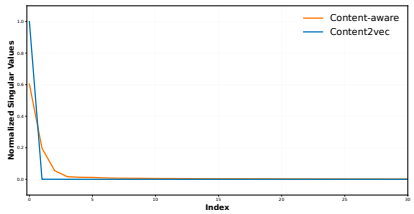
[5]PAI: Platform of Articial Intelligence https://pai.base.shuju.aliyun.com/

Table 1: Offline performance of CLE-QR and its ablation models.

Methods	*Recall*@6000	$Recall_p$@6000
LTRQ	0.9438	0.9790
LTRQ-RE	0.9527(+0.89%)	0.9832(+0.42%)
LTRQ-SR	0.9482(+0.44%)	0.9811(+0.21%)
LTRQ-OR	0.9476(+0.38%)	0.9804(+0.14%)
CLE-QR	0.9560(+1.22%)	0.9860(+0.70%)



(a) Content2vec (b) Content-aware



(c) Singular Values

Figure 4: The *t-SNE* visualization and *SVD* numerical analysis results of the unsupervised representations of *Content2vec* and *Content-aware*.

5.4.2 Qualitative Results of Retrieval. As 90% of the words have low frequencies according to Zipf's law, *Word2vec* suffers from the representation degeneration problem [13]. As mentioned in Section 3.2.2, the *content-aware* method we previously used is *content2vec* [15]. We found that it suffers from the same problem as *word2vec*, which dramatically degrades query representation quality. As an illustration, Figures 4 (a) and (b) show the 2-dimensional *t-SNE* [37] visualizations of the *content2vec* and *content-aware* representations obtained from the same 50, 000 queries. As can be seen, *content2vec* embeds the queries to a small space, and with many collapses. On the other hand, contrastive learning expands the embedding space of *content-aware*, leading to better alignment and uniformity [38].

Gao et al. [13] suggest that a more uniform singular value distribution of the embedding matrix can lead to better quality. Figure 4 (c) shows the normalized singular values of the *content2vec* and *content-aware* representations. As can be seen, the singular values

Table 2: Some queries rewrites retrieved by methods of *CF, Context/Content2vec* and *Cotent/Interative-aware.*

Original Query	*Collaborative Filtering*	*Context2vec*	*Content2vec*	*Content-aware*	*Interaction-aware*
水果味薄荷糖 (fruit flavored mints)	亲劲 (a mint brand)	有个圈的薄荷糖 (mints with a circle)	口香糖 (chewing gum)	水果薄荷糖 (fruit mints)	果味糖 (-)
	果味软糖 (fruity fudge)	mints (mints)	mints海盐薄荷糖 (sea salt mints)	无糖水果薄荷糖 (sugar-free fruit mints)	水果糖 (fruit candy)
	益美滋 (a mint brand)	水蜜桃薄荷糖 (peach mints)	提神薄荷糖 (refreshing mints)	百香果薄荷糖 (passion Fruit Mints)	水果薄荷糖 (fruit mints)
jordan1 童鞋 (jordan1 kids shoes)	6351 ar (nike shoes model)	6066 bq (nike shoes model)	耐克空军一号 (nike air force 1)	jordan 童鞋 (jordan kids shoes)	aj1 童鞋 (aj1 kids shoes)
	nike aj 童鞋 (nike aj kids shoes)	耐克aj儿童 (nike aj kids)	aj14 (aj14)	jordan11 童鞋 (jordan11 kids shoes)	aj儿童鞋 (aj kids shoes)
	6066 bq (nike shoes model)	aj13童鞋 (aj13 kids shoes)	耐克儿童鞋子 (nike kids shoes)	airforce 1 童鞋 (airforce1 kids shoes)	乔丹童鞋 (jordan kids shoes)

of *content2vec* are mainly concentrated in the first few entries, while the singular values of *content-aware* decay much more slowly. Note that the supervised learning methods *context2vec* and *interaction-aware* do not suffer from representation degeneration.

5.4.3 Case Study of Retrieval. Table 2 shows the top-3 rewrites retrieved by the different methods for two randomly sampled queries. *Collaborative Filtering* can match beyond text similarity, and can return candidates that do not match literally but are actually relevant. *Context2vec* can effectively reduce the semantic gap and focus more on semantic matching than *Collaborative Filtering*. *Content-aware* concentrates on semantic matching, which is beneficial for torso and tail queries. *Interaction-aware* learns a mapping from a given query to well-performing queries in terms of its candidates' highest click-through rates and search frequencies. *Content2vec* performs poorly due to representation degeneration. Overall, the retrieval stage improves the relevance and diversity of the candidate queries.

Table 3: Comparison of the training paradigms on manually annotated data. Number in parentheses shows the performance when statistical features are not used.

Paradigm	Methods	AUC
Finetune	GoogleBERT	0.7177 (-0.024)
	StructBERT	0.7624 (-0.015)
Continual Pre-Training + Finetune	SL-StructBERT+S1	0.7688 (-0.010)
	SL-StructBERT+S2	0.7660 (-0.012)
	SL-StructBERT+Mix	0.7672(-0.013)
	CL-StructBERT+S1	0.7680 (-0.014)
	CL-StructBERT+S2	**0.7709** (-0.013)
	CL-StructBERT+Mix	0.7705(-0.015)

5.4.4 Quantitative Results of Semantic Relevance Ranking. We study the improvements on relevance by different types of user feedback in the supervised learning (*SL*) and contrastive learning (*CL*) continual pre-training paradigms mentioned in Section 3.3. Specifically, we first categorize user feedback data into the following three groups: (i) *S1*, which contains $(query, rewq)$ pairs that product is retrieved by both of the $query$ and $rewq$ and has been clicked. These pairs are relevant both literally (terms) and semantically, and contain less noise, (ii) *S2*, which contains $(query, rewq)$ pairs that product is only retrieved by the $rewq$ ones and has been clicked. These pairs are more relevant in semantics than terms, showing a more general relevance pattern but with more noise, and (iii) *Mix*, which contains *S1+S2*. We then separately apply *SL* and *CL* continual pre-training paradigms on the three groups of data above. For example, *CL-StructBERT+S1* means that we continually pre-train *StructBERT* with contrastive learning loss in Eq. (5) on *S1* data. Finally, we finetune the model parameters on the human-labeled data.

As shown in Table 3, *StructBERT* significantly outperforms *GoogleBERT* (Chinese version), indicating its SOTA performance. Also, it is clear that the continual pre-training process improves fine-tuning's performance. Comparing the SL and CL paradigms, although the SL loss is the same as the fine-tuning task of relevance ranking, we observe that CL achieves better performance due to the noisy labels in the user feedback data. Moreover, SL achieves the best performance on *S1* data, indicating that supervised learning is less effective in capturing generalized patterns from noisy data. However, CL achieves the best performance on *S2* data, showing that the *S2* data have more relevance-improving effect than *S1*, and contrastive learning is robust enough to capture it.

5.5 Online Experimental Results

5.5.1 Ablation Study. As mentioned in Section 5.2.2, for online performance evaluation, we report the number of products returned by the inverted index engine (i.e., Num_{rec}) and participated in the ranking stage (i.e., Num_{rank}). In addition to *CLE-QR*'s three ablation models (*LTRQ-RE*, *LTRQ-SR* and *LTRQ-OR*), we also include *LTRQ*'s performance for comparison.

As shown in Table 4, all the three ablation modules can retrieve more item candidates that enter the ranking stage (higher Num_{rank}), indicating the effectiveness of the proposed retrieval methods (*content/interactive-aware* matching), relevance-oriented contrastive pre-training, and online ranking, again. Specifically, *LTRQ-RE* increases Num_{rec} by 45.15% and reduces P_{r_good} by 0.5%, showing that it improves the volume of recalls as well as introducing some invalid results. This is because the online traffic covered

Table 4: Online performance of CLE-QR and its ablation models.

Methods	Num_{rec}	Num_{rank}	P_{r_good}
LTRQ	25916	4007	78.52%
LTRQ-RE	37617(+45.15%)	4088(+2.05%)	77.68%(-0.84%)
LTRQ-SR	23760(-8.32%)	4027(+0.51%)	79.46%(+0.94%)
LTRQ-OR	22337(-13.81%)	4023(+0.42%)	78.77%(+0.25%)
CLE-QR	35970(+38.80%)	4153(+3.66%)	79.01%(+0.49%)

Table 5: Online A/B test of CLE-QR on Mobile Taobao Search. The improvements are averaged over 18 days. ‡ and † indicate that the improvement is statistically significant with $p < 0.01$ and $p < 0.05$, respectively.

	GMV	#Trans	CTR	CVR	P_{good}
CLE_QR	+0.80%‡	+0.41%‡	+0.01pt‡	+0.003pt†	+0.30%

by *LTRQ-RE* is 20% higher than that of *LTRQ*. With the same query, *LTRQ-RE* increases P_{r_good} by 1.56%. Furthermore, both the *SR* and *OR* module can reduce invalid recalls by improving relevance (lower Num_{rec} and higher P_{r_good}), indicating that they can retrieve more relevant products. Comparing *CLE-QR* and *LTRQ*, we can conclude that the proposed method can retrieve more supplementary and relevant products for our following ranking stage.

5.5.2 Online A/B Tests. Finally, we report the 18-day average of GMV improvements (removing cheating traffic by the business analyst) achieved by *CLE-QR*. We also include the following metrics: (i) number of transactions (#Trans), (ii) click-through rate (CTR), and conversion rate (CVR). Table 5 shows that *CLE-QR* improves GMV and #Trans by 0.80% and 0.41%, respectively, indicating that the daily increase in transaction amount is tens of millions. CTR and CVR also increase by 0.01pt and 0.003pt, respectively, which demonstrates that *CLE-QR* can better satisfy users. Note that while we have already deployed the embedding-based product retrieval method, *CLE-QR* still achieves some success, showing that query rewriting provides a valuable candidate item set that cannot be returned by the vector-recall method. We argue that vector-recall suffers from the Matthew effect [30] (popular items are displayed more and become even more popular) since it is driven by the clicked/purchased items, while *QR* retrieves products based on term matching, which can improve diversity of the candidate item set. In addition, to statistically analyze the performance, we perform a pairwise comparison for online baseline *LTRQ* against *CLE-QR*. Specifically, we perform the Wilcoxon signed-rank test [9] to measure the significance of the difference. As shown by the marks in Table 5, improvements on GMV, #Trans, and CTR are statistically significant at the level of $p < 0.01$, while improvement on CVR is statistically significant at the level of $p < 0.05$.

Furthermore, we sort all queries in descending order of their search frequencies in 30 days and categorize them into three types: top, torso, and tail. As Table 6 shows, *CLE-QR* works for all query types, especially for the top and torso queries.

Table 6: Online performance of CLE-QR on different type of queries.

Query_Type	No.	#Trans	GMV
Top	1	-0.13%	0.59%
	2	0.78%	-0.04%
Torso	3	0.94%	2.03%
	4	1.09%	3.07%
Tail	5	1.33%	0.28%

5.5.3 Case Study of Personalization. To demonstrate the effect of personalization in *CLE-QR* on the inverted index engine, we simulate a female user who submits the query "运动鞋 (sneakers)". *CLE-QR* rewrites it as the search query ("运动鞋 (sneakers)" OR "可爱运动鞋 (cute sneakers)" OR "女运动鞋 (female sneakers)" OR "女高帮运动鞋 (female high-top sneakers)" OR "长跑运动鞋 (long-distance running sneakers)" OR "健身运动鞋 (fitness sneakers)") and then conducts search. Alternatively, when we remove the **O**nline **R**anking module of *CLE-QR* (named *CLE-QR w/o OR*), the rewritten query is ("运动鞋 (sneakers)" OR "男运动鞋 (male sneakers)" OR "女运动鞋 (female sneakers)" OR "女款运动鞋 (women's sneakers)" OR "男鞋运动鞋 (men's sneakers)" OR "软底运动鞋 (soft sole sneakers)"). For the female user, *CLE-QR w/o OR* inevitably performs invalid search and wastes computing resources, while *CLE-QR* rewrites some candidates that are more in line with the user characteristics. Furthermore, the number of items retrieved by *CLE-QR* that participate in the ranking stage is 2870, while that of *CLE-QR w/o OR* is 2786, verifying that personalization in query rewriting can retrieve more relevant items. Giving the hundreds of millions of requests every day, such an improvement definitely improves resource utilization and performance of the search system.

6 CONCLUSION

This paper introduces our large-scale query rewriting solution, Contrastive Learning Enhanced Query Rewriting (*CLE-QR*). It addresses the representation degeneration during the retrieval stage (section 5.4.2, 5.4.3) and discrepancy between the pre-train and finetuned paradigms (section 5.4.4) in the previous query rewriting system. We further adopt an online ranking module to capture personalization so as to retrieve more relevant products (section 5.2.1, 5.5.1, 5.5.3). Offline experiments and online A/B tests verify the effectiveness of *CLE-QR*. We have deployed the proposed system on Taobao Product Search to serve hundreds of millions of users in real-time. Meanwhile, we share the lessons learned from improving the performance of lexical matching, including the optimization tricks, design of query rewriting system and its effect on each stage of the search system, its deployment scheme, and the online environment of our retrieval system.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful feedbacks.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al.

2016. Tensorflow: A system for large-scale machine learning. In *12th OSDI*. 265–283.
[2] Ioannis Antonellis, Hector Garcia-Molina, and Chi-Chao Chang. 2008. Simrank++ query rewriting through link analysis of the clickgraph. In *17th WWW*. 1177–1178.
[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv Preprint ArXiv:1409.0473* (2014).
[4] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *31st SIGIR*. 243–250.
[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *22nd SIGKDD*. 785–794.
[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*. 1597–1607.
[7] Zheng Chen, Xing Fan, and Yuan Ling. 2020. Pre-training for query rewriting in a spoken language understanding system. In *ICASSP*. 7969–7973.
[8] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. 2002. Probabilistic query expansion using query logs. In *11th WWW*. 325–332.
[9] Janez Demsar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *JMLR* 7 (2006), 1–30.
[10] 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).
[11] Doug Downey, Susan Dumais, and Eric Horvitz. 2007. Heads and tails: studies of web search with common and rare queries. In *30th SIGIR*. 847–848.
[12] Bruno M Fonseca, Paulo Golgher, Bruno Pôssas, Berthier Ribeiro-Neto, and Nivio Ziviani. 2005. Concept-based interactive query expansion. In *14th CIKM*. 696–703.
[13] Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Representation Degeneration Problem in Training Natural Language Generation Models. In *ICLR*.
[14] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. *ArXiv Preprint ArXiv:2104.08821* (2021).
[15] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context- and content-aware embeddings for query rewriting in sponsored search. In *38th SIGIR*. 383–392.
[16] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *25th CIKM*. 1443–1452.
[17] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *26th SIGKDD*. 2553–2561.
[18] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *15th WWW*. 387–396.
[19] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *43rd SIGIR*. 39–48.
[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980* (2014).
[21] Mu-Chu Lee, Bin Gao, and Ruofei Zhang. 2018. Rare query expansion through generative adversarial networks in search advertising. In *24th SIGKDD*. 500–508.
[22] Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, and Qianli Ma. 2021. Embedding-Based Product Retrieval in Taobao Search. In *27th SIGKDD*. 3181–3189.
[23] Yijiang Lian, Zhijie Chen, Jinlong Hu, Kefeng Zhang, Chunwei Yan, Muchenxuan Tong, Wenying Han, Hanju Guan, Ying Li, Ying Cao, et al. 2019. An end-to-end Generative Retrieval Method for Sponsored Search Engine–Decoding Efficiently into a Closed Target Domain. *ArXiv Preprint ArXiv:1902.00592* (2019).
[24] Xusheng Luo, Le Bo, Jinhang Wu, Lin Li, Zhiy Luo, Yonghua Yang, and Keping Yang. 2021. AliCoCo2: Commonsense Knowledge Extraction, Representation and Application in E-commerce. In *27th SIGKDD*. 3385–3393.
[25] Saurav Manchanda, Mohit Sharma, and George Karypis. 2019. Intent term weighting in e-commerce queries. In *28th CIKM*. 2345–2348.
[26] Aritra Mandal, Ishita K Khan, and Prathyusha Senthil Kumar. 2019. Query Rewriting using Automatic Synonym Extraction for E-commerce Search.. In *eCOM@ SIGIR*.
[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NeurIPS*. 3111–3119.
[28] Akash Kumar Mohankumar, Nikit Begwani, and Amit Singh. 2021. Diversity driven Query Rewriting in Search Advertising. *ArXiv Preprint ArXiv:2106.03816* (2021).
[29] Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic product search. In *25th SIGKDD*. 2876–2885.
[30] Matjaz Perc. 2014. The Matthew effect in empirical data. *Journal of the Royal Society Interface* 11, 98 (2014), 20140378–20140378.
[31] 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. In *37th ICDE*. IEEE, 2435–2446.
[32] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).
[33] Joseph Rocchio. 1971. Relevance feedback in information retrieval. *The Smart Retrieval System-experiments in Automatic Document Processing* (1971), 313–323.
[34] Amit Singhal et al. 2001. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.* 24, 4 (2001), 35–43.
[35] Zhenqiao Song, Jiaze Chen, Hao Zhou, and Lei Li. 2021. Triangular Bidword Generation for Sponsored Search Auction. In *14th WSDM*. 707–715.
[36] Tao Tao and ChengXiang Zhai. 2006. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *29th SIGIR*. 162–169.
[37] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, 11 (2008).
[38] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*. 9929–9939.
[39] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020. StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding. In *ICLR*.
[40] Yaxuan Wang, Hanqing Lu, Yunwen Xu, Rahul Goutam, Yiwei Song, and Bing Yin. 2021. QUEEN: Neural Query Rewriting in E-commerce. (2021).
[41] Jiewen Wu, Ihab Ilyas, and Grant Weddell. 2011. A study of ontology-based query expansion. In *Technical Report CS-2011–04*.
[42] 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 and Semantic Matching for e-Commerce. In *12th WSDM*. 402–410.
[43] Jinxi Xu and W Bruce Croft. 2017. Quary expansion using local and global document analysis. In *Acm Sigir Forum*, Vol. 51. 168–175.
[44] Xiaoyong Yang, Yadong Zhu, Yi Zhang, Xiaobo Wang, and Quan Yuan. 2020. Large Scale Product Graph Construction for Recommendation in E-commerce. *ArXiv Preprint ArXiv:2010.05525* (2020).
[45] 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 *28th CIKM*. 309–318.
[46] 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 *30th WWW*. 2890–2899.
[47] Han Zhang, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, and Wen-Yun Yang. 2020. Towards Personalized and Semantic Retrieval: An End-to-End Solution for E-commerce Search via Embedding Learning. In *43rd SIGIR*. 2407–2416.