Embedding-based Product Retrieval in Taobao Search

Sen Li, Fuyu Lv*
Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng
Alibaba Group
Hangzhou, China
{lisen.lisen,fuyu.lfy,taiwei.jtw,
guli.lingl,shaoyao,yuanhan}@alibaba-inc.com

Xiao-Ming Wu[1], Qianli Ma[2]
[1]The Hong Kong Polytechnic University
Hong Kong, China
[2]South China University of Technology
Guangzhou, China
csxmwu@comp.polyu.edu.hk,qianlima@scut.edu.cn

ABSTRACT

Nowadays, the product search service of e-commerce platforms has become a vital shopping channel in people's life. The retrieval phase of products determines the search system's quality and gradually attracts researchers' attention. Retrieving the most relevant products from a large-scale corpus while preserving personalized user characteristics remains an open question. Recent approaches in this domain have mainly focused on embedding-based retrieval (EBR) systems. However, after a long period of practice on Taobao, we find that the performance of the EBR system is dramatically degraded due to its: (1) low relevance with a given query and (2) discrepancy between the training and inference phases. Therefore, we propose a novel and practical embedding-based product retrieval model, named Multi-Grained Deep Semantic Product Retrieval (MGDSPR). Specifically, we first identify the inconsistency between the training and inference stages, and then use the softmax cross-entropy loss as the training objective, which achieves better performance and faster convergence. Two efficient methods are further proposed to improve retrieval relevance, including smoothing noisy training data and generating relevance-improving hard negative samples without requiring extra knowledge and training procedures. We evaluate MGDSPR on Taobao Product Search with significant metrics gains observed in offline experiments and online A/B tests. MGDSPR has been successfully deployed to the existing multi-channel retrieval system in Taobao Search. We also introduce the online deployment scheme and share practical lessons of our retrieval system to contribute to the community.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

Embedding-based retrieval system; E-commerce search

ACM Reference Format:
Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, and Xiao-Ming Wu[1], Qianli Ma[2]. 2021. Embedding-based Product Retrieval in Taobao Search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467101

*Equal contribution

1 INTRODUCTION

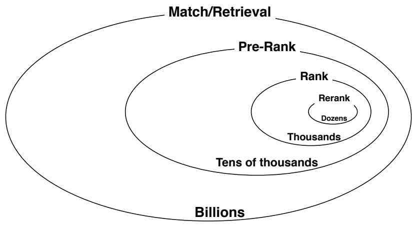

Figure 1: Overview of the product search system in Taobao. The head of each circle denotes different phase. The bottom is the scale of the corresponding candidate set.

Nowadays, online shopping has become a daily habit in people's lives. The top E-commerce shopping platforms (such as eBay, Amazon, Taobao, and JD) have hundreds of millions of daily active users and thus facilitate billion-level transaction records [17, 26, 34]. Therefore, product search engines are designed to discover products that satisfy users, which is also our working goal. As shown in Figure 1, our search system uses the "match-prerank-rank-rerank" architecture to screen and sort thousands of products from billions of candidates to possess controllable and high-efficiency characteristics. We finally return dozens of products to display to users. Obviously, the match (retrieval) phase plays an important role in determining the quality of the item candidate set fed to the follow-up ranking stage. The problem gradually receives more and more attention from academia and industry.

Search retrieval in e-commerce poses different challenges than in web (document) search: the text in e-commerce is usually shorter and lacks grammatical structure, while it is important to consider the massive historical user behaviors [2, 3]. The lexical matching engine (typically an inverted index [21, 24, 38]), despite its widely criticized semantic gap issue [14, 24, 31], remains a vital part of current retrieval systems due to its reliability and controllability of search relevance (exactly matching query terms). However, it hardly distinguishes users' interests in the same query and cannot flexibly capture user-specific characteristics. Hence, how to effectively retrieve the most relevant products satisfying users while considering the relationship between query semantics and historical user behaviors is the main challenge facing e-commerce platforms.

With the development of deep learning [35], Amazon [21] and JD [34] built their respective two-tower embedding-based retrieval (EBR) systems to provide relevant or personalized product candidates in their e-commerce search engines. Both of them reported the success of EBR without further discussing its low controllability of search relevance (compared to the lexical matching engine). We have also built an EBR system that can dynamically capture the relationship between query semantics and personalized user behaviors, and launched it on Taobao[1] Product Search for quite a long time. In the first deployment, it can achieve a good improvement in various metrics. However, after long observation, we have found that the embedding-based method's controllability of relevance is relatively low due to the inability to exactly matching query terms [11], resulting in increasing user complaints and bad cases that cannot be fixed. To improve its controllability of relevance (*i.e.*, resolving bad cases), we have adopted a relevance control module to filter the retrieved products. The control module only keeps those products that meet the relevance standards of exact matching signals and feed them to the follow-up ranking module. However, we statistically find it usually filters out thirty percent of candidates due to the low relevance of retrieved products. It is quite a waste of online computing resources because the filtered products cannot participate in the ranking stage, thus degrading the EBR system's performance. Therefore, the practical challenge for our search system is to enable the embedding-based model to retrieve more relevant products and increase the number of participants in the subsequent ranking stage.

Moreover, random negative samples are widely used to train large-scale deep retrieval models to ensure the sample space in training is consistent with that of the inference phase [14, 34]. Nevertheless, there remains a discrepancy in existing e-commerce product search methods [21, 34] due to the inconsistent behavior between the training and inference stages. Specifically, during inference, the model needs to select the top-K products closest to the current query from all candidates, requiring the ability for global comparison. However, [21] and [34] both adopt hinge (pairwise) loss as the training objective, which can only do local comparison.

This paper introduces the design of the proposed Multi-Grained Deep Semantic Product Retrieval (MGDSPR) model, its effect on each stage of the search system, and the lessons learned from applying it to product search. To tackle the above problems, we first use the softmax cross-entropy loss as the training objective to equip the model with global comparison ability, making training and inference more consistent. We further propose two effective methods without extra training procedures to enable MGDSPR to retrieve more relevant products. Specifically, we smooth the relevance noise introduced by using user implicit feedback (*i.e.*, click data) logs as training data [29, 31] by including a temperature parameter to the softmax function. Also, we mix the positive and random negative samples to generate relevance-improving hard negative samples. Moreover, we adapt the relevance control module to enhance the EBR system's controllability of search relevance. The effectiveness of MGDSPR is verified by an industrial dataset collected from the Taobao search system and online A/B tests.

The main contributions of this work are summarized as follows:

- We propose a Multi-Grained Deep Semantic Product Retrieval (MGDSPR) model to dynamically capture the relationship between user query semantics and his/her personalized behaviors and share its online deployment solution.
- We identify the discrepancy between training and inference in existing e-commerce retrieval systems and suggest using the softmax cross-entropy loss as the training objective to achieve better performance and faster convergence.
- We propose two methods to make the embedding-based model retrieve more relevant products without additional knowledge and training time. We further adapt the relevance control module to improve the EBR system's controllability of relevance.
- Experiments conducted on a large-scale industrial dataset and online Product Search of Taobao demonstrate the effectiveness of MGDSPR. Moreover, we analyze the effect of MGDSPR on each stage of the search system.

2 RELATED WORK

2.1 Deep Matching in Search

With the booming interest in deep NLP techniques, various neural models have been proposed to address the semantic gap problem raised by traditional lexical matching in the last few years. Those approaches fall into two categories: representation-based learning and interaction-based learning. The two-tower structure is the typical characteristic of representation-based models, such as DSSM [15], CLSM [25], LSTM-RNN [22], and ARC-I [13]. Each tower uses a siamese/distinct neural network to generate semantic representations of query/document. Then a simple matching function (*e.g.*, inner product) is applied to measure the similarity between the query and document. Interaction-based methods learn the complicated text/relevance patterns between the query and document. Popular models include MatchPyramid [23], Match-SRNN [28], DRMM [11], and K-NRM [32]. Other than semantic and relevance matching, more complex factors/trade-offs, *e.g.*, user personalization [2, 3, 10] and retrieval efficiency [5], need to be considered when applying deep models to a large-scale online retrieval system.

2.2 Deep Retrieval in Industry Search

Representation-based models with an ANN (approximate near neighbor) algorithm have become the mainstream trend to efficiently deploy neural retrieval models in industry. For social networks, Facebook developed an EBR system to take text matching and searcher's context into consideration [14]. They introduced various tricks and experiences (*e.g.*, hard mining, ensemble embedding, and inverted index-based ANN) to achieve hybrid retrieval (fuzzy matching). For display advertising, Baidu proposed MOBIUS [8] for CPM (cost per mile) maximization in the web ads retrieval phase, reducing the objective distinction between ranking and matching. For web search, Google [30] adopted transfer learning to learn semantic embeddings from data in recommendation systems to alleviate the cold start problem. Due to more text features and fewer user behaviors, their search scenarios are characterized by strong semantic matching and weak personalization. For e-commerce search, Amazon developed a two-tower model to address the semantic gap issue in a lexical matching engine for semantic product retrieval [21],

[1] https://www.taobao.com/

where one side uses n-gram query features and the other side exploits item features, without considering user personalization. Recently, JD [34] proposed a deep personalized and semantic retrieval model (DPSR) to combine text semantics and user behaviors. However, DPSR aggregates user behaviors through average pooling, weakening personalization characteristics. Furthermore, neither Amazon nor JD studies the problem of insufficient product relevance caused by the EBR method. This paper will discuss the low relevance issue of the EBR system encountered in Taobao Product Search and propose our solution.

3 MODEL

Here, we introduce our model called Multi-Grained Deep Semantic Product Retrieval (MGDSPR) to simultaneously model query semantics and historical behavior data, aiming at retrieving more products with good relevance. The general structure of MGDSPR is illustrated in Figure 2. We first define the problem and then introduce our design of the two-tower model, including the user tower and the item (product) tower. Finally, we elaborate on the training objective and proposed methods to retrieve more relevant products.

3.1 Problem Formulation

We first formulate the e-commerce product retrieval problem and our solution as well as the notations used in this paper. Let $\mathcal{U} = \{u_1, ..., u_u, ..., u_N\}$ denote a collection of N users, $\mathcal{Q} = \{q_1, ..., q_u, ..., q_N\}$ denote the corresponding queries, and $\mathcal{I} = \{i_1, ..., i_i, ..., i_M\}$ denote a collection of M items (products). Also, we divide the user u's historical behaviors into three subsets according to the time interval from the current time t: real-time (denoted as $\mathcal{R}^u = \{i_1^u, ..., i_t^u, ..., i_T^u\}$, before the current time step), short-term (denoted as $\mathcal{S}^u = \{i_1^u, ..., i_t^u, ..., i_T^u\}$, before $\mathcal{R}$ and within ten days) and long-term sequences (denoted as $\mathcal{L}^u = \{i_1^u, ..., i_t^u, ..., i_T^u\}$, before $\mathcal{S}$ and within one month), where T is the length of the sequence.

We now define the task. Given the historical behaviors $(\mathcal{R}^u, \mathcal{S}^u, \mathcal{L}^u)$ of a user $u \in \mathcal{U}$, after he/she submits query q_u at time t, we would like to return a set of items $i \in \mathcal{I}$ that satisfy his/her search request. Typically, we predict top-K item candidates from $\mathcal{I}$ at time t based on the scores z between the user (query, behaviors) and items, i.e.,

$$z = \mathcal{F}(\phi(q_u, \mathcal{R}^u, \mathcal{S}^u, \mathcal{L}^u), \psi(i)), \tag{1}$$

where $\mathcal{F}(\cdot)$, $\phi(\cdot)$, $\psi(\cdot)$ denote the scoring function, query and behaviors encoder, and item encoder, respectively. Here, we adapt the two-tower retrieval model for efficiency. We instantiate $\mathcal{F}$ with the inner product function. In the following, we introduce the design of the user and item towers, respectively.

3.2 User Tower

3.2.1 Multi-Granular Semantic Unit. Queries in Taobao search are usually in Chinese. After query segmentation, the average length of the segmentation result is less than three. As such, we propose a multi-granular semantic unit to discover the meaning of queries from multiple semantic granularities and enhance the representation of queries. Given a query's segmentation result $q_u = \{w_1^u, ..., w_n^u\}$ (*e.g.*, {红色, 连衣裙}), each $w^u = \{c_1^u, ..., c_m^u\}$ (*e.g.*, {红, 色}), and its historical query $q_{his} = \{q_1^u, ..., q_k^u\} \in \mathbb{R}^{k \times d}$ (*e.g.*, {绿色, 半身裙, 黄色, 长裙}), where $w_n \in \mathbb{R}^{1\times d}$, $c_m \in \mathbb{R}^{1\times d}$ and $q_k \in \mathbb{R}^{1\times d}$, we can obtain its six granular representations $Q_{mgs} \in \mathbb{R}^{6\times d}$. It is done by concatenating q_u's unigram mean-pooling $q_{1\_gram} \in \mathbb{R}^{1\times d}$, 2-gram mean-pooling $q_{2\_gram} \in \mathbb{R}^{1\times d}$, word segmentation mean-pooling $q_{seg} \in \mathbb{R}^{1\times d}$, word segmentation sequence $q_{seg\_seq} \in \mathbb{R}^{1\times d}$, historical query words $q_{his\_seq} \in \mathbb{R}^{1\times d}$, and mixed $q_{mix} \in \mathbb{R}^{1\times d}$ representations. d, n, and m denote the embedding size, the number of word segmentation, and the number of words in each segment, respectively. Formally, the **M**ulti-**G**ranular **S**emantic representation Q_{mgs} is obtained as follows:

$$q_{1\_gram} = mean\_pooling(c_1, ..., c_m), \tag{2}$$

$$q_{2\_gram} = mean\_pooling(c_1c_2, ..., c_{m-1}c_m), \tag{3}$$

$$q_{seg} = mean\_pooling(w_1, ..., w_n), \tag{4}$$

$$q_{seg\_seq} = mean\_pooling(Trm(w_1, ..., w_n)), \tag{5}$$

$$q_{his\_seq} = softmax(q_{seg} \cdot (q_{his})^T) q_{his}, \tag{6}$$

$$q_{mix} = q_{1\_gram} + q_{2\_gram} + q_{seg} + q_{seg\_seq} + q_{his\_seq}, \tag{7}$$

$$Q_{mgs} = concat(q_{1\_gram}, q_{2\_gram}, q_{seg}, q_{seg\_seq}, q_{his\_seq}, q_{mix}), \tag{8}$$

where Trm, $mean\_pooling$, and $concat$ denote the Transformer [27], average, and vertical concatenation operation, respectively. We average all the outputs of the last layer of the Transformer in Eq. (5).

3.2.2 User Behaviors Attention. User behaviors are recorded by their history of items clicked (or bought). Taking user u's short-term behaviors $\mathcal{S}^u$ as an example, $i_t^u \in \mathcal{S}^u$ denotes the user clicks on item i at time t, and each item i is described by its ID and side information $\mathcal{F}$ (*e.g.*, leaf category, first-level category, brand and, shop) [18]. Specifically, each input item $i_t^u \in \mathcal{S}^u$ is defined by:

$$e_i^f = W_f \cdot x_i^f, \tag{9}$$

$$i_t^u = concat(\{e_i^f | f \in \mathcal{F}\}), \tag{10}$$

where W_f is the embedding matrix and x_i^f is a one-hot vector. $e_i^f \in \mathbb{R}^{1\times d_f}$ is the corresponding embedding vector of size d_f and $\cdot$ denotes matrix multiplication. We concatenate the embeddings of item i's ID and side information in Eq. (10). We use the same way to embed items in real-time $\mathcal{R}^u$ and long-term $\mathcal{L}^u$ sequences.

Unlike the target-item attention [9, 36, 37] used in advertising and recommendation, here we use query attention to capture user history behaviors related to the current query semantics. Moreover, inspired by [2], we put an all-zero vector into user behavior data to remove potential noise and deal with situations where the historical behaviors may not be related to the current query. In the following, we introduce the fusion of real-time, short-term, and long-term sequences, respectively.

For real-time sequences $\mathcal{R}^u = \{i_1^u, ..., i_t^u, ..., i_T^u\}$, we apply Long Short-Term Memory (LSTM) [12] to capture the evolution and collect all hidden states $\mathcal{R}_{lstm}^u = \{h_1^u, ..., h_t^u, ..., h_T^u\}$. Next, we use multi-head self-attention to aggregate multiple potential points of interest [18] in $\mathcal{R}_{lstm}^u$ to get $\mathcal{R}_{self\_att}^u = \{h_1^u, ..., h_t^u, ..., h_T^u\}$. Then, we add a zero vector at the first position of $\mathcal{R}_{self\_att}^u$, resulting in $\mathcal{R}_{zero\_att}^u = \{0, h_1^u, ..., h_t^u, ..., h_T^u\} \in \mathbb{R}^{(T+1)\times d}$. Finally, the real-time personalized representation $H_{real} \in \mathbb{R}^{6\times d}$ related to the current query is obtained by the attention operation (Q_{mgs} is analogous to

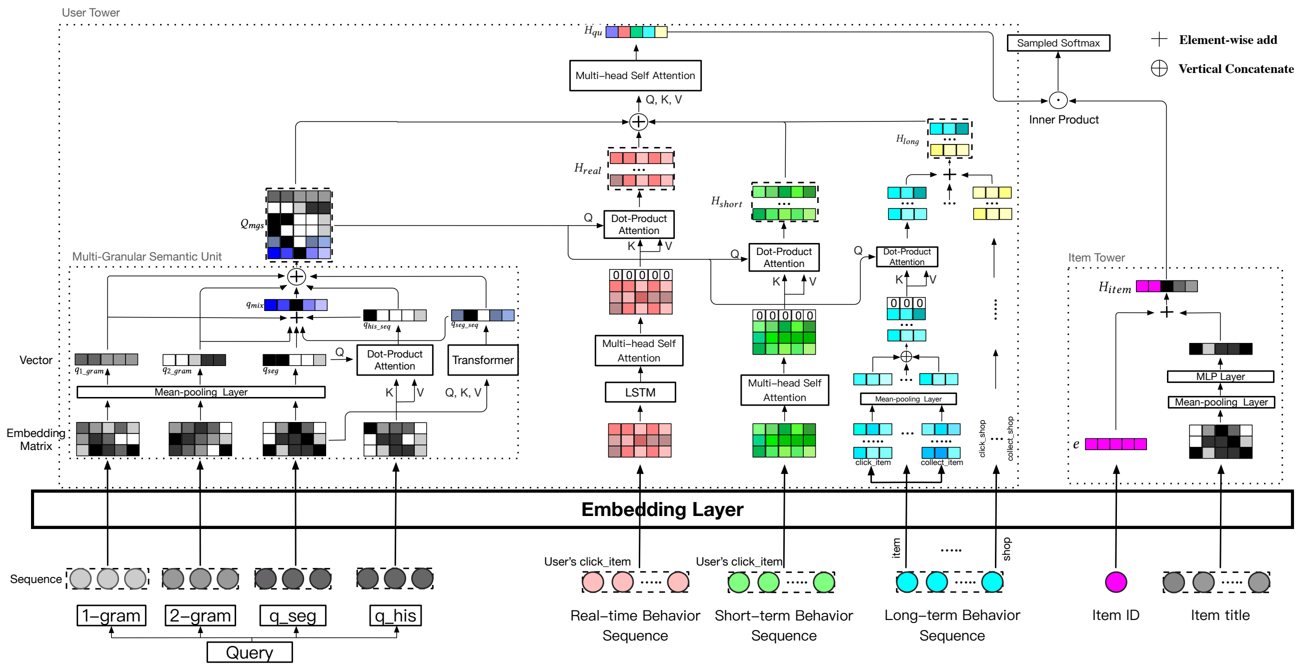

Figure 2: General architecture of the proposed Multi-Grained Deep Semantic Product Retrieval model (MGDSPR).

Q in the attention mechanism), which is defined by:

$$H_{real} = softmax(Q_{mgs} \cdot R^T_{zero\_att}) \cdot R^T_{zero\_att}. \quad (11)$$

For short-term sequences $\mathcal{S}^u = \{i^u_1, ..., i^u_t, ..., i^u_T\}$, we apply multi-head self-attention to aggregate $\mathcal{S}^u$ into $\mathcal{S}^u_{self\_att} = \{h^u_1, ..., h^u_t, ..., h^u_T\}$. We add a zero vector at the first position of $\mathcal{S}^u_{self\_att}$, resulting in $\mathcal{S}^u_{zero\_att} = \{0, h^u_1, ..., h^u_t, ..., h^u_T\} \in \mathbb{R}^{(T+1)\times d}$. Finally, the short-term personalized representation $H_{short} \in \mathbb{R}^{6\times d}$ is defined by:

$$H_{short} = softmax(Q_{mgs} \cdot S^T_{zero\_att}) \cdot S^T_{zero\_att}. \quad (12)$$

The real-time and short-term sequences are composed of click sequences.

We use four attribute behaviors to describe the long-term sequence (within one month), including *item* ($\mathcal{L}^u_{item}$), *shop* ($\mathcal{L}^u_{shop}$), *leaf category* ($\mathcal{L}^u_{leaf}$) and *brand* ($\mathcal{L}^u_{brand}$). Each attribute behavior is described by a user's click, buy and collecting actions. For example, $\mathcal{L}^u_{item}$ consists of multiple action sequences: $\mathcal{L}_{click\_item}$, $\mathcal{L}_{buy\_item}$ and $\mathcal{L}_{collect\_item}$. Entries in each action sequence are embedded by Eq. (9) and aggregated into a vector through mean-pooling with consideration of quick response in online environment, resulting in $\mathcal{L}^u_{item} = \{0, h_{click}, h_{buy}, h_{collect}\}$. The representation of item attribute behavior $H_{a\_item} \in \mathbb{R}^{6\times d}$ is then defined by:

$$H_{a\_item} = softmax(Q_{mgs} \cdot L^T_{item}) \cdot L^T_{item}. \quad (13)$$

Finally, the long-term personalized representation $H_{long} \in \mathbb{R}^{6\times d}$ is defined as follows:

$$H_{long} = H_{a\_item} + H_{a\_shop} + H_{a\_leaf} + H_{a\_brand}, \quad (14)$$

where H_{a_shop}, H_{a_leaf}, and H_{a_brand} denote the representation of the attribute behaviors of *shop*, *leaf category*, and *brand* respectively.

3.2.3 *Fusion of Semantics and Personalization.*

To retrieve products relevant to the current user's query and preserve personalized characteristics, we take the multi-granular semantic representation Q_{mgs} and personalized representations ($H_{real}, H_{short}, H_{long}$) as the input of self-attention to dynamically capture the relationship between the two. Specifically, we add a "[CLS]" token at the first position of the input $I = \{[CLS], Q_{mgs}, H_{real}, H_{short}, H_{long}\}$ of self-attention and regard the output as the user tower's representation $H_{qu} \in \mathbb{R}^{1\times d}$, which is defined as follows:

$$H_{qu} = Self\_Att^{first}([[CLS], Q_{mgs}, H_{real}, H_{short}, H_{long}]). \quad (15)$$

3.3 Item Tower

For the item tower, we experimentally use item ID and title to obtain the item representation H_{item}. Given the representation of item i's ID, $e_i \in \mathbb{R}^{1\times d}$, and its title segmentation result $T_i = \{w^i_1, ..., w^i_N\}$, $H_{item} \in \mathbb{R}^{1\times d}$ is calculated as follows:

$$H_{item} = e + tanh(W_t \cdot \frac{\sum_{i=1}^N w_i}{N}), \quad (16)$$

where W_t is the transformation matrix. We empirically find that applying LSTM [12] or Transformer [27] to capture the context of the title is not as effective as simple mean-pooling since the title is stacked by keywords and lacks grammatical structure.

3.4 Loss Function

To make the sample space where the model is trained consistent with that of online inference, Huang et al. [14], Nigam et al. [21], and Zhang et al. [34] use random samples as negative samples. However, they use pairwise (hinge) loss as the training objective, making training and testing behavior inconsistent. Specifically, during inference, the model needs to pick the top-K items that are closest to the current query from all candidates, which requires the model to have the ability of global comparison. However, hinge loss can only do local comparison. Also, hinge loss introduces a cumbersome tuning margin, which has a significant impact on performance [14]. Here, we adapt the softmax cross-entropy loss as the training objective, achieving faster convergence and better performance without additional hyper-parameter tuning.

Given a user u and his/her query q_u, the positive item i^+ is the item clicked by u under q_u. The training objective is defined by:

$$\hat{y}_{(i^+|q_u)} = \frac{\exp(\mathcal{F}(q_u, i^+))}{\sum_{i' \in I} \exp(\mathcal{F}(q_u, i'))}, \tag{17}$$

$$L(\nabla) = -\sum_{i \in I} y_i \log(\hat{y}_i), \tag{18}$$

where $\mathcal{F}$, I, i^+, and q_u denote the inner product, the full item pool, the item tower's representation H_{item}, and the user tower's representation H_{qu}, respectively. Note that Eq. (17) endows the model with global comparison ability. The softmax involves calculating an expensive partition function, which scales linearly to the number of items. In practice, we use sampled softmax (an unbiased approximation of full-softmax) [4, 16] for training. Similar to [34], we also experimentally find that using the same set of random negative samples for every training example in the current batch results in similar performance as using a different set for each one. We adopt the former training method to reduce computing resources.

To improve the EBR system's relevance in retrieval and increase the number of products participating in the follow-up ranking stage while maintaining high efficiency, we propose two efficient methods without relying on additional knowledge to make our model retrieve more relevant products.

3.4.1 *Smoothing Noisy Training Data.*

In e-commerce search, users' click and purchase records are used as supervisory signals to train a model. However, these signals are noisy since they are influenced not only by query-product relevance but also by images, prices, and user preferences [29, 31]. Hence, we introduce a temperature parameter τ into softmax to smooth the overall fitted distribution of the training data. If τ->0, the fitted distribution is close to one-hot distribution, which means that the model completely fits the supervisory signals. The model will be trained to push positive items far away from negative ones, even if the relevance of a positive item is low. If τ->∞, the fitted distribution is close to a uniform distribution, indicating that the model does not fit the supervisory signals at all. We can increase τ to reduce the noise in training data and thus alleviate the impact of low relevance caused by fully fitting users' click records, which does not require additional knowledge and is verified by our experiments. Formally, the softmax function with the temperature parameter τ is defined as follows:

$$\hat{y}_{(i^+|q_u)} = \frac{\exp(\mathcal{F}(q_u, i^+)/\tau)}{\sum_{i' \in I} \exp(\mathcal{F}(q_u, i')/\tau)}. \tag{19}$$

3.4.2 *Generating Relevance-improving Hard Negative Samples.*

Unlike prior works [20] that require additional annotated training data and training process, we propose a method to generate relevance-improving hard negative samples in the embedding space. Specifically, given a training example (q_u, i^+, i^-), where i^- denotes a set of random negative samples sampled from item pool I. For simplicity, we use q_u, i^+, and i^- to refer to their respective representations. We first select the negative items of i^- that have the top-N inner product scores with q_u to form the hard sample set I_{hard}, and then mix $i^+ \in \mathbb{R}^{1\times d}$ and $I_{hard} \in \mathbb{R}^{N\times d}$ by interpolation to obtain the generated sample set $I_{mix} \in \mathbb{R}^{N\times d}$, which is defined as follows:

$$I_{mix} = \alpha i^+ + (1-\alpha) I_{hard}, \tag{20}$$

where $\alpha \in \mathbb{R}^{N\times 1}$ is sampled from the uniform distribution $U(a, b)$ $(0 \le a < b \le 1)$. The closer α is to 1, the closer the generated sample is to the positive samples i^+ in the embedding space, indicating the harder the generated sample is. We take I_{mix} as the set of relevance-improving hard negative samples and include it in the denominator of the softmax function to make the model distinguish the positive sample i^+ and its nearby samples. Formally, the softmax function with relevance-improving hard samples I_{mix} is defined as follows:

$$\hat{y}_{(i^+|q_u)} = \frac{\exp(\mathcal{F}(q_u, i^+)/\tau)}{\sum_{i' \in I \cup I_{mix}} \exp(\mathcal{F}(q_u, i')/\tau)}. \tag{21}$$

Note that we can tune the maximum b and minimum a of the uniform distribution U to determine the "hardness" of the generated relevance-improving negative samples. This generation process only needs a linear interpolation after calculating the inner product scores between the current query q_u and the negative samples of i^-, which is quite efficient.

4 SYSTEM ARCHITECTURE

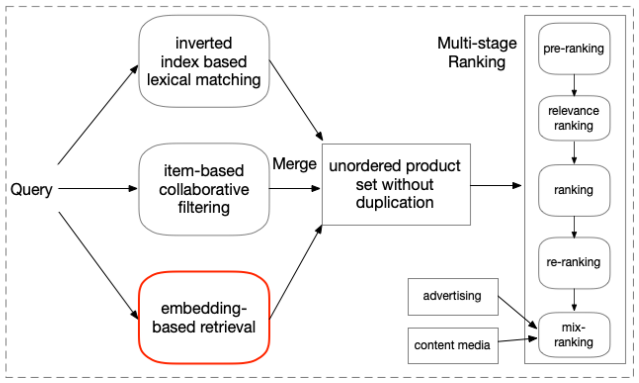
Query
inverted index based lexical matching
item-based collaborative filtering
embedding-based retrieval
Merge
unordered product set without duplication
Multi-stage Ranking
pre-ranking
relevance ranking
ranking
re-ranking
mix-ranking
advertising
content media

Figure 3: Overview of Taobao search engine.

As illustrated in Figure 3, at a high level, the Taobao search engine works as follows: a user issues a query, which triggers a multi-channel retrieval system, producing an unordered candidate set without duplication. Before the most relevant items are finally displayed to users, the candidates are passed through multi-stages of ranking, including pre-ranking, relevance ranking (removing products that are inconsistent with the predictions of the query's category), ranking, re-ranking, and mix-ranking. Our embedding-based retrieval module is the third matching channel as a supplement to the existing two-channel retrieval. In the following, we introduce how to deploy MGDSPR in the production environment once we have trained the model and the relevance control module on the EBR system.

4.1 Offline Training and Indexing

We use search logs in the past one week to train the model by distributed Tensorflow [1] and update the model parameters daily. Note that we do not use sequential training [33] to do the A/B test. Since the base model has been trained by lots of data (several months or even one year), it is difficult to catch up with the same volume of data for the new testing model. As illustrated in Figure 4, the deployment system of MGDSPR is an offline to online architecture. At the offline phrase, *build service* optimizes and constructs a user/query network extracted from the user tower, which is passed to *real-time prediction* platform. All the item embeddings are simultaneously exported from the item tower and transmitted to an approximate near neighbor (ANN) indexing system. The total number of items is about one hundred millions. They are placed in multiple columns (6 in our system) because of the enormous amounts. Each column of the ANN builds indexes of embeddings by HC (hierarchical clustering) algorithm with K-means and INT8 quantization to promote storage and search efficiency. The training sample size is 4 million for HC, and the max scan ratio is 0.01.

4.2 Online Serving

The user/query network and item embedding indexes are published in an online environment after offline indexing. When a user issues a query, user history behaviors and the query are fed into a real-time prediction platform for online inference. The *ANN search module* then distributively seeks top-K ($K = 9600$ in our system) results from indexes of multi-columns (referred to as $n = 6$ columns). Each column returns the same size of K/n. The indexing retrieval accuracy is 98% accompanied with 10 milliseconds of retrieval latency.

4.3 Relevance Control

After a long period of practice, we find that although embedding-based retrieval has advantages in personalization and fuzzy matching, it often leads to more search bad cases due to lack of exact matching [**?**] to the key terms of a query. The *key terms of a query* are referred to as words of brand, type, color, etc., which are significant to product search relevance. For instance, a user is searching for *Adidas sports shoes*. Items of *Nike sports shoes* are similar to the query in the embedding space and hence will appear in the top-K results with high probability. However, this is not the user intent and will harm user experience. Hence, we add an inverted index based *boolean matching* module on top of the ANN results. Boolean matching aims to filter out items that do not contain key query terms in their titles. The final search results can then be expressed as:

```
(ANN results) and (Brand: Adidas) and (Category: Shoes).
```

Generally, we predefine the rule of key terms according to query understanding, *e.g.*, words of brand, color, style, and audience. Note that Facebook [14] uses an embedding-based method to enhance boolean expression and achieves fuzzy matching, while we use boolean matching to improve retrieval relevance of the EBR system.

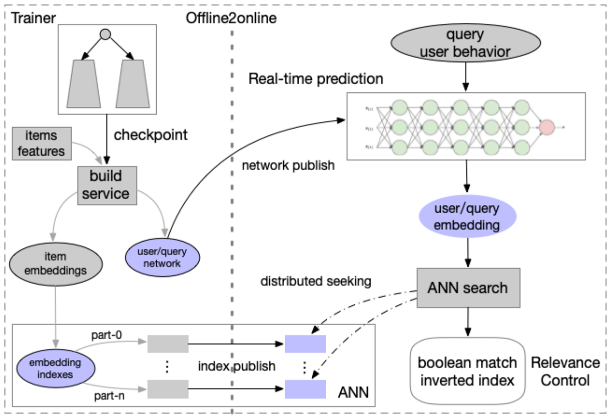

Figure 4: Deployment system of our MGDSPR model.

5 EXPERIMENTS

Here, we introduce evaluation metrics, implementation details, datasets, and offline and online experimental results of our method including its effect in the search system.

5.1 Evaluation Metrics

5.1.1 Offline Evaluation. We use the metric of Recall@K to evaluate the offline performance. 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 a model are regarded as the retrieval set $I = \{i_1, ..., i_K\}$. Recall@K is defined as

$$Recall@K = \frac{\sum_{i=1}^{K} i_i \in T}{N}. \tag{22}$$

Empirically, during the retrieval phase, we find that the AUC metric has no positive correlation with the online Gross Merchandise Volume (GMV) metric, while the recall metric does. Also, we add T_{rec}, the records relevant to the current query that were not purchased in search but elsewhere (*e.g.*, recommender system) in Taobao Mobile App, to the testing set.

In Taobao search, we also pay attention to the relevance of retrieved products (related to user experience). Due to the large amount of test data, we use an online well-trained relevance model (its AUC for human-labeled data is **0.915**) instead of expensive human evaluation to calculate the proportion of products with good relevance (abbreviated as good rate and denoted as P_{good}) in the retrieval set I, which is defined as

$$P_{good} = \frac{\sum_{i=1}^{K} \mathbb{I}(i_i)}{K}, \tag{23}$$

where $\mathbb{I}(\cdot)$ is an indicator function. When item i is rated as good by the relevance model, the function value is 1, otherwise 0. It is not appropriate to use the AUC metric to evaluate whether our model can return more products with good relevance because it evaluates the order of the set elements rather than the number of the "good" elements. K is experimentally set to be $1,000$.

Meanwhile, to analyze the effect of our model on each stage of the search system, we also count the number of items in the retrieval set I that participate in each follow-up stage. Given a retrieval set $I = \{i_1, ..., i_K\}$, every time it goes through a stage (such as the relevance control module, pre-ranking, and ranking), the number of items will decrease, resulting in $I_{left} = \{i_1, ..., i_k\}$, $k < K$. Therefore, we calculate the number of items in I_{left} after going through each phase, and use Num_{prank} and Num_{rank} to denote the number of items that enter the pre-ranking and ranking stages. For a total retrieval set $\mathcal{I} = \{I_1, ..., I_i, ..., I_N\}$ of N queries, the calculation of Num_{prank} and Num_{rank} are averaged by N.

5.1.2 Online Evaluation. We consider the most important online metrics: **GMV**, $\mathbf{P_{good}}$, and $\mathbf{P_{h\_good}}$. GMV is the Gross Merchandise Volume, which is defined as

$$GMV = \#\text{pay amount}. \qquad (24)$$

In addition to the amount of online income, we also consider user search experience by the $\mathbf{P_{good}}$ and $\mathbf{P_{h\_good}}$ metrics (defined in Eq. (23)). Precisely, both P_{good} and P_{h_good} calculate the good rate of the item set displayed to users, but P_{good} is determined by the relevance model while P_{h_good} is determined by humans.

5.2 Implementation Details

The maximum length T of real-time, short-term, and long-term sequences are 50, 100, and 100, respectively. We use attention with a mask to calculate those sequences whose length is less than T. The dimensions of the user tower, item tower, behavior sequence, and hidden unit of LSTM are all set to 128. The batch size is set to 256. We use LSTM of two layers with dropout (probability 0.2) and residual network [19] between vertical LSTM stacks. The number of heads in self-attention is set to 8. The parameters a and b of uniform distribution U and the number of generated samples N are set to 0.4, 0.6 and 684, respectively. The temperature parameter τ of softmax is set to 2. All parameters are orthogonally initialized and learned from scratch. The experiments are run on the distributed TensorFlow platform [1] using 20 parameter servers and 100 GPU (Tesla P100) workers. The AdaGrad optimizer [7] is employed with an initial learning rate of 0.1, which can improve the robustness of SGD for training large-scale networks [6]. We also adopt gradient clip when the norm of gradient exceeds a threshold of 3. The training process converges at about 35 million steps for about 54 hours.

5.3 Datasets

5.3.1 Large-scale Industrial Offline DataSet. We collect search logs of user clicks and purchases for 8 consecutive days from online **Mobile Taobao App** in December 2020, and filter the spam users. The training set comprises samples from the first 7 consecutive days (a total of 4.7 billion records). For evaluation, we randomly sample 1 million search records T and 0.5 million purchase logs T_{rec} from the recommender system in the 8-th day. We have also tried to extend the timeframe of training data to 10 days, but there is no significant benefit, indicating billions of data can effectively prevent the model from overfitting. The size of the candidate item set is consistent with the online environment, *i.e.*, about 100 million.

5.3.2 Online Dataset. We deploy a well-trained MGDSPR in the Taobao search production environment containing hundreds of millions of user query requests. The size of the item candidate set is about 100 million, covering the most active products at Taobao.

Table 1: Comparison with the strong baseline α-DNN on a large-scale industrial offline dataset. Num_{prank} is the number of products that flow into the follow-up pre-ranking phase. P_{good} is the good rate. Relative improvements are shown in parentheses.

Methods	Recall@1000	P_{good}	P_{f_good}	Num_{prank}
α-DNN [5]	82.6%	70.6%	83.2%	769
MGDSPR	84.7%(+2.5%)	80.0%(+13.3%)	84.1%(+1.1%)	815(+6.0%)

5.4 Offline Experimental Results

Previously, our embedding-based retrieval system adopts the DNN architecture proposed in [5], but uses more user behaviors and statistical features (inherited from the ranking model), which has been experimentally verified to be effective to some extent. Specifically, we concatenate the vectors of user behaviors (obtained by mean-pooling) and statistical features (*e.g.*, Unique Visitor (UV), Item Page View (IPV)) and feed it into a multi-layer feed-forward neural network. We refer to it as a strong baseline α-DNN. In addition, adding statistical features to MGDSPR has no benefit in the metric of *recall*, so we delete them but keep the user behavior sequences.

5.4.1 Comparison with the Strong Baseline. As mentioned in Section 5.1.1, we report the metrics of Recall@K, good rate, and Num_{prank}. Note that we report P_{good} on both the retrieval set I (denoted as P_{good}) and the filtered set I_{left} (denoted as P_{f_good}). As shown in Table 1, MGDSPR improves over α-DNN by 2.5%, 13.3% and 6.0% in Recall@1000, P_{good} and P_{f_good} respectively, indicating it can retrieve more products with good relevance and improve the quality of the retrieval set. Comparing P_{good} and P_{f_good} shows our relevance control module enhances retrieval relevance.

Table 2: Ablation study of MGDSPR.

Methods	Recall@1000	P_{good}
MGDSPR	85.6%	71.2%
MGDSPR + *mgs*	86.0%	71.6%
MGDSPR + *trm*	86.4%	71.4%
MGDSPR + τ	85.5%	79.0%
MGDSPR + *mgs* + *trm* + τ	86.8%	79.2%
MGDSPR + I_{mix}	83.6%	75.6%
MGDSPR + all	84.7%	80.0%

5.4.2 Ablation Study. We study the effectiveness of each component of MGDSPR by adding only one component at a time. Specifically, MGDSPR have the following four components: 1) Multi-Granular Semantic unit (*i.e.*, Eq. (8), denoted as *mgs*); 2) dynamic fusion of semantics and personalization (*i.e.*, Eq. (15), denoted as *trm*); 3) the temperature parameter τ of softmax (denoted as τ); 4) the relevance-improving hard negative samples (denoted as I_{mix}). Note that here we focus on the model's performance, so good rate P_{good} is calculated on the retrieval set I instead of I_{left}.

As shown in Table 2, both the multi-granular semantics unit *mgs* and *trm* can improve the metrics of Recall@1000 and P_{good}, indicating the effectiveness of multi-granular semantics and dynamic fusion. The temperature parameter τ and relevance-improving hard negative samples I_{mix} make the model retrieve more relevant products in terms of much higher good rate P_{good}. Comparing MGDSPR+all and MGDSPR or MGDSPR+mgs+trm+τ, we observe there is a trade-off between recall and relevance even in search scenarios, which may indicate excessive personalization in our system.

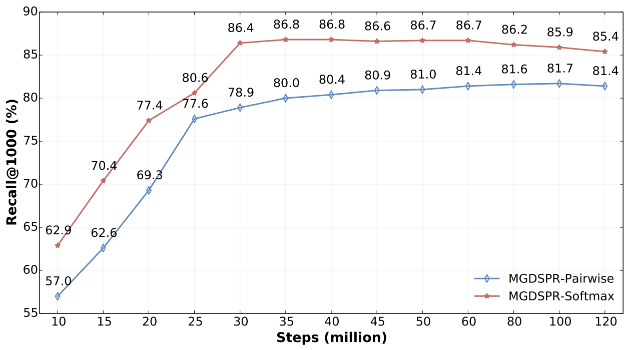

Figure 5: Convergence comparison of the softmax cross-entropy and hinge (pairwise) loss functions. The X-axis denotes the number of training steps, and the Y-axis denotes the corresponding test Recall@1000 score.

5.4.3 Convergence Analysis. We investigate the performance of MGDSPR using softmax cross-entropy and pairwise loss [21, 34] as the training objective, respectively. We report the test Recall@1000 score with respect to the nubmer of training steps. As shown in Figure 5, the softmax function's global comparison capability make training and testing more consistent, achieving faster convergence and better performance. In fact, it only takes about three days for the softmax loss to converge while about six days for the pairwise loss. Note that the margin parameter used in the hinge loss has been carefully tuned.

5.4.4 Hyper-parameter Analysis. We perform an investigation of the hyper-parameters τ (for noise smoothing) and N (the number of generated relevance-improving hard negative samples) to demonstrate how they affect the good rate P_{good}. We conduct the evaluation by varying τ (or N) while fixing the other parameters.

As mentioned in Section 3.4.1, we can increase τ to smooth the noisy training data and thus alleviate the effect of insufficient relevance due to overfitting users' click records. As shown in Figure 6, $\tau = 0.1$ decreases relevance, indicating that the training data does have noise. Also, every non-zero value of N gives better relevance than $N = 0$, showing that the generated hard negative samples can improve good rate P_{good}. Further, the good rate P_{good} reaches its maximum at $N = 684$ and then decreases, indicating that simply increasing the number of samples cannot bring more benefits.

5.5 Online A/B Test

We deploy MGDSPR on Taobao Product Search and compare it with the strong baseline α-DNN. As aforementioned, to improve user experience, our relevance control module (introduced in Section 4.3) will filter out some products retrieved by the EBR system, resulting in low utilization of online computing resources. Therefore, apart from **GMV**, $\mathbf{P_{good}}$, and $\mathbf{P_{h\_good}}$, we report the number of products that participate in the pre-ranking and ranking phases (denoted as Num_{prank} and Num_{rank}) to analyze the model's effect on our search system.

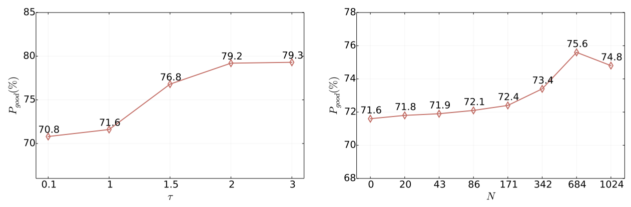

Figure 6: The influence of τ and N on the good rate P_{good}.

Table 3: The improvements of MGDSPR in Num_{prank}, Num_{rank}, P_{good}, and P_{h_good} compared with the previous model deployed on Taobao Product Search. The last two columns only report relative values that are calculated on the exposed item set.

Methods	Num_{prank}	Num_{rank}	P_{good}	P_{h_good}
Baseline	4070	1390	-	-
MGDSPR	4987(+22.53%)	1901(+36.76%)	+1.0%	+0.35%

As shown in Table 3, after being filtered by the relevance control module, the number of products retrieved by MGDSPR that enter the pre-ranking and ranking phases increases by 22.53% and 36.76%, respectively. Obviously, MGDSPR retrieves more products with good relevance and effectively improves the utilization of computing resources. Besides, MGDSPR achieves higher good rates P_{good} and P_{h_good} of exposure relevance, and thus can display more relevant products to users.

Table 4: Online A/B test of MGDSPR. The improvements are averaged over 10 days in Jan 2021.

Launched Platform	GMV	#Transactions
Taobao Search on Mobile	+0.77%	+0.33%

Finally, we report the 10-day average of GMV improvements (by removing cheating traffic) achieved by MGDSPR. We also include the corresponding number of transactions (denoted as #Transactions) to increase results confidence. As shown in Table 4, MGDSPR improves GMV and #Transactions by 0.77% and 0.33%, respectively. Considering the billions of transaction amounts per day in Taobao Search, 0.77% improvement is already tens of millions of transaction amounts, indicating MGDSPR can significantly better satisfy users.

6 CONCLUSION

This paper proposes a practical embedding-based product retrieval model, named Multi-Grained Deep Semantic Product Retrieval

(MGDSPR). It addresses model performance degradation and online computing resource waste due to the low retrieval relevance in the previous EBR system of Taobao Product Search. Meanwhile, we share the lessons learned from solving those problems, including model design and its effect on each stage of the search system, selection of offline metrics and test data, and relevance control of the EBR system. We verify the effectiveness of MGDSPR experimentally by offline and online A/B tests. Furthermore, we have deployed MGDSPR on Taobao Product Search to serve hundreds of millions of users in real time. Moreover, we also introduce the online architecture of our search system and the deployment scheme of the retrieval model to promote development of the community.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful feedbacks. This research was supported by the Alibaba Innovative Research project $P0034058$ (ZGAL).

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 {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Qingyao Ai, Daniel N Hill, SVN Vishwanathan, and W Bruce Croft. 2019. A zero attention model for personalized product search. In *Proceedings of the 28th ACM International Conference on Information & Knowledge Management*. 379–388.

[3] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W Bruce Croft. 2017. Learning a hierarchical embedding model for personalized product search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 645–654.

[4] Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks* 19, 4 (2008), 713–722.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 191–198.

[6] Jeffrey Dean, Greg S Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V Le, Mark Z Mao, Marc' Aurelio Ranzato, Andrew Senior, Paul Tucker, et al. 2012. Large scale distributed deep networks. (2012).

[7] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, 7 (2011).

[8] Miao Fan, Jiacheng Guo, Shuai Zhu, Shuo Miao, Mingming Sun, and Ping Li. 2019. MOBIUS: towards the next generation of query-ad matching in baidu's sponsored search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2509–2517.

[9] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. *arXiv preprint arXiv:1905.06482* (2019).

[10] Songwei Ge, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing search results using hierarchical RNN with query-aware attention. In *Proceedings of the 27th ACM International Conference on Information & Knowledge Management*. 347–356.

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

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

[13] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2015. Convolutional neural network architectures for matching natural language sentences. *arXiv preprint arXiv:1503.03244* (2015).

[14] 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 *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.

[15] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using click through data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. 2333–2338.

[16] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[17] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. 2017. Cascade ranking for operational e-commerce search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1557–1565.

[18] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *Proceedings of the 28th ACM International Conference on Information & Knowledge Management*. 2635–2643.

[19] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182* (2017).

[20] Thanh V Nguyen, Nikhil Rao, and Karthik Subbian. 2020. Learning Robust Models for e-Commerce Product Search. *arXiv preprint arXiv:2005.03624* (2020).

[21] 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 *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2876–2885.

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

[23] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[24] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge.

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

[26] Daria Sorokina and Erick Cantu-Paz. 2016. Amazon search: The joy of ranking products. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 459–460.

[27] 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).

[28] Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-srnn: Modeling the recursive matching structure with spatial rnn. *arXiv preprint arXiv:1604.04378* (2016).

[29] Wenjie Wang, Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2020. " Click" Is Not Equal to" Like": Counterfactual Recommendation for Mitigating Clickbait Issue. *arXiv preprint arXiv:2009.09945* (2020).

[30] Tao Wu, Ellie Ka-In Chio, Heng-Tze Cheng, Yu Du, Steffen Rendle, Dima Kuzmin, Ritesh Agarwal, Li Zhang, John Anderson, Sarvjeet Singh, et al. 2020. Zero-Shot Heterogeneous Transfer Learning from Recommender Systems to Cold-Start Search Retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2821–2828.

[31] 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 *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 402–410.

[32] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 55–64.

[33] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 269–277.

[34] 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. *arXiv preprint arXiv:2006.02282* (2020).

[35] Jing Zhang and Dacheng Tao. 2020. Empowering Things with Intelligence: A Survey of the Progress, Challenges, and Opportunities in Artificial Intelligence of Things. *IEEE Internet of Things Journal* (2020).

[36] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5941–5948.

[37] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[38] Justin Zobel and Alistair Moffat. 2006. Inverted files for text search engines. *ACM Computing Surveys (CSUR)* 38, 2 (2006), 6–es.