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

Xiaoming Wu
The Hong Kong Polytechnic University
Hong Kong, China
csxmwu@comp.polyu.edu.hk

Qianli Ma
South China University of Technology
Guangzhou, China
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 the product determines the search system's quality and gradually attracts researchers' attention. Retrieving the most relevant products from the 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 found that EBR system performance is dramatically degraded due to its: (1) low relevance with a given query and 2) discrepancy between the training and inference phase. To this end, we propose a novel and practical embedded product retrieval model, named Multi-Grained Deep Semantic Product Retrieval (MGDSPR). Specifically, we first identify the inconsistency between the training and inference, and use the softmax function as the training objective, achieving better performance and faster convergence. Two efficient methods are further proposed to promote the model relevance, including smoothing noisy training data and generating relevance hard negative samples without requiring extra knowledge and training procedures. We evaluated MGDSPR on Taobao Product Search with significant metrics gains observed in offline experiments and online A/B test. 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.

KEYWORDS

product retrieval system, e-commerce search, deep learning

ACM Reference Format:
Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiaoming Wu, and Qianli Ma. 2021. Embedding-based Product Retrieval in Taobao Search. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Both authors contributed equally to this research.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

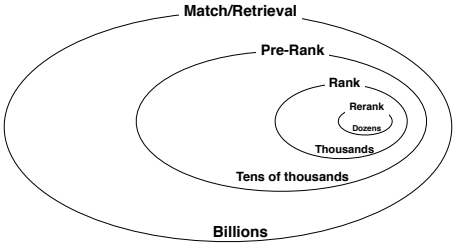

Figure 1: The overview of the product search system in Taobao. The head of each circle denotes different phases. The bottom is the corresponding number of candidate sets.

1 INTRODUCTION

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 [18, 27, 35]. 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 product 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 item candidate sets fed into the follow-up ranking stage. It is gradually receiving more and more attention from academia and industry.

Search in e-commerce retrieval 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 take into account the massive historical user behaviors [2, 3]. The lexical matching engine (typically an inverted index [22, 25, 38]), despite its widely criticized semantic gap issue [15, 25, 32], is still a vital part of the current retrieval system 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 with considering the relationship between the query semantics and historical user behaviors is the main challenge of the e-commerce platform.

With the development of deep learning, Amazon [22] and JD [35] build their two-tower embedding-based retrieval (EBR) system to

provide relevant or personalized product candidate set in their e-commerce search engine. 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 match query terms [11], resulting in increasing user complaints and bad cases that cannot be fixed. To improve its controllability of relevance (*i.e.*, resolve bad cases), we have adopted a relevance control module to filter the recalled products. The control module only keeps those products that met the relevance standards of exact matching signals and fed them to follow-up ranking. However, we statistically found it usually filters out thirty percent of the products. On the one hand, it wastes online computing resources because the recalled products cannot participate in the ranking. On the other hand, it degrades the EBR system's performance due to the low relevance of retrieval products. Therefore, our search system's practical challenge is to enable the embedding-based model to recall more relevant products and increase the amount of participants in the subsequent ranking.

Moreover, random negative samples are widely used to train large-scale deep retrieval models to ensure the sample space used in training is consistent with the inference phase [15, 35]. Nevertheless, there is still a discrepancy in existing e-commerce product search methods [22, 35] due to the inconsistent behavior between the training and inference. Specifically, during inference, the model needs to select the top k products closest to the current query from all candidates, which requires the model to have global comparison ability. Both Nigam et al. [22] and Zhang et al. [35] adapt hinge (pairwise) loss as training objective, which can only provide local ones.

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 problems above, we first use the softmax function as the training objective to make the model possess 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 insufficient relevance noise introduced by using user implicit feedback (*i.e.*, click data) logs as training data [30, 32] by including a temperature parameter to the softmax function. Also, we mix the positive and random negative samples to generate hard relevance 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 the industrial dataset collected from the Taobao search system and online A/B test.

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 behavior and share its online deployment solution.
- We identify the discrepancy between training and inference in existing e-commerce retrieval work and suggest using the softmax function 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 search scenarios of Taobao demonstrated 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 rapid development of deep NLP techniques, various neural search 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 and interaction-based learning. Two tower structure is the typical characteristic of representation based models, such as DSSM [16], CLSM [26], LSTM-RNN [23], and ARC-I [14]. Each side of the tower first uses a siamese/distinct neural network to generate semantic representations of query or document text features. 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 matching patterns between the query and document. Classical models are MatchPyramid [24], Match-SRNN [29], DRMM [12], and K-NRM [33]. Except for 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 Search Deep Retrieval in Industry

Representation-based models in companies with an ANN (approximate near neighbor) algorithm have become the mainstream trend to deploy neural retrieval models in the industry efficiently. For social networks, Facebook developed an EBR system to take text matching and searcher's context into consideration [15]. They introduced various tricks and experiences (*i.e.*, hard mining, ensemble embedding, and inverted index-based ANN) to achieve hybrid retrieval (fuzzy matching). For advertising display, Baidu proposed MOBIUS [8] for CPM (cost per mile) maximization in the web ads retrieval phase, decreasing the objective distinction between ranking and matching. For web search, Google [31] adopt transfer learning to learn semantic embedding from the recommendation system's data 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

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

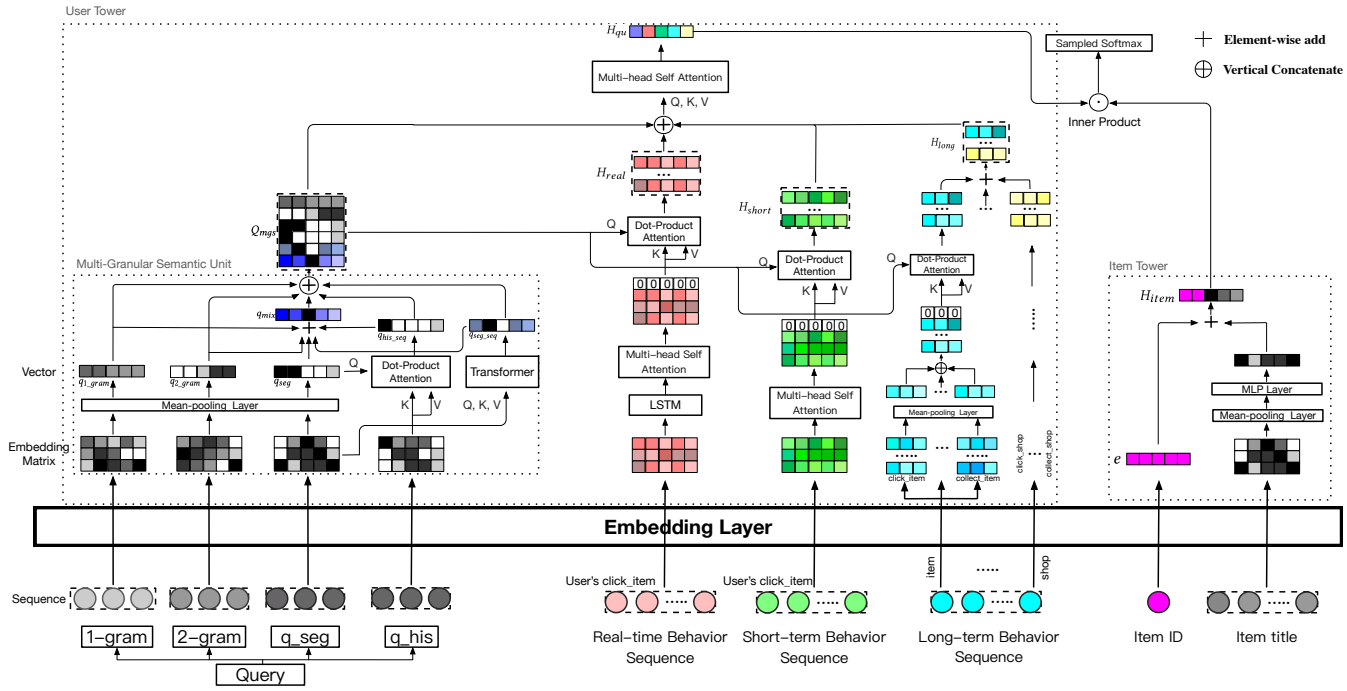

Figure 2: The general architecture of the proposed Multi-Grained Deep Semantic Product Retrieval (MGDSPR).

for product semantic retrieval [22], where one side with n-grams average pooling and the other comprising of items only without considering user personalization, to address the semantic gap issue exist in the lexical match engine. Recently, JD [35] proposed a deep personalized and semantic retrieval model (DPSR) to combine text semantic and user behavior. However, DPSR aggregates user behavior through average pooling, weakening the personalization characteristic. Furthermore, neither of them study the problem of insufficient product relevance caused by the EBR method. This paper will discuss the low relevance 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 and retrieve more products with good relevance. The general structure of MGDSPR is illustrated in Figure 2. We first formulate the problem and then introduce the design of the two-tower model, including the user tower and the item (product) tower. Finally, we elaborate on the model's training objective and methods to recall more relevant products.

3.1 Problem Formulation

We first formulate the e-commerce product retrieval problem and our solution and mathematical notations used in our paper. Let $\mathcal{U} = \{u_1, ..., u_u, ..., u_N\}$ denotes a collection of N users, $\mathcal{Q} = \{q_1, ..., q_u, ..., q_N\}$ denotes the corresponding queries, and $\mathcal{I} = \{i_1, ..., i_i, ..., i_M\}$ denotes a collection of M items (products). Also, according to the time interval from the current time t, we divide the user u's historical behavior into 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.

Based on these preliminaries, we can define our task. Given his/her historical behavior $(\mathcal{R}^u, \mathcal{S}^u, \mathcal{L}^u)$, after the user $u \in \mathcal{U}$ submits query q_u at time t, we would like to return items $i \in \mathcal{I}$ that satisfy his/her. To do that, we typically predict top K item candidates from $\mathcal{I}$ at time t based on the scores z between the user query and its behaviour and items. The mathematical definition is as follows:

$$z = \mathcal{F}(\phi(q, u), \psi(i)) \tag{1}$$

where $\mathcal{F}(\cdot)$, $\phi(\cdot)$, $\psi(\cdot)$ denote the score, query embedding, and item embedding functions, respectively. Here we adapt two-tower retrieval model for maintaining efficiency. We instanced $\mathcal{F}$ with the inner product function and introduce the design of user and item tower, respectively.

3.2 User Tower

3.2.1 Multi-Granular Semantic Unit. Queries in Taobao search are usually composed of Chinese. After query segmentation, the average length of the word segmentation result set is less than three. We propose a multi-granular semantic unit to discover the meaning of queries from multiple semantic granularities and enhance the embedded 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 $Q_{mgs} \in \mathbb{R}^{6 \times d}$ by concatenating its 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 embedded size, the number of word segmentation, and the number of words in each word, respectively. Formally, **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 Transformers [28], average and vertically concat operation, respectively. Eq. (5) denotes we average all the output of the last layer of transformers.

3.2.2 User Behaviors Attention. The user behaviors are recorded by their history of click or buy items. Taking user u's short-term behavior $\mathcal{S}^u$ as an example, each $i_t^u \in \mathcal{S}^u$ denotes that the user clicks on item i at time t, and each item i is described by its ID and side information $\mathcal{F}$ (*i.e.*, leaf category, first-level category, brand and shop) [19]. Specifically, each input item $i_t^u \in \mathcal{S}^u$ is defined as follows:

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

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

where W_f is the feature f's embedded 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 with size d_f and $\cdot$ is the matrix multiplication. Eq. (10) denotes we concatenate its ID and side information embedding in the last axis. Note that we use the same way to embed items in real-time $\mathcal{R}^u$ and long-term $\mathcal{L}^u$ sequence.

Unlike the target-item attention [9, 36, 37] in the advertising and recommendation domains, we here use query attention to capture the personalized behavior representations related to the current query semantics to maintain the user's query intention. Moreover, inspired by [2], we put an all-zero vector into the user behavior data to remove the potential noise in the behavior and deal with situations where the historical behavior may not be related to the current query. 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) [13] to capture the evolution and collect all hidden states $\mathcal{R}_{lstm}^u = \{h_1^u, ..., h_t^u, ..., h_T^u\}$. Then we use multi-head self-attention to aggregate multiple potential points of interest [19] in $\mathcal{R}_{lstm}^u$ to get $\mathcal{R}_{self\_att}^u = \{h_1^u, ..., h_t^u, ..., h_T^u\}$. Next, we add a zero vector at the first position in $\mathcal{R}_{self\_att}^u$, 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 representations $H_{real} \in \mathbb{R}^{6 \times d}$ related to the current query is obtained by the attention operation (regarding the multi-granular semantic representation Q_{mgs} as the Q), which is defined as follows:

$$H_{real} = softmax(Q_{mgs} \cdot R_{zero\_att}^T) \cdot R_{zero\_att}^T \tag{11}$$

For short-term sequences $\mathcal{S}^u = \{i_1^u, ..., i_t^u, ..., i_T^u\}$, we apply multi-head self-attention to aggregate $\mathcal{S}^u$ to get $\mathcal{S}_{self\_att}^u = \{h_1^u, ..., h_t^u, ..., h_T^u\}$. Next, we add a zero vector at the first position in $\mathcal{S}_{self\_att}^u$, resulting in $\mathcal{S}_{zero\_att}^u = \{0, h_1^u, ..., h_t^u, ..., h_T^u\} \in \mathbb{R}^{(T+1) \times d}$. Finally, the short-term personalized representations $H_{short} \in \mathbb{R}^{6 \times d}$ is defined as follows:

$$H_{short} = softmax(Q_{mgs} \cdot S_{zero\_att}^T) \cdot S_{zero\_att}^T \tag{12}$$

Note that the real-time and short-term sequence is composed of click sequences.

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

$$H_{a\_item} = softmax(Q_{mgs} \cdot L_{item}^T) \cdot L_{item}^T \tag{13}$$

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

$$H_{long} = H_{a\_item} + H_{a\_shop} + H_{a\_leaf} + H_{a\_brand} \tag{14}$$

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

3.2.3 Fusion of Semantics and Personalization. To retrieve the products relevant to the current user's query and preserve personalized characteristics, we take 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 in the input $I = \{[CLS], Q_{mgs}, H_{real}, H_{short}, H_{long}\}$ of self-attention and regard its output as user tower's representation H_{qu}. $H_{qu} \in \mathbb{R}^{1 \times d}$ is defined as follows:

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

3.3 Item Tower

For the item tower, we experimentally use the item id and its title to get the embedded representation H_{item}. Given the i-th item's ID embedded representation $e_i \in \mathbb{R}^{1 \times d}$ and its title segmentation

result $T_i = \{w_1^i, ..., w_N^i\}$, the $H_{item} \in \mathbb{R}^{1\times d}$ is calculated as follows:

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

where W_t is the transformation matrix. Note that we experimentally found that applying LSTM [13] or Transformer [28] to capture the sequence dependency of the title is not as effective as mean-pooling simply 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 is consistent with the online inference, Huang et al. [15], Nigam et al. [22], and Zhang et al. [35] use random samples as negative samples. However, they adapt pairwise (hinge) loss as a 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 global comparison ability. However, hinge loss can only provide local ones. Also, hinge loss introduces a cumbersome tuning margin, which has a significant impact on performance [15]. Here we adapt the softmax function as a training objective, achieving faster convergence speed and better performance without additional tuning hyper-parameter.

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 as follows:

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

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

where $\mathcal{F}$, I, and q_u denote the inner product, the full item pool, and the user tower's representation H_{qu}, respectively. Note that Eq. (17) enables model possesses global comparison ability. The softmax involves calculating an expensive partition function, which scales linearly to the number of items. In practice, we use the sampled softmax (an unbiased approximation version of full-softmax) [4, 17] for training. Similar to [35], we also experimentally found that the performance of using the same random negative set for all training examples in the batch is similar to use pure random negatives. The former training method is used to reduce computing resources.

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

3.4.1 Smoothing Noisy Training Data. In e-commerce search, users' click and purchase records are usually used as supervisory signals to train a model. However, this signal is noisy since it is influenced not only by query-product relevance but also by the image, price, and user preference [30, 32]. Therefore, we introduce temperature parameters τ into softmax to smooth the overall fitted distribution of the training data. If τ->0, the fitted distribution is close to the one-hot distribution, which means that the model does its best to fit the supervisory signals. The model will be trained to push the positive items far away from negative ones, even the positive item's relevance 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 training data's noise and thus alleviate the impact of insufficient relevance due to fully fitting users' click records, which do not require additional knowledge and is verified by our experiments. Formally, the softmax with 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 Hard Relevance Negative Samples. Unlike the requirement of additional annotated training data and training process in [21], we propose a method to generate hard relevance negative samples in the embedding representations level. Specifically, given a training example (q_u, i^+, i^-), where i^- denotes random negative item set sampled from item pool I. Note that, for simplicity, we here use q_u, i^+, and i^- to refer to their representations. We first regard the items with the top N among the inner product score of q_u and i^- as 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 relevance the generated sample is. We take the set I_{mix} as negative samples and put it in the denominator of softmax to make the model distinguish the positive sample i^+ and its nearby samples. Formally, the softmax function with hard relevance 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 how difficult it is to generate relevance negative samples. This generation process only needs to add a linear interpolation after calculating the inner product scores between the current query q_u and its sampled negative samples, which is quite efficient.

4 SYSTEM ARCHITECTURE

At a high level, the Taobao search engine, as illustrated in Figure 3, 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 pass through multi-stages of ranking, including pre-ranking, relevance ranking (removing those products that are inconsistent with the query's category prediction results), 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. Now 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.

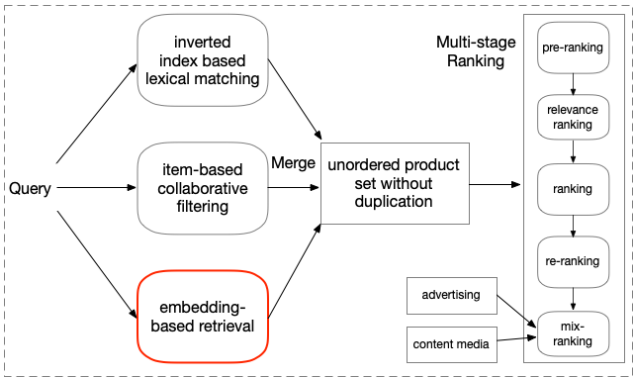

Figure 3: The overview of Taobao search engine.

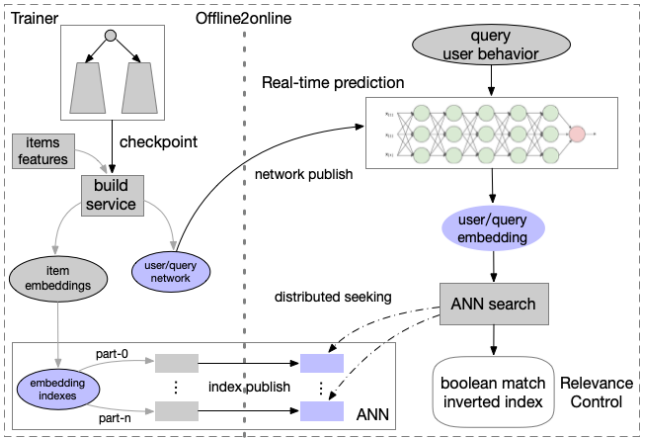

Figure 4: Deployment system of embedding-based retrieval.

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 new parameters daily. Note that we do not use sequential training [34] to do the A/B test fairly. Once 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 data for the new testing model. The deployment system of MGDSPR is an offline to online architecture. At the offline phrase, *build service* optimizes and constructs user/query network extracted from the user tower, which is passed to *real-time prediction* platform. All item embeddings are simultaneously exported from the item tower, transmitted to an approximate near neighbor (ANN) indexing system. The ANN system builds indexes of embeddings through hierarchical cluster algorithm and INT8 quantization to promote storage and searching efficiency. Indexes are placed in multiple columns because of the enormous amounts of items (*i.e.*, one hundred millions or so).

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 behaviors and query features are fed into a real-time prediction platform for online embedding inference. *ANN search module* then distributively seeks top-k KNN results from indexes of multi-columns (referred to n columns). Each column returns the same size of k/n.

4.3 Relevance Control

After a long period of practice, we found that although embedding-based retrieval has advantages in personalization and fuzzy matching, it often leads to more search bad cases due to its lack of exact matching [11] to the key terms of a query. The *key terms of query* are referred to as words of the brand, type, color, etc., which are significant to product search relevance. For instance, a user is searching *Adidas sports shoes*. Items of *Nike sport shoes* are similar to the query in embedding space and are returned in top-k results with high probability. However, this is not the user intent and will decrease user experience. Hence, we add an inverted index based *boolean matching* module on the top of ANN results. The boolean matching aims to filter out items that do not contain key query terms in their titles. The final search results can 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 [15] uses the embedding-based method to enhance boolean expression, achieving fuzzy matching, while we further use boolean matching to improve the EBR system's relevance.

5 EXPERIMENTS

Here we introduce evaluation metrics, implementation details, datasets, and offline and online experimental results, including our method's 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 item clicked or purchased by the user u is 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 $I = \{i_1, ..., i_K\}$.

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

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

In Taobao search, we also pay attention to the relevance of recalled products (related to user experience). Due to a large amount of test data, we use the 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 our

retrieval set I, which is defined as follows:

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

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

Simultaneously, to analyze the effect of the proposed model on each stage of the search system, we also count the number of items in retrieval set I to participate in each ranking stage. Given a retrieval set $I = \{i_1, ..., i_K\}$, every time it goes through a stage (such as relevance system, 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 I that goes through each phase, and use Num_{prank} and Num_{rank} to denote the number of items entering the pre-ranking and ranking. For a total retrieval set $\mathcal{I} = \{I_1, ..., I_i, ..., I_N\}$ of N query, Num_{prank} and Num_{rank}'s calculation are averaged over 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 follows:

$$GMV = \#\text{pay amount} \tag{24}$$

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 in the item set displayed to users, but P_{good} is determined by the relevance model, while humans determine P_{h_good}.

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 [20] 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 adapt gradient clip when its norm exceeded a threshold of 3. The training process converges at about 35 million steps for about 48 hours.

5.3 Datasets

5.3.1 Large-scale Industrial Offline DataSet. We collected search logs of user clicks and purchases for 8 consecutive days from online **Mobile Taobao App** in December 2020, and filter those spam users. The training set comprises samples from the first 7 consecutive days (a total of 4.7 billion records). In the test phase, 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 for evaluation. The size of the candidate item set is consistent with the online environment, *i.e.*, about 100 million.

5.3.2 Online Dataset. We released 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.

5.4 Offline Experimental Results

The previous embedding-based retrieval in our recall system adopts the DNN proposed by [5], but with more user behavior and statistical features, which has been experimentally verified to achieve some success. Specifically, we concatenate the vectors of user behavior (obtained by mean-pooling) and statistical features (*e.g.*, age, sex) and feed it into a multi-layer feed-forward neural network. We here refer to it as a strong baseline α-DNN.

Table 1: Comparisons with Strong Baseline α-DNN on Large-scale Industrial Offline Dataset. Num_{prank} is the number of products that flow into the follow-up ranking phase. P_{good} is the good rate (relative improvements are in parentheses).

Methods	Recall@1000	P_{good}	P_{f_good}	Num_{prank}
a-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%)

5.4.1 Comparison with Strong Baseline. As mentioned in Section 5.1.1, we report the metrics of $recall@K$, good rate, and Num_{prank}. Note that we both report the P_{good} on the retrieval set I (denoted as P_{good}) and the filter set I_{left} (denoted as P_{f_good}). As shown in Table 1, MGDSPR increases by 2.5%, 13.3% and 6.0% on metrics $recll@1000$, P_{good} and P_{f_good} respectively, indicating it can retrieve more products with good relevance and improve the retrieval set's quality. Comparing P_{good} and P_{f_good}, our relevance control can enhance the relevance of any model.

5.4.2 Ablation Study. We studied 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) dynamically fusion of semantics and personalization (*i.e.*, Eq. (15), denoted as *trm*); 3) the temperature parameters τ of softmax (denoted as τ); 4) the hard relevance negative samples (denoted as I_{mix}). Note that we are here focusing 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 semantic unit (*mgs*) and *trm* can improve metrics of $recall@1000$ and P_{good}, indicating the effectiveness of multi-granular semantics and dynamically fusion. The temperature parameter τ and hard relevance negative samples I_{mix} make the model recall more relevant products in terms of much better good rate P_{good}. Comparing MGDSPR+all and MGDSPR (or MGDSPR+*mgs*+*trm*+τ), we observe that there is

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%

a trade-off between recall and relevance even in search scenarios, which may indicate excessive personalization in our system.

5.4.3 Convergence Analysis. We investigate the performance of MGDSPR using softmax and pairwise loss [22, 35] as training targets, respectively. Specifically, we report the test *recall*@1000 score with increasing training steps. As shown in Figure 5, the softmax function's global comparison capability make the training and testing behaviors more consistent, achieving faster convergence and better performance. Specifically, the number of days required for the softmax function (about three days) to converge is half less than the pairwise one (about six days). Note that the margin used in the reported hinge loss results has been carefully tuned.

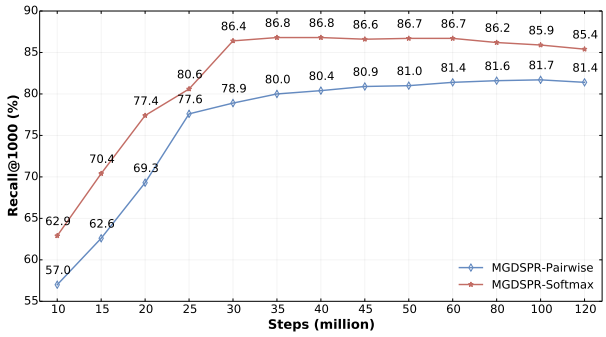

Figure 5: Convergence comparison of softmax and hinge (pairwise) loss. The X-axis denotes the number of training steps, and the Y-axis denotes the corresponding test Recall@1000 score.

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

As shown in Figure 6, 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 fitting users' click records. Remarkably, $\tau = 0.1$ decreases relevance, indicating that the training data does have noise. Also, every non-zero value of N gives better relevance than $N = 0$, so generated hard relevance samples improves good rate P_{good}. Further, the good rate P_{good} reaches its maximum at $N = 684$ and then decreases, indicating that only increases the number of samples cannot bring more benefits.

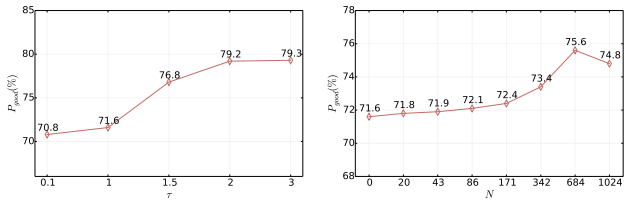

Figure 6: The influence of τ and N on the good rate P_{good}.

Table 3: The improvements on Num_{prank}, Num_{rank}, P_{good}, and P_{h_good} of MGDSPR compared with the previous system deployed on Taobao product search. The last two columns only report the relative value and 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%

5.5 Online A/B Test

We deployed MGDSPR on Taobao Product Search and compared it with the online baseline α-DNN. As mentioned at the first beginning, to improve user experience, our relevance control module (introduced in Section 4.3) will filter out some products retrieved by embedding-based methods, resulting in low utilization of online computing resources. Therefore, apart from **GMV**, $\mathbf{P_{good}}$, and $\mathbf{P_{h\_good}}$, we also report the number of products that have participated in the pre-ranking (denoted as Num_{prank}) and ranking phase (denoted as Num_{rank}) to analyze the model's effect on our search system better.

As shown in Table 3, after being filtered by the relevance control module, the number of products retrieved by MGDSPR that entered the pre-ranking and ranking phase increased by 22.53% and 36.76%, respectively. Obviously, MGDSPR recalled more products with good relevance and effectively improved the utilization of computing resources. Besides, in the set of exposed products, MGDSPR achieves a higher good rate P_{good} and P_{h_good}, providing a more relevant item candidate set. MGDSPR has a positive effect on every stage of the existing search system.

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

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

Finally, we report the 10-day average GMV improvement (removing cheating traffic) achieved by MGDSPR. We also include the corresponding number of transactions (denoted as #Transactions) to increase the results' confidence. As shown in Table 4, MGDSPR achieved 0.77% GMV and 0.33% #Transactions improvement. Considering the billions of transactions per day in Taobao search, an improvement of 0.77% is already tens of millions of transactions, which means that our model better satisfies people.

6 CONCLUSION

This paper proposes a practical embedded product retrieval model, named Multi-Grained Deep Semantic Product Retrieval (MGDSPR), addressing model performance degradation and online computing resource waste due to low relevance in the previous EBR system of the Taobao product search. Meanwhile, we shared the lessons learned from solving these problems, including the model's design and its effect on each stage of the search system, selecting offline metric and test data, and the relevance control of the EBR system. We verify the effectiveness of MGDSPR experimentally. Furthermore, we have released MGDSPR on Taobao product search, one of the largest e-commerce platforms, to serve hundreds of millions of users in real-time. Moreover, we also introduced the search system's used online architecture and our deep retrieval model's deployment scheme to promote the community's development.

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 and 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 and 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 on Conference on Information and Knowledge Management*. New York, NY, USA, 5564. https://doi.org/10.1145/2983323.2983769

[12] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. 55–64.

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

[14] 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).

[15] 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.

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

[17] 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).

[18] 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.

[19] 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 and Knowledge Management*. 2635–2643.

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

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

[22] 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.

[23] 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.

[24] 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.

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

[26] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*. 101–110.

[27] 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.

[28] 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).

[29] 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).

[30] 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).

[31] 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.

[32] Rong Xiao, Jianhui Ji, Baoliang Cui, Haihong Tang, Wenwu Ou, Yanghua Xiao, Jiwei Tan, and Xuan Ju. 2019. Weakly Supervised Co-Training of Query Rewriting andSemantic Matching for e-Commerce. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 402–410.

[33] 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.

[34] 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.

[35] 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-EndSolution for E-commerce Search via Embedding Learning. *arXiv preprint arXiv:2006.02282* (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.