From Semantic Retrieval to Pairwise Ranking: Applying Deep Learning in E-commerce Search

Rui Li, Yunjiang Jiang, Wenyun Yang, Guoyu Tang, Songlin Wang, Chaoyi Ma, Wei He, Xi Xiong, Yun Xiao, Eric Yihong Zhao
JD.com, Mountain View, CA
{rui.li, yunjiang.jiang, wenyun.yang, tangguoyu, wangsonglin3, machaoyi, hewei92, xiongxi, xiaoyun1, ericzhao}@jd.com

ABSTRACT

We introduce deep learning models to the two most important stages in product search at JD.com, one of the largest e-commerce platforms in the world. Specifically, we outline the design of a deep learning system that retrieves semantically relevant items to a query within milliseconds, and a pairwise deep re-ranking system, which learns subtle user preferences. Compared to traditional search systems, the proposed approaches are better at semantic retrieval and personalized ranking, achieving significant improvements.

CCS CONCEPTS

• **Information systems** → **Information retrieval**.

KEYWORDS

Neural networks; Semantic Search; Personalized Ranking

1 INTRODUCTION

Over past decades, online shopping platforms have become ubiquitous in people's daily life. In a typical e-commerce search engine, two major stages are involved to answer a user query, namely **Candidate Retrieval**, which uses inverted indexes to efficiently retrieve candidates based on term matching, and **Candidate Ranking**, which orders those candidates based on factors, such as relevance and predicted conversion ratio. Here we share our experiences and promising results of applying deep learning for both stages in e-commerce search at JD.com.

For candidate retrieval, we introduce the Deep Semantic Retrieval (DSR) system, in addition to traditional term matching based retrieval systems. DSR encodes queries and items into a semantic space based on click logs and human supervision data, and uses an efficient K-nearest neighbor search algorithm to retrieve relevant items from billions of candidates within milliseconds. For candidate ranking, we introduce the Deep Pairwise Ranking (DPR) system to leverage both hand-crafted numerical features and raw sparse features. DPR uses a Siamese architecture to effectively learn preference between a pair of items under the same query from billions of search log data and efficiently score each item individually (without enumerating all the pairs) online.

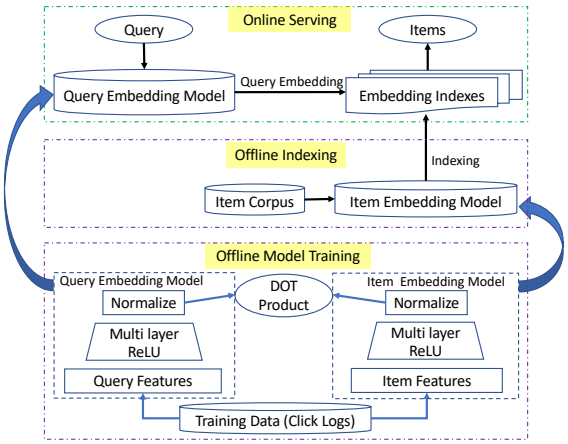

Figure 1: Overview of deep semantic retrieval system.

2 DEEP SEMANTIC RETRIEVAL

The most critical challenge for this stage is enabling semantic retrieval beyond exact term matches. Based on our analysis, this problem impacts around 20% of our search traffic.

System Overview Neural network models naturally address this problem via semantic term embeddings. While many deep neural network models (*e.g.*, DSSM [2]) have been proposed for search relevance ranking, our DSR system explores the rare territory of candidate retrieval. As Figure 1 shows, DSR has three parts.

Offline Training process learns a query, item two tower model. Each tower consists of 1) an input layer that concatenates input features of a query or an item, 2) an encoding ReLU multi-layer, and 3) a normalization layer to ensure L2 normalized output embeddings.

Offline Indexing module computes all the item embeddings from the item corpus, then builds an embedding index offline for efficient online embedding retrieval.

Online Serving module loads the query part of the two tower model and computes an embedding for any query, which then retrieves the K nearest/most similar items from the item embedding index.

Model Overview Now, we give a overview of our proposed model. Given a training set $\mathcal{D} = \{(q_i, s_j^+, s_k^-) \mid i, j, k\}$, where q_i stands for a query, s_j^+ stands for a positive item that is relevant to q_i, and s_k^- stands for a negative item that is irrelevant to q_i, we learn the query and item towers jointly by minimizing the following loss function

$$L(\mathcal{D}) = \sum_{(q_i, s_j^+, s_k^-) \in \mathcal{D}} \max\left(0, \delta - \left(Q(q_i)^T S(s_j^+) - Q(q_i)^T S(s_k^-)\right)\right)$$

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331434

	UCTR	UCVR	GMV	QRR
overall	+1.61%	+0.98%	+0.54%	−4.34%
long tail	+9.7%	+10.03%	+7.5%	−9.99%

Table 1: Online A/B test improvements.

	indexing (sec.)	search (ms)	QPS
CPU	3453	9.92	100
GPU	499	0.74	1422

Table 2: Efficiency for indexing and serving.

query	retrieved item
奶粉 大童 (milk powder big kid)	美赞臣 安儿健A+ 4段 (Enfamil A+ level-4 for 3-6 yr old)
学习自由泳器材 (learn free-style swimming equipment)	英发/yingfa 划臂 (yingfa hand paddle)

Table 3: Good cases from DSR

where $\delta \geq 0$ is the margin used in hinge loss, $Q(\cdot)$ denotes the query tower, $S(\cdot)$ denotes the item tower and the superscript $\cdot^T$ denotes transpose operation. Intuitively, this function employs the hinge loss to encourage the dot product between a query q_i and a positive item s_j^+ to be larger than the product between the query q_i and a negative item s_k^- by a certain margin δ.

Experimental Results Table 1 shows relative improvement of DSR in online A/B tests in terms of four core business metrics, including user click through rate (UCTR), user conversion rate (UCVR), and gross merchandise value (GMV), as well as query rewrite rate (QRR), which is believed to be a good indicator of search satisfaction. Table 2 reports the time consumed by DSR for indexing and searching 120 million items with Nvidia Tesla P40 GPU and Intel 64-core CPU: GPU achieves ~10x speed up. Table 3 shows a few sample queries to better illustrate the power of DSR at bridging semantic gaps between queries and relevant items.

3 DEEP PAIRWISE RANKING

Traditional e-commerce search systems apply gradient boosted trees (xgboost) models [1] to rank products based on predicted conversion rates. Relying mainly on numeric features, those models cannot effectively explore sparse features, such as user's click/purchase history. Deep learning(DL) naturally address this problem as it can leverage those raw features via embeddings. However, to apply DL in ranking, we need to address 1) big computation overhead and 2) sufficient amount of good training data required by DL.

Overview To address those challenges, we first introduce a *re-ranking* stage, which only scores the top K (100) items selected from earlier phases with a complex DL model, to reduce the required computation, as we observe that 90% of clicks/orders are within top 75 positions in existing results. Second, we choose to train a pairwise ranking model with a large amount of noisy logs. While the top 100 results are usually all reasonably relevant, the focus is personalized preference, which is hard to quantify. Logs, though abundant, are only good for approximating theses subtle preferences among presented items. Thus, instead of learning conversion rate, as most e-commerce ranking models do, we learn user preference between a pair of items for a given query.

Model Details In order to effectively learn preference between a pair of items under the same query offline and efficiently score each item individually (without enumerating all the pairs) online, we design a pairwise Siamese model, as shown in Figure 2. The training data consists of triples (q, a, b) where a, b are two items co-occurring under the same session with query q. The Siamese structure has two identical towers (sharing the same parameters) for item a and item b. Each tower takes all the input features from user, query, and item, goes through 3 ReLU layers, and outputs a logit. The difference of the two logits is then combined with the binary label in a cross-entropy loss function. During online serving, only one of the two identical towers is used to score each item.

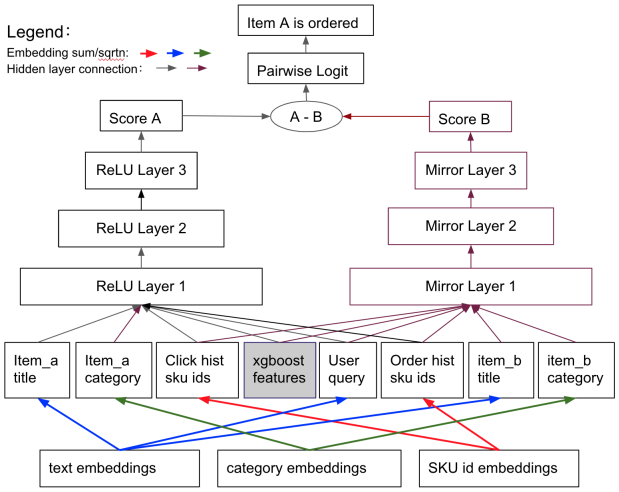

Figure 2: Simplified Siamese ranking model.

	session AUC	order NDCG@5
Xgboost	0.852	0.522
Siamese DL	0.870	0.573

Table 4: Offline validation metrics.

	UCTR	UCVR	GMV	AOP
overall	+1.16%	+1.99%	+1.04%	−2.10%

Table 5: Online A/B test improvements.

The model uses the following features: 1) hand picked numeric features, such as user's purchasing power and past CTR/CVR/sale volume, 2) raw text features for queries and items, tokenized into unigrams and bigrams, and 3) raw user action features, such as historical click streams. The latter two use sum-pooled embeddings.

Experimental Results Our top-line validation metric is order-based average session AUC. Our DPR model, which uses not only all the Xgboost features (about 130 total) but also sparse features, performs significantly better than Xgboost methods (Table 4). Table 5 shows results of online A/B tests. DPR significantly improves all the core business metrics, including UCTR, UCVR, GMV, and average order position (AOP). As DPR uses orders as positive examples during our training, it achieves the most gain on UCVR. However, similar gains on UCTR/GMV are also observed.

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *KDD*. ACM, 785–794.
[2] 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 *CIKM*. 2333–2338.