UniERF: A Uniform Embedding-based Retrieval Framework for E-Commerce Search

Hao Jiang
haojiang@stu.hit.edu.cn
JD.COM
Beijing, China

Xiaoyu He*
hexiaoyu5@jd.com
JD.COM
Beijing, China

Fanyi Qu
qufanyi1@jd.com
JD.COM
Beijing, China

Congcong Liu*
cliubh@connect.ust.hk
JD.COM
Beijing, China

Xue Jiang
jiangxue@jd.com
JD.COM
Beijing, China

Changping Peng
pengchangping@jd.com
JD.COM
Beijing, China

Zhangang Lin
linzhangang@jd.com
JD.COM
Beijing, China

Ching Law
lawching@jd.com
JD.COM
Beijing, China

Jingping Shao
shaojingping@jd.com
JD.COM
Beijing, China

Abstract

E-commerce has become an integral part of daily life, and the ability to effectively retrieve items relevant to a user's query is crucial for enhancing the shopping experience. Embedding-based retrieval (EBR) has proven to be an effective approach in industrial e-commerce search systems. This method involves training models to generate high-quality representations of queries and items, followed by the use of efficient approximate nearest neighbor (ANN) search techniques to find relevant items. However, current EBR methods face several critical limitations: (1) multiple EBR branches often retrieve overlapping or redundant result sets; (2) allocation and utilization of computational resources remain suboptimal, hindering performance; (3) the differentiation modeling of features is somewhat neglected, which restricts the system's ability to retrieve diverse and representative results. These issues hinder the retrieval performance of online search systems. In this paper, we introduce a novel e-commerce search framework called the **Uni**form **E**mbedding-based **R**etrieval **F**ramework (`UniERF`). This framework is meticulously designed to incorporate diverse samples for joint model training, enabling the model to effectively leverage both the semantic information of queries and the personalized features of different users. Extensive offline and online experiments demonstrate that `UniERF` surpasses baseline methods across various evaluation metrics. `UniERF` has been successfully implemented in the existing retrieval system at JD.COM, a renowned online shopping website.

*Corresponding author.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737270

CCS Concepts

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

Keywords

Embedding-based retrieval; E-commerce search; Neural networks

ACM Reference Format:
Hao Jiang, Xiaoyu He, Fanyi Qu, Congcong Liu, Xue Jiang, Changping Peng, Zhangang Lin, Ching Law, and Jingping Shao. 2025. UniERF: A Uniform Embedding-based Retrieval Framework for E-Commerce Search. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3711896.3737270

1 Introduction

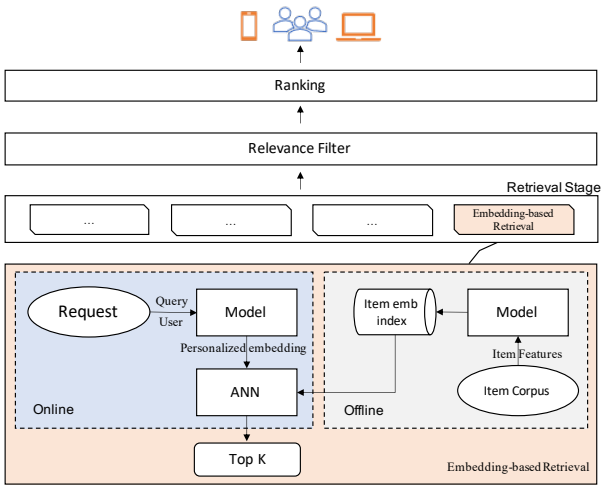


Figure 1: Overview of search advertising system at JD.COM

Online shopping has become an essential part of daily life, with leading e-commerce companies like Taobao, JD, and Amazon serving millions of users who generate browsing and shopping activity

every day [19, 32, 42]. To meet these users' demands, retrieval systems must provide fast feedback while ensuring that the results align with user needs. In search systems, search advertising is crucial not only for satisfying advertisers, but also for delivering results that enhance the user shopping experience. The structure of JD's search advertising system, illustrated in Figure 1, comprises three main processes: multiple retrieval, relevance filtering, and ranking. When faced with a large volume of user requests, the search advertising system initially employs the multiple retrieval module to select a subset of candidate products. The relevance filter module then ensures that these candidates are pertinent to the user's query. Finally, the ranking module processes and orders the filtered collection, presenting the most relevant results to users. As an initial step in the system, the retrieval module plays a critical role in preselecting candidates, effectively narrowing down the product set for further refinement in subsequent stages. Consequently, it has attracted significant attention from both academia and industry.

With the advancement of deep learning [5], many companies are leveraging neural networks to address retrieval challenges. Unlike traditional retrieval methods, embedding-based retrieval (EBR) primarily exploits the neural network's robust representation capabilities to extract deep semantic information from queries and obtain personalized user representations. Consequently, numerous EBR methods [12, 17, 25, 42] have been developed for application in search retrieval. We have also implemented the EBR method within JD's search advertising system, creating distinct branches for personalized retrieval, semantic retrieval, and similar product retrieval to cater to various scenarios and objectives. However, after multiple iterations and updates, we have identified several major challenges associated with this strategy.

- **Insufficient utilization of computing resources** Different branches of EBR often retrieve overlapping sets of candidate products, leading to inefficiencies in computing and memory resource usage.
- **Differentiation modeling ability is insufficient** The relevance between the user's query and the product is a crucial modeling goal shared by all EBR retrieval branches. Additionally, while the modeling objectives of each EBR retrieval branch are generally aligned, they operate independently, which complicates the process of iterative optimization.
- **Locally optimal of each retrieval branch** Currently, independent training of each branch results in local optimization for each branch. Due to the lack of inter-branch awareness, optimizing one retrieval branch often results in a decline in the performance of others.

These challenges have limited the overall retrieval performance in online search advertising systems. To address these issues, we introduce joint multiple scenario training into EBR models to retrieve diverse products. The existing EBR methods in search systems fall mainly into three categories: (1) constructing EBR models based solely on user information to capture personalized features during product retrieval [44]; (2) focusing on query semantics to build EBR models that capture user query information [25]; and (3) combining text semantics and user behavior to propose deep semantic and personalized retrieval models [17, 42]. These methods face challenges such as training models with random or batch negative samples, which do not adequately address different retrieval scenarios. Current mainstream multi-channel retrieval strategies involve training multiple retrieval models from scratch, each based on distinct insights, and subsequently merging their candidate sets. However, these models operate independently, unaware of each other, which limits the diversity of results. Additionally, in search scenarios, certain features (e.g., queries) are redundantly inferred by multiple retrieval models, leading to inefficiencies.

In this paper, we propose a novel e-commerce search framework called the **Uni**form **E**mbedding-based **R**etrieval **F**ramework (`UniERF`). Within this framework, we designed a **differentiation-sample construction** strategy to address various scenarios and task objectives, allowing the retrieval of more diverse results across different search branches. As a data-centric task, the quality of training samples determines the retrieval performance of models [40, 41]. Using online log data, we generated multiple different positive and negative samples customized for each retrieval scenario in `UniERF`. We conducted extensive offline and online experiments to verify the retrieval performance. The experimental results demonstrate that `UniERF` significantly outperforms baseline models across all offline metrics and online performance measures. We have successfully deployed `UniERF` in the JD online search advertising system.

The main contributions of this work are summarized as follows:

- We propose a novel uniform EBR framework `UniERF` for e-commerce search, designed for the joint training of multiple retrieval modules to enhance the utilization of computing resources in online search advertising systems.
- We developed a differentiation-sample construction strategy and a diversity loss function for training within `UniERF` to enhance the distinctiveness of different retrieval branches. This approach improves both the diversity and quality of the retrieval candidate product collection.
- Under the `UniERF`, we propose a general retrieval optimization strategy and model optimization objectives.
- Experiments conducted on a large-scale industrial dataset and online A/B test demonstrate the effectiveness of `UniERF`.

2 Related Works

2.1 Deep Relevance Matching

With the rapid advancement of deep learning, various neural network-based methods have been proposed to enhance the retrieval of relevant information in response to user queries. Leveraging the powerful representational capabilities of neural networks, deep relevance matching models (DRRMs) improve the relevance of search results by matching query and retrieval result representations at a deeper and more fine-grained level, which traditional information retrieval (IR) methods cannot achieve [8, 14, 23]. Specifically, the classic deep structured semantic model (DSSM) introduced a two-tower model structure to map user queries and documents into a common semantic space, using cosine similarity to retrieve relevant results [13]. This efficient two-tower retrieval model is widely employed in designing models for IR tasks [11, 27, 30, 31, 37]. Additionally, some works have attempted to integrate traditional IR lexical matching signals (e.g., exact matching, query word importance) into neural networks to develop new methods [8, 24].

2.2 Embedding Based Retrieval

Recently, EBR methods have gained increasing attention and have been applied to industrial search, recommendation, and advertising tasks. These methods not only ensure the relevance of retrieval results but also take into account users' personalized characteristics to enhance the user experience during searches. EBR methods rely on a nearest neighbor search engine, which identifies the Top-K nearest neighbors in a candidate corpus. In this framework, only query embeddings need to be computed in real-time, while product or document embeddings can be precomputed offline. An Approximate Nearest Neighbor (ANN) index is then constructed for the embeddings of the corpus [15, 16, 18, 22].

Many companies in the industry have adopted EBR methods for their search systems. Facebook, for instance, formalized the search task as a recall optimization problem and designed hard negative augmented samples to aid model training, deploying this EBR system for online social network retrieval [12]. Google addressed video retrieval as a multi-category problem [39] and introduced Mixed Negative Sampling (MNS) to enhance model generalization for web search [38]. Baidu integrated traditional CTR prediction models into the retrieval layer and introduced MOBIUS [6]. To tackle SSB and long-tail issues in EBR models, they explored various sample optimization strategies, which improved the performance of online display advertising [7]. Huawei introduced a cross-batch negative sampling strategy [36]. For e-commerce retrieval, JD added personalized features based on semantic relevance and designed a method that balances personalization and relevance [42], developing a more sophisticated sample construction strategy to enhance retrieval performance [35]. Alibaba proposed MGDSPR [17] to dynamically capture the relationship between user search query semantics and personalized interaction history, considering both semantics and personalization. Additionally, they introduced MOPPR [44] to optimize sample construction strategies and model objectives.

3 Problem Formulation

We can formulate the retrieval problem of the e-commerce product as follows. Let R represents any requests, an associated set Z exists in the product material library S, $\hat{Z}$ denotes the collection retrieved by the system through EBR methods. Suppose the EBR method is imperfect (i.e., there is an R that makes $Z - \hat{Z} \neq \emptyset$), we want to optimize the following goal:

$$\max_{R} \mathbb{E} |Z \cap \hat{Z}| \tag{1}$$

The optimization goal of retrieval is how to make the candidate collection $\hat{Z}$ contain more elements of Z.

Next, we give a formal definition of the EBR method. Let $\mathcal{U} = \{u_1, u_2, ..., u_N\}$ denote the collection of user features for N users, $\mathcal{Q} = \{q_1, q_2, ..., q_N\}$ denote the user queries, and $\mathcal{I} = \{i_1, i_2, ..., i_M\}$ denote a collection of different M products. Let $\mathcal{G} = \{g_1, g_2, ..., g_N\}$ denote trigger products (i.e. the items exposed on the previous page for similar retrieval), $\mathcal{B} = \{b_1, b_2, ..., b_N\}$ denote users' historical behaviors according to the time interval from the current time t. The user history behavior consists of a sequence of historical interaction items and historical search queries, where $b = (q'_1, \ldots, q'_T, i'_1, \ldots, i'_T)$ and T is the length of the behavior sequence. Given the search request $r \in R$, we can obtain a query $q \in \mathcal{Q}$, the trigger product $g \in \mathcal{G}$, the user information $u \in \mathcal{U}$ and his/her history behavior sequence $b \in \mathcal{B}$, we would like the model to retrieve a set of candidate items $C \subset \mathcal{I}$ that satisfy user demand. Specifically, the EBR method predicts a candidate collection with top-K items from $\mathcal{I}$ based on the scores y with user(query, historical behaviors), trigger products and items as follows:

$$y = \mathcal{T}(\mathcal{E}_U(u, b, q, g; \theta), \mathcal{E}_I(i; \theta)) \tag{2}$$

where $\mathcal{E}_U$, $\mathcal{E}_I$, and θ denote user-query tower, item tower, and model parameters, respectively. Let $\mathcal{T}$ denote the scoring function (e.g. cosine similarity), which can be used as the basis for subsequent ANN retrieval of Top-K candidate products.

4 Method

In this section, we present our uniform EBR model `UniERF`, designed to accommodate various retrieval scenarios. The overall structure of `UniERF` is depicted in Figure 2, featuring a two-tower architecture. We provide a detailed introduction to `UniERF` as follows: (1) Section 4.1 covers the implementation details of the user-query tower. (2) Section 4.2 outlines the implementation specifics of the item tower. (3) Section 4.3 delves into the strategy for constructing training samples. (4) Section 4.4 discusses the loss function used in the `UniERF` training process.

4.1 User-Query Tower

The user-query tower is primarily composed of three modules: the personalized retrieval module, the semantic retrieval module, and the similar item retrieval module. To generate high-quality user-query representations, three foundational expert models are deployed within this tower: the query-expert model ϕ_{q}, the user-expert model ϕ_{u}, and the trigger item-expert model ϕ_{trig}. Initially, we pre-trained these expert models using user search logs from JD.com as the training data. To enhance the inference efficiency of the online retrieval system and improve model generalization, we distilled the parameters from the pre-trained expert models into the foundational expert model ϕ [10].

4.1.1 Semantic retrieval module. The semantic retrieval module is designed to model the retrieval based on semantic relationship. This module uses only the query text as input features to capture deep semantic information. For a given set of queries, we utilize the query-expert model ϕ_{q} to encode the user query.

$$e_q = \phi_{\text{q}}(q; \theta) \tag{3}$$

where $e_q \in \mathbb{R}^{1 \times d_q}$ is the embedding vector of size d_q of query. Then e_q is inputted to semantic representation aggregation network to obtain deep semantic embedding H^{sem}.

$$f = \text{ReLU}(\text{LayerNorm}(Wx + b)) \tag{4}$$

$$H^{\text{sem}} = l2_{norm}(f(e_q)) \tag{5}$$

where $H^{\text{sem}} \in \mathbb{R}^{1 \times d}$ denotes semantic embedding vector of size d.

4.1.2 Personalized retrieval module. Personalized retrieval module is designed to model personalized features and retrieve items that match the user's interests. The insight of personalized retrieval module is to obtain a representation that matches the users' personalized preferences and the users' query search intent. For a given set of queries $\{q_1, q_2, ..., q_n\} \subseteq \mathcal{Q}$, the corresponding user personalized

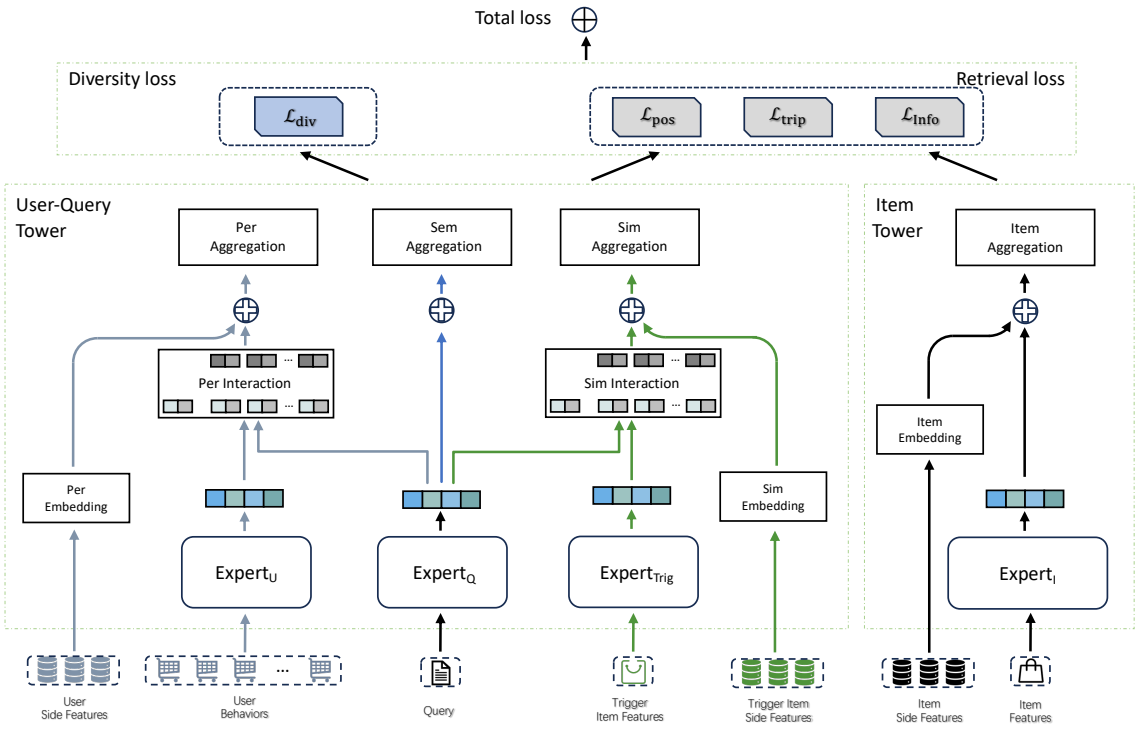


Figure 2: Overall framework of the proposed UniERF. a). For simplicity, we use *per*, *sem*, and *sim* to represent the personalized retrieval module, semantic retrieval module and similar item retrieval module, respectively. b). We use Expert_{U}, Expert_{Q} and $\text{Expert}_{\text{Trig}}$ to denote user expert model ϕ_{u}, query expert model ϕ_{q} and trigger item expert model ϕ_{trig}. c). For additional clarity and emphasis, we use different colors to represent different retrieval branches, blue grey for personalized retrieval, blue for semantic retrieval, green for similar item retrieval, and black for shared module.

feature is $\{u_1, u_2, ..., u_n\} \subseteq \mathcal{U}$. The corresponding user historical behavior is $\{b_1, b_2, \ldots, b_n\} \subseteq \mathcal{B}$, where $b_j = \{i_{j1}, \ldots, i_{jt}, a_{j1}, \ldots, a_{jt}, q_{j1}, \ldots, q_{jt}\}$, for $j = 1, 2, \ldots, n$ and t denotes the length of user historical behavior. Specifically, i_j represents the historical product with which the user interacts, q_j represents the historical query that the user has searched, and a_j denotes the type of user's interaction with the product, such as browsing, clicking or buying. We rank all historical user behaviors in chronological order and use them as input to the user-expert model ϕ_{u}. Then we can obtain the embedding vector as follows

$$e_u = \varphi_{\text{u}}(u; \theta); \quad e_b = \phi_{\text{u}}(b; \theta) \tag{6}$$

where e_u and e_b represent the embedding vector of user personalized and user historical behavior feature, respectively. And φ_{u} and ϕ_{u} represent the embedding layers for user personalized features and the expert model of user historical behaviors, respectively. To capture the user's personalized preferences while aligning with the user's current search query intent, we designed a personalized interaction network $\mathcal{E}_{\text{per}}$, where the query is treated as Q, and the user's historical actions serve as K and V. This setup enables the model to extract features that align with the user's current search intent. The implementation details are shown in Figure 3.

$$e_{\text{inter}} = \mathcal{E}_{\text{per}}(q, b; \theta) \tag{7}$$

where e_{inter} represents user search intent interaction features captured from the historical search queries and the historical interaction items. To better combine user characteristics with search query intent, we designed multi-head attention fusion function instead of using concatenate function in personalized aggregation network.

$$E = \begin{bmatrix} e_u \\ e_b \\ e_q \\ e_{\text{inter}} \end{bmatrix} \in \mathbb{R}^{4\times d} \tag{8}$$

$$Q_i = EW_i^Q, \quad K_i = EW_i^K, \quad V_i = EW_i^V \tag{9}$$

$$\alpha_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) \tag{10}$$

$$\text{head}_i = \alpha_i V_i \tag{11}$$

$$e_{\text{fusion}} = W^O \left[\text{head}_1 \| \text{head}_2 \| \ldots \| \text{head}_h\right] \tag{12}$$

where e_{fusion} represents an embedding vector that combines query intent, user personalized features, and historical behavior features. Then e_{fusion} is inputted to personalized representation network to obtain personalized embedding.

$$H^{\text{per}} = l2_{norm}(f(e_{\text{fusion}})) \tag{13}$$

where $H^{\text{per}} \in \mathbb{R}^{1\times d}$ represents personalized embedding of size d.

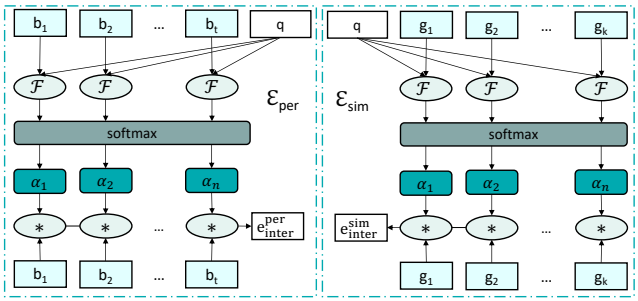


Figure 3: The detail of personalized and similar item interaction network, the former focuses on the sequence of user queries and historical behaviors, while the latter explores the relationships between queries and similar items.

4.1.3 Similar item retrieval module. Similar item retrieval module is designed for modeling item-to-item retrieval, which is suitable for commodity oriented application scenarios. We found that current EBR methods did not pay attention to modeling explicitly the relationship between item and other similar items, which is important in some special online scenarios like product orientation. For a given set of queries $\{q_1, q_2, ..., q_n\} \subseteq \mathcal{Q}$. We can obtain the corresponding trigger items $\{g_1, g_2, \ldots, g_n\} \subseteq \mathcal{G}$ from the products exposed on the previous page, where $g_j = (i_{j1}, i_{j2}, \ldots, i_{jk})$, for $j = 1, 2, \ldots, n$ and k denote the number of trigger items. Specifically, i_j includes the textual features and numeric features of trigger item.

$$e^{\text{sim}}_{\text{num}} = \text{mean\_pooling}(\varphi_{\text{trig}}(g;\theta)) \tag{14}$$

$$e^{\text{sim}}_{\text{text}} = \text{mean\_pooling}(\phi_{\text{trig}}(g;\theta)) \tag{15}$$

where $e^{\text{sim}}_{\text{num}}$ and $e^{\text{sim}}_{\text{text}}$ represent the numerical and textual feature embedding vectors of the trigger item, respectively. Additionally, we designed a similar item interaction network $\mathcal{E}_{\text{sim}}$ to capture the relationship between search query and trigger items.

$$e^{\text{sim}}_{\text{inter}} = \mathcal{E}_{\text{sim}}(q, g; \theta) \tag{16}$$

$$H^{\text{sim}} = l2_{norm}(f(\text{concat}(e_q, e^{\text{sim}}_{\text{inter}}, e^{\text{sim}}_{\text{num}}, e^{\text{sim}}_{\text{text}})) \tag{17}$$

where $H^{\text{sim}} \in \mathbb{R}^{1\times d}$ represents similar item embedding of size d.

4.2 Item Tower

In the item tower, we use item-related textual features and numeric features to obtain the item representation, H^{item}. To ensure that these items and trigger items are in the same latent space, we distill ϕ_{trig} and ϕ_{item} from the same pre-trained item-expert model.

$$e_{\text{num}} = \varphi_{\text{item}}(i;\theta); \quad e_{\text{text}} = \phi_{\text{item}}(i;\theta) \tag{18}$$

Then, the final item representation is defined as:

$$H^{\text{item}} = l2_{norm}(f(\text{concat}(e_{\text{num}}, e_{\text{text}})) \tag{19}$$

4.3 Differentiation Sample Construction Strategy

The selection of training samples directly influences the retrieval performance of the EBR methods. To enhance the retrieval diversity of our model, we developed a specific sample construction strategy for UniERF based on online logs. In practice, we constructed multiple positive and negative samples tailored to different scenarios and tasks. Let $\mathcal{S}$ denote a training batch; the construction strategy for positive and negative samples is as follows.

4.3.1 Multiple Positives. In each sample, the positive item is typically the one that the user has clicked on or purchased. When optimizing for the relevance between the query and the item, items that were exposed but not clicked can also serve as positive items for the semantic retrieval module.

$$\mathcal{P} = \{(x_i, x_i^+) \mid x_i \in \mathcal{S}\} \tag{20}$$

4.3.2 Batch Negatives and Random Negatives. As traditional negative construction strategy, batch negative sampling can make full use of each sample in each batch $\mathcal{S}$ and increase training efficiency. We collect batch negative set $\mathcal{N}_i^{\text{batch}}$ as follows.

$$\forall x_i \in \mathcal{S}, \quad \mathcal{N}_i^{\text{batch}} = \{x_j^- \mid x_j^- \in \mathcal{S}, j \neq i\} \tag{21}$$

For random negatives, given a candidate negative item corpus of size (N), we sample a random integer j from a uniform distribution $j \sim \text{Uniform}(1, N)$, and select the j-th element from the item set as a random negative. To enhance the model's ability to discriminate between positives and negatives, we also collect a subset of negatives that belong to the same category as the positive item, treating these as more challenging random negatives.

$$\mathcal{N}_i^{\text{rand}} = \{x_{j_1}^-, x_{j_2}^-, \ldots, x_{j_k}^- \mid x_{j_1}^-, x_{j_2}^-, \ldots, x_{j_k}^- \sim \text{Uniform}(\mathcal{I})\} \tag{22}$$

4.3.3 Hard Negatives. To enhance the diversity of the retrieval results and differentiate the various retrieval modules in UniERF, we designed distinct hard negative items for each module. In practice, we found that batch negatives and random negatives often serve as simple negatives. These typically exhibit clear categorical or relevance differences from positive samples, making it straightforward for the model to learn the boundary between negative and positive samples. Beyond these simple negatives, we introduced a subset of hard negatives to improve the model's generalization and discriminative capabilities. For the personalized retrieval module, we identified items with low scores during the post-link partial sorting phase as hard negatives, which can be denoted as $\mathcal{N}_i^{\text{per}^-}$.

$$\mathcal{N}_i^{\text{per}} = \{x_j^- \mid \mathcal{F}_{\text{ranking}}(q, x_j^-; \theta) < \epsilon, \quad x_j^- \in \mathcal{I}\} \tag{23}$$

where $\mathcal{F}_{\text{ranking}}$ denotes the post-link ranking model. These items can help improve the modeling of user's personalized features in the process of model training. For the deep semantic retrieval module, we evaluated the relevance scores between query and item title. Then selected those items with low relevance score as hard negatives, which can be denoted as follows.

$$\mathcal{N}_i^{\text{sem}} = \{x_j^- \mid \mathcal{T}_{\text{rel}}(q, x_j^-) < \tau, \quad x_j^- \in \mathcal{I}\} \tag{24}$$

For similar item retrieval module, we constructed specialized positive item to meet the demanding of modeling item-to-item target. Specifically, we extracted the image and title features of different items. Then leveraged it as similarity measure, the items with low similarity are taken as similar negative items.

$$\mathcal{N}_i^{\text{sim}} = \{x_j^- \mid \mathcal{F}_{\text{sim}}(x_i^{\text{img}+}, x_i^{\text{title}+}, x_j^{\text{img}-}, x_j^{\text{title}-}) < \lambda, \quad x_j^- \in \mathcal{I}\} \tag{25}$$

where $\mathcal{F}_{sim}$ denotes a multimodal model that combines images to screen out similar items based on their attributes and appearance.

Then, we can obtain the complete negative set $\mathcal{N}$ as follows

$$\mathcal{N}_i = \mathcal{N}_i^{\text{batch}} \cup \mathcal{N}_i^{\text{rand}} \cup \mathcal{N}_i^{\text{per}} \cup \mathcal{N}_i^{\text{sem}} \cup \mathcal{N}_i^{\text{sim}} \tag{26}$$

4.4 Optimization Objective

We designed multiple optimization objectives to meet different scenario demands. Specifically, the basic optimization objectives contained three parts, including the positive loss function, the triplet loss function [28] and InfoNCE loss function [26]. The positive loss function was used to encourage the similarity score between positive samples and the query exceeds a specified threshold β.

$$e_{\text{u-q}} = \mathcal{E}_U(u, b, q, c; \theta) \tag{27}$$

$$\mathcal{L}_{\text{pos}} = \max\left\{\beta - \mathcal{T}\left[e_{\text{u-q}}, \mathcal{E}_I(i^+; \theta)\right], 0\right\} \tag{28}$$

where $e_{\text{u-q}}$ denotes the output embedding vector of user-query tower $\mathcal{E}_U$. In our practice, we set clicked and exposed but not clicked items as positives to model relevance objective.

$$\mathcal{L}_{\text{trip}} = \max\left\{\mathcal{T}\left[e_{\text{u-q}}, \mathcal{E}_I(i^-; \theta)\right] - \mathcal{T}\left[e_{\text{u-q}}, \mathcal{E}_I(i^+; \theta)\right] + \gamma, 0\right\} \tag{29}$$

where γ denotes the margin.

$$\mathcal{L}_{\text{Info}} = -\frac{1}{N}\sum_{k=1}^{N} \log\left[\frac{\exp(\frac{\mathcal{T}(e_{\text{u-q}}, \mathcal{E}_I(i^+;\theta))}{\tau})}{\sum_{j=1}^{N}\exp(\frac{\mathcal{T}(e_{\text{u-q}}, \mathcal{E}_I(i^-;\theta))}{\tau})}\right] \tag{30}$$

For the triplet loss and InfoNCE loss functions, we treat clicked items and exposed but unclicked items as positives, with $\mathcal{N}$ as the negative sample, to model the relevance optimization objective. For the CTR optimization objective, we consider clicked items as positives and exposed but unclicked items as negatives. The key insight of this strategy is to ensure that the clicked item receives a higher score than the exposed unclicked item, i.e., $y_{\text{click}} > y_{\text{unclicked}} > y_{\mathcal{N}}$. Unlike other EBR methods that focus solely on relevance as the modeling goal, our approach incorporates CTR as a modeling goal, ensuring the consistency of search links and meeting the specific requirements of search advertisements.

Furthermore, we proposed diversity optimization objective to increase diversity of different retrieval branches.

$$\mathcal{L}_{\text{diversity}} = \log\left[1 + \sum_{(i,k)\ ret_i \neq ret_k} \exp\left[\tau \cdot \eta_n \cdot (s_{ik} - \Delta)\right]\right] \tag{31}$$

where s_{ik} represents the similarity between retrieval branch i and retrieval branch k, Δ denotes the similarity margin, and τ is the scaling factor. We define $\eta_n = [s_{ik} - \alpha]_+$, where $[]_+$ denotes the clamp operation and α serves as the truncation threshold. When s_{ik} is less than α, the gradient is truncated. The rationale behind this design is to prevent excessively low similarity between different retrieval branches. Empirically, we believe that representations of different retrieval branches should maintain a certain degree of similarity, since they share the query as an input feature. The final optimization objective can then be expressed as follows

$$\mathcal{L} = \mathcal{L}_{\text{pos}} + \mathcal{L}_{\text{trip}} + \mathcal{L}_{\text{Info}} + \mathcal{L}_{\text{diversity}} \tag{32}$$

4.5 Theoretical Analysis

In this section, we discuss the feasibility of UniERF. For the joint training process of different retrieval branches, we can regard it as a multi-task learning (MTL) process.

Initially, given a training set $\mathcal{D}_{\text{train}}$, a training example $\mathbf{z} = (\mathbf{x}_j, \mathbf{y}_j^+, \mathbf{y}_j^-)$ is defined as a group of a data point $\mathbf{x}_j$ and corresponding positive and negative sample $\mathbf{y}_j^+$ and $\mathbf{y}_j^-$. Let $\mathcal{D}_{\text{train}}^{\setminus j}$ denotes a new training set by removing the jth element. The uniform stability and generalization error of the model are given as follows [1].

Definition 1 (Uniform Stability) *A model $\mathcal{F}$ has uniform stability β with respect to the loss function $\mathcal{L}$ if the following holds*

$$\forall j \in \{1, ...n\},\ \mathbb{E}_{\mathcal{D}_{\text{train}}}[\|(\mathcal{F}_{\mathcal{D}_{\text{train}}}, \mathbf{z}) - (\mathcal{F}_{\mathcal{D}_{\text{train}}^{\setminus j}}, \mathbf{z})\|] \leq \beta \tag{33}$$

The generalization error of model $\mathcal{F}$ which is trained on $\mathcal{D}_{\text{train}}$ is

$$R(\mathcal{F}, \mathcal{D}_{\text{train}}) = \sum_{i=1}^{m} \mathbb{E}[\mathcal{L}(\mathcal{F}_{\mathcal{D}_{\text{train}}}, \mathbf{z}_j)] \tag{34}$$

where $\mathbb{E}[\cdot]$ denotes the expectation operator. The empirical risk is

$$R_{\text{emp}}(\mathcal{F}, \mathcal{D}_{\text{train}}) = \sum_{i=1}^{m} \frac{1}{n_i} \sum_{j=1}^{n_i} \mathcal{L}(\mathcal{F}_{\mathcal{D}_{\text{train}}}, \mathbf{z}_j) \tag{35}$$

Lemma 1 *Let $\mathcal{F}$ be a model with uniform stability β with respect to a loss function such that $0 \leq \mathcal{L}(\mathcal{F}_{\mathcal{D}_{train}}, \mathbf{z}) \leq M$ for all $\mathbf{z}$ and $\mathcal{D}_{train}$. Then, for any $n \geq 1$ and any $\delta \in (0, 1)$, the following bounds hold with probability at least $1 - \delta$ over the random draw of the sample.*

$$R(\mathcal{F}, \mathcal{D}_{\text{train}}) \leq R_{\text{emp}}(\mathcal{F}, \mathcal{D}_{\text{train}}) + 2\beta + (4n\beta + M)\sqrt{\frac{ln(1/\delta)}{2n}} \tag{36}$$

Then, we can prove the following generalization bound of UniERF. The proving progress can be found in Appendix A.1.

Theorem 1 Let m be number of retrieval branches, the generalization error bound of UniERF based on shared expert models is lower than training retrieval branches independently.

$$R^{\text{Uni}}(\mathcal{F}, \mathcal{D}_{\text{train}}) \leq \sum_{i=1}^{m} R_i^{Ret}(\mathcal{F}_i, \mathcal{D}_{\text{train}}) \tag{37}$$

Based on Theorem 1, we can see that the UniERF can be helpful in improving the performance of retrieval task compared to the previous retrieval with multiple branches.

5 Experiments

In this section, we first introduce online deployment in Section 5.1 and evaluation metrics in Section 5.2. Then we report the implementation details of our UniERF in Section 5.3, including hyperparameters settings and model architecture. Next, we introduce the details of datasets in Section 5.4. Furthermore, we report the experiment results of offline experiment and online A/B test in Section 5.5 and Section 5.6, respectively.

5.1 Online Deployment

As illustrated in Figure 4, the deployment system of UniERF is an offline to online architecture. At the online phrase, the request is retrieved in different segments and the results are obtained. To further ensure the semantic relevance of the retrieved results, we

Table 1: Comparison of UniERF to the strong baseline Adapted-DNN based on a large-scale industrial offline dataset.

Methods	$\text{Precision}_{\text{click}}$	Recall@1	Recall@100	Recall@1000
Adapted-DNN	0.439	0.018	0.468	0.644
UniERF(OURS)	0.585	0.036	0.552	0.772

apply a relevance model to filter the recall results. More details about online system can be found in Appendix A.4.

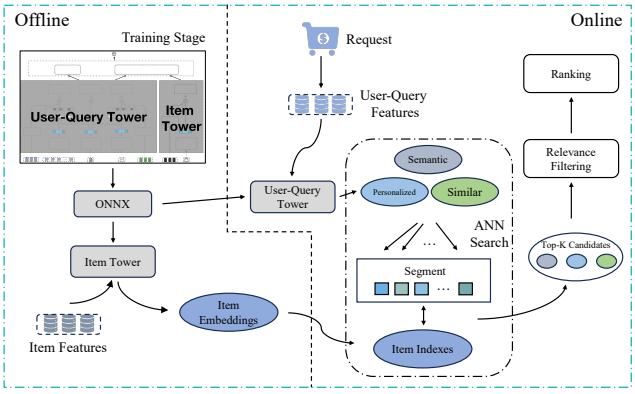


Figure 4: The online deployment system for our UniERF model. (To maintain clarity and conciseness, some details have been omitted.)

5.2 Evaluation Metrics

5.2.1 Offline Metrics. We utilize the Recall@K metric to assess the offline performance of various methods. Specifically, for a given query q, the relevant items are considered as the target set T. The model retrieves the *top-K* items i from the product material library. Recall@K is defined as follows

$$Recall@K = \frac{\sum_{j=1}^{K} i_j \in T}{N} \tag{38}$$

In practice, unlike the search scenario where relevance and GMV (Gross Merchandise Volume) are the primary objectives for model optimization, search advertising must also consider metrics such as CTR, CPC, and Conversion Rate. In the retrieval phase of JD search advertising scenario, we focus not only on the relevance between retrieved products and the query but also on enhancing click and exposure rates. To assess the impact of our model's recall results on CTR, we use the metric $\text{Precision}_{\text{click}}$ to evaluate offline performance. $\text{Precision}_{\text{click}}$ considers clicked items as positive samples and is defined as follows

$$\text{Precision}_{\text{click}} = \frac{1}{N}\sum_{j=1}^{N} \mathbb{I}(y_i^{\text{click}} > y_j^{\text{unclick}}) \tag{39}$$

where y denotes the score of samples and $\mathbb{I}$ denotes a indicator function that takes the value 1 when $y_i^{\text{click}} > y_j^{\text{unclick}}$ and 0 otherwise.

5.2.2 Online Metrics. In the search advertising scenario, we focus on three key metrics: **IMP**, **CLK** and **COST**. IMP (Impressions) indicates the total number of times an advertisement is displayed. In our online system, retrieval results are filtered through a relevance model for downstream tasks, so the IMP metric directly reflects the quality of retrieval results. A high number of impressions suggests better exposure for the advertisement. CLK (Click Times of Advertising) measures the ad's ability to attract user clicks, with a higher CLK generally indicating that the ad content is highly relevant to the user's search intent. COST (Actual Cost Through Click) assesses the cost-effectiveness of advertising campaigns.

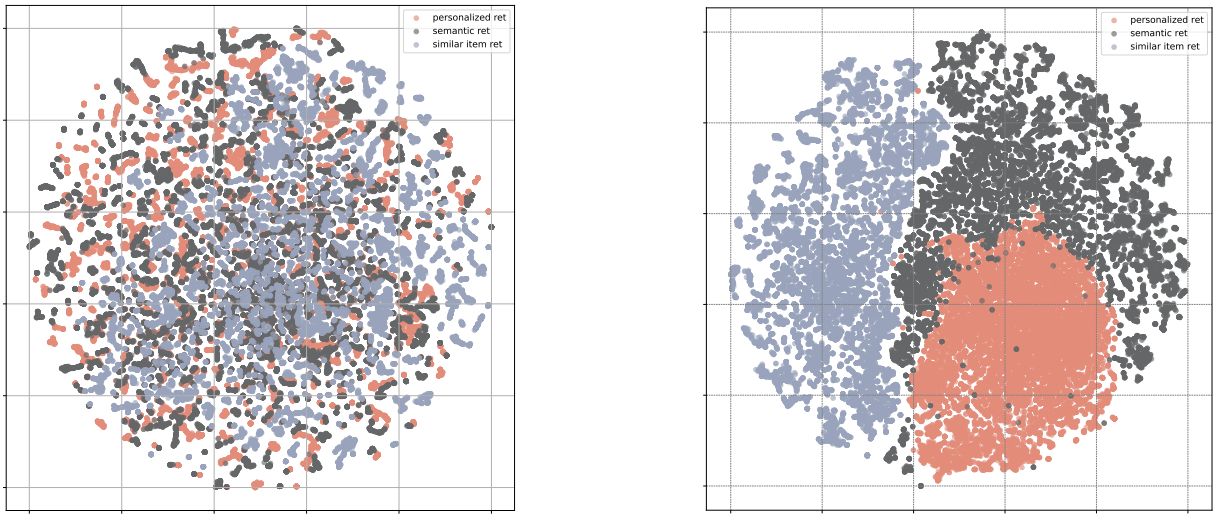
5.3 Implementation Details

5.3.1 Model Architecture. For the model architecture, we first introduce the design of the expert model. Our UniERF has four main expert models, (1) query-expert model was a 3-layer BERT [4], distilled from a 12-layer pre-trained BERT. The pre-trained model was trained on billions of query data collected from online logs on our website; (2) user-expert model was a modified 4-layer Transformer [34], distilled from a 12-layer pre-trained Transformer that was trained on online user historical behaviors. We modified the original architecture by adding time information encoding for the user's historical behavior to model the user's interests and preferences; (3) item-expert model and trigger item-expert model were 4-layer BERT, distilled from a 12-layer pre-trained BERT that was trained with item textual features, including product title, category textual information and shop name. We concatenate the aforementioned information as the input to the item-expert model. Then as for aggregation layer, we adopt the structure of *Concatenation* Layer, *MLP* layer and *L2 Normalization* layer.

5.3.2 Hyper-parameter Setting. For the training hyper-parameter part, we set the output embedding size of User-Query tower and Item tower as 64. The batch size is set to 128. The number of heads in multi-head attention layer of interaction layer is set to 4. All dropout probability are set at 0.2. We currently utilize short-term user behavior history with a sequence length of 120. The parameters α and Δ of diversity loss are set to 0.6 and 0.8, respectively. In the differentiation sampling strategy described in Sec. 4.3, the hyperparameters ϵ, τ, and λ are set based on the last 30% of the total score. The AdamW optimizer [20] is employed, the initial learning rate is 0.0001 for the expert models and 0.001 for the rest.

5.4 Datasets

5.4.1 Large-scale Industrial Offline Dataset. We collected search logs over a 10-day period from the JD E-commerce website, resulting in a dataset exceeding one billion entries. We preprocessed this data to prepare it for model training, categorizing input features into ID, attribute, textual, and statistical types. The first nine days of data were designated as the training dataset. During preprocessing,



(a) Visualization of embedding from UniERF(w.o. $\mathcal{L}_{\text{DIV}}$/DIFF-SAM).

(b) Visualization of embedding from UniERF.

Figure 5: Visualization of query-user embedding from different model. Different colors denote multiple retrieval branches, pastel orange for personalized embedding, dark grey for semantic embedding and dusty blue for similar item embedding.

we meticulously cleaned the data by removing any corrupted or irrelevant entries. Following the differentiation sample construction strategy outlined in Section 4.3, we sampled challenging negative examples from the search logs and integrated them into the training dataset. For evaluation purposes, we randomly extracted millions of samples from the data collected on the final day.

5.4.2 Online Dataset. We deployed a well-trained UniERF in the JD E-commerce search product advertising environment containing hundreds of millions of user search requests, covering the most active product advertising at JD.COM.

5.5 Offline Experimental Results

In our previous work, we implemented the DNN architecture [3] for the deployment of JD's online search advertising system. We tailored the model to accommodate various retrieval branches. Specifically, for personalized retrieval, we enriched the model with extensive user historical behavior data and user profiles, integrating these embeddings into a multi-layer feed-forward neural network. For semantic retrieval, we excluded user historical behavior to better capture semantic features. In the case of similar item retrieval, we incorporated trigger item features to align with commodity-oriented scenarios. We refer to this customized model as the Adapted-DNN, which has demonstrated its effectiveness to a certain extent in the context of online search advertising deployment.

5.5.1 Offline Comparison with the Baseline. Table 1 shows the overall offline performances of our proposed method UniERF. We can observe that UniERF achieves a significant improvement with most evaluation metrics. UniERF improves over Adapted-DNN by 1.8%, 8.4% and 12.8% in Recall@1, Recall@100 and Recall@1000 respectively. It indicates that UniERF can retrieve more products with good relevance and better quality. And we can see UniERF improved over Adapted-DNN by 14.6% for $\text{Precision}_{\text{click}}$, indicating that UniERF can prioritize the retrieval of products that meet user needs and are more willing to click on while ensuring the retrieval of more high-quality products. Our method is able to capture the users' preference relation for different items to some extent, which can verify the consistency of links and offer better candidates for subsequent coarse and fine ranking stages.

5.5.2 Ablation Study. We study the effectiveness of our proposed UniERF by ablation experiments. Specifically, at first, we remove the diversity loss $\mathcal{L}_{\text{diversity}}$ from the total optimization objective and get rid of different hard negative sample strategies for each retrieval branch. Then we train the model from scratch and keep the other configurations the same as the normal UniERF. We use the trained model to inference the query embedding, which is visualized with T-SNE [33]. The results are shown in Figure 5, indicating that the effectiveness of differentiated sampling strategy and diversity loss $\mathcal{L}_{\text{diversity}}$. In the architecture of joint training, using exactly the same training sample will cause different retrieval branches to end up entangled, limiting the diversity of retrieval items.

Table 2: Ablation study of UniERF. For simplicity, we use $\mathcal{L}_{\text{DIV}}$ and DIFF-SAM to represent diversity loss and differentiated sampling strategy, respectively.

Methods	$\text{Precision}_{\text{click}}$	Recall@1000
UniERF(w.o. $\mathcal{L}_{\text{DIV}}$ / DIFF-SAM)	0.545	0.729
UniERF(all)	0.585	0.772

As shown in Table 2, compared with fully configured UniERF, the $\text{Precision}_{\text{click}}$ and Recall@1000 performance of ablation experiment groups are attenuated up to a point. Coupled multiple retrieval branch embedding limits the diversity of retrieval candidate sets of products, which led to repeated retrieval of some high quality products between different retrieval branches. Differential hard negative sample sampling strategy enables the model to retrieve

a more diverse set of high products. And the introduction of hard negative samples is beneficial for the model to learn more accurate user preferences. In addition, the experiment result shows that the introduction of different expert model increases the feature expression ability of the model to some extent. The analysis of time and space complexity can be found in the Appendix A.5.

5.6 Online A/B Test

After verifying the validity of UniERF offline, we launch it in search advertising system on multiple platforms in JD. These platforms include the JD App on mobile phones, JD Search Website, etc. Then we conduct several A/B tests, and the online results are reported in Table 3. Comparing with the lastest model in the online real environment, we can note that the performance of UniERF increases by 0.96%, 1.37%, 2.24% in terms of **IMP**, **CLK** and **COST**, respectively.

Table 3: The improvements on IMP, CLK, and COST of UniERF compared with the previous advertising system deployed on different websites/apps. The results are based on our 7-day surveillance of the entire online traffic.

Launched Platform	IMP	CLK	COST
JD APP on Mobiles	+0.67%	+1.21%	+2.20%
JD Search Products	**+1.13%**	**+1.59%**	**+2.41%**
JD Whole Search	+0.96%	+1.37%	+2.24%

6 Conclusion

In this paper, we proposes a novel and practical embedding-based retrieval model, called the **Uni**form **E**mbedding-based **R**etrieval **F**ramework (UniERF). It addresses computing resource waste and retrieval performance optimization due to the repeated candidates of multiple retrieval branches and interaction effect among different branches of JD Product Search Advertising. In the process, we proposed multiple expert model joint training strategy, differential hard negative sample construction strategy, and diversity loss to solve the above problems. Through offline experiments and online A/B tests, we verify the effectiveness of UniERF. In addition, we have deployed UniERF on JD Search Advertising System.

References

[1] Olivier Bousquet and André Elisseeff. 2002. Stability and generalization. *Journal of machine learning research* 2, Mar (2002), 499–526.

[2] Yair Censor. 1977. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization* 4, 1 (1977), 41–59.

[3] 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.

[4] Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Shi Dong, Ping Wang, and Khushnood Abbas. 2021. A survey on deep learning and its applications. *Computer Science Review* (2021).

[6] 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*.

[7] Hongliang Fei, Shulong Tan, Pengju Guo, Wenbo Zhang, Hongfang Zhang, and Ping Li. 2020. Sample optimization for display advertising. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.

[8] 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*.

[9] Moritz Hardt, Ben Recht, and Yoram Singer. 2016. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*. PMLR.

[10] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems* (2020).

[11] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. *Advances in neural information processing systems* (2014).

[12] 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*.

[13] 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*.

[14] Aaron Jaech, Hetunandan Kamisetty, Eric Ringger, and Charlie Clarke. 2017. Match-tensor: a deep relevance model for search. *arXiv preprint arXiv:1701.07795* (2017).

[15] Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. 2011. Searching in one billion vectors: re-rank with source coding. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* (2019).

[17] Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, and Qianli Ma. 2021. Embedding-based product retrieval in taobao search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

[18] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2019), 1475–1488.

[19] 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*.

[20] I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[21] Dinh The Luc. 2008. Pareto optimality. *Pareto optimality, game theory and equilibria* (2008).

[22] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[23] Bhaskar Mitra, Nick Craswell, et al. 2018. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval* (2018).

[24] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th international conference on world wide web*.

[25] 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*.

[26] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[27] 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* (2016).

[28] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[29] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

[30] 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*.

[31] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd international conference on world wide web*.

[32] 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*.

[33] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* (2008).

[34] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[35] Binbin Wang, Mingming Li, Zhixiong Zeng, Jingwei Zhuo, Songlin Wang, Sulong Xu, Bo Long, and Weipeng Yan. 2023. Learning Multi-Stage Multi-Grained Semantic Embeddings for E-Commerce Search. In *Companion Proceedings of the ACM Web Conference 2023.*

[36] Jinpeng Wang, Jieming Zhu, and Xiuqiang He. 2021. Cross-batch negative sampling for training two-tower recommenders. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval.*

[37] R Ward. 2014. Semantic modelling with long-short-term memory for information retrieval. *arXiv preprint arXiv:1412.6629* (2014).

[38] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion proceedings of the web conference 2020.*

[39] 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.*

[40] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, and Xia Hu. 2023. Data-centric ai: Perspectives and challenges. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM).* SIAM.

[41] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, Zhimeng Jiang, Shaochen Zhong, and Xia Hu. 2023. Data-centric artificial intelligence: A survey. *arXiv preprint arXiv:2303.10158* (2023).

[42] Han Zhang, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, and Wen-Yun Yang. 2020. Towards personalized and semantic retrieval: An end-to-end solution for e-commerce search via embedding learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[43] Yu Zhang. 2015. Multi-task learning and algorithmic stability. In *Proceedings of the AAAI Conference on Artificial Intelligence.*

[44] Yukun Zheng, Jiang Bian, Guanghao Meng, Chao Zhang, Honggang Wang, Zhixuan Zhang, Sen Li, Tao Zhuang, Qingwen Liu, and Xiaoyi Zeng. 2022. Multi-Objective Personalized Product Retrieval in Taobao Search. *arXiv preprint arXiv:2210.04170* (2022).

A Research Methods

A.1 Proof of theoretical analysis

According to **Lemma 1**, we can get the upper bound of the generalization error of the previous multiple retrieval branches R^{Ret} as follows

$$R^{\text{Ret}} \leq R^{\text{Ret}}_{\text{emp}} + 2\sum_{i=1}^{m}\beta_i^{\text{Ret}} + (4n\sum_{i=1}^{m}\beta_i^{\text{Ret}} + mM)\sqrt{\frac{ln(m/\delta)}{2n}} \quad (40)$$

According to [43], we can give Generalized McDiarmid's Inequality as

Lemma 2 *Let $X_1, ..., X_n$ be n independent random variables taking values from some set C, and assume that $f : C^n \rightarrow \mathbb{R}$ satisfies the following bounded differences condition when changing any q input arguments:*

$$\sup_{x_1,...,x_m,\hat{x}_{j_1},...,\hat{x}_{j_q}} |f(x_1, ..., x_{j_1}, ..., x_{j_q}, ..., x_n) - \quad (41)$$

$$f(x_1, ..., \hat{x}_{j_1}, ..., \hat{x}_{j_q}, ..., x_n)| \leq a \quad (42)$$

where $\{j_1, ..., j_q\} \subset \{1, ...n\}$, n is assumed to be divisible by q, and a is a fixed constant. Then for any $\varepsilon > 0$, we have

$$p\big[f(X_1, ..., X_n) - \mathbb{E}[f(X_1, ..., X_n)] \geq \varepsilon\big] \leq \exp\big\{-\frac{2q\varepsilon^2}{na^2}\big\} \quad (43)$$

Then, we assume that the numbers of training data points in different tasks are the same, i.e., $n_i = n_0$ for $i = 1, ..., m$. Based on the Generalized McDiarmid's Inequality in Lemma 2, we can obtain the generalization error bound of UniERF as

$$R^{\text{Uni}} \leq R^{\text{Uni}}_{\text{emp}} + 2\beta^{\text{Uni}} + (4n\beta^{\text{Uni}} + mM)\sqrt{\frac{ln(1/\delta)}{2n}} \quad (44)$$

Then we can analyze Eq. (40) and Eq. (44). For the empirical risks in the two bounds of R^{Uni} and R^{Ret}, we can assume the empirical risk of UniERF $R^{\text{Uni}}_{\text{emp}}$ is comparable with even lower than $R^{\text{Ret}}_{\text{emp}}$. Because UniERF can obtain more training data than training multiple retrieval model independently and access more expressive power shared foundational expert models by reducing repeated inference.

In the bound of multiple retrieval branches training independently, there is an additional constant m appearing in the numerator of the logarithm function of confidence part. So if the uniform stability of UniERF β^{Uni} is smaller than the sum of all the multiple retrieval branches $\sum_{i=1}^{m}\beta_i^{\text{Ret}}$, the generaliztion error bound of UniERF is smaller. According to [1, 9, 29], models with more expressive power can better capture the underlying structure of the data when the training data is sufficient. At the same time, as the amount of training data increases, the model's dependence on a single sample decreases, that is, the model's uniform stability increases. In UniERF, by means of joint training and sharing the foundation models, we are able to use the saved computing resources to build more expressive expert models. So we can obtain that $\beta^{\text{Uni}} \leq \sum_{i=1}^{m}\beta_i^{\text{Ret}}$. And then we can obtain that $R^{\text{Uni}}(\mathcal{F}, \mathcal{D}_{\text{train}}) \leq \sum_{i=1}^{m} R_i^{Ret}(\mathcal{F}_i, \mathcal{D}_{\text{train}})$.

A.2 Algorithm

The pseudo-code of our UniERF is shown in Alg. 1.

A.3 Compared with other methods

We compare the design insights of our UniERF with those of mainstream EBR (Embedding-Based Retrieval) approaches, which is shown in Table 4. A key innovation in UniERF is the explicit modeling of item-item relationships within the model, which significantly enhances its ability to retrieve items with similar properties. Furthermore, under the cooperative training strategy of UniERF, we introduce specialized diversity-enhancing techniques, such as differentiated sample construction and diversity loss functions. These strategies enable the multi-channel retrieval branches to collaboratively optimize, resulting in more diverse and comprehensive retrieval outcomes.

A.4 Online System

As illustrated in Figure A.4, the deployment system of UniERF operates in two phases. In the offline phase, the system routinely trains the model using data from the previous two weeks. Subsequently, the user-query tower component of the model is extracted and deployed for real-time prediction, while the item tower component is used to generate item embeddings. These embeddings are then transmitted to an Approximate Nearest Neighbor (ANN) indexing system for storage and retrieval. In the online phase, when a user issues a query, the system processes the input features and query to generate multi-channel retrieval embeddings. To enhance retrieval efficiency, the system is configured to perform retrievals across different segments (we use categories to divide segments in our system). Finally, the ANN search module distributively identifies the Top-K candidates from the item indexes.

Table 4: Comparison of UniERF to other mainstream retrieval methods.

Methods	DPSR	MGDSPR	MOPPR	UniERF(Ours)
Semantic Relevance	✓	✓	✓	✓
Individuation	✓	✓	✓	✓
Multi-channel Retrieval	✓			✓
Cooperative Training				✓
Diversity Strategy				✓
Hard Negative Samples		✓	✓	✓

Algorithm 1: UniERF: A Uniform Embedding-based Retrieval Framework for E-Commerce Search

Input : Training dataset $\mathcal{S} = \{(q_j, u_j, b_j, g_j, i_j)\}_{j=1}^{N}$, Batch size B, Learning Rate $\eta > 0$, Epoch T

Output : Retrieval Results $\hat{Z}$

Initialization $\theta \sim \mathcal{N}(0, \sigma^2)$;
$f_i = \text{ReLU}(\text{LayerNorm}(W_i x + b_i))$; $M = \lceil N/B \rceil$

for *epoch* $= 1$ *to* T **do**
  $\mathcal{S} \leftarrow \text{shuffle}(\mathcal{S})$
  for $m \in \{1, ...M\}$ **do**
    /* User-Query Tower */
    $e_{q_m} = \phi_q(q_m; \theta)$
    // Semantic Retrieval
    $H_m^{\text{sem}} = l2_{norm}(f_1(e_{q_m}))$
    // Personalized Retrieval
    $e_{u_m} = \varphi_{\text{u}}(u_m; \theta)$
    $e_{b_m} = \phi_{\text{u}}(b_m; \theta)$
    $e_{\text{inter}_m} = \mathcal{E}_{\text{per}}(q_m, b_m; \theta)$
    $H_m^{\text{per}} = l2_{norm}(f_2(e_{\text{fusion}}))$
    // Similar Retrieval
    $e_{\text{num}_m}^{\text{sim}} = \text{mean\_pooling}(\varphi_{\text{trig}}(g_m; \theta))$
    $e_{\text{text}_m}^{\text{sim}} = \text{mean\_pooling}(\phi_{\text{trig}}(g_m; \theta))$
    $e_{\text{inter}_m}^{\text{sim}} = \mathcal{E}_{\text{sim}}(q_m, g_m; \theta)$
    $H_m^{\text{sim}} = l2_{norm}(f_3(\text{concat}(e_{q_m}, e_{\text{inter}_m}^{\text{sim}}, e_{\text{num}_m}^{\text{sim}}, e_{\text{text}_m}^{\text{sim}}))$
    /* Item Tower */
    $e_{\text{num}_m} = \varphi_{\text{item}}(i_m; \theta)$
    $e_{\text{text}_m} = \phi_{\text{item}}(i_m; \theta)$
    $H_m^{\text{item}} = l2_{norm}(f_4(\text{concat}(e_{\text{num}_m}, e_{\text{text}_m}))$
    $\mathcal{L}_m = \frac{1}{B} \sum L(H_m^{\text{sem}}, H_m^{\text{per}}, H_m^{\text{sim}}, H_m^{\text{item}})$
    $\theta \leftarrow \theta - \eta \bigtriangledown_\theta \mathcal{L}_m$
  end
end

/* Online Inference State */
// For a request, get representations of the different retrieval branches via UniERF: h^{sem}, h^{per}, h^{sim}

$\text{TopK}(h) = \text{argmax}_{h_j \in H^{\text{item}}}^{K} \text{sim}(h, h_j)$

$\hat{Z} = \{\text{TopK}(h^{\text{sem}}), \text{TopK}(h^{\text{per}}), \text{TopK}(h^{\text{sim}})\}$

A.5 Performance

We compare our UniERF with previous online systems that utilized multiple embedding-based retrieval branches, as illustrated in Figure 5. Let n denote the number of retrieval branches, and T represent the time complexity of a single branch. Let D denote the size of the dataset, V the number of model parameters in a single retrieval branch, and H the number of model parameters in the aggregation layer, where $H \ll V$. For the conventional multi-channel retrieval strategy, the space complexity can reach $O(nV + nD)$. In contrast, UniERF reduces the space complexity to $O(V + nH + D)$, facilitated by cooperative training that aligns multiple retrieval branches within the same latent space. Moreover, both the time complexity of UniERF and the actual time consumption in the online system outperform the previous multi-retrieval branch strategy.

Table 5: A comparison of the time and space performance between UniERF and the traditional multi-channel retrieval approach.

Methods	Multi-EBR	UniERF
Time Complexity	$O(nT)$	$O(T)$
Space Complexity	$O(nV + nD)$	$O(V + nH + D)$
Real-time performance(ms)	40.82	24.66

A.6 Environment

We used Pytorch for code development and converted the trained model weights into TensorRT files for online deployment. We used 8 H800s as GPU resources for model training.

B Future Work

For future work, many potential directions can be explored. For example, we expect to introduce multi-objective optimization for better coordination of multiple retrieval branches and optimization objectives. The way our UniERF uses joint training is actually a multi-objective optimization problem. The Pareto Optimal theory [2, 21] can be considered to further optimize the model to improve the offline performance and the benefits of online systems.

Furthermore, there are some researches at multimodal retrieval scenarios currently. The introduction of multimodal features in UniERF will be one of our future explorations. For example, in the JD search system, there are very rich product picture information. The visual information is of great benefit for modeling user preference features, enriching the item-side feature representation, and improving the relevance between query and retrieval results.