pEBR: A Probabilistic Approach to Embedding Based Retrieval

Han Zhang[1], **Yunjiang Jiang**[2], **Mingming Li**[3],
Haowei Yuan[4], **Yiming Qiu**[1†], **Wen-Yun Yang**[1†]
[1]JD.com, Beijing, China [2]Meta, USA [4]Shanghai Jiaotong University
[3]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{zhanghang33, qiuyiming3}@jd.com limingming@iie.ac.cn
{yunjiangster, wenyun.yang}@gmail.com yhw@sjtu.edu.cn

Abstract

Embedding-based retrieval aims to learn a shared semantic representation space for both queries and items, enabling efficient and effective item retrieval through approximate nearest neighbor (ANN) algorithms. In current industrial practice, retrieval systems typically retrieve a fixed number of items for each query. However, this fixed-size retrieval often results in insufficient recall for head queries and low precision for tail queries. This limitation largely stems from the dominance of frequentist approaches in loss function design, which fail to address this challenge in industry. In this paper, we propose a novel **p**robabilistic **E**mbedding-**B**ased **R**etrieval (**pEBR**) framework. Our method models the item distribution conditioned on each query, enabling the use of a dynamic cosine similarity threshold derived from the cumulative distribution function (CDF) of the probabilistic model. Experimental results demonstrate that pEBR significantly improves both retrieval precision and recall. Furthermore, ablation studies reveal that the probabilistic formulation effectively captures the inherent differences between head-to-tail queries.

1 Introduction

Embedding retrieval is the core technology in search (Liu et al., 2021; Zhao et al., 2024; Bao et al., 2024) and recommendation(Covington et al., 2016; Li et al., 2020; Xing et al., 2025). The classic approaches are DSSM-like models (Huang et al., 2013; Zhang et al., 2020; Qiu et al., 2022; Li et al., 2021; Wang et al., 2023; Bao et al., 2024) *etc.*, where the query and item are embedded into dense vectors in the same semantic space. The relevant items are then retrieved by calculating the cosine similarity between the query vectors and the item vectors.

† Corresponding authors.

However, we notice that existing works on embedding-based retrieval do not pay enough attention to the model training algorithm, especially the loss function, which in practice causes some tricky problems as follows: 1) **Heuristic and suboptimal retrieval cutoffs.** Existing embedding-based retrieval systems rely on heuristic retrieval cutoffs, such as a fixed number of items or a fixed cosine similarity threshold. This leads to imbalanced retrieval performance: head queries suffer from low recall, while tail queries suffer from low precision. For instance, "iPhone 16" is a very specific query with few relevant items, while "gift" is broad, covering many products across categories. Thus, one can easily find that retrieving a fixed number of items would be suboptimal here, since if the number is small we will miss many relevant products for "gift" and if the number is large we will introduce many irrelevant items for "iPhone 16". On the other hand, it is also suboptimal to use a fixed cosine threshold, since all the existing model training algorithms have never taken that into account in their design. 2) **Ignoring Query Popularity Bias.** Current two-tower training algorithms treat all queries equally and lack a probabilistic formulation to model item relevance distributions. As a result, they fail to account for query popularity differences (head vs. tail queries) and struggle to generalize, especially for long-tail queries with limited training data.

Before we introduce our new approach to address the above shortcomings, let us first review previous related works: probabilistic models, or more precisely probabilistic graphical models, which have a long history of success in machine learning (Murphy, 2012; Koller and Friedman, 2009). Several models have significantly changed the landscape of machine learning in recent decades, such as the Baum-Welch algorithm for training (McCallum, 2004), LDA (Blei et al., 2003), and probabilistic modeling with neural network (Nguyen et al., 2017; Malinin, 2019; Kohl et al., 2018).

Inspired by the previous successes of probabilistic spirits, here we propose a redesign of the embedding-based retrieval to address the above challenges by using probabilistic modeling. Our contributions and the overview of the following sections can be summarized as follows.

- We first formalize the conventional two-tower model with a frequentist perspective and systematically analyze its inherent limitations.
- We then propose two novel probabilistic methods: one based on maximum likelihood estimation and another based on contrastive estimation; the latter includes two instances, ExpNCE and BetaNCE.
- We conduct both offline and online experiments to demonstrate the effectiveness of our model, and perform ablation studies to help understand how the model works.

2 Problem and Limitations

In this section, we first formulate the embedding retrieval problem based on a two-tower architecture. Next, we review the existing frequentist approaches and present limitations.

2.1 Problem Definition and Notations

The embedding retrieval model is typically composed of a query (or a user) tower and an item tower. For a given query q and an item d, the scoring output of the model is:

$$f(q, d) = s(\mathbf{v}_q, \mathbf{v}_d), \tag{1}$$

where $\mathbf{v}_q \in \mathbb{R}^n$ and $\mathbf{v}_d \in \mathbb{R}^n$ denotes the embedding of the query and the item, respectively. $s(.,.)$ computes the final score between the query and the item, such as the cosine similarity function, equivalently inner product between normalized vectors, *i.e.*, $s(\mathbf{v}_q, \mathbf{v}_d) = \mathbf{v}_q^\top \mathbf{v}_d$, where the superscript $^\top$ denotes matrix transpose. The objective is to select the top k items from a pool of candidate items for each given query $\mathbf{v}_q$.

2.2 Existing Frequentist Approaches

Most existing embedding retrieval adopt a frequentist approach to loss function design, which can be categorized into two types: point-wise loss and pair-wise loss. We discuss each category in detail.

2.2.1 Point-Wise Loss

The point-wise based method converts the original retrieval task into a binary classification, whose goal is to optimize the embedding space where the similarity between the query and its relevant item is maximized, while the similarity between the query and the irrelevant items is minimized. It usually adopts the sigmoid cross-entropy to train the model:

$$L_{\text{pointwise}}(\mathcal{D}) = -\sum_i \log \sigma(s(\mathbf{v}_{q_i}, \mathbf{v}_{d_i^+})) + \sum_{ij} \log \sigma(s(\mathbf{v}_{q_i}, \mathbf{v}_{d_{ij}^-})),$$

where d_i^+ denotes the relevant items for query q_i and d_{ij}^- denotes the irrelevant ones, $\sigma(x) = 1/(1 + \exp(-x))$ is the standard sigmoid function.

2.2.2 Pair-Wise Loss

This kind of method aims to learn the partial order relationship between positive and negative items from the perspective of metric learning. The classical work contains triple loss, margin loss, A-Softmax loss, and several variants (margin angle cosine). Without loss of generality, we formulate the loss as softmax cross-entropy:

$$L_{\text{pairwise}}(\mathcal{D}) = -\sum_i \log \left(\frac{\exp(s(\mathbf{v}_{q_i}, \mathbf{v}_{d_i^+})/\tau)}{\sum_j \exp(s(\mathbf{v}_{q_i}, \mathbf{v}_{d_{ij}})/\tau)} \right),$$

where τ is the so-called temperature parameter: a lower temperature is less forgiving of misprediction of positive items by the model. In the same direction, researchers later proposed more advanced loss functions, by introducing max margin in angle space (Liu et al., 2017), in cosine space (Wang et al., 2018) and so on.

2.2.3 Limitations

Both point-wise and pair-wise loss functions have their advantages and limitations. Point-wise loss functions are relatively simpler to optimize, while they may not capture the partial order relationships effectively. In contrast, Pair-wise loss functions explicitly consider the relative ranking between items. As a result, pair-wise loss functions usually achieve better performance in embedding retrieval tasks. While both of them are frequentist approaches, in the sense that no underlying probabilistic distribution are learned and consequently there is no cutoff threshold in principle when retrieving items. Thus, we propose the following probabilistic approach to a more theoretically well founded retrieval.

3 Method

In this section, we explore the probabilistic approach, which contains the choice of the item distribution function and our carefully designed loss function based on the item distribution.

3.1 Retrieval Embeddings as a Maximum Likelihood Estimator

Given a query q, we propose to model the probability of retrieving item d:

$$p(d|q) \propto p(r_{d,q}|q),$$

where $r_{d,q}$ represents the relevance between the query q and document d. For all relevant items $\mathbf{d}^+$, we assume $r_{d^+,q}$, where $d^+ \in \mathbf{d}^+$, follows a distribution whose probability density function is f_θ. For all irrelevant items $\mathbf{d}^-$, we assume $r_{d^-,q}$, where $d^- \in \mathbf{d}^-$, follows a distribution of whose probability density function is h_θ. The likelihood function can be defined as

$$\begin{aligned} L(\theta) &= \prod_q \left(\prod_{\mathbf{d}^+} p(d^+|q) \prod_{\mathbf{d}^-} p(d^-|q) \right) \\ &\propto \prod_q \left(\prod_{\mathbf{d}^+} p(r_{d^+,q}|q) \prod_{\mathbf{d}^-} p(r_{d^-,q}|q) \right) \\ &= \prod_q \left(\prod_{\mathbf{d}^+} f_{\theta,q}(r_{d^+,q}) \prod_{\mathbf{d}^-} h_{\theta,q}(r_{d^-,q}) \right). \end{aligned}$$

Importantly, the density for relevant items $p(r_{d^+,q}|q) = f_{\theta,q}(r_{d^+,q})$ and irrelevant items $p(d^-|q) = h_{\theta,q}(r_{d^-,q})$ are both query dependent. This is a useful generalization from fixed density since different queries have different semantic scopes. Finally the objective function can be defined as the log-likelihood:

$$l(\theta) = \sum_q \left(\sum_{\mathbf{d}^+} \log f_{\theta,q}(r_{d^+,q}) + \sum_{\mathbf{d}^-} \log h_{\theta,q}(r_{d^-,q}) \right).$$

When we choose different distributions for relevant and irrelevant items, the loss function can resemble a point-wise loss, which may lead to suboptimal performance compared to pair-wise loss functions. To address the limitation, we propose new probabilistic loss functions based on the principles of contrastive loss, which is a well-known pair-wise loss function.

3.2 Retrieval Embeddings as a Noise Contrastive Estimator

Apart from the above maximum likelihood estimator, the most widely used technique for retrieval model optimization is based on the InfoNCE loss (Oord et al., 2018), which is a form of noise contrastive estimator of the model parameter. In that setting, we need to choose two distributions: positive sample distribution $p(d^+|q)$, and background (noise) proposal distribution $p(d)$, where they are related in theory, if we know the joint distribution of d and q. But in practice, we can either treat them as separate or simply hypothesize their ratio as the scoring function $r(d,q) := p(d|q)/p(d)$, without knowing them individually. The loss we are minimizing is thus the following negative log-likelihood of correctly identifying the positive item within the noise pool

$$L_r = -\sum_i \log \frac{r(d_i^+, q_i)}{\sum_j r(d_{ij}, q_i)}. \qquad (2)$$

By minimizing the loss function, the model aims to maximize $p(d^+|q)$ for the query q and one of its relevant items d^+, while pushing away q and its irrelevant items d_j^+, thus assign higher similarities to relevant items compared to irrelevant items. Based on the definition, we propose two types of distributions as the basis of the estimator, the truncated exponential distribution ExpNCE and the Beta distribution BetaNCE.

3.2.1 Parametric Exponential InfoNCE (ExpNCE)

Here we propose to use the following truncated exponential distribution density function as the scoring function $r(d,q) \propto \exp(\cos(\mathbf{v}_\mathrm{d}, \mathbf{v}_\mathrm{q})/\tau_\mathrm{q})$ where the function cos stands for the cosine similarity between the two vectors and the temperature τ_q is query dependent. This offers an interesting alternative to the standard InfoNCE loss with log bi-linear distribution as

$$L_{\mathrm{ExpNCE}} = \sum_i \log \left(1 + \frac{\sum_j \exp(\cos(\mathbf{v}_{\mathrm{q_i}}, \mathbf{v}_{\mathrm{d_{ij}^-}})/\tau_\mathrm{q})}{\exp(\cos(\mathbf{v}_{\mathrm{q_i}}, \mathbf{v}_{\mathrm{d_i^+}})/\tau_\mathrm{q})} \right).$$

A nice property of the above probabilistic modeling is that we can use a simple cumulative density function (CDF) to decide the cutoff threshold. The CDF for the above truncated exponential distribution can be derived as follows.

$$\mathbb{P}_{\text{ExpNCE}}(x < t) = \frac{\int_{-1}^{t} e^{x/\tau_q} dx}{\int_{-1}^{1} e^{x/\tau_q} dx} = \frac{e^{t/\tau_q} - e^{-1/\tau_q}}{e^{1/\tau_q} - e^{-1/\tau_q}},$$

which can be easily computed.

3.2.2 Parametric Beta InfoNCE (BetaNCE)

We call a probability distribution compactly supported if its cumulative distribution function F satisfies $F((-\infty, -x]) = 1 - F([x, \infty)) = 0$ for some $x > 0$. In other words, all the mass is contained in the finite interval $[-x, x]$. The best-known family of compact distributions in statistics is probably the Beta distributions, whose pdf are defined on $[0, 1]$. Since the cosine similarity used in two-tower model has value range of $[-1, 1]$, we expand the definition range of Beta distributions to $[-1, 1]$ and define its density as

$$f_{\alpha,\beta}(x) = \frac{\Gamma(\alpha+\beta)}{2\Gamma(\alpha)\Gamma(\beta)} \left(\frac{1+x}{2}\right)^{\alpha-1} \left(\frac{1-x}{2}\right)^{\beta-1}.$$

An interesting special case is when $\alpha = \beta = 1$, $F_{1,1}(x) = \frac{x+1}{2}$, which gives the following uniform CDF on $[-1, 1]$.

One difficulty working with Beta distributions is that its CDF has no closed form, but rather is given by the incomplete Beta function[1]: $B(t; \alpha, \beta) = \int_0^t x^{\alpha-1}(1-x)^{\beta-1} dx$ is the incomplete Beta integral. The CDF for the above Beta distribution can be derived as follows.

$$\begin{aligned}\mathbb{P}_{\text{BetaNCE}}(x < t) &= \frac{\int_{-1}^{t}((1+x)/2)^{\alpha-1}((1-x)/2)^{\beta-1} dx}{\int_{-1}^{1}((1+x)/2)^{\alpha-1}((1-x)/2)^{\beta-1} dx} \\ &= \frac{\int_0^{\frac{t+1}{2}} x^{\alpha-1}(1-x)^{\beta-1} dx}{\int_0^1 x^{\alpha-1}(1-x)^{\beta-1} dx} \\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} B(\frac{t+1}{2}; \alpha, \beta).\end{aligned}$$

Fortunately we only need the PDF during training; the CDF is needed only once, to determine the retrieval threshold after training (see Appendix A). Then we assume $p(d|q)$ and $p(d)$ follow beta distributions like:

$$p(d|q) = g_q(s) \propto z^{\alpha_g(q)}(1-z))^{\beta_g(q)},$$

$$p(d) = k_q(s) \propto z^{\alpha_k(q)}(1-z))^{\beta_k(q)},$$

[1]https://www.tensorflow.org/api_docs/python/tf/math/betainc

where $z = \frac{1+s}{2}$, s is the cosine similarity between the query and item, $\alpha_g(q)$ and $\beta_g(q)$ are encoders based on the query representation to distinguish the learning of item distribution, and so do $\alpha_k(q)$ and $\beta_k(q)$. According to the definition of $r(d, q)$ in Section 3.2, we can get

$$r(d, q) = \frac{g_q(s)}{k_q(s)} \propto z^{\alpha_g(q)-\alpha_k(q)}(1-z)^{\beta_g(q)-\beta_k(q)},$$

which is again a Beta density. Note that α, β correspond to the number of successes and failures that lead to the Beta distribution. We can view both $p(d|q)$ and $p(d)$ as having the same number of failures (being a negative document for q), but $p(d|q)$ has some additional successes. Thus we can hypothesize that $\beta_g = \beta_k$, and therefore get the final BetaNCE loss of log-likelihood in Equation (2) as

$$\begin{aligned}L_{\text{BetaNCE}} &= -\sum_i \log\left(\frac{z(\mathbf{v}_{q_i}, \mathbf{v}_{d_i^+})^{\alpha_g(q)-\alpha_k(q)}}{\sum_j z(\mathbf{v}_{q_i}, \mathbf{v}_{d_{ij}})^{\alpha_g(q)-\alpha_k(q)}}\right) \\ &= -\sum_i \log\left(\frac{e^{\log z(\mathbf{v}_{q_i}, \mathbf{v}_{d_i^+})/\tau_q}}{\sum_j e^{\log z(\mathbf{v}_{q_i}, \mathbf{v}_{d_{ij}})/\tau_q}}\right),\end{aligned}$$

where $\tau_q = (\alpha_g(q) - \alpha_k(q))^{-1}$. Thus compared with InfoNCE, the main difference is the logarithmic transformation applied to the cosine similarity. Since the cosine similarity s is empirically always bounded away from -1, the logarithm presents no numerical difficulty.

After the training, we can get back the Beta distribution parameters from the learned τ_q by fixing the background distribution parameters $\alpha_k(q) \equiv 1$, $\beta_k(q) \equiv 1$, that is, the uniform distribution on the unit interval:

$$\alpha_g(q) = \tau_q^{-1} \text{ and } \beta_g(q) = 1. \qquad (3)$$

4 Experiments

In this section, we first explain the details of the dataset, experimental setup, baseline methods and evaluation metrics. Then we show the comparison of experimental results between the baseline methods and our proposed method pEBR. Finally we show the ablation study to intuitively illustrate how pEBR could perform better.

4.1 Experimental Setup

4.1.1 Dataset and Setup

Our model was trained on a randomly sampled dataset consisting of 87 million user click logs collected over a period of 60 days. We trained the

Table 1: Comparison of retrieval quality between our proposed method and baseline methods.

	All Queries		Head Queries		Torso Queries		Tail Queries	
	P@1500	R@1500	P@1500	R@1500	P@1500	R@1500	P@1500	R@1500
DSSM-topk	0.327%	93.29%	0.782%	90.38%	0.211%	94.36%	0.126%	94.36%
DSSM-score	0.435%	93.64%	0.841%	90.77%	0.339%	94.64%	0.275%	94.57%
pEBR	**0.583%**	**94.08%**	**0.945%**	**91.42%**	**0.513%**	**95.00%**	**0.450%**	**94.73%**

Table 2: Online A/B testing.

	UCVR	UCTR
Relative improvement	**+0.190%**	**+0.093%**
P-value	**0.0183**	**0.0213**

model on an Nvidia GPU A100 and employed the AdaGrad optimizer with a learning rate of 0.05, batch size of 1024, and an embedding size of 128. The training process converged after approximately 200,000 steps, which took around 12 hours.

4.1.2 Comparative Backbone

Our method is compatible with most two-tower retrieval models since it primarily modifies the loss function rather than the model architecture. Thus, we choose a classic two-tower DSSM (Huang et al., 2013; Qiu et al., 2022) as the backbone model. Then, we focus on cutting off items retrieved by the DSSM model with three methods:

- **DSSM-topk** refers to the method that cuts off items using a fixed threshold of number.
- **DSSM-score** refers to the method that cuts off items with a fixed threshold of relevance score. Items with relevance scores below the threshold are discarded.
- **pEBR** refers to our proposed probabilistic method which cuts off items with a threshold derived by a probabilistic model, specifically the CDF value of the learned item distribution. We are using a default CDF value 0.5 if not further specified. In this experiment, we focus on using BetaNCE with $\beta = 1$ as shown in Equation (3) to demonstrate the effectiveness of the methods. But our method is general enough to accommodate other distributions, such as ExpNCE.

As our method is model-agnostic and can be applied to any two-tower architecture, we also conduct experiments with a BGE (Xiao et al., 2024) (transformer-based) backbone to verify this compatibility. Details of these additional experiments are presented in Appendix B.

4.1.3 Experimental Metrics

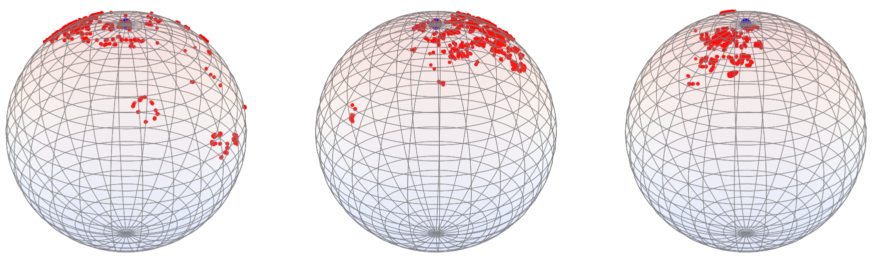
We use two widely reported metrics, Recall@k and Precision@k, to evaluate the quality and effectiveness of retrieval models (Zhang et al., 2021; Zhu et al., 2018; Yuan et al., 2023). There are some nuances that need a little more explanation. In DSSM-topk, a fixed number k of items is retrieved for each query to calculate R@k and P@k. While in DSSM-score and pEBR, a threshold of relevance score or CDF value is used to cut off retrieved items, which results in a variable number of items for each query. To ensure a fair comparison, we slightly adjust the threshold so that the average number of retrieved items across all queries equals k when computing the evaluation metrics.

Moreover, there is a tradeoff between precision and recall when varying the k value. In the experiments, we choose $k = 1500$ to optimize for recall, since it is the main goal of a retrieval system. Thus, the precision value is relatively low.

4.2 Experimental Results

Since retrieval performance varies considerably across head, torso, and tail queries, we adopt a tripartite dataset partitioning strategy for stratified evaluation. As shown in Table 1, we can draw the following conclusions:

- DSSM-score performs better than DSSM-topk on both R@1500 and P@1500, which is as expected since the retrieved items of DSSM-score have better relevance scores than DSSM-topk, because items with lower relevance scores are cut off.
- pEBR outperforms both DSSM-topk and DSSM-score, because pEBR learns varying item distributions for different queries, allowing for the determination of dynamic and optimal relevance score thresholds, thus leading to enhanced overall retrieval performance.
- pEBR achieves better performance on separated evaluation sets, *i.e.* head queries, torso queries and tail queries. Moreover, we observe that the amount of improvement



(a) “phone” (b) “Huawei phone 5g” (c) “Huawei mate50pro”

Figure 1: Relevant item distributions projected to a sphere for queries in head, torso, and tail categories, respectively.

achieved by pEBR varies systematically with respect to query popularities. Specifically, compared to DSSM-topk, pEBR achieves improvements on head queries, torso queries and tail queries respectively. This is because head queries normally have many more relevant items than tail queries; thus, a dynamic cutoff threshold could benefit head queries to retrieve more items significantly. What's more, tail queries normally do not have many relevant items, typically just tens of items. Thus, retrieving a fixed number k of items hurts the precision significantly. pEBR appears to solve this problem nicely.

4.3 Online A/B Testing

To assess the effectiveness of the proposed method in a real-world scenario, we performed comprehensive online A/B testing within an e-commerce search engine with hundreds of millions of users and billions of products.

Through a continuous week-long controlled experiment, table 2 shows that our proposed method achieved consistant improvements over the previous baselines, yielding a 0.19% increase in User Conversion Rate (UCVR) (p-value < 0.02), translating into substantial revenue gains for the company.

4.4 Ablation Study

4.4.1 Visual Illustration of Distributions

As shown in Figure 1, we visualize the relevant item distribution for three queries with different frequencies. We first need to apply a dimension reduction technique, specifically t-SNE (Van der Maaten and Hinton, 2008) here to reduce the dimension to 3-D. Then we normalize the vectors as unit vectors and plot them on a sphere. The head query, “phone”, is a quite general one that can retrieve phones of various brands and models. As a result, the retrieved items are widely distributed and dispersed across the surface of the sphere. The torso query “Huawei phone 5g” is a more specific one as it focuses on phones from the brand Huawei and with 5G capability. Consequently, the item distribution is more concentrated and has a narrower scope compared to the query “phone”. The tail query, “Huawei mate50pro”, is highly specific as it specifies the brand (Huawei) and model (mate50pro), thus the number of relevant items is very small and the distribution becomes even more concentrated. These differences in item distributions reaffirm the conclusion that cutting off retrieved items by a fixed threshold of item numbers or a fixed relevance score is suboptimal for embedding retrieval.

4.4.2 Dynamic Retrieval Effect

To investigate the effect of dynamic retrieval, we analyze the retrieval results from two perspectives. First, we examine the distribution of the number of retrieved items under a fixed probabilistic CDF threshold. Second, we evaluate how the number of retrieved items varies across different probabilistic CDF thresholds. These analyses provide insights into how dynamic retrieval impacts the overall retrieval results.

In Figure 2, we show the histogram of retrieved item numbers cut off by CDF value 0.985 for both head, torso, and tail queries. In detail, we filter out the items with the cosine similarity x bellowing a threshold t to ensure the equation $\mathbb{P}(x >= t) = 0.985$. In general, head queries have a more uniform distribution and the numbers lie mostly in the range of [500, 1500], while torso and tail queries share a similar skewed distribution and the numbers lie in the range of [0, 1000] and [0, 500], respectively. This indicates that head queries retrieve more items than torso and tail queries, which confirms the assumption that queries with

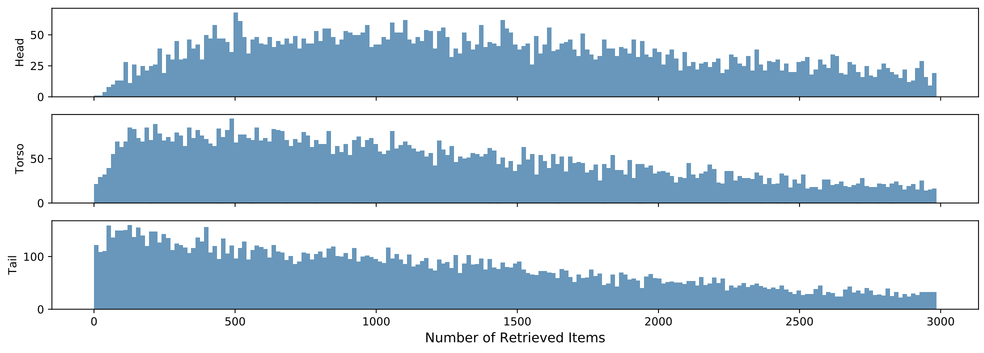


Figure 2: Histogram of the retrieved item numbers for head, torso, and tail queries, with the CDF value set to 0.985.

Table 3: Average numbers of retrieved items for different cutoff CDF values.

Cutoff CDF Value	0.99	0.95	0.9	0.8	0.7	0.6	0.5	0.4
Head Queries	2253.12	1443.51	1073.79	761.06	600.31	441.51	227.42	66.36
Torso Queries	1897.53	961.90	617.27	334.10	200.17	111.28	48.37	12.89
Tail Queries	1804.80	813.24	473.40	215.50	113.99	58.86	21.46	4.60

higher popularity need more candidates and queries with lower popularity need fewer.

In Table 3, we present the average number of retrieved items under different CDF values with the same cutting off strategy in Figure 2. As the probability decreases, queries claim a higher relevance level for items, thus the average numbers of retrieved items decreases accordingly. Comparing different queries, head queries retrieve more items than torso and tail queries under all CDF thresholds, which again confirms the conclusion above.

5 Conclusion

In this paper, we have proposed a novel probabilistic embedding based retrieval model, namely pEBR, to address the challenges of insufficient retrieval for head queries and irrelevant retrieval for tail queries. Based on the noise constrastive estimator, we have proposed two instance models: ExpNCE and BetaNCE, with the assumption that relative items follow truncated exponential distribution or beta distribution, which allow us to easily compute a probabilistic CDF threshold instead of relying on fixed thresholds, such as a fixed number of items or a fixed relevance score. Comprehensive experiments and ablation studies show that the proposed method not only achieves better performance in terms of recall and precision metrics, but also present desirable item distribution and number of retrieved items for head and tail queries.

6 Limitations

While our probabilistic model demonstrates significant improvements in retrieval accuracy and system performance, it remains inherently constrained by the two-tower and late-interaction retrieval paradigms (e.g., DSSM architectures). Notably, the emerging paradigm of generative retrieval (Kuai et al., 2024; Li et al., 2024; Deng et al., 2025) powered by large language models (LLMs) introduce new opportunities and challenges that our current framework does not address. We identify two promising directions for future research: 1) developing hybrid architectures that integrate the robustness and reliability of our probabilistic modeling with the semantic flexibility of generative retrieval, particularly for dynamic cutoff optimization. 2) Exploring constrained generation techniques that retain the precision and advantages of our current approach while mitigating LLM hallucinations through adaptive distribution modeling, thereby enhancing both accuracy and reliability in retrieval tasks.

7 Acknowledgments

We would like to express our sincere gratitude to Lin Liu, Songlin Wang, and Sulong Xu for their valuable guidance and technical support throughout this research. Their insightful suggestions and continuous encouragement were instrumental in shaping the direction of our work.

References

Wentian Bao, Hu Liu, Kai Zheng, Chao Zhang, Shunyu Zhang, Enyun Yu, Wenwu Ou, and Yang Song. 2024. Beyond relevance: Improving user engagement by personalization for short-video search. *arXiv preprint arXiv:2409.11281*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198.

Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. 2025. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *Preprint*, arXiv:2502.18965.

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*, pages 2333–2338.

Simon Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R Ledsam, Klaus Maier-Hein, SM Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. 2018. A probabilistic u-net for segmentation of ambiguous images. *Advances in neural information processing systems*, 31.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press.

Zhirui Kuai, Zuxu Chen, Huimu Wang, Mingming Li, Dadong Miao, Wang Binbin, Xusong Chen, Li Kuang, Yuxing Han, Jiaxing Wang, and 1 others. 2024. Breaking the hourglass phenomenon of residual quantization: Enhancing the upper bound of generative retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 677–685.

Mingming Li, Huimu Wang, Zuxu Chen, Guangtao Nie, Yiming Qiu, Guoyu Tang, Lin Liu, and Jingwei Zhuo. 2024. Generative retrieval with preference optimization for e-commerce search. *arXiv preprint arXiv:2407.19829*.

Mingming Li, Shuai Zhang, Fuqing Zhu, Wanhui Qian, Liangjun Zang, Jizhong Han, and Songlin Hu. 2020. Symmetric metric learning with adaptive margin for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4634–4641.

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*, pages 3181–3189.

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6738–6746.

Yiqun Liu, Kaushik Rangadurai, Yunzhong He, Siddarth Malreddy, Xunlong Gui, Xiaoyi Liu, and Fedor Borisyuk. 2021. Que2search: fast and accurate query and document understanding for search at facebook. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3376–3384.

Andrey Malinin. 2019. *Uncertainty estimation in deep learning with application to spoken language assessment*. Ph.D. thesis, University of Cambridge.

Andrew McCallum. 2004. Hidden markov models baum welch algorithm. *Introduction to Natural Language Processing CS585; University of Massachusetts Amherst: Massachusetts, USA*.

Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Dai Quoc Nguyen, Dat Quoc Nguyen, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2017. A mixture model for learning multi-sense word embeddings. *arXiv preprint arXiv:1706.05111*.

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

Yiming Qiu, Chenyu Zhao, Han Zhang, Jingwei Zhuo, Tianhao Li, Xiaowei Zhang, Songlin Wang, Sulong Xu, Bo Long, and Wen-Yun Yang. 2022. Pre-training tasks for user intent detection and embedding retrieval in e-commerce search. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4424–4428.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

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*, pages 411–415.

Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Zhifeng Li, Dihong Gong, Jingchao Zhou, and Wei Liu. 2018. Cosface: Large margin cosine loss for deep face recognition. pages 5265–5274.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In

Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval, pages 641–649.

Haibo Xing, Kanefumi Matsuyama, Hao Deng, Jinxin Hu, Yu Zhang, and Xiaoyi Zeng. 2025. Esans: Effective and semantic-aware negative sampling for large-scale retrieval systems. In *Proceedings of the ACM on Web Conference 2025*, pages 462–471.

Chunyuan Yuan, Yiming Qiu, Mingming Li, Haiqing Hu, Songlin Wang, and Sulong Xu. 2023. A multi-granularity matching attention network for query intent classification in e-commerce retrieval. In *Companion Proceedings of the ACM Web Conference 2023*, pages 416–420.

Han Zhang, Hongwei Shen, Yiming Qiu, Yunjiang Jiang, Songlin Wang, Sulong Xu, Yun Xiao, Bo Long, and Wen-Yun Yang. 2021. Joint learning of deep retrieval model and product quantization based embedding index. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1718–1722.

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. In *SIGIR*, pages 2407–2416.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4):1–60.

Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1079–1088.

A Inference Truncation

The major goal of using query dependent item distribution in neural retrieval is to give statistical meaning to the retrieved candidate set. Previously a somewhat arbitrary combination of cosine similarity threshold and top-K threshold are used.

$$S_{t,K}(q) = \mathrm{Top}_{\mathrm{K}}^{\cos(\mathrm{q},\cdot)}\left(\{\mathrm{d_i} : \cos(\mathrm{q}, \mathrm{d_i}) \geq \mathrm{t}\}\right)$$

The cosine threshold does not account for the variability of item distributions across different queries, while the top-K threshold is mainly to save inference cost.

Now for a given q-isotropic spherical distribution H, whose marginal density with respect to $\cos(\mathrm{q}, \cdot)$ is given by $h : [-1, 1] \to \mathbb{R}_+$, we can compute its CDF as follows

$$\mathbb{P}(H < t) = \frac{\int_{-1}^{t} h(x)(1 - x^2)^{(n-3)/2} dx}{\int_{-1}^{1} h(x)(1 - x^2)^{(n-3)/2} dx}. \quad (4)$$

For a given h and t, (4) can be computed numerically. For the two special h we are concerned here, we can give semi-closed forms:

- For the Beta density (BetaNCE) $h(x) \propto (1 + x)^{\alpha-1}(1 - x)^{\beta-1}$, the normalization constant, after the rescaling $[-1, 1] \to [0, 1]$, is a complete Beta integral

  $$\int_0^1 x^{\alpha+\frac{n-5}{2}}(1-x)^{\beta+\frac{n-5}{2}} dx = B(\alpha + \frac{n-3}{2}, \beta + \frac{n-3}{2}) = \frac{\Gamma(\alpha + \frac{n-5}{2})\Gamma(\beta + \frac{n-5}{2})}{\Gamma(\alpha + \beta + n - 4)},$$

  while the numerator is proportional to an incomplete Beta integral $\int_0^{\frac{1+t}{2}} x^{\alpha+\frac{n-5}{2}}(1-x)^{\beta+\frac{n-5}{2}} dx =: B_{\frac{1+t}{2}}(\alpha + \frac{n-3}{2}, \beta + \frac{n-3}{2})$.

- For the truncated exponential density (corresponding to InfoNCE), $h(x) \propto e^{x/\tau}$, the integral we need to compute is the following modified Bessel integral $\int_{-1}^{t} e^{x/\tau}(1-x^2)^{\frac{n-3}{2}} dx = \int_{-1}^{t} e^{x/\tau}(1+x)^{\frac{n-1}{2}-1}(1-x)^{\frac{n-1}{2}-1} dx.$ This admits a closed form solution for any t provided $n > 2$ is odd, however the solution has alternating signs, which is numerically unstable especially for large n.

Due to the difficulty of exact solutions, we resort to numeric quadratures:

```
import scipy.integrate as integrate
import scipy.special as special
import math

def BetaInt(alpha, beta, t):
    return integrate.quad(
        lambda x: (1+x)**(alpha-1) * \
            (1-x) ** (beta-1), -1, t)

def BetaExpInt(alpha, beta, tau, t):
    return integrate.quad(
        lambda x: (1+x)**(alpha-1) * \
            (1-x)**(beta-1) * \
            math.exp(x/tau), -1, t)

cache = {}

def CosineInvCDF(p, quad_fn, *args):
    values = cache.setdefault(
        quad_fn, {}).get(tuple(args))
    if not values:
        denom = quad_fn(*args, 1)[0]
        values = \
            cache[quad_fn][tuple(args)] \
          = [quad_fn(*args, i/500-1)[0] \
            / denom for i in range(1001)]
    if p >= values[-1]:
        return 1
    if p <= values[0]:
        return -1
    right = min(i for i,
       v in enumerate(values) if v >= p)
    left = right - 1
   return (right * (p-values[left])+left*
        (values[right]-p)) / \
      (values[right]-values[left])/500-1


def BetaInvCDF(p, alpha, beta):
    return CosineInvCDF(
        p, BetaInt, alpha, beta)

def BetaExpInvCDF(p, alpha, beta, tau):
    return CosineInvCDF(
        p, BetaExpInt, alpha, beta, tau)

def InfoNCEInvCDF(p, n, tau):
    return CosineInvCDF(
       p,BetaExpInt,(n-1)/2,(n-1)/2,tau)
```

B Backbone Experiments

Our proposed pEBR method is orthogonal to the underlying two-tower architecture and can be seamlessly integrated into various backbone models. To further validate the generalizability of our method, we conducted additional experiments using BGE, a transformer-based embedding model, as the backbone for our two-tower setup. As shown in table 4, our method consistently outperforms both top-K and fixed-score truncation approaches in terms of Recall@1500 across different backbone models, demonstrating the generalizability and effectiveness of pEBR.

Table 4: Recall@1500 on different methods.

	All Queries	**Head**	**Torso**	**Tail**
BGE-topk	0.9592	0.9556	0.9622	0.9585
BGE-score	0.9352	0.9415	0.9320	0.9347
BGE-pEBR	0.9691	0.9730	0.9731	0.9641

C Different Corpus Sizes

We have conducted supplementary experiments with varying training data sizes. The tabel 5 shows that while absolute recall@1500 decreases as training data is reduced, the relative improvement brought by our method remains stable. This suggests that the query distribution modeling enabled by pEBR converges efficiently, and our method is robust even with moderate-sized datasets.

Table 5: Recall@1500 on different corpus sizes.

	87M	**40M**	**20M**
BGE-topk	0.9592	0.9535	0.9482
BGE-score	0.9352	0.9153	0.9102
BGE-pEBR	0.9691	0.9672	0.9524

D Different Recall Cutoffs

To provide a more comprehensive evaluation, we also experimented with smaller k values (e.g., k=500 and k=1000). Detailed results are presented in the Table 6.

Table 6: Performance at different recall cutoffs.

	R@1500	**R@1000**	**R@500**
BGE-topk	0.9592	0.9350	0.8833
BGE-score	0.9352	0.8598	0.7344
BGE-pEBR	0.9691	0.9589	0.9226