CROLoss: Towards a Customizable Loss for Retrieval Models in Recommender Systems

Yongxiang Tang
Alibaba Group
Beijing, China
tangyongxiang94@gmail.com

Wentao Bai
Alibaba Group
Beijing, China
bwt1997@126.com

Guilin Li
Alibaba Group
Beijing, China
liguilin.lgl@lazada.com

Xialong Liu
Alibaba Group
Beijing, China
xialong.lxl@alibaba-inc.com

Yu Zhang
Alibaba Group
Beijing, China
daoji@lazada.com

ABSTRACT

In large-scale recommender systems, retrieving top N relevant candidates accurately with resource constrain is crucial. To evaluate the performance of such retrieval models, Recall@N, the frequency of positive samples being retrieved in the top N ranking, is widely used. However, most of the conventional loss functions for retrieval models such as softmax cross-entropy and pairwise comparison methods do not directly optimize Recall@N. Moreover, those conventional loss functions cannot be customized for the specific retrieval size N required by each application and thus may lead to sub-optimal performance. In this paper, we proposed the **C**ustomizable **R**ecall@N **O**ptimization **Loss** (**CROLoss**), a loss function that can directly optimize the Recall@N metrics and is customizable for different choices of Ns. This proposed CROLoss formulation defines a more generalized loss function space, covering most of the conventional loss functions as special cases. Furthermore, we develop the Lambda method, a gradient-based method that invites more flexibility and can further boost the system performance. We evaluate the proposed CROLoss on two public benchmark datasets. The results show that CROLoss achieves SOTA results over conventional loss functions for both datasets with various choices of retrieval size N . CROLoss has been deployed onto our online E-commerce advertising platform, where a fourteen-day online A/B test demonstrated that CROLoss contributes to a significant business revenue growth of 4.75%.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Top-k retrieval in databases**.

Yongxiang Tang and Wentao Bai contributed equally.
Code implementation is available at https://github.com/WDdeBWT/CROLoss/

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557274

KEYWORDS

Recommender systems, information retrieval, Recall@N optimization

ACM Reference Format:
Yongxiang Tang, Wentao Bai, Guilin Li, Xialong Liu, and Yu Zhang. 2022. CROLoss: Towards a Customizable Loss for Retrieval Models in Recommender Systems. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3511808.3557274

1 INTRODUCTION

Large-scale recommender systems are widely adopted in commercial applications, such as media recommendation [10, 34], commodities recommendation [6, 30] and computational advertising [36, 37]. With the scale of users and items on such popular platforms expanding rapidly, multi-stage recommender systems [10, 16, 34] have become an important strategy. The multi-stage recommender system is consist of the retrieval stage [15, 23, 37] that selects hundreds of candidates from the whole item set and the ranking stage [8, 12] that sorts the candidates in a more precise order before final exposure. In this paper, we focus on optimizing the retrieval stage, of which the goal is to optimize the Recall@N metrics, which is defined as the frequency of positive samples being retrieved in the top N ranking.

Popular retrieval models such as [6, 10, 13, 16, 20, 28, 30, 34, 35, 37] mainly adopt softmax cross-entropy or pairwise comparison methods [26, 27] as their loss functions. The softmax cross-entropy method transforms the retrieval scores into a probability distribution and minimizes the cross-entropy loss. The pairwise comparison methods including the triplet loss [27] and some learning to rank methods such as RankNet [3] and BPR loss [26] optimize the retrieval model by learning the order between positive and negative items for each user. Although these methods have achieved great success in practical recommender systems, neither the the cross-entropy loss or the pairwise comparison methods is directly optimizing the Recall@N metrics. Moreover, as well known, a proper choice of retrieval size N provides a trade-off between system efficiency and accuracy for different commercial usage. However, these conventional losses are also not adaptable for different choice of retrieval size N, which may lead to sub-optimal solutions for practical commercial usage.

In this paper, we propose the **C**ustomizable **R**ecall@N **O**ptimization **Loss** (**CROLoss**), which is derived to directly optimize the Recall@N metrics. We first rewrite the Recall@N metrics in the form of pairwise sample comparison. By leveraging a pairwise comparison kernel function ϕ, this objective function is derived as a differentiable loss functional space. We also invite a weighting function w_α to allow this loss function customizable for a different choice of N. Besides, it can be shown that the proposed CROLoss function space covers the conventional cross-entropy loss, triplet loss, and BPR loss as special cases. Furthermore, by analyzing the gradient of the CROLoss, we find that the comparison kernel ϕ plays two different roles. To further improve this loss function, we develop the Lambda method, a gradient-based method that allows a different choice of kernel ϕ_1 and ϕ_2 for these two roles, and further boost the system performance.

Extensive experiments are conducted on two public benchmark datasets. Experimental results demonstrate that the proposed CROLoss can significantly improve traditional loss functions on both datasets for all different choices of N. These experiments also verify the customizability of CROLoss for different retrieval sizes N through adjusting the comparison kernels and weighting parameters. Furthermore, a fourteen-day online A/B is performed on our online E-commerce advertising platform, where CROLoss improved Recall@N by 6.50% over the cross-entropy loss and contributes to a significant business revenue growth of 4.75%.

The main contributions of our paper are summarized as follows:

- We propose the CROLoss, to our best knowledge, it is the first method that directly optimizes the Recall@N metrics and can be customized for different retrieval sizes N.
- This CROLoss covers a more generalized loss function space, allowing users to tailor it for their specific usage by tuning the kernel and the weighting parameter. We proved that cross-entropy loss, triplet loss and BPR loss are special cases of our method.
- We develop the Lambda method, a gradient-based method that allows a more flexible choice of the kernel function and further boost the system performance.
- We conduct extensive experiments on two real-world public datasets, which show that CROLoss outperforms existing loss functions significantly.

2 RELATED WORK

The retrieval stage aims to retrieve N items from a large corpus with both the efficiency and the accuracy as system objective [10]. In recent years, extensive work has been developed to improve the retrieval model by learning high-quality user and item representation vectors [6, 18, 20, 30, 35, 37] or by organizing the training data in a wise manner [21, 34]. Limited attention has been paid to the design of a more suitable loss function for this task. Most conventional models employ two categories of retrieval loss: the softmax cross-entropy loss and the pairwise method. However, all these losses do not directly optimize Recall@N which may lead to suboptimal solutions. In our practice, we also found that a fixed loss form may not be suitable for every different choice of N.

2.1 Retrieval Model Optimization Losses

The softmax cross-entropy loss treats the optimization of the retrieval model as a classification problem [6, 28, 34, 37]. Suppose item v is a positive sample for user u taken from a set of positive user-item pairs C, and items $v_1, \cdots, v_n$ are negative samples for user u. The softmax function transforms the retrieval scores $S(u, v_i)$ to a probability distribution, and we have the cross-entropy minimization objective:

$$\mathcal{L}_{softmax} = -\sum_{(u,v)\in C} \log \frac{e^{S(u,v)}}{e^{S(u,v)} + \sum_i e^{S(u,v_i)}}, \tag{1}$$

of which solution is the maximum likelihood estimation of the classification problem.

The second category is the pairwise methods [16, 35], which model the order between positive and negative items for each user by pairwise comparison on retrieval scores. For example, when using the user as the anchor, the triplet loss [27] is:

$$\mathcal{L}_{triplet} = \sum_{(u,v)\in C} \sum_i (S(u, v_i) - S(u, v) + m)_+, \tag{2}$$

where $(\cdot)_+$ is the positive part function. Since the margin parameter m appears in the loss, the method can enlarge the gap between the positive and negative retrieval scores. However, both (1) and (2) do not directly optimize Recall@N, and cannot be customized for the retrieval size N.

2.2 Learning to Rank

Learning To Rank (LTR) is a method for directly learning the ordering of samples, which can be classified into three categories: pointwise method, pairwise method, and listwise method. In Section 3, we show that our CROLoss can also be viewed as a pairwise LTR method.

The pointwise methods[9, 19] are not an optimal choice for retrieval task since the retrieval stage generally does not need to accurately estimate the ranking score. The listwise methods[4, 5, 11, 29, 32, 33] are generally used in scenarios where the correlation of each ranking position can be accurately determined, which is also not completely consistent with the object of the retrieval stage. The pairwise methods[2, 3, 7, 24–26] approximate the LTR problem to the classification problem, which produce ranking results by learning the relative order of each sample pair. The triplet loss mentioned above can also be seen as a form of the pairwise LTR. RankNet loss [3] is the classic form of pairwise LTR loss. If S is a set of sample pairs, including the rank order label between samples a and b, we have:

$$\mathcal{L}_{ranknet} = -\sum_{a,b\in S} t_{ab} \log(\sigma(s_a - s_b)) + (1 - t_{ab}) \log(1 - \sigma(s_a - s_b)), \tag{3}$$

where s_a and s_b are the scores predicted by the ranking model, σ is logistic sigmoid function, t_{ab} is equal to 1 if sample a is ranked higher than b, and equal to 0 otherwise. If samples a and b have the same rank, then t_{ab} is equal to 0.5. In the retrieval stage of the recommender system, due to the large number of candidate items, it generally uses the item clicked by the user and the item of random negative sampling to construct a sample pair, therefore, it is not necessary to consider pairs of equal order. If we manually specify

that the sample v is the positive sample and the sample v_i is the negative sample, we can get the BPR loss [26]:

$$\mathcal{L}_{bpr} = -\sum_{(u,v)\in C}\sum_{i}\log(\sigma(S(u,v) - S(u,v_i))), \tag{4}$$

which is an LTR method widely used in recommender systems [13, 31]. There are some pairwise LTR methods that use specific models [7, 25] or require special supervision signals [2, 24] and are not suitable for the retrieval stage of recommender systems.

In this paper, we will prove that $\mathcal{L}_{bpr}$, as well as the aforementioned $\mathcal{L}_{softmax}$ and $\mathcal{L}_{triplet}$, are special cases of our CROLoss, and will provide a new perspective to interpret them from the object of Recall@N optimization.

3 CUSTOMIZABLE RECALL@N OPTIMIZATION LOSS

In this section, we propose the Customizable Recall@N Optimization Loss, abbreviated as CROLoss and marked as $\mathcal{L}$. We first formulate the Recall@N optimization problem and rewrite it in the form of pairwise sample comparison using a non-differentiable comparison function ψ. Next, we obtain the $\mathcal{L}_{\alpha,\psi}$ by introducing a weighting function $w_\alpha(N)$ to customize the retrieval size N. Then, with help from the pairwise comparison kernel ϕ, we get $\mathcal{L}_{\alpha,\phi}$, which makes CROLoss differentiable. Furthermore, we develop the Lambda method, which is an improvement on the gradient form of CROLoss and can better optimize our objective.

3.1 The base model

Our approach is model-agnostic. To illustrate the idea, we adopt the widely used two-tower retrieval model [10] as our base model, which has a good balance between accuracy and online processing speed. The model structure is shown Figure 1. The two-tower model can be generally divided into two parts: User Tower and Item Tower. The User Tower processes the user profile features and user behavior features to produce the user representation vector u, and the Item Tower processes the target item features to produce the item representation vector v. Finally, the retrieval score $S_\theta(u,v)$ of the user and the target item is generally obtained by inner product or vector similarity calculation.

3.2 Problem Formulation

Suppose the whole set of items is $\mathcal{I} = \{v_1, ..., v_{|\mathcal{I}|}\}$ and the corpus of our retrieval model is $C = \{(u,v)^{(j)}\}_{j=1}^{|C|}$, which is a set of positive user-item pairs. The retrieval model $S_\theta(u,v)$ is a scoring function for any user-item pair, and θ is its parameter. The Recall@N statistic determined by model $S_\theta(u,v)$ and corpus C is

$$\text{Recall@N} = \frac{|\{(u,v)|S_\theta(u,v) \in \text{Top}_\text{N}\{S_\theta(u,v_i)|v_i \in \mathcal{I}\}, (u,v) \in C\}|}{|C|}, \tag{5}$$

where $\text{Top}_\text{N}(\cdot)$ returns a set's largest N elements. Our purpose is to find a retrieval model S_θ which maximizes (5).

In order to rewrite the Recall@N metrics in a simpler form, we define the ranking of $S_\theta(u,v)$ among the whole user-item scoring set $\{S_\theta(u,v_i)|v_i \in \mathcal{I}\}$ as the ranking statistic of (u,v):

$$R_\psi(u,v;\theta) = 1 + \sum_{\substack{v_i\in\mathcal{I}\\ v_i\neq v}} \psi\left(S_\theta(u,v_i) - S_\theta(u,v)\right), \tag{6}$$

where ψ is the unit step function, $\psi(x) = 0$ when $x < 0$ and $\psi(x) = 1$ when $x >= 0$. Then item v is retrievable by top-N retrieval scores of user u if and only if $R_\psi(u,v;\theta) \le N$, and we can rewrite Recall@N as

$$\text{Recall@N} = \frac{1}{|C|}\sum_{(u,v)\in C}\mathbf{1}_{\{R_\psi(u,v;\theta)\le N\}}, 1 <= N <= |\mathcal{I}|. \tag{7}$$

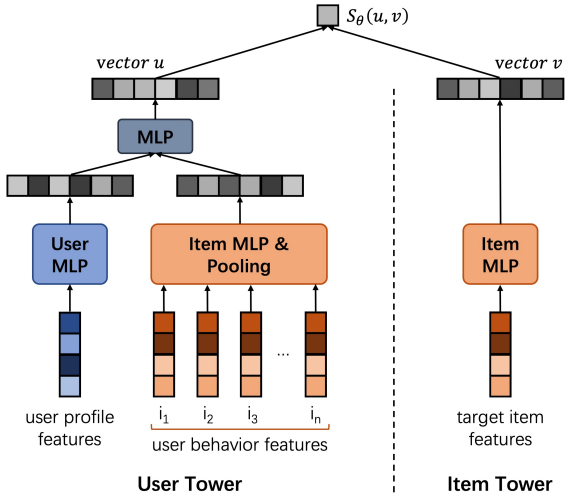

Figure 1: Architecture of two tower retrieval model

3.3 Customizable Retrieval Range

According to the target of our retrieval model, we should pay attention to some specific ranges of N when we try to optimize the Recall@N metrics. We can achieve the requirement by introducing a probability density function

$$w\colon [1, |\mathcal{I}|+1) \to [0, +\infty), \tag{8}$$

which satisfies

$$\int_1^{|\mathcal{I}|+1} w(x)\,dx = 1. \tag{9}$$

We call w the weighting function, and the value of $w(x)$ resembles the importance of Recall@x metric, where Recall@x = Recall@$\lfloor$x$\rfloor$ is the generalized form of Recall@N for $x \in \mathbb{R}$.

By using $w(x)$ to weight the importance of Recall@x metrics for $x \in [1, |\mathcal{I}|+1)$, we can optimize the Recall@N metrics for all possible Ns by defining our loss function as the negative expectation of Recall@x with respect to the PDF $w(x)$ as follows,

$$\begin{aligned}\mathcal{L}(\theta) &= -\,\mathbb{E}_w\,\text{Recall@x}\\ &= -\int_1^{|\mathcal{I}|+1} w(x)\text{Recall@x}\,dx\end{aligned} \tag{10}$$

By (7), we can simplify the expression of $\mathcal{L}(\theta)$ as:

$$\begin{aligned}\mathcal{L}(\theta) =& -\int_1^{|\mathcal{I}|+1} w(x) \sum_{(u,v)\in C} \frac{1}{|C|} \mathbf{1}_{\{R_\psi(u,v;\theta)\le x\}}\, dx,\\ =& \frac{-1}{|C|} \sum_{(u,v)\in C} \int_1^{|\mathcal{I}|+1} w(x) \mathbf{1}_{\{R_\psi(u,v;\theta)\le x\}}\, dx\\ =& \frac{-1}{|C|} \sum_{(u,v)\in C} \int_{R_\psi(u,v;\theta)}^{|\mathcal{I}|+1} w(x)\, dx\\ =& \frac{-1}{|C|} \sum_{(u,v)\in C} \left(1 - \int_1^{R_\psi(u,v;\theta)} w(x)\right) dx\\ =& \frac{1}{|C|} \sum_{(u,v)\in C} W(R_\psi(u,v;\theta)) - 1,\end{aligned} \tag{11}$$

where $W(x) = \int_1^x w(z)\, dz$ is the cumulative distribution function of $w(x)$.

We generally hope the value of $w(x)$ is larger for smaller x since for most recommender systems, the retrieval size is far less than the size of the candidate set $|\mathcal{I}|$, making most of the Recall@N metrics with greater N unimportant. Here, we take the power function $w_\alpha(x)$ as our weighting function:

$$w_\alpha(x) = \frac{1}{Z} x^{-\alpha} \tag{12}$$

where $\alpha \le 0$ is the weighting parameter and Z is a regularization parameter so that $\int_1^{|\mathcal{I}|+1} w_\alpha(x)\, dx = 1$. Then $W(N)$ becomes

$$W_\alpha(N) = \int_1^N w_\alpha(x)\, dx = \begin{cases} \dfrac{\log(N)}{\log(|\mathcal{I}|+1)} & \text{if } \alpha = 1,\\ \dfrac{1 - N^{1-\alpha}}{1 - (|\mathcal{I}|+1)^{1-\alpha}} & \text{else.}\end{cases} \tag{13}$$

Therefore, by neglecting the constant -1 in (11), we have the loss function controlled by α:

$$\mathcal{L}_{\alpha,\psi}(\theta) = \sum_{(u,v)\in C} W_\alpha(R_\psi(u,v;\theta)). \tag{14}$$

For a larger α, the weighting function $w_\alpha(x)$ decays faster and guides our model to optimize Recall@N on smaller Ns. Hence, we can customize the Recall@N optimization problem by tuning the weighting parameter α.

3.4 Pairwise Kernel-Based Objective

We notice that our objective in (14) is not differentiable, which means it is impossible to solve the problem via a gradient descent algorithm. To solve this issue, we modify the unit step function ψ into a differentiable substitution $\phi(x)$, then we have the approximation of the ranking statistic in (6):

$$R_\phi(u,v;\theta) = 1 + \sum_{\substack{v_i\in\mathcal{I}\\ v_i\ne v}} \phi(S_\theta(u,v_i) - S_\theta(u,v)). \tag{15}$$

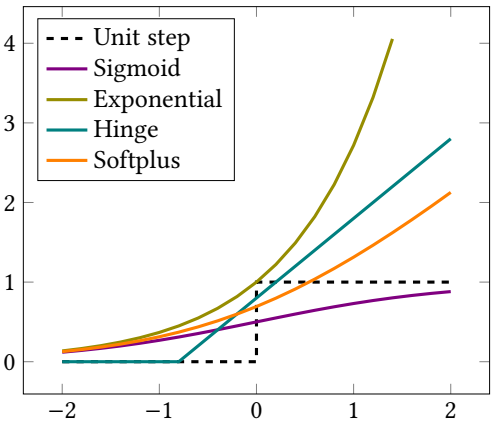

Figure 2: Example of optional comparison kernel functions.

By (15), we have the approximation of $\mathcal{L}_{\alpha,\psi}(\theta)$ in (14):

$$\begin{aligned}\mathcal{L}_{\alpha,\phi}(\theta) &= \sum_{(u,v)\in C} W_\alpha\left(R_\phi(u,v;\theta)\right)\\ &= \sum_{(u,v)\in C} W_\alpha\left(1 + \sum_{\substack{v_i\in\mathcal{I}\\ v_i\ne v}} \phi(S_\theta(u,v_i) - S_\theta(u,v))\right),\end{aligned} \tag{16}$$

which is the formal form of the CROLoss.

We call function $\phi(x)$ the comparison kernel if it satisfies the following conditions:

i) $\phi(x)$ is differentiable almost everywhere;
ii) $\phi(x)$ is monotonically increasing;
iii) $\lim_{x\to-\infty} \phi(x) = 0$;
iv) $0.5 <= \phi(0) <= 1$;
v) $\lim_{x->\infty} \phi(x) >= 1$.

With these properties, ϕ can approximates ψ well for $x < 0$. However, we do not strictly constraint $\phi(x) \le 1$ for $x \ge 0$, since the optimization can be easier when $\phi(x)$ is convex. For a convex kernel ϕ, larger gaps in $G_u = \{S_\theta(u,v_i) - S_\theta(u,v) | v_i \in \mathcal{I}, v_i \ne v\}$ can decay faster via SGD [1] due to the convexity of ϕ, letting positive gap $g \in G_u$ close to 0 and $\phi(g)$ close to $\phi(0)$, where $\phi(0) \in [0.5, 1]$. Thus, it prevents (16) go too far from the original $\mathcal{L}_{\alpha,\psi}$. As shown in Figure 2, there are many popular activation functions can be regarded as the comparison kernel. We will verify the adaptability of these kernels to CROLoss in the experimental section.

3.5 Relation to Existing Methods

In this section, we will prove that by choosing specific comparison kernel and alpha, the softmax cross-entropy loss $\mathcal{L}_{softmax}$, the triplet loss $\mathcal{L}_{triplet}$ and the BPR loss $\mathcal{L}_{bpr}$ can be regarded as special cases of our CROLoss.

By taking $\phi(x) = \exp(x) = e^x$ and $\alpha = 1$, we have

$$\begin{aligned}\mathcal{L}_{1,\exp}(\theta) &= \sum_{(u,v)\in C} \log\left(1 + \sum_{\substack{v_i\in\mathcal{I}\\ v_i\neq v}} e^{S_\theta(u,v_i)-S_\theta(u,v)}\right)\\ &= \sum_{(u,v)\in C} \log\left(\frac{\sum_{v_i\in\mathcal{I}} e^{S_\theta(u,v_i)}}{e^{S_\theta(u,v)}}\right)\\ &= -\sum_{(u,v)\in C} \log\left(\frac{e^{S_\theta(u,v)}}{\sum_{v_i\in\mathcal{I}} e^{S_\theta(u,v_i)}}\right),\end{aligned} \tag{17}$$

which is identical to the softmax cross-entropy loss shown in (1).

By taking $\phi(x) = \text{hinge}(x) = (x+m)_+$ and $\alpha = 0$, we have

$$\begin{aligned}\mathcal{L}_{0,\text{hinge}}(\theta) &= \frac{1}{|\mathcal{I}|}\left(1 + \sum_{(u,v)\in C}\sum_{\substack{v_i\in\mathcal{I}\\ v_i\neq v}} (S_\theta(u,v_i) - S_\theta(u,v) + m)_+\right)\\ &= \frac{1}{|\mathcal{I}|}\sum_{(u,v)\in C}\sum_{\substack{v_i\in\mathcal{I}\\ v_i\neq v}} (S_\theta(u,v_i) - S_\theta(u,v) + m)_+ + const,\end{aligned} \tag{18}$$

which is identical to the triplet loss shown in (2).

By taking $\phi(x) = \text{softplus}(x) = \log(1+e^x)$ and $\alpha = 0$, we have

$$\begin{aligned}\mathcal{L}_{0,\text{softplus}}(\theta) &= \frac{1}{|\mathcal{I}|}\left(1 + \sum_{(u,v)\in C}\sum_{\substack{v_i\in\mathcal{I}\\ v_i\neq v}} \log\left(1 + e^{S_\theta(u,v_i)-S_\theta(u,v)}\right)\right)\\ &= \frac{1}{|\mathcal{I}|}\sum_{(u,v)\in C}\sum_{\substack{v_i\in\mathcal{I}\\ v_i\neq v}} -\log\left(\frac{1}{1+e^{-(S_\theta(u,v)-S_\theta(u,v_i))}}\right) + const,\end{aligned} \tag{19}$$

which is identical to the BPR loss shown in (4).

Therefore, we provide a new perspective to interpret these conventional methods in the form of CROLoss. We will further analyze the impact of their selection of specific kernels and alphas on performance in the experiments.

3.6 The Lambda Method

Introducing the comparison kernel is helpful for optimizing the CROLoss. However, the range of most comparison kernels is $(0,\infty)$, causing the prediction of the ranking statistic $R_\phi(u,v;\theta)$ imprecise and thus inability to accurately estimate the gradient of the parameters through the weight function. To find an appropriate method to optimize the model parameters, we focus on the gradient of CROLoss to the parameters. We let $g_u^{(i)} = S_\theta(u,v_i) - S_\theta(u,v)$ to represent the gap between negative score $S_\theta(u,v_i)$ and positive score $S_\theta(u,v)$. Then, according to (16) we have the partial derivative of $\mathcal{L}_{\alpha,\phi}(\theta)$ with respect to $g_u^{(i)}$:

$$\frac{\partial\mathcal{L}_{\alpha,\phi}(\theta)}{\partial g_u^{(i)}} = \left\{w_\alpha(R_\phi(u,v;\theta))\right\}\left\{\phi'(g_u^{(i)})\right\}. \tag{20}$$

We use curly brackets to divide the right-hand side of (20) into two parts, the first part can be regarded as the approximated weighting density, and the second part can be regarded as the descendent velocity of each gap $g_u^{(i)}$. The choice of the comparison kernel function ϕ is the key to the performance of CROLoss. For the first part, we want to estimate $R_\phi(u,v;\theta)$ accurately and map it to an appropriate weighting density. As shown in Figure 2, taking ϕ as the logistic sigmoid or the unit step function can bring $R_\phi(u,v;\theta)$ closer to the authentic ranking statistic $R_\psi(u,v;\theta)$. For the second part, we hope that ϕ can properly penalize on larger $g_u^{(i)}$ to speed up model fitting, so the choice of ϕ can be softplus function, hinge function, or exponential function. Therefore, the comparison kernel ϕ actually plays two roles, and their goals are inconsistent. Inspired by [2], which directly converts the non-differentiable metric into a multiplier on the gradient of the objective function, we propose the Lambda method to resolve this conflict.

First, we use a multiplier $\lambda^{(u)}$ to replace $w_\alpha(R_\phi(u,v;\theta))$ in (20), assuring $\lambda^{(u)}$ to be a suitable estimation of the actual weighting density $w_\alpha(R_\psi(u,v;\theta))$. We take

$$\lambda^{(u)}_{\alpha,\phi_1} = w_\alpha\left(R_{\phi_1}(u,v;\theta)\right) = w_\alpha\left(1 + \sum_{\substack{v_i\in\mathcal{I}\\ v_i\neq v}} \phi_1(S_\theta(u,v_i) - S_\theta(u,v))\right), \tag{21}$$

which is regarded as a non-differentiable value during training.

Then, we take a suitable kernel ϕ_2 to replace the kernel ϕ of $\phi'(g_u^{(i)})$ in (20), and the partial derivative form shown in (20) can be replaced by

$$\frac{\partial\mathcal{L}_{\alpha,\phi_1,\phi_2}(\theta)}{\partial g_u^{(i)}} = \lambda^{(u)}_{\alpha,\phi_1}\phi_2'(g_u^{(i)}), \tag{22}$$

where $\mathcal{L}_{\alpha,\phi_1,\phi_2}$ is the new loss function controlled by two independent comparison kernel ϕ_1 and ϕ_2.

Therefore, we can configure different choices for ϕ_1 and ϕ_2, and even choose a non-differentiable function such as a unit step function directly in ϕ_1. Based on the form of partial derivative controlled by ϕ_1 and ϕ_2 in (22), we can derive the original loss function of the lambda method:

$$\begin{aligned}\mathcal{L}_{\alpha,\phi_1,\phi_2}(\theta) &= \sum_{(u,v)\in C}\lambda^{(u)}_{\alpha,\phi_1}\left(1 + \sum_{\substack{v_i\in\mathcal{I}\\ v_i\neq v}}\phi_2\left(g_u^{(i)}\right)\right)\\ &= \sum_{(u,v)\in C}\text{SG}\left(w_\alpha\left(R_{\phi_1}(u,v;\theta)\right)\right)R_{\phi_2}(u,v;\theta),\end{aligned} \tag{23}$$

where SG resembles the stop-gradient method, that avoids the gradient backpropagation from $\lambda^{(u)}_{\alpha,\phi_1}$ to the model parameter θ.

4 EXPERIMENT

In this section, we conduct extensive experiments on two benchmark datasets, as well as an online A/B test in our advertising system, to answer the following questions:

RQ1: Could we boost the performance of retrieval models with different retrieval size Ns by using CROLoss?

RQ2: How to select the comparison kernel ϕ and weighting parameter α for different retrieval size Ns?

RQ3: Whether the Lambda method can further improve the performance of CROLoss, and how to choose suitable kernel functions for ϕ_1 and ϕ_2?

RQ4: Can CROLoss improve the performance of current losses in a live recommender system?

4.1 Experimental Setup

Table 1: Dataset basic statistics

Dataset	#users	#items	#behaviors
Amazon Books	459,133	313,966	8,898,041
Taobao	976,779	1,708,530	85,384,110

Datasets. We conduct experiments on two public real-world datasets. The statistics of the two datasets are shown in Table 1.

- **Amazon Books** [22] is the *Books* category of the Amazon dataset, which is consist of product reviews from Amazon.
- **Taobao** [37] is a public dataset collected users' click logs from Taobao's recommender systems.

We split all users into training/validation/test sets by the proportion of 8:1:1 and use the user's first k behaviors to predict the $(k+1)$-th behavior in each training sample. We truncate the length of user behavior sequences to 20 for the Amazon Book dataset and 50 for the Taobao dataset.

Compared Methods. To evaluate the performance of our proposed method, we compare the CROLoss with the following conventional methods:

i) softmax cross-entropy loss shown in (1);
ii) triplet loss shown in (2);
iii) BPR loss shown in (4).

Besides, we test the performance of CROLoss under different comparison kernels and weighting parameters to reveal their effects. For the Lambda method, we regard it as a version of CROLoss and include it in the comparison. The optional kernel functions are as follows:

i) hinge kernel $\phi_{hinge}(x) = (x+m)_+$;
ii) logistic sigmoid kernel $\phi_{sigmoid}(x) = 1/(1+e^{-x})$;
iii) exponential kernel $\phi_{exponential}(x) = e^x$;
iv) softplus kernel $\phi_{softplus}(x) = \log(1+e^x)$.

Implementation Details. We take $S(u,v) = \tau\langle u, v\rangle$, where $\langle u, v\rangle$ is the cosine similarity of user vector u and item vector v. τ is the scale parameter to map the $S(u,v)$ to the appropriate range of the kernel function, we generally set it to 10 to achieve the best performance.

To speed up model training, we uniformly sample n_{rn} random items for each positive sample and share these random items inside a mini-batch of size n_{bs}, letting each positive sample be attached with a set of $n_{rn} * n_{bs}$ random items acting as negative items. For uniformly sampled subset $\mathcal{I}'$ of $\mathcal{I}$, a biased but useful estimation of the original ranking statistic in (6) is

$$\hat{R}_\phi(u,v;\theta) = \frac{|\mathcal{I}|}{|\mathcal{I}'|}\left\{1 + \sum_{\substack{v_i \in \mathcal{I}' \\ v_i \neq v}} \phi(S_\theta(u,v_i) - S_\theta(u,v))\right\}, \qquad (24)$$

which substitutes $R_\phi(u,v;\theta)$ defined in (16) and (23) in our original model. We set n_{rn} and n_{bs} as 10 and 256 for Amazon Books dataset, and as 10 and 512 for the larger Taobao dataset. The number of embedding dimensions and hidden dimensions are set to 32. The margin m of hinge comparison kernel is set to 5. We use the Adam optimizer [17] with learning rate 0.02 for training.

Evaluate Metrics. We use Recall@N shown in (5) as the metric to evaluate the performance of the retrieval model. To verify the customization ability of our method for different retrieval ranges, we set N as 50, 100, 200, and 500, respectively. All the results are percentage numbers with '%' omitted.

4.2 Performance Comparison (RQ1)

Table 2: Experimental results of comparison between CROLoss and conventional methods. R is short for Recall Metrics.

Datasets	Methods	R@50	R@100	R@200	R@500
Amazon	cross-entropy loss	9.68	13.24	17.46	24.26
	triplet loss	7.53	11.21	15.93	24.11
	BPR loss	8.24	12.08	16.96	25.21
	CROLoss[1]	**10.20**	14.03	18.63	26.06
	CROLoss-lambda[2]	10.17	**14.07**	**18.81**	**26.20**
Taobao	cross-entropy loss	4.71	6.59	9.01	13.13
	triplet loss	2.46	3.71	5.43	8.84
	BPR loss	2.89	4.33	6.35	10.25
	CROLoss[3]	4.75	6.65	9.06	13.13
	CROLoss-lambda[4]	**5.27**	**7.35**	**10.01**	**14.57**

1. Use softplus as kernel and set α to 1.0.
2. Use sigmoid as kernel 1 and softplus as kernel 2 and set α to 1.0.
3. Use exponential as kernel and set α to 1.4.
4. Use sigmoid as kernel 1 and exponential as kernel 2 and set α to 1.4.

We compare the best performing CROLoss and CROLoss-lambda with the softmax cross-entropy loss, the triplet loss, and the BPR loss. As Table 2 shows, by applying suitable comparison kernel ϕ and weighting parameter α, our method significantly outperforms these three conventional methods. Furthermore, in almost all cases, the Lambda method helps CROLoss-lambda achieve even better performance. Among the three conventional methods, softmax cross-entropy loss performs the best, while the other two, especially triplet loss, perform not well. We suppose that this is due to the fact that in most practices using triplet loss, the hard triplet mining method is used [14, 16, 35], while we use the uniform negative sample method in our experiments. We will analyze this phenomenon from the perspective of CROLoss later, and provide an efficient method equivalent to the hard triplet mining to improve their performance.

4.3 Influence of Kernel and Alpha (RQ2)

We test the CROLoss under different comparison kernels ϕ and weighting parameters α. The experimental results are shown in Table 3, where the row of lambda kernel represents the performance of CROLoss using the Lambda method under the best choices of

Table 3: Experimental results of CROLoss with different kernels and weights.

Datasets	Kernels	Recall@50				Recall@100				Recall@200				Recall@500			
		α=0.6	α=0.8	α=1.0	α=1.2	α=0.6	α=0.8	α=1.0	α=1.2	α=0.6	α=0.8	α=1.0	α=1.2	α=0.6	α=0.8	α=1.0	α=1.2
Amazon	hinge	9.03	9.50	9.88	9.53	12.95	13.35	13.55	12.87	17.64	18.04	17.94	16.76	25.56	25.63	24.99	22.91
	sigmoid	9.57	9.94	9.96	10.05	13.63	13.77	13.94	13.88	18.51	18.56	18.59	18.38	**26.59**	26.35	26.09	25.66
	exponential	8.90	9.39	9.68	9.99	12.44	12.87	13.24	13.56	16.75	17.11	17.46	17.71	23.81	24.13	24.26	24.33
	softplus	9.63	9.97	**10.20**	10.19	13.54	13.85	14.03	14.03	18.30	18.59	18.63	18.49	26.20	26.19	26.06	25.51
	lambda[1]	9.65	9.80	10.17	10.19	13.58	13.78	**14.07**	14.01	18.39	18.50	**18.81**	18.46	26.29	26.25	26.20	25.96
		α=1.0	α=1.2	α=1.4	α=1.6	α=1.0	α=1.2	α=1.4	α=1.6	α=1.0	α=1.2	α=1.4	α=1.6	α=1.0	α=1.2	α=1.4	α=1.6
Taobao	hinge	3.75	3.96	4.02	3.72	5.37	5.62	5.62	5.09	7.51	7.82	7.68	6.84	11.35	11.64	11.18	9.63
	sigmoid	3.89	4.04	4.11	4.13	5.62	5.84	5.87	5.87	7.91	8.21	8.15	8.13	11.93	12.19	12.13	11.96
	exponential	4.71	4.74	4.75	4.61	6.59	6.71	6.65	6.42	9.01	9.20	9.06	8.69	13.13	13.33	13.13	12.44
	softplus	4.21	4.30	4.36	4.38	5.98	6.12	6.21	6.15	8.32	8.55	8.52	8.44	12.51	12.70	12.54	12.25
	lambda[2]	4.55	5.04	5.27	**5.31**	6.39	7.06	7.35	**7.44**	8.75	9.68	**10.01**	9.98	12.90	14.28	**14.57**	14.54

1. Use sigmoid as ϕ_1 and softplus as ϕ_2.
2. Use sigmoid as ϕ_1 and exponential as ϕ_2.

ϕ_1 and ϕ_2, and will be discussed in detail in the next subsection. For clarity, we underline the best performance of each comparison kernel at each retrieval size N, and bold the best performance among all kernels.

By comparing the performance of different comparison kernels in the two datasets, we can see that the performance of the exponential kernel on the Taobao dataset, which contains a larger candidate set, is significantly better than its performance on the Amazon dataset. Figure 3 shows a similar phenomenon that the performance of exponential kernel gradually decreases with the increase of N compared to other kernels. Therefore, we suppose that the exponential function is an ideal kernel for the scenario where the ratio of the retrieval size N to the size of the candidate set $|\mathcal{I}|$ is very small. For other comparison kernels with relatively stable gradients, the retrieval size has no significant effect on its performance. Specifically, the softplus kernel is a good choice and can always achieve the best or sub-best performance; the sigmoid kernel performs slightly worse, but it can best match the weighting parameter α to be customized for different retrieval sizes; the hinge kernel is not recommended because it performs poorly and has more hyperparameters.

By observing the performance under different α, we can find that in most cases, a larger α makes CROLoss pay more attention to the top-ranked samples, resulting in better performance of Recall@N under smaller retrieval size N. Conversely, a smaller α can achieve better performance with a larger N. Among all the comparison kernels, the logistic sigmoid performs best in this customizability, which we suppose is due to it can estimate the $R_\phi(u, v; \theta)$ most accurately. Finally, we test CROLoss on a wider range of N using the sigmoid kernel, and the results reported in Figure 4 visually demonstrate that CROLoss can be customized for different retrieval sizes by simply adjusting α.

4.4 Performance of Lambda Method (RQ3)

The performance of CROLoss using the Lambda method has been shown in Tables 2 and 3. Compared with the original CROLoss,

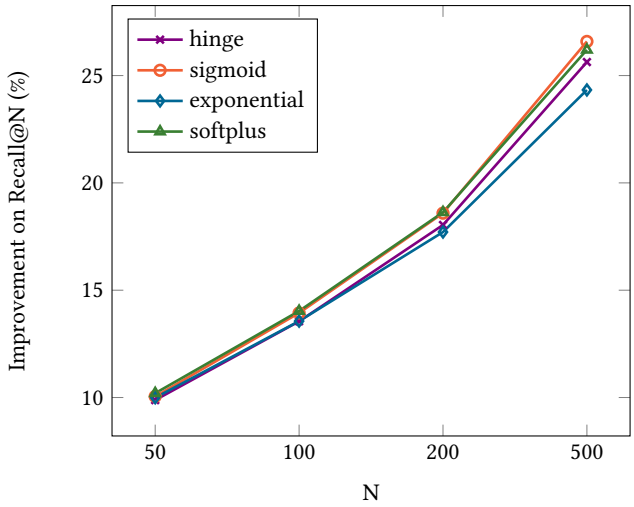

Figure 3: The Recall@N metrics of four different comparison kernels of CROLoss at different retrieval size N on Amazon dataset.

it has a greater improvement in the Recall@N metric, especially on the Taobao dataset. Moreover, since the selection of ϕ_1 and ϕ_2 is independent, CROLoss-lambda can choose a suitable kernel function such as sigmoid to accurately estimate $R_{\phi_1}(u, v; \theta)$, which inherits the good customizability of CROLoss using the sigmoid kernel. Table 4 shows the impact of different ϕ_1 and ϕ_2 choices on CROLoss-lambda, where the choice of ϕ_2 confirms the previous conclusions: the exponential function is suitable for the Taobao dataset that contains a larger candidate set, and the softplus kernel performs well in the Amazon dataset. For the choice of ϕ_1, the experimental results show that although the unit step function can calculate $R_{\phi_1}(u, v; \theta)$ more accurately, the smoother logistic sigmoid function can achieve the best performance in most cases.

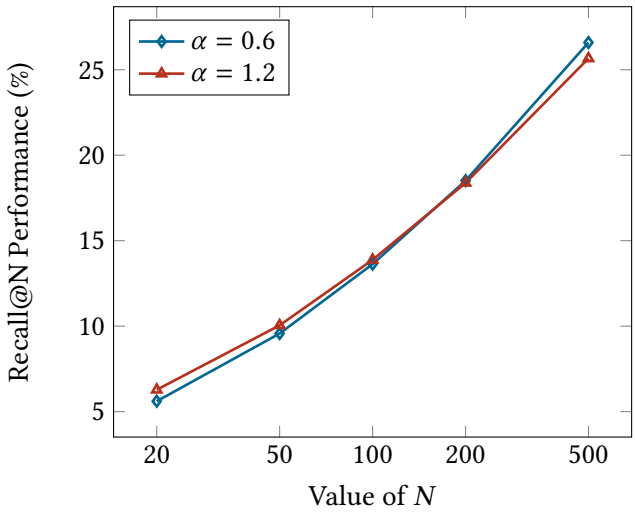

Figure 4: The Recall@N metrics of CROLoss with sigmoid kernel and different weighting parameter α on the Amazon dataset.

Table 4: Experimental results of the Lambda method with different kernels. We set α as 1.0 for Amazon dataset and 1.4 for Taobao dataset. R is short for Recall Metrics.

Dataset	Kernel ϕ_1	Kernel ϕ_2	R@50	R@100	R@200	R@500
Amazon	unit step	hinge	9.07	12.865	17.46	24.96
	unit step	exponential	8.74	12.15	16.35	23.20
	unit step	softplus	9.98	13.85	18.53	25.96
	sigmoid	hinge	9.435	13.305	17.995	25.69
	sigmoid	exponential	8.79	12.10	16.26	22.90
	sigmoid	softplus	**10.17**	**14.07**	**18.81**	**26.20**
Taobao	unit step	hinge	3.30	4.79	6.79	10.30
	unit step	exponential	4.85	6.81	9.36	13.70
	unit step	softplus	4.25	6.07	8.47	12.62
	sigmoid	hinge	3.82	5.49	7.70	11.60
	sigmoid	exponential	**5.27**	**7.35**	**10.01**	**14.57**
	sigmoid	softplus	4.74	6.71	9.26	13.62

4.5 Industrial Results (RQ4)

We deploy the CROLoss method onto the retrieval stage of our large-scale online advertising system. As mentioned above, the two-tower recommendation model served as the base retrieval model in our system. After completing the model training, we infer and store the representation vectors of all advertised items. When serving online, our model first computes the representation vector of visiting users and then retrieves top-N items from the item vector pool by a fast nearest neighbor searching method. We deployed both the model trained with CROLoss and softmax cross-entropy loss on the A/B test platform. In a fourteen-day online A/B test, our new method improved the Recall metric by 6.50%, and increased revenue by 4.75% in Cost-per-click advertising.

4.6 A New Perspective on Hard Negative Mining

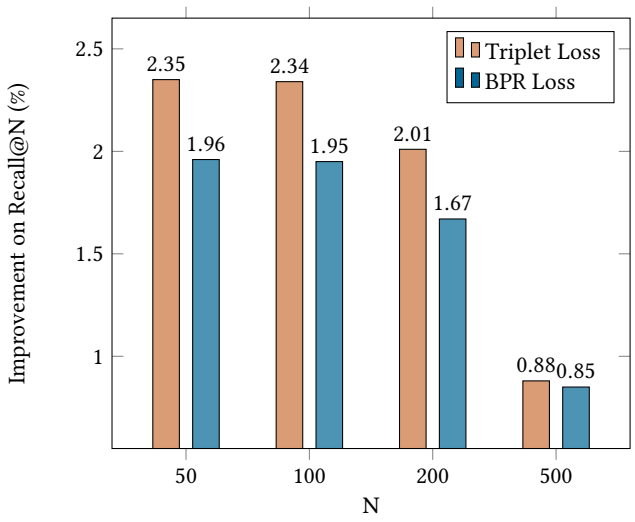

Figure 5: Performance improvement of using CROLoss for hard negative mining on triplet loss and BPR loss.

As shown in (17), (18), and (19), the softmax cross-entropy loss, the triplet loss, and the BPR loss can be regarded as special cases of CROLoss. This provides a new perspective to understanding why hard negative mining (triplet mining) [16, 27] is more commonly used in the practice of triplet loss than in cross-entropy loss. In addition to different kernels, the difference between triplet loss and cross-entropy loss is that the former is equivalent to setting $\alpha = 0$ and the latter's $\alpha = 1$. BPR loss can also be classified into the category with $\alpha = 0$.

If we regard the weighting function $w_\alpha(x)$ as the probability density function of an importance sampling, when $\alpha = 0$, it is equivalent to uniform sampling, and when $\alpha > 0$, it is equivalent to paying more attention to the top-ranked samples of (u, v). For these (u, v), most of the easy negative samples are already well optimized, so the gradient is mostly brought by the hard negative samples. Therefore, the CROLoss with $\alpha > 0$ can be seen as an efficient hard negative mining method for hard negatives, which interprets why cross-entropy does not require additional work on hard negative mining but triplet loss does. Figure 5 shows the performance improvement over the original loss by using the same kernel as triplet loss and BPR loss in CROLoss and setting $\alpha = 1$. Specifically, this effective mining method can significantly improve the performance of both loss functions, especially when N is smaller.

5 CONCLUSION

This paper proposes the novel Customizable Recall@N Optimization Loss, an objective direct optimize the Recall@N metrics and can customized for the retrieval size N required by different applications. We achieve the goal by rewriting the definition of Recall@N via a ranking statistic $R_\psi(u, v; \theta)$ and assigning an importance weight for each Recall@N metric via the weighting function w_α. To optimize the CROLoss, we introduce the comparison kernel function ϕ that makes the CROLoss differentiable. Furthermore,

we develop the Lambda method that solves the CROLoss more efficiently. We test our method on two real-world public datasets and our online advertising system, and the result shows our method outperforms the conventional objectives on Recall@N metrics and proves the ability to customize N for the Recall@N optimization problem.

REFERENCES

[1] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 177–186.

[2] Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems* 19 (2006).

[3] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*. 89–96.

[4] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[5] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[6] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2942–2951.

[7] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[9] David Cossock and Tong Zhang. 2008. Statistical analysis of Bayes optimal subset ranking. *IEEE Transactions on Information Theory* 54, 11 (2008), 5140–5154.

[10] 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.

[11] Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and Heung Yeung Shum. 2010. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. 295–303.

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[14] Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* (2017).

[15] Michael E Houle and Michael Nett. 2014. Rank-based similarity search: Reducing the dimensional dependence. *IEEE transactions on pattern analysis and machine intelligence* 37, 1 (2014), 136–150.

[16] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 2615–2623.

[19] Ping Li, Qiang Wu, and Christopher Burges. 2007. Mcrank: Learning to rank using multiple classification and gradient boosting. *Advances in neural information processing systems* 20 (2007).

[20] Zheng Liu, Jianxun Lian, Junhan Yang, Defu Lian, and Xing Xie. 2020. Octopus: Comprehensive and elastic user representation for the generation of recommendation candidates. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 289–298.

[21] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled self-supervision in sequential recommenders. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 483–491.

[22] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.

[23] Marius Muja and David G Lowe. 2014. Scalable nearest neighbor algorithms for high dimensional data. *IEEE transactions on pattern analysis and machine intelligence* 36, 11 (2014), 2227–2240.

[24] Ivo FD Oliveira, Nir Ailon, and Ori Davidov. 2018. A new and flexible approach to the analysis of paired comparison data. *The Journal of Machine Learning Research* 19, 1 (2018), 2458–2486.

[25] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 3–11.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[27] 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*. 815–823.

[28] 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*. 373–374.

[29] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. 77–86.

[30] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 839–848.

[31] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[32] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*. 1192–1199.

[33] Jun Xu and Hang Li. 2007. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 391–398.

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

[35] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[36] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1059–1068.

[37] 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*. 1079–1088.