From Pairwise to Ranking: Climbing the Ladder to Ideal Collaborative Filtering with Pseudo-Ranking

Yuhan Zhao[1, 2], Rui Chen[1*], Li Chen[2], Shuang Zhang[1], Qilong Han[1], Hongtao Song[1*]

[1] College of Computer Science and Technology, Harbin Engineering University
[2] Department of Computer Science, Hong Kong Baptist University
{csyhzhao,lichen}@comp.hkbu.edu.hk, {ruichen, zhangshuang, hanqilong, songhongtao}@hrbeu.edu.cn

Abstract

Intuitively, an ideal collaborative filtering (CF) model should learn from users' full rankings over all items to make optimal top-K recommendations. Due to the absence of such full rankings in practice, most CF models rely on pairwise loss functions to approximate full rankings, resulting in an immense performance gap. In this paper, we provide a novel analysis using the multiple ordinal classification concept to reveal the inevitable gap between a pairwise approximation and the ideal case. However, bridging the gap in practice encounters two formidable challenges: (1) none of the real-world datasets contains full ranking information; (2) there does not exist a loss function that is capable of consuming ranking information. To overcome these challenges, we propose a pseudo-ranking paradigm (PRP) that addresses the lack of ranking information by introducing pseudo-rankings supervised by an original noise injection mechanism. Additionally, we put forward a new ranking loss function designed to handle ranking information effectively. To ensure our method's robustness against potential inaccuracies in pseudo-rankings, we equip the ranking loss function with a gradient-based confidence mechanism to detect and mitigate abnormal gradients. Extensive experiments on four real-world datasets demonstrate that PRP significantly outperforms state-of-the-art methods.

Introduction

Collaborative filtering (CF) has been one of the most fundamental techniques in recommender systems due to its simplicity and effectiveness (Mao et al. 2021). It leverages user-item interactions to learn user preferences and make top-K recommendations (He et al. 2017; Wang et al. 2019; He et al. 2020). To find the top-K items, a CF model normally needs to generate a full ranking over the entire item universe. To fulfill this task, ideally, given a set of users U, one should derive a CF model that maximizes the posterior probability $\prod_{u \in U} p(\Theta \mid \Psi_u)$, where Θ denotes the set of the model's parameters that need to learn, and Ψ_u is user $u \in U$'s full ranking of all items. However, this ideal optimization goal is unattainable in real-world scenarios, as it is impractical for users to fully rank a vast number of items.

In practice, most CF methods (He et al. 2020; Wu et al. 2021; Yu et al. 2022) rely on pairwise loss functions to approximate full rankings. We refer to these as pairwise CF. Its core idea is to make positive item i_p (i.e., an item interacted by a user) more similar to user u than negative item i_n. For example, Bayesian personalized ranking (BPR) (Rendle et al. 2009) pairs each positive item with a negative item to form a pairwise relationship for training. Formally, a pairwise loss function forms a pairwise relationship $i_p >_u i_n$ (i.e., u prefers item i_p to i_n) as the ground truth. In this context, the optimization objective is to maximize the posterior probability $\prod_{(u,i_p,i_n) \in D} p(\Theta \mid i_p >_u i_n)$ based on a set of pairwise relationships, where D is the training dataset. Despite the strong performance of pairwise CF models, approximating full rankings with pairwise relationships introduces a significant gap compared to the ideal CF model. In this paper, we provide a novel analysis using the concept of multiple ordinal classification to *reveal the inherent gap between the pairwise approximation and the ideal case*. However, bridging the gap in practice encounters two formidable challenges:

- **None of the real-world datasets contains full ranking information**: Obtaining full rankings requires extensive user participation, a feat that is unachievable in most practical scenarios.
- **Lack of a loss function that is capable of handling ranking information**: Current CF loss functions focus on non-ranking relationships, such as pairwise or listwise (Xia et al. 2008), and are not designed to effectively and efficiently process ranking data.

To overcome the aforementioned challenges, we introduce a novel pseudo-ranking paradigm (PRP). For the first challenge, we propose a ranker module to generate a pseudo-ranking of given items. However, training the ranker demands supervised signal (i.e. ranking information), presenting a classic chicken-and-egg dilemma. We innovatively propose a noise injection mechanism to add different levels of noise to positive samples to construct a ranking in line with user preferences. The intuition is that when the injection noise is very small, the semantics of the constructed i_p^2 will not change significantly. When the injection noise is large, the semantics of the positive item i_p^3 will be seriously broken (He et al. 2018; Zhao et al. 2023a), and users are difficult to love the item. Accordingly, $i_p >_u i_p^2 >_u i_p^3$ is constructed as a supervision. For the second challenge, inspired by the multiple ordinal classification concept, we then put forward a ranking loss

*Corresponding authors

function grounded in classification principles. This function ensures that an item's score is proportional to its ranking order: the higher the rank, the higher the score. To further ensure our method's robustness against potential inaccuracies in pseudo-rankings provided by the ranker, we equip the ranking loss function with an original gradient-based confidence mechanism, which detects outliers by assessing gradient density. By reducing the weights of outliers, the impact of inaccurate ranking information on training can be alleviated. We summarize our main contributions as follows:

- To the best of our knowledge, we are the first to use the multiple ordinal classification concept to approximate ideal CF rather than traditional pairwise loss. We provide a new direction for future development of loss functions.
- We propose a novel ranker, which can generate pseudo-rankers according to user preferences through a noise injection mechanism, effectively solving the problem that the existing dataset does not contain ranking information.
- We propose a novel loss fucnion, which can directly deal with ranking data that traditional loss fucnions cannot directly deal with, and a gradient-based confidence mechanism, which can judge outliers by gradients and alleviate inaccurate ranking information.
- We conduct extensive experiments on four real-world datasets that confirm that PRP can achieve significant improvements over representative state-of-the-art methods. Moreover, when combined with PRP, a wide variety of mainstream CF models can consistently and substantially boost their performance.

Understanding Ideal Collaborative Filtering

According to maximum likelihood estimation, an ideal CF model should be trained by maximizing the posterior probability $\prod_{u \in U} p(\Theta \mid \Psi_u)$, where Ψ_u represents the full ranking of all items by user $u \in U$. However, in real-world scenarios, this ideal approach is impractical due to the absence of complete ranking information in datasets. Consequently, conventional CF models approximate this by using pairwise loss functions, shifting the optimization objective to maximizing the posterior probability $\prod_{(u,i_p,i_n) \in D} p(\Theta \mid i_p >_u i_n)$. A classic example of this is BPR:

$$\mathcal{L}_{BPR} = -\ln[\sigma(s_u(i_p) - s_u(i_n))], \tag{1}$$

where $s_u(\cdot)$ denotes the score of an item for user u, and $\sigma(\cdot)$ is the sigmoid function. Models trained with pairwise loss functions are termed pairwise CF models.

To model the ideal CF, inspired by works such as (Yang and Koyejo 2020; Li, Wu, and Burges 2007), we propose a novel approach: transforming CF ranking into a multiple ordinal classification problem. Specifically, we hypothesize the existence of N distinct labels $(top1, top2, top3, \ldots, topN)$. For each user u, the ideal CF model would assign each item to the appropriate label based on user preference. When making recommendations, these labels are treated as ordinal, with lower $top*$ values indicating higher recommendation priority. Figure 1 illustrates this concept. The classification loss in this context can be expressed as:

$$\mathcal{L}_{class} = -\log \frac{\exp(z_t)}{\sum_{v=1}^{N} \exp(z_v)}, \tag{2}$$

where z_t denotes the score of the target class. Under this framework, pairwise CF can be viewed as a process where each user u considers certain labels (e.g., $top1, top2, \ldots, topK$) as positive samples, while the remaining labels are negative. This comparison highlights that existing pairwise CF methods fall short of achieving ideal CF, exposing a significant gap. This observation raises a critical question: **Is there a connection between optimizing ideal CF and existing metrics (e.g., NDCG)**? While these metrics are not without flaws, they have proven effective in both practical and theoretical settings. Therefore, it is neither practical nor advisable to disregard them entirely (Pu et al. 2024). Understanding the relationship between ideal CF and traditional metrics is essential.

Theorem 1. *Optimizing ideal CF is a sufficient but not necessary condition for optimizing NDCG.*

Proof. For convenience, we analyze discounted cumulative gain (DCG, with NDCG being its normalized version). DCG is defined as:

$$DCG_K = \sum_{j=1}^{K} \frac{\mathbb{I}(i_j \in \mathcal{P})}{\log(j+1)}, \tag{3}$$

where $\mathbb{I}(\cdot)$ is an indicator function, and $\mathcal{P}$ represents the ground truth set of positive items.

Sufficient Condition (if part): If Ψ_u achieves ideal CF, it will rank all positive items at the top, thereby satisfying:

$$\{\Psi_u(1), \Psi_u(2), \ldots, \Psi_u(K)\} = \mathcal{P}. \tag{4}$$

This implies that ideal CF $\implies \max DCG_K$.

Necessary Condition (only if part): We can construct a different ranking Ψ'_u such that:

$$\forall (j < K), \Psi'_u(j) = \Psi_u(j+1), \Psi'_u(K) = \Psi_u(1). \tag{5}$$

Clearly, $\Psi'_u(1), \Psi'_u(2), \Psi'_u(3), \ldots \Psi'_u(K) = \mathcal{P}$, satisfying the maximum DCGK. However, $\Psi_u \neq \Psi'_u$, meaning that maximizing DCGK does not necessarily imply ideal CF. □

Methodology

In this section, we introduce a concrete implementation of the PRP. Note that these steps can be implemented by different methods and thus the overall paradigm is generic. The implementation hinges on two fundamental components: a ranker for generating pseudo-rankings and a ranking loss function tailored to effectively leverage the ranking information.

Ranker

The absence of ranking information in the dataset presents a significant challenge. To address this, we introduce a ranker module capable of generating accurate pseudo-rankings to facilitate training. The items to be ranked are represented as $M = \{i_1, i_2, i_3, \ldots, i_k\}$, which are randomly sampled from the dataset. The ranker aims to learn a user-specific mapping

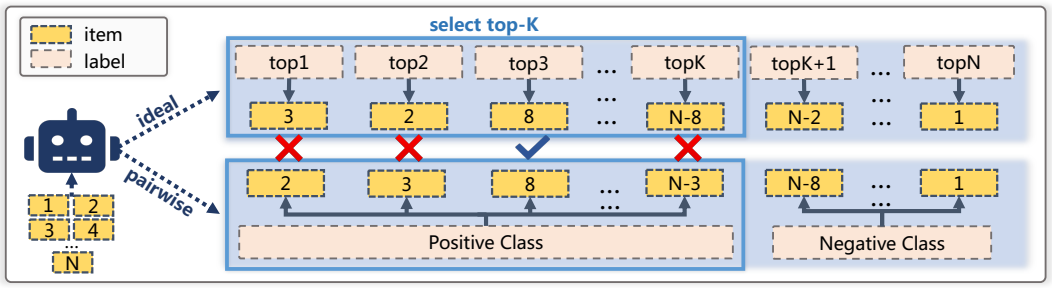


Figure 1: A comparison between ideal CF and pairwise CF.

$\gamma_u : M \to \pi_u$, where π_u denotes the ranking result. Specifically, $\pi_u(v)$ represents the item at position v in the ranking, and $\gamma_u(i)$ denotes the position of item i in π_u. This mapping is required to satisfy the condition: $\forall i_v, i_w \in M$, if $\gamma_u(i_v) < \gamma_u(i_w)$, then $i_v >_u i_w$ and $s_u(i_v) > s_u(i_w)$. We also impose an additional constraint aligned with a common assumption in CF: interacted items should receive higher scores than non-interacted ones (Pan and Chen 2013; Rendle et al. 2009; Mao et al. 2021).

To realize this objective, an intuitive approach is to design a neural network that takes user u and set M as input, outputting scores for all items to enable ranking:

$$\begin{aligned} s_u(i_1), s_u(i_2), \ldots, s_u(i_k) &= Rank_u(\mathbf{e}_u, \mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_k), \\ \pi_u &= argsort(s_u(i_1), s_u(i_2), \ldots, s_u(i_k)). \end{aligned} \tag{6}$$

The $Rank_u$ module can employ any network architecture. However, for efficiency, we use a simple multi-layer perceptron (MLP) in this work. Despite its simplicity, this approach lacks a specific training objective, making it insufficient to guarantee the acquisition of reliable ranking knowledge. Consequently, there is no direct assurance that the network's output will correspond to the desired ranking outcome. Therefore, an additional training objective is required to guide the network's learning process. However, the lack of ranking information in the dataset creates a classic chicken-and-egg dilemma.

To circumvent this issue, we propose a noise injection mechanism to construct ranking signals based on interacted items, denoted as $\mathbf{e}_p^1$. The core intuition is that injecting a small amount of noise into an item $\mathbf{e}_p$ produces a new item $\mathbf{e}_p^2$ with slightly altered semantics, reflecting a slight decrease in user preference. Given the substantial noise magnitude, resulting in $\mathbf{e}_p^3$, the item's semantics degrade significantly, leading to a further reduction in user preference (He et al. 2018; Zhao et al. 2023a). We formalize this by applying two different magnitudes of noise to the original item $\mathbf{e}_p$, generating a set $\mathcal{P}_n$:

$$\begin{aligned} \mathcal{P}_n &= \{i_p^m | \mathbf{e}_p^m = \mathcal{T}_{\theta_m}(\mathbf{e}_p), m = 1, 2, 3\}, \\ \mathcal{T}_{\theta_m}(\mathbf{e}_p) &= \mathbf{e}_p + \theta_m \epsilon_u, \end{aligned} \tag{7}$$

$\mathcal{T}_{\theta_m}$ represents the noise injection function, ϵ_u denotes the noise, and θ_m denotes the noise magnitude. We set $\theta_1 = 0, \theta_3 \gg \theta_2$. While random noise injection is a straightforward idea, it is unreliable. If a user's preferences are robust, significant noise is necessary to alter their ranking preferences. Consequently, the noise distribution should be user-dependent. Assuming no additional prior knowledge, we model the noise as following a Gaussian distribution, with the parameters generated as follows:

$$\mu_u = MLP_1(\mathbf{e}_u), \tag{8}$$

$$\log \sigma_u^2 = MLP_2(\mu_i). \tag{9}$$

We opt to fit $\log \sigma_u^2$ instead of σ_u^2 directly due to the non-negativity constraint on σ_u^2, which would otherwise require an activation function (Yang et al. 2023). Once we obtain the mean μ_u and variance σ_u^2, the latent noise ϵ_u is generated by sampling from $\mathcal{N}(\mu_u, \sigma_u^2)$. Direct optimization of this process is infeasible due to its non-differentiable nature. To address this, we employ the reparameterization trick (Chen et al. 2022):

$$\epsilon_u = \mu_u + \sigma_u \cdot \eta, \tag{10}$$

$$\eta \sim \mathcal{N}(0, \mathbf{I}), \tag{11}$$

where $\eta \sim \mathcal{N}(0, \mathbf{I})$, with $\mathbf{I}$ as the identity matrix, is Gaussian noise. The resulting ranking is then processed by our ranking loss function, denoted as $\mathcal{L}_p$, to supervise the ranker's training:

$$\mathcal{L}_p = \mathcal{L}_{rank}^{\alpha}(u, \gamma_u(\mathcal{P}_n)). \tag{12}$$

Ranking Loss Function

With the ranker generating pseudo-ranking data, the challenge shifts to designing an appropriate loss function. Existing loss functions, typically following a pairwise paradigm, are inadequate for capturing complex ranking information. Our ideal CF theory suggests that recommendation tasks can be viewed as multiple ordinal classification problems. While a classification loss might seem applicable, the absence of explicit labels such as $(top1, top2, top3, \ldots, topN)$ renders this approach unfeasible. Nevertheless, the classification perspective inspires the design of our loss function. In classification tasks, the classical objective is to output scores such that the score for the target class exceeds that for non-target classes:

$$\max \sum_{v=1}^{k} \sum_{w=1, w \neq v}^{k} \max(z(v) - z(w), 0). \tag{13}$$

This formulation, however, only addresses the classification task and fails to account for the ordinal nature of ranking. Moreover, without label information, computing $z(\cdot)$ is impractical. To incorporate ordinal properties, we reformulate the problem to ensure that $\forall v < w, s_u(\pi_u(v)) > s_u(\pi_u(w))$:

$$\mathcal{L}_{rank} = \sum_{v=1}^{k} \sum_{w=v+1}^{k} \max(s_u(\pi_u(w)) - s_u(\pi_u(v)), 0) \tag{14}$$

$$= \sum_{v=1}^{k} \max(\max_{w \neq v}\{s_u(\pi_u(w))\} - s_u(\pi_u(v)), 0). \tag{15}$$

Equation 14 effectively decomposes the ranking problem into multiple sub-ranking problems. For example, if the ranking is (i_3, i_1, i_2), the scores should satisfy the sub-rankings $s_u(i_3) > s_u(i_1) > s_u(i_2)$ and $s_u(i_1) > s_u(i_2)$. By exploiting the principle of transitivity, we establish that if the conditions $s_u(i_3) > s_u(i_1)$ and $s_u(i_1) > s_u(i_2)$ hold, we can directly infer the ordering $s_u(i_3) > s_u(i_1) > s_u(i_2)$, which ultimately leads to Equation 15. However, the non-differentiability of the max function renders it unsuitable as a loss function, necessitating further modification:

$$\mathcal{L}_{rank} \approx \sum_{v=1}^{k} \log(1 + \exp(\max_{w \neq v}\{s_u(\pi_u(w))\} - s_u(\pi_u(v)))) \tag{16}$$

$$\approx \sum_{v=1}^{k-1} \log(1 + \exp(s_u(\pi_u(v+1)) - s_u(\pi_u(v)))). \tag{17}$$

In the derivation above, we first replace the non-differentiable $max(\cdot, 0)$ with softplus function $log(1 + \exp(\cdot))$. Thanks to the ranker's result, we can then simply use $v + 1$ to represent $\max_{w \neq v}\{s_u(\pi_u(w))\}$. This substitution eliminates the need for significant additional computational effort to find $\max_{w \neq v}\{s_u(\pi_u(w))\}$, thereby significantly enhancing the algorithm's efficiency. For instance, in the ranking (i_3, i_1, i_2) ensuring $s_u(i_3) > s_u(i_1)$ and $s_u(i_1) > s_u(i_2)$ naturally constructs $s_u(i_3) > s_u(i_1) > s_u(i_2)$.

Given that the pseudo-ranking from the ranker may not always be accurate, we introduce a confidence mechanism. Specifically, we introduce a confidence coefficient α_v. If a ranking is deemed inaccurate, we reduce its corresponding coefficient to lessen its impact on the overall loss:

$$\mathcal{L}^{\alpha}_{rank} = \sum_{v=1}^{k-1} \alpha_v \log(1 + \exp(s_u(\pi_u(v+1)) - s_u(\pi_u(v)))). \tag{18}$$

An intuitive approach to designing α_v would be to set it as a hyperparameter, but this would require extensive expert knowledge and dataset-specific tuning. Inspired by previous works (Gao et al. 2022; Li, Liu, and Wang 2019; Wang et al. 2022b), we propose learning α_v based on gradient. If a sub-ranking produces a gradient that deviates from typical training values, it is likely incorrect, and we accordingly reduce its weight. First, we calculate the absolute gradient value $g_v = |\nabla_{\pi_u(v)}\mathcal{L}_{rank}|$ for all sub-rankings and identify the largest gradient value $max(g_v)$. We then divide the gradients into ten groups $G_1, G_2, \ldots, G_{10}$ according to the interval $[0, max(g_v)]$. For a particular gradient g_v, if it belongs to group G_v, we calculate the number N_v of gradients in that group and the ratio of the sum of gradients across all groups:

$$\alpha_v = \frac{N_v}{\sum_{y=0}^{10} N_y}. \tag{19}$$

This approach ensures that if the number of gradients in a group is small, indicating that they are outliers, α_v is reduced, aligning with our objective. This concept can be interpreted as a form of statistical gradient density. Given that the gradient is inherently a continuous quantity, direct calculation of the density might yield $1/N$ for each gradient density. To address this, we approximate the density by grouping the gradients. The final loss function is thus formulated as follows:

$$\mathcal{L} = \mathcal{L}^{\alpha}_{rank}(u, \pi_u) + \beta\mathcal{L}_p, \tag{20}$$

where $\mathcal{L}_{rank}(u, \pi_u)$ processes the ranking information from the ranker. $\mathcal{L}_p$ aids the ranker in learning ranking knowledge. β is a hyperparameter to control the relative weights.

Discussion

Our newly proposed loss function offers a more effective approximation of the ideal CF. This prompts further reflection: *is there an intuitive connection between our loss function and the classic loss?*

Theorem 2. *BPR is a special case of ranking loss where $k = 2$. BPR with hard negative sampling is a special case of longest sub-ranking loss where $k > 2$.*

Proof.

$$\mathcal{L}_{rank} = -\sum_{v=1}^{k-1} \log\left(\frac{1}{\exp(s(\pi(v)) - s(\pi(v+1))) + 1}\right) \tag{21}$$

$$= -\sum_{v=1}^{k-1} \log(\sigma(s(\pi(v)) - s(\pi(v+1)))) \tag{22}$$

$$= -\sum_{v=1}^{k-1} \log(\sigma(s(\pi(v)) - \max_{w \neq v}\{s(\pi(w))\})). \tag{23}$$

We omit the u subscript for brevity. It shows that if we set $k = 2$, our method is equivalent to BPR, where $\pi_u(v)$ is a positive sample, and $\max_{w \neq v}\{s_u(\pi_u(w))\}$ is equal to a negative sample. When $k > 2$, obviously, BPR only considers a certain longest sub-ranking, where $\pi_u(v)$ is still a positive sample, $\max_{w \neq v}\{s_u(\pi_u(w))\}$ is equivalent to performing **hard negative sampling** from $M - \{\pi_u(v)\}$ and selecting the hardest sample.

□

Dataset	Methods	HR@10	HR@20	Recall@10	Recall@20	NDCG@10	NDCG@20
ML-1M	**BPR**	0.7419	0.8374	0.1635	0.2502	0.2554	0.2619
	SRNS	0.7611	0.8533	0.1746	0.2618	0.2639	0.2687
	MixGCF	0.7601	0.8483	0.1739	0.2604	0.2657	0.2702
	UIB	0.7346	0.8318	0.1559	0.2406	0.2453	0.2520
	SimpleX	0.7586	0.8518	0.1709	0.2573	0.2626	0.2675
	ANS	0.7675	0.8583	0.1759	0.2625	0.2624	0.2683
	PRP	**0.7796**	**0.8657**	**0.1866**	**0.2765**	**0.2822**	**0.2874**
	Improvement	**5.08%**	**3.38%**	**14.13%**	**10.51%**	**10.49%**	**9.74%**
Gowalla	**BPR**	0.1989	0.2763	0.0938	0.1389	0.0674	0.0804
	SRNS	0.2295	0.3177	0.1171	0.1711	0.0814	0.0972
	MixGCF	0.2401	0.3254	0.1189	0.1728	0.0853	0.1008
	UIB	0.2275	0.3161	0.1140	0.1695	0.0794	0.0954
	SimpleX	0.2192	0.3082	0.1074	0.1617	0.0736	0.0892
	ANS	0.2467	0.3317	0.1206	0.1756	0.0869	0.1027
	PRP	**0.2528**	**0.3395**	**0.1229**	**0.1777**	**0.0884**	**0.1040**
	Improvement	**27.10%**	**22.87%**	**31.02%**	**27.93%**	**31.16%**	**29.35%**
Foursquare	**BPR**	0.1717	0.2336	0.0262	0.0381	0.0283	0.0328
	SRNS	0.1948	0.2596	0.0314	0.0418	0.0351	0.0392
	MixGCF	0.1994	0.2752	0.0316	0.0458	0.0351	0.0405
	UIB	0.1662	0.2299	0.0265	0.0375	0.0280	0.0322
	SimpleX	0.1801	0.2650	0.0267	0.0421	0.0267	0.0329
	ANS	0.1958	**0.2798**	0.0303	0.0461	0.0321	0.0386
	PRP	**0.2124**	0.2752	**0.0332**	**0.0461**	**0.0367**	**0.0413**
	Improvement	**23.70%**	**17.81%**	**26.72%**	**20.73%**	**29.68%**	**25.91%**
Yelp	**BPR**	0.1537	0.2398	0.0427	0.0717	0.0334	0.0432
	SRNS	0.1834	0.2779	0.0520	0.0862	0.0406	0.0522
	MixGCF	0.1891	0.2781	0.0538	0.0871	0.0424	0.0536
	UIB	0.1592	0.2504	0.0445	0.0756	0.0350	0.0455
	SimpleX	0.1850	0.2780	0.0519	0.0861	0.0410	0.0524
	ANS	0.1844	0.2777	0.0524	0.0862	0.0414	0.0527
	PRP	**0.2000**	**0.2940**	**0.0577**	**0.0928**	**0.0464**	**0.0581**
	Improvement	**30.12%**	**22.60%**	**35.13%**	**29.43%**	**38.92%**	**34.49%**

Table 1: The performance comparison between PRP and other methods. The best results are boldfaced, and the improvements are significant over the BPR under a two-sided t-test with $p < 0.05$.

Experiments

In this section, we conduct comprehensive experiments to demonstrate the performance of PRP.

Experimental Settings

Datasets. We evaluate PRP on four widely used public real-world datasets: (1) **MovieLens** contains user ratings on movies. We use 1M versions and treat rating movies as interacted items. (2) **Yelp** contains user reviews of restaurants and bars after Jan.1st, 2018. (3) **Gowalla** is a check-in dataset where users share their locations. (4) **Foursquare** consists of check-ins in NYC and Tokyo over approximately 10 months. These datasets have different statistical properties, which can reliably validate the performance of a model (Chin, Chen, and Cong 2022).

Baselines. To assess the effectiveness of PRP, we compare it against several competitive methods that represent different research directions: **BPR** (Rendle et al. 2009), **SimpleX** (Mao et al. 2021), **UIB** (Zhuo et al. 2022), **SRNS** (Ding et al. 2020),**MixGCF** (Huang et al. 2021), **ANS** (Zhao et al. 2023b). Furthermore, to validate the broad applicability of PRP, we integrate it with various mainstream CF models, including: **MF** (Rendle et al. 2009), **NGCF** (Wang et al. 2019), and **LightGCN** (He et al. 2020).

All data preprocessing, dataset partitioning and implementation of the methods are carried out using the RecBole v1.1.1 framework (Zhao et al. 2021).

Overall Performance

Table 1 presents the performance results of PRP compared to several baseline methods. We also integrated PRP with various mainstream CF models, as shown in Table 2. The

Dataset	Metric	MF		LightGCN		NGCF	
		w/o	w	w/o	w	w/o	w
Yelp	Recall@10	0.0427	**0.0577**	0.0543	**0.0655**	0.0439	**0.0514**
	Recall@20	0.0717	**0.0928**	0.0884	**0.1044**	0.0738	**0.0834**
	HR@10	0.1537	**0.2000**	0.1896	**0.2210**	0.1593	**0.1811**
	HR@20	0.2398	**0.2940**	0.2847	**0.3232**	0.2460	**0.2705**
	NDCG@10	0.0334	**0.0464**	0.0432	**0.0525**	0.0345	**0.0406**
	NDCG@20	0.0432	**0.0581**	0.0546	**0.0655**	0.0447	**0.0512**
	Improvement		**31.78%**		**18.38%**		**14.33%**
Gowalla	Recall@10	0.0938	**0.1236**	0.1280	**0.1360**	0.1011	**0.1225**
	Recall@20	0.1389	**0.1787**	0.1849	**0.1967**	0.1490	**0.1745**
	HR@10	0.1989	**0.2534**	0.2553	**0.2712**	0.2110	**0.2468**
	HR@20	0.2763	**0.3366**	0.3432	**0.3626**	0.2900	**0.3321**
	NDCG@10	0.0674	**0.0896**	0.0917	**0.0981**	0.0722	**0.0856**
	NDCG@20	0.0804	**0.1052**	0.1079	**0.1155**	0.0859	**0.1010**
	Improvement		**29.91%**		**6.42%**		**17.65%**

Table 2: Experimental results of different CF models with (w) or without (w/o) the PRP.

reported improvements are statistically significant under a two-sided t-test with $p < 0.05$. Our observations are as follows: (1) PRP consistently achieves state-of-the-art results across almost all cases. Notably, it significantly outperforms traditional methods like BPR, with the highest improvement of 38.9% observed on the Yelp dataset. This underscores the substantial performance gains achieved by incorporating ranking information into the model. (2) Negative sampling (NS) methods achieve sub-optimal results. While NS methods can capture information from sub-rankings where $k > 2$, BPR is limited to $k = 2$. This ability of NS methods to utilize more extensive sub-ranking information explains their improved performance over BPR. However, NS methods still overlook other sub-ranking information, leading to their lower performance compared to PRP. (3) Methods such as ANS and MixGCF generate harder synthetic negative samples i_n^{syn}, leading to impressive results. This is reasonable, as generating harder i_n^{syn} implicitly aligns with the ranking information $i_n^{syn} > i_n$. However, synthesizing these negative samples requires careful design and involves significant computational overhead. Moreover, since only a specific ranking is considered, there remains a performance gap when compared to PRP. (4) Methods such as SimpleX use large numbers of negative samples to construct more pairwise interactions. Although they have the potential to recover ranking information if we have $\Omega(|U|N^2)$ accurate pairs (Alon et al. 1994; Maystre and Grossglauser 2017), this is impractical in real-world scenarios (Liu et al. 2021; Lin et al. 2024). (5) As shown in Table 2, integrating PRP leads to significant performance improvements for all base models in all cases. Surprisingly, even with a simple BPR-MF model, PRP achieves performance comparable to more sophisticated GNN models. This demonstrates the superior effectiveness and broad applicability of the PRP approach.

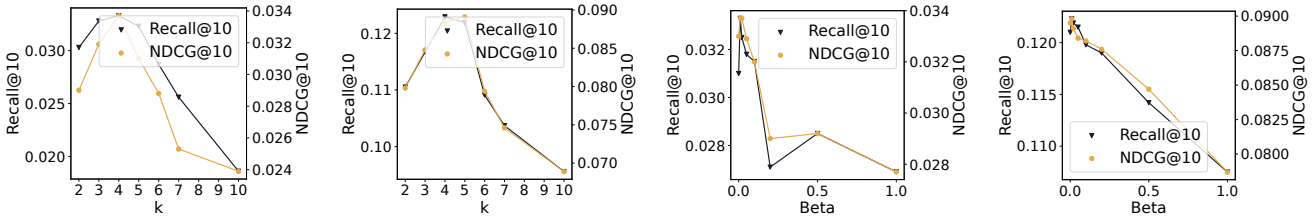
In-depth Analysis

Ranking Length. The parameter k determines the number of items to be ranked. Intuitively, the larger the k, the closer it approximates the full ranking needed for ideal CF, potentially enhancing performance. However, as k increases, ranking becomes more challenging. Due to data quality limitations, model capacity, and inherent noise, it is impractical to let k increase indefinitely. Experimental results presented in Figure 2(a), 2(b) confirm this, showing a sharp performance drop for $k > 5$, indicating the ranker's inability to accurately rank and the detrimental impact of incorrect rankings on model performance. Additionally, a larger k affects the method's efficiency, necessitating careful selection based on the specific application scenario.

Supervised Magnitude. The parameter β adjusts the supervised magnitude of L_p on the ranker. Experimental results, presented in Figure 2(c), 2(d), show that performance initially increases with β but decreases beyond a certain point. Initially, a higher β allows L_p to provide more information for ranker training, improving pseudo-ranking quality. However, an excessively high β simplifies the noise injection ranking problem too much, making L_p optimization easier than the recommendation task. This shifts the model's focus away from the recommendation task, degrading performance.

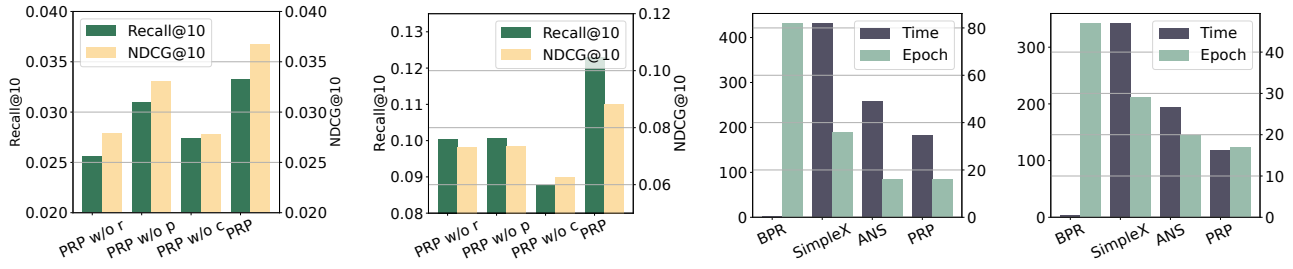
Ablation Study. We analyze the effectiveness of different components through the following variants: (1) PRP without ranker (PNN w/o r), (2) PRP without $\mathcal{L}_p$ (PNN w/o p), (3) PRP without confidence (PNN w/o c). The results, presented in Figure 3(a), 3(b) indicate that each component positively contributes to model performance. In essence, these three components ensure that the model effectively utilizes accurate ranking information, underscoring the importance of constructing a reasonable ranking. If the ranking is incorrect, the ranking loss becomes meaningless.

Efficiency Analysis. Consistent with previous works (Mao



(a) Foursquare (b) Gowalla (c) Foursquare (d) Gowalla

Figure 2: Experiments to demonstrate the influence of ranking length and supervised magnitude.



(a) Foursquare (b) Gowalla (c) Yelp (d) Gowalla

Figure 3: Experimental results to demonstrate the ablation study and efficiency analysis.

et al. 2021; Wang et al. 2022a; Zhao et al. 2023b), we compare the efficiency of PRP with other representative methods (BPR, SimpleX, and ANS). Figure 3(c), 3(d) present the average training time (sec.) per epoch and the number of epochs required to converge. As anticipated, BPR is the most efficient due to its simplicity. SimpleX, on the other hand, shows the least efficiency, attributed to its dependence on massive negative samples and filtering of uninformative ones. PRP emerges as the second most efficient method. Given its outstanding performance, we assert that PRP stands out as a highly desirable choice.

Related Work

Pairwise loss functions lie in the core of existing implicit CF. Since they aim to maximize the posterior probability $\prod_{(u,i_p,i_n)\in D} p(\Theta \mid i_p >_u i_n)$, there have been two major research directions. The first line focuses on how to effectively form pairwise relationships (i.e., $>_u$). BPR (Rendle et al. 2009; Sun et al. 2020; Yu et al. 2021) assumes that an interacted item is more similar to the user than a randomly sampled uninteracted item. CML (Hsieh et al. 2017) encodes not only users' preferences but also the user-user and item-item similarity. SimpleX (Mao et al. 2021) introduces massive negative samples and removes uninformative items. UIB (Zhuo et al. 2022) introduces a user interest boundary to penalize samples crossing a threshold. The second line aims to find better negative items to contrast positive items. PNN (Zhao et al. 2024) introduces a neutral class and a novel positive-neutral-negative learning paradigm. DNS (Zhang et al. 2013) selects negative items that are more similar to a user. IRGAN (Wang et al. 2017) utilizes a generative adversarial network to compute the probabilities of negative samples by a min-max game. ReinforcedNS (Ding et al. 2019) uses reinforcement learning to guide the sampling process. MixGCF (Huang et al. 2021) integrates information from a graph structure and positive samples to enhance the hardness of negative items. GDNS (Zhu et al. 2022) develops a gain-aware function to calculate the probability of an item being a real negative sample. SRNS (Ding et al. 2020) favors low-variance samples to avoid false negative samples. DENS (Lai et al. 2023) disentangles relevant and irrelevant factors of samples to select appropriate negative samples. ANS (Zhao et al. 2023b) proposes to generate synthetic negative samples to improve performance.

Conclusion

In this work, we creatively apply the concept of multiple ordinal classification to highlight the inherent gap between pairwise methods and the ideal CF. Based on this theory, we introduce an innovative pseudo-ranking paradigm (PRP). Our implementation features a novel ranker that generates pseudo-rankings, supervised by a noise mechanism. We also propose a new loss function to handle ranking information effectively. Furthermore, this loss is enhanced with a gradient-based confidence mechanism that mitigates abnormal rankings. Extensive experiments demonstrate that PRP significantly outperforms state-of-the-art methods. This work marks a significant advancement in CF modeling, shifting the focus from pairwise comparisons to a more comprehensive ranking approach, and represents a major step toward realizing ideal CF models.

Acknowledgements

This work was supported by the Heilongjiang Key R&D Program of China under Grant No. GA23A915 and partially supported by Hong Kong Baptist University IG-FNRA project (RC-FNRA-IG/21-22/SCI/01) and Key Research Partnership Scheme (KRPS/23-24/02).

References

Alon, N.; Bollobas, B.; Brightwell, G.; and Janson, S. 1994. Linear Extensions of a Random Partial Order. *The Annals of Applied Probability*, 4.

Chen, J.; Lian, D.; Jin, B.; Huang, X.; Zheng, K.; and Chen, E. 2022. Fast Variational Autoencoder with Inverted Multi-Index for Collaborative Filtering. In *WWW*, 1944–1954.

Chin, J. Y.; Chen, Y.; and Cong, G. 2022. The Datasets Dilemma: How Much Do We Really Know About Recommendation Datasets? In *WSDM*, 141–149.

Ding, J.; Quan, Y.; He, X.; Li, Y.; and Jin, D. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *IJCAI*, 2230–2236.

Ding, J.; Quan, Y.; Yao, Q.; Li, Y.; and Jin, D. 2020. Simplify and Robustify Negative Sampling for Implicit Collaborative Filtering. *NeurIPS*, 33.

Gao, Y.; Du, Y.; Hu, Y.; Chen, L.; Zhu, X.; Fang, Z.; and Zheng, B. 2022. Self-Guided Learning to Denoise for Robust Recommendation. In *SIGIR*, 1412–1422.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*, 639–648.

He, X.; He, Z.; Du, X.; and Chua, T.-S. 2018. Adversarial Personalized Ranking for Recommendation. In *SIGIR*, 355–364.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural Collaborative Filtering. In *WWW*, 173–182.

Hsieh, C.; Yang, L.; Cui, Y.; Lin, T.; Belongie, S. J.; and Estrin, D. 2017. Collaborative Metric Learning. In *WWW*, 193–201.

Huang, T.; Dong, Y.; Ding, M.; Yang, Z.; Feng, W.; Wang, X.; and Tang, J. 2021. MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems. In *KDD*, 665–674.

Lai, R.; Chen, L.; Zhao, Y.; Chen, R.; and Han, Q. 2023. Disentangled Negative Sampling for Collaborative Filtering. In *WSDM*, 96–104.

Li, B.; Liu, Y.; and Wang, X. 2019. Gradient Harmonized Single-Stage Detector. In *AAAI*, 8577–8584.

Li, P.; Wu, Q.; and Burges, C. 2007. McRank: Learning to Rank Using Multiple Classification and Gradient Boosting. *NeurIPS*, 20.

Lin, G.; Gao, C.; Zheng, Y.; Li, Y.; Chang, J.; Niu, Y.; Song, Y.; Gai, K.; Li, Z.; Jin, D.; et al. 2024. Inverse Learning with Extremely Sparse Feedback for Recommendation. In *WSDM*, 396–404.

Liu, W.; Su, J.; Chen, C.; and Zheng, X. 2021. Leveraging Distribution Alignment via Stein path for Cross-Domain Cold-Start Recommendation. *NeurIPS*, 19223–19234.

Mao, K.; Zhu, J.; Wang, J.; Dai, Q.; Dong, Z.; Xiao, X.; and He, X. 2021. SimpleX: A Simple and Strong Baseline for Collaborative Filtering. In *CIKM*, 1243–1252.

Maystre, L.; and Grossglauser, M. 2017. Just Sort It! A Simple and Effective Approach to Active Preference Learning. In *ICML*, 2344–2353.

Pan, W.; and Chen, L. 2013. GBPR: Group Preference based Bayesian Personalized Ranking for One-Class Collaborative Filtering. In *IJCAI*, 2691–2697.

Pu, Y.; Chen, X.; Huang, X.; Chen, J.; Lian, D.; and Chen, E. 2024. Learning-Efficient Yet Generalizable Collaborative Filtering for Item Recommendation. In *ICML*.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*, 452–461.

Sun, J.; Guo, W.; Zhang, D.; Zhang, Y.; Regol, F.; Hu, Y.; Guo, H.; Tang, R.; Yuan, H.; He, X.; et al. 2020. A Framework for Recommending Accurate and Diverse Items using Bayesian Graph Convolutional Neural Networks. In *KDD*, 2030–2039.

Wang, C.; Yu, Y.; Ma, W.; Zhang, M.; Chen, C.; Liu, Y.; and Ma, S. 2022a. Towards Representation Alignment and Uniformity in Collaborative Filtering. In *KDD*, 1816–1825.

Wang, J.; Yu, L.; Zhang, W.; Gong, Y.; Xu, Y.; Wang, B.; Zhang, P.; and Zhang, D. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *SIGIR*, 515–524.

Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T.-S. 2019. Neural Graph Collaborative Filtering. In *SIGIR*, 165–174.

Wang, Y.; Wang, H.; Shen, Y.; Fei, J.; Li, W.; Jin, G.; Wu, L.; Zhao, R.; and Le, X. 2022b. Semi-Supervised Semantic Segmentation using Unreliable Pseudo-Labels. In *CVPR*, 4248–4257.

Wu, J.; Wang, X.; Feng, F.; He, X.; Chen, L.; Lian, J.; and Xie, X. 2021. Self-Supervised Graph Learning for Recommendation. In *SIGIR*, 726–735.

Xia, F.; Liu, T.-Y.; Wang, J.; Zhang, W.; and Li, H. 2008. Listwise Approach to Learning to Rank: Theory and Algorithm. In *ICML*, 1192–1199.

Yang, F.; and Koyejo, S. 2020. On the Consistency of Top-K Surrogate Losses. In *ICML*, 10727–10735.

Yang, Y.; Wu, Z.; Wu, L.; Zhang, K.; Hong, R.; Zhang, Z.; Zhou, J.; and Wang, M. 2023. Generative-Contrastive Graph Learning for Recommendation. In *SIGIR*, 1117–1126.

Yu, J.; Yin, H.; Gao, M.; Xia, X.; Zhang, X.; and Viet Hung, N. Q. 2021. Socially-Aware Self-Supervised Tri-training for Recommendation. In *KDD*, 2084–2092.

Yu, J.; Yin, H.; Xia, X.; Chen, T.; Cui, L.; and Nguyen, Q. V. H. 2022. Are Graph Augmentations Necessary? Simple Graph Contrastive Learning for Recommendation. In *SIGIR*, 1294–1303.

Zhang, W.; Chen, T.; Wang, J.; and Yu, Y. 2013. Optimizing Top-N Collaborative Filtering via Dynamic Negative Item Sampling. In *SIGIR*, 785–788.

Zhao, P.; Pan, Y.; Li, X.; Chen, X.; Tsang, I. W.; and Liao, L. 2023a. Coarse-to-Fine Contrastive Learning on Graphs. *IEEE Transactions on Neural Networks and Learning Systems*.

Zhao, W. X.; Mu, S.; Hou, Y.; Lin, Z.; Chen, Y.; Pan, X.; Li, K.; Lu, Y.; Wang, H.; Tian, C.; et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *CIKM*, 4653–4664.

Zhao, Y.; Chen, R.; Han, Q.; Song, H.; and Chen, L. 2024. Unlocking the Hidden Treasures: Enhancing Recommendations with Unlabeled Data. In *RecSys*, 247–256.

Zhao, Y.; Chen, R.; Lai, R.; Han, Q.; Song, H.; and Chen, L. 2023b. Augmented Negative Sampling for Collaborative Filtering. In *RecSys*, 256–266.

Zhu, Q.; Zhang, H.; He, Q.; and Dou, Z. 2022. A Gain-Tuning Dynamic Negative Sampler for Recommendation. In *WWW*, 277–285.

Zhuo, J.; Zhu, Q.; Yue, Y.; and Zhao, Y. 2022. Learning Explicit User Interest Boundary for Recommendation. In *WWW*, 193–202.