Implicit Feedbacks are Not Always Favorable: Iterative Relabeled One-Class Collaborative Filtering against Noisy Interactions

Zitai Wang[1,2], Qianqian Xu[3*], Zhiyong Yang[1,2], Xiaochun Cao[1,4], Qingming Huang[3,5,6,7*]
[1]State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, CAS, Beijing, China
[4]School of Cyber Science and Technology, Sun Yat-sen University, Shenzhen, China
[5]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China
[6]Key Laboratory of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, Beijing, China
[7]Artificial Intelligence Research Center, Peng Cheng Laboratory, Shenzhen, China
wangzitai@iie.ac.cn,xuqianqian@ict.ac.cn,{yangzhiyong,caoxiaochun}@iie.ac.cn,qmhuang@ucas.ac.cn

ABSTRACT

Due to privacy concerns, there is a rising favor in Recommender System community for the One-class Collaborative Filtering (OCCF) framework, which predicts user preferences only based on binary implicit feedback (*e.g.,* click or not-click, rated or unrated). The major challenge in OCCF problem stems from the inherent noise in implicit interaction. Previous approaches have taken into account the noise in unobserved interactions (*i.e.,* not-click only means a missing value, rather than negative feedback). However, they generally ignore the noise in observed interactions (*i.e.,* click does not necessarily represent positive feedback), which might induce performance degradation. To attack this issue, we propose a novel iteratively relabeling framework to jointly mitigate the noise in both observed and unobserved interactions. As the core of the framework, the iterative relabeling module exploits the self-training principle to dynamically generate pseudo labels for user preferences. The downstream module for a recommendation task is then trained with the refreshed labels where the noisy patterns are largely alleviated. Finally, extensive experiments on three real-world datasets demonstrate the effectiveness of our proposed methods.

CCS CONCEPTS

• **Information systems → Collaborative filtering**; • **Human-centered computing → Social recommendation**; • **Computing methodologies → Learning from implicit feedback**.

KEYWORDS

Collaborative Filtering, Implicit Feedback, Preference Learning

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MM '21, October 20–24, 2021, Virtual Event, China
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8651-7/21/10…$15.00
https://doi.org/10.1145/3474085.3475446

ACM Reference Format:
Zitai Wang, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, Qingming Huang. 2021. Implicit Feedbacks are not Always Favorable: Iterative Relabeled One-Class Collaborative Filtering against Noisy Interactions. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3475446

1 INTRODUCTION

Nowadays, a large amount of multimedia content (*e.g.,* microblogging, music, short videos) has surged into the Internet, which greatly improves our online experience on online platforms. However, the constantly growing information might be a double-edged sword in the end. As the information explodes, it becomes difficult for users to discover interesting content, which makes them dissatisfied, decrease activities and even drop out of the platforms [16]. Therefore, it has become crucial for the platforms to analyze personalized user performance and then customize the content accurately. As a popular way to provide personalized content, Recommender System (RS) has become a standard component of large amounts of online platforms in recent years [5, 7, 36].

Among RS approaches, Collaborative Filtering (CF) is known to be one of the most popular and effective representatives [15]. The basic idea behind CF is to predict user preferences based on their historical feedback [31, 38]. Since involving no user information, CF framework is inherently friendly to privacy concerns and some scenarios such as cold-start. Early research on CF focuses on explicit feedback, such as 1-5 star ratings in Netflix [46]. Recently, more attention is shifting towards an alternative kind of information called implicit feedback which only contains user-item interactions (*e.g.,* click, browsing, bookmark history) [11]. As a significant advantage, implicit feedback covers abundant collaborative information in a much simpler manner than explicit forms. Correspondingly, there arises a specific variant of CF called **One-class Collaborative Filtering** (OCCF) which directly deals with such implicit feedback when it is organized in a binary form (*e.g.,* click or not-click, rated or unrated) [4, 25].

One main challenge in OCCF problem stems from the inevitable noise in implicit interactions. The classical OCCF approach has realized the fact that unobserved interactions are a mixture of negative feedback and missing values. For example, Pan et al. [25] develop a sampling strategy positing that if a user has clicked more items, her unobserved interactions are negative with higher probability. Lin et al. [20] design a weighting scheme that explicitly models the likelihood of an unobserved interaction being negative feedback. Chen et al. [4] propose a ranking-based regularizer, which averts the estimation for the noise in unobserved interactions. He et al. [9] center on utilizing DNNs to model the implicit interactions.

Despite their great attention to the noise in unobserved interactions, the existing studies generally ignore the noise from observed interactions, which might as well induce performance degradation. More specifically, a label 1 only represents the observed interaction, rather than explicit user preference. For example, we might click a news page due to the exaggerated headline only to find that the content was extremely boring. We might purchase some items just as gifts, and such behaviors did not represent our own preference. Considering the above-mentioned instances, we argue that the noise in observed interactions is also non-negligible.

But, how should we eliminate both types of noise in implicit interactions? When we assume both observed and unobserved interactions are noisy, **no valid supervision information is available.** Meanwhile, we only have access to historical feedback, which is different from common denoising tasks where auxiliary information could help recognizes noise. To tackle the challenges above, we take inspiration from self-supervised learning [12] and propose a novel iterative relabeling framework to mitigate the noise in implicit feedback. Specifically, the proposed framework consists of an iterative relabeling module and a downstream module. As the core of the framework, the iterative relabeling module exploits the self-training principle to dynamically generate promising labels for user preferences. On one hand, as the generated labels become less noisy, the predicted scores can better indicate user preferences. On the other hand, with more reliable predictions, the module can better filter out the noise in implicit interactions. The iterative relabeling process terminates when the labels become stable. Based on these pseudo labels, the downstream module could better capture user preferences and thus improves the overall ranking performance.

The main contributions of this work are listed as follows:

- We propose a novel iterative relabeling framework for OCCF to eliminate the noise in both observed and unobserved interactions.
- The core of the framework lies the interactive relabeling module which aims to generate promising labels for user preferences by exploiting the self-training principle. With these pseudo labels, the downstream module can better capture the user preference and thus improve the ranking performance.
- The empirical experiments on three real-world datasets illustrate the effectiveness of our proposed framework, and validate the necessity to mitigate the noise in observed interactions.

2 RELATED WORK

2.1 One-Class Collaborative Filtering

According to the way to model interactions, existing OCCF approaches generally fall into the following camps: (1) Matrix Factorization (MF) is one of the most representative CF frameworks. Generally, MF methods calculate interaction scores via the inner product of latent vectors of users and items [15]. As a variant for OCCF, Weighted Regularized MF (WRMF) methods correct the bias of treating unobserved interactions as negative examples based on weighted low-rank approximation [11, 25]. Bayesian Personalized Ranking (BPR) [30] is another variant that converts OCCF into a ranking problem by positing that users prefer observed items to unobserved ones. A large amount of research focuses on improving BRP [24, 26, 27]. (2) Neighbourhood methods estimate interaction scores by computing neighborhood averages iteratively [3, 32]. Lee et al. [17] further adopt belief propagation (BP) and random walk with restart (RWR) to handle the noise in unobserved interactions. (3) Sparse Linear Lodel (SLIM) directly learns an item-similarity matrix Ning and Karypis [22]. Further, Sedhain et al. [34] develop a variant of SILM to produce user-specific predictions. Sedhain et al. [33] address the computational bottleneck of such linear models by reducing dimensionality with a randomized SVD. Wu et al. [41] then replace SVD with Noise Contrastive Estimation (NCE) to correct the popularity bias exhibited by naive SVD embedding. (4) Recently, a growing body of work suggests applying neural networks to model non-linear user preferences to boost the recommendation performance [9, 18, 19, 35, 42, 45].

Previous arts generally take into account the noise in unobserved interactions. Although Wang et al. [39] consider the noise in observed interactions, the proposed adaptive loss functions do not apply to ranking scenarios, which might limit their recommendation performance.

2.2 Self-Supervised Learning

Self-Supervised Learning (SSL) is a popular form of unsupervised learning where data itself provides supervision signals [12]. The key idea behind SSL is to design a pretext task that guidances the model to learn the prior knowledge of interest.

In Computer Vision, a general task is to recover the reorganized data. For example, Zhang et al. [43] train ConvNets to produce plausible colorization for the input grayscale images, which could model the statistical dependencies between the semantics, textures, and color versions. Noroozi and Favaro [23] develop a jigsaw puzzle as the pretext task to learn visual features and spatial information. Pathak et al. [29] inpaint the missing image region to capture the semantics of visual structures. Gidaris et al. [6] propose to recognize the geometric transformation applied to the input image, which requires the pretext model to understand the concept of the objects depicted in the image, such as location, type, and pose. Agrawal et al. [1] predict the camera transformation between two images to learn features that are adept at establishing visual correspondence. Korbar et al. [14] leverage the natural correlation between the visual channels and the audio of a video by the pretext task that entails deciding whether given audio and an image sequence are in-sync. To discover more pretext tasks, please refer to the survey [12].

Taking inspiration from SSL, we design a pretext task to dig out reliable user preference hidden behind the noisy implicit data. The learned information could help boost the downstream learning process and achieve better ranking performance for user preference.

3 PRELIMINARY

We first introduce the notations and further briefly illustrate how the noise in implicit interactions induces performance degradation.

3.1 Learn from implicit feedback

Let n_{user} and n_{item} denote the number of users and items, respectively. In a general OCCF scenario, the historical user-item interactions are stored in a matrix $\mathbf{Y} \in \mathbb{R}^{n_{\text{user}} \times n_{\text{item}}}$, where

$$y_{u,i} = \begin{cases} 1, & \text{if interaction (user } u\text{, item } i\text{) is observed;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

To make the recommendation, it becomes crucial to estimate the scores of unobserved interactions in $\mathbf{Y}$. The classical OCCF approaches project users and items into the same metric space, where different latent vectors and score functions lead to different recommendation results. Formally, let $\boldsymbol{v}_u, \boldsymbol{v}_i \in \mathbb{R}^d$ be the latent vectors of user $u \in [n_{\text{user}}]$ and item $i \in [n_{\text{item}}]$, where $[n]$ represents the set $\{1, \ldots, n\}$; d denotes the dimension of the latent vectors. Then, the interaction score is defined by the following:

$$s_{u,i} = f(\boldsymbol{v}_u, \boldsymbol{v}_i | \theta_f), \tag{2}$$

where θ_f denotes the parameters of the score function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Naturally, $s_{u,i} > s_{u,j}$ indicates that user u prefers item i to item j.

Estimating the parameters $\theta = \{\boldsymbol{v}_u, \boldsymbol{v}_i, \theta_f\}$ is critical in making reasonable recommendation. But, how should these parameters be estimated or learned? According to the loss function, classical OCCF approaches generally fall in two camps:

Pointwise loss. These methods regard $y_{u,i}$ as the label for user preference. In other words, $s_{u,i}$ should be close to $y_{u,i}$ as much as possible. To achieve such a goal, they commonly perform a regression with a weighted squared loss [10, 11]:

$$\min_{\theta} \sum_{u} \sum_{i \in \mathcal{I}_u} w_{u,i} \left(y_{u,i} - s_{u,i} \right)^2 \tag{3}$$

where $\mathcal{I}_u = \mathcal{I}_{u+} \cup \mathcal{I}_{u-}$ represents the training set of user u, and $\mathcal{I}_{u+} = \left\{ i | y_{u,i} = 1 \right\}, \mathcal{I}_{u-} = \left\{ i | y_{u,i} = 0 \right\}$ denote the set of observed and sampled unobserved items, respectively; $w_{u,i}$ is the hyperparameter that weights each training instance.

In a practical recommendation scenario, the ranking of scores is more essential than the score values themselves. However, pointwise methods fail to model such partial ordering relation explicitly.

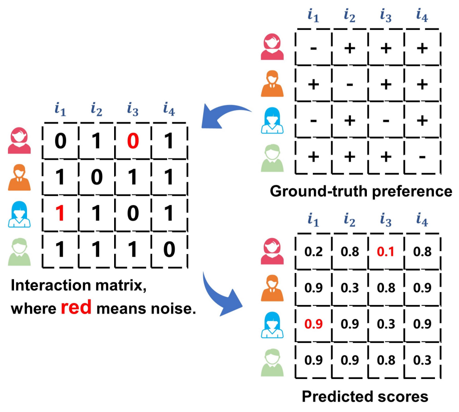
Pairwise loss. Contrary to pointwise methods forcing s_{ui} to 0/1, these methods expand the margin between the observed and unobserved interactions by optimizing the BPR loss function [24, 26, 27, 30, 37]:

$$\min_{\theta} - \sum_{(u,i,j) \in \mathcal{T}} \log \sigma \left(s_{u,i} - s_{u,j} \right), \tag{4}$$

where $\mathcal{T} = \{(u, i, j) | i \in \mathcal{I}_{u+}, j \in \mathcal{I}_{u-}\}$ denotes the set of pairwise comparisons between the observed and unobserved interactions; $\sigma : \mathbb{R} \mapsto (0, 1)$ is the sigmoid function.

3.2 Noise in implicit interactions

The implicit interactions are inherently noisy. More specifically, $y_{u,i} = 1$ only represents the historical user-item interaction (*e.g.*, click, browsing, bookmark history). Such implicit behaviors do not necessarily mean user u actually likes item i. Similarly, unobserved interactions are a mixture of negative feedback and missing values. We cannot directly infer whether user u likes item i or not when only given $y_{u,i} = 0$.



	i_1	i_2	i_3	i_4
	0	1	0	1
	1	0	1	1
	1	1	0	1
	1	1	1	0

	i_1	i_2	i_3	i_4
	-	+	+	+
	+	-	+	+
	-	+	-	+
	+	+	+	-

	i_1	i_2	i_3	i_4
	0.2	0.8	0.1	0.8
	0.9	0.3	0.8	0.9
	0.9	0.9	0.3	0.9
	0.9	0.9	0.8	0.3

Figure 1: An example illustrates the non-negligible noise in observed interactions. From ground-truth preference matrix, u_1 prefers item i_3 to item i_1. However, the noise in observed interactions provides the opposite and erroneous result, which results in a large ranking loss for u_1.

As mentioned above, classical OCCF methods have taken into account the noise in unobserved interactions. However, they generally regard all the observed interactions as positive feedback, which ignores the noise in observed interactions. We argue that such a strategy oversimplifies the complicated problem and might induce performance degradation. As shown in Figure 1, it is easy to find that user u_1 prefers item i_3 to item i_1 from the ground-truth preferences, where + and − denote positive and negative preferences, respectively. However, the noisy observation collected in the dataset provides the opposite and thus erroneous result. More specifically, due to the noisy interactions (*i.e.*, $y_{3,1} = 1$ and $y_{1,3} = 0$), item i_1 is labeled as 1 more times than item i_3. As a result, the model might predict that $s_{1,1} > s_{1,3}$, which is contrary to the fact and results in a large ranking loss for u_1.

4 ITERATIVE RELABELING FRAMEWORK FOR OCCF

So far, we find that the noise in implicit interactions might greatly degrade the ranking performance. In this section, we aim to eliminate the non-negligible noise in both observed and unobserved interactions. As a way to filter out such noise, we could relabel the noisy interactions based on confident positive or negative feedback. Unfortunately, based on the original observation neither the observed nor unobserved interactions could be considered as confident information. To attack this issue, we propose a self-correction framework to dig out the reliable information hidden behind the noisy datasets with an iterative relabeling model. In our framework the prediction model is trained together with the labels. In each iteration of this method, we first train a ranking model based on the current label, then we relabel the data with the output of the finer model trained in the current iteration. In the long run, it leverages a reciprocal interplay of label prediction and label sanitization in a sense that the prediction model benefits from the gradual improvement of the label confidence and the labeling process benefits from

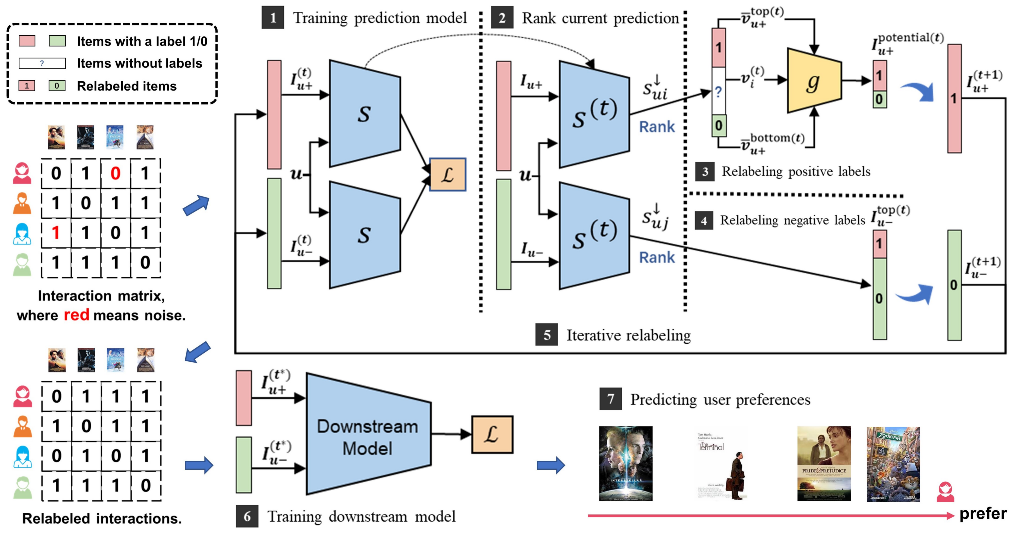

Items with a label 1/0
Items without labels
Relabeled items
Interaction matrix, where red means noise.
Relabeled interactions.
1 Training prediction model
2 Rank current prediction
3 Relabeling positive labels
4 Relabeling negative labels
5 Iterative relabeling
6 Training downstream model
7 Predicting user preferences
Downstream Model
Rank
Rank
prefer

Figure 2: Our framework aims to mitigate the noisy patterns in implicit interactions: In each epoch, we first train a prediction model based on the current label (*i.e.,* step 1). Then, we relabel the observed and unobserved interactions with the ranking list of the current prediction (*i.e.,* step 2, 3, 4). We leverage the relabeling process iteratively to gradually improve the label confidence (*i.e.,* step 5). Finally, the downstream model learns user preferences from the refreshed labels where the noise has been largely alleviated (*i.e.,* step 6, 7).

a gradually finer trained prediction model. We will elaborate on this framework in the remainder of this section.

4.1 Iterative relabeling module

We first present the iterative relabeling module, which aims at mitigating the noisy signals in implicit interactions.

Training the prediction model. We first train a prediction model p_θ based on the current label to evaluate the label confidence. To this end, we employ a popular OCCF method BPR as the prediction model p_θ since this MF-based approach directly models ranking-related preferences:

$$\min_{\theta} - \sum_{(u,i,j)\in\mathcal{T}} \log\sigma\left(\boldsymbol{v}_u^T\boldsymbol{v}_i - \boldsymbol{v}_u^T\boldsymbol{v}_j\right). \tag{5}$$

Let $\theta^* = \{\boldsymbol{v}_u^*, \boldsymbol{v}_i^*\}$ be the parameters when the prediction model p_θ reaches convergence. Then, the learned score can be defined as:

$$s_{u,i} = f(\boldsymbol{v}_u^*, \boldsymbol{v}_i^*) = \boldsymbol{v}_u^{*T}\boldsymbol{v}_i^*. \tag{6}$$

Relabeling the positive labels. In order to obtain reliable positive and negative interactions, we first rank the observed items for each user:

$$s_{u,1}^{\downarrow} > s_{u,2}^{\downarrow} > \cdots > s_{u,|\mathcal{I}_{u+}|}^{\downarrow}, u \in [n_{\text{user}}], \tag{7}$$

where $|\mathcal{I}_{u+}|$ denotes the size of the observed item set $\mathcal{I}_{u+}$. Since the top items have the highest likelihood to be positive, we define the set of confident positive labels as:

$$\mathcal{I}_{u+}^{\text{top}} = \left\{i : s_{u,i} \geq s_{u,n_+}^{\downarrow}\right\}, \tag{8}$$

where n_+ is a hyperparameter representing the number of top positive items we choose to preserve in the list. Similarly,

$$\mathcal{I}_{u+}^{\text{bottom}} = \left\{i : s_{u,i} \leq s_{u,n_-}^{\downarrow}\right\} \tag{9}$$

contains the observed items that are most likely to be negative, where n_- is a hyperparameter that denotes the number of confident negative items we select from the list. Thus, the corresponding pseudo label $\hat{y}_{u,i}$ can be defined as:

$$\hat{y}_{u,i} = \begin{cases} 1, & i \in \mathcal{I}_{u+}^{\text{top}}; \\ 0, & i \in \mathcal{I}_{u+}^{\text{bottom}}. \end{cases} \tag{10}$$

Given confident positive and negative feedback, we could predict the labels of items in $\mathcal{I}_{u+}\backslash(\mathcal{I}_{u+}^{\text{top}} \cup \mathcal{I}_{u+}^{\text{bottom}})$, where the interactions also cover abundant collaborative information. One strategy is to learn a classifier $g : \mathbb{R}^d \mapsto \{0,1\}$ based on $\boldsymbol{v}_i^*, \mathcal{I}_{u+}^{\text{top}}, \mathcal{I}_{u+}^{\text{bottom}}$. Since the size of $\mathcal{I}_{u+}^{\text{top}}$ and $\mathcal{I}_{u+}^{\text{bottom}}$ list should not be too much, we adopt a simple prototype-based method classifier. Specifically, we define the prototype of the positive and negative class as $\overline{\boldsymbol{v}}_{u+}^{\text{top}}$ and $\overline{\boldsymbol{v}}_{u+}^{\text{bottom}}$ the corresponding centroids of the embeddings:

$$\begin{aligned} \overline{\boldsymbol{v}}_{u+}^{\text{top}} &= \frac{1}{n_+}\sum_{i\in\mathcal{I}_{u+}^{\text{top}}} \boldsymbol{v}_i^*, \\ \overline{\boldsymbol{v}}_{u+}^{\text{bottom}} &= \frac{1}{n_-}\sum_{i\in\mathcal{I}_{u+}^{\text{bottom}}} \boldsymbol{v}_i^*. \end{aligned} \tag{11}$$

Then we assign the pseudo labels for the items in $\mathcal{I}_{u+}\backslash(\mathcal{I}_{u+}^{\text{top}} \cup \mathcal{I}_{u+}^{\text{bottom}})$ to their closest prototype:

$$\hat{y}_{u,i} = g(\boldsymbol{v}_i^*) = \begin{cases} 1, & \alpha||\boldsymbol{v}_i^* - \overline{\boldsymbol{v}}_{u+}^{\text{top}}||_2 < ||\boldsymbol{v}_i^* - \overline{\boldsymbol{v}}_{u+}^{\text{bottom}}||_2; \\ 0, & \text{otherwise}, \end{cases} \tag{12}$$

where α is the threshold parameter. Then the potential positive set consists of the items assigned a pseudo label 1 by the classifier g:

$$\mathcal{I}_{u+}^{\text{potential}} = \{i : \hat{y}_{u,i} = 1, i \in \mathcal{I}_{u+}\backslash(\mathcal{I}_{u+}^{\text{top}} \cup \mathcal{I}_{u+}^{\text{bottom}})\}. \tag{13}$$

Relabeling the negative labels. Next we proceed to mitigate the noise among unobserved interactions. Similarly, we rank the unobserved interactions for each user:

$$s^{\downarrow}_{u,1} > s^{\downarrow}_{u,2} > \cdots > s^{\downarrow}_{u,|\mathcal{I}_{u-}|}, u \in [n_{\text{user}}]. \tag{14}$$

Then we define the labels for unobserved interactions $\mathcal{I}_{u-}$ as:

$$\hat{y}_{u,i} = \begin{cases} 1, & s_{u,i} \geq s^{\downarrow}_{u,n_+^{\text{unobserved}}}; \\ 0, & otherwise, \end{cases} \tag{15}$$

where $n_+^{\text{unobserved}}$ is a hyperparameter that denotes the number of positive items we select from unobserved interactions. In other words, we give the top unobserved interactions a label 1. This scheme is inspired by two facts: (1) The unobserved interactions with a high score are also likely to be positive, which helps eliminate the noise in unobserved interactions. (2) It is inevitable to introduce more noise in positive samples when labeling more unobserved interactions as 1. Since the top interactions in $\mathcal{I}_{u-}$ tend to be most confident, we set the maximum number of positive instances to $n_+^{\text{unobserved}}$.

Finally, we get the set of generated labels in the first epoch:

$$\mathcal{T}^{(1)} = \{(u,i,j)\,|\hat{y}_{u,i} = 1, \hat{y}_{u,j} = 0\}. \tag{16}$$

Iterative relabeling. As mentioned above, as the labels become less noisy, the predicted scores would be more consistent with user preferences. Let $\theta^{(t)} = \{\boldsymbol{v}_u^{(t)}, \boldsymbol{v}_i^{(t)}, f_\theta^{(t)}\}$ denote the parameters when the prediction model p_θ reaches convergence in epoch t, and $s_{u,i}^{(t)} = f(\boldsymbol{v}_u^{(t)}, \boldsymbol{v}_i^{(t)} | f_\theta^{(t)})$ represent the corresponding score values. Then, it becomes a natural strategy to repeat the steps in the first epoch to further eliminate the noise in implicit feedback. We first rank the observed and unobserved items:

$$\begin{aligned} &s_{u,1}^{(t)} > \cdots > s_{u,k}^{(t)} > \cdots > s_{u,|\mathcal{I}_{u+}|}^{(t)}, u \in [n_{\text{user}}], k \in \mathcal{I}_{u+} \\ &s_{u,1}^{(t)} > \cdots > s_{u,k}^{(t)} > \cdots > s_{u,|\mathcal{I}_{u-}|}^{(t)}, u \in [n_{\text{user}}], k \in \mathcal{I}_{u-} \end{aligned} \tag{17}$$

With the above ranking lists, the labels generated by the relabeling model g_μ in epoch t can be defined as:

$$y_{u,i}^{(t)} = g_\mu(u,i) = \begin{cases} 1, & i \in \mathcal{I}_{u+}^{\text{top}(t)} \cup \mathcal{I}_{u+}^{\text{potential}(t)} \cup \mathcal{I}_{u-}^{\text{top}(t)}; \\ 0, & otherwise, \end{cases} \tag{18}$$

where:

- $\mathcal{I}_{u+}^{\text{top}(t)} = \{i : s_{u,i} \geq s_{u,n_+}^{(t)}, i \in \mathcal{I}_{u+}\}$ denotes the set of confident positive items for user u in epoch t;
- $\mathcal{I}_{u+}^{\text{potential}(t)} = \{i : g(\boldsymbol{v}_i^{(t)}) = 1, i \in \mathcal{I}_{u+} \backslash (\mathcal{I}_{u+}^{\text{top}(t)} \cup \mathcal{I}_{u+}^{\text{bottom}(t)})\}$ represents the set of predicted positive items in observed interactions for user u in epoch t;
- $\mathcal{I}_{u-}^{\text{top}(t)} = \{i : s_{u,i} \geq s_{u,n_+^{\text{unobserved}}}^{(t)}, i \in \mathcal{I}_{u-}\}$ is the positive item set we select from the unobserved interactions for user u in epoch t;
- $\mu = \{n_+, n_-, n_+^{\text{unobserved}}, \alpha\}$ contains the parameters of the relabeling model g_μ;
- $y_{u,i}^{(1)}$ is the pseudo label $\hat{y}_{ui}$ that is generated in the first epoch.

Finally, the set of pseudo labels generated in epoch t is denoted as:

$$\mathcal{T}^{(t)} = \{(u,i,j)\,|y_{u,i}^{(t)} = 1, y_{u,j}^{(t)} = 0\}. \tag{19}$$

Algorithm 1 Training algorithm for the iterative relabeling framework

```
Input: prediction model $p_\theta$; relabeling model $g_\mu$; downstream
    model $d_\phi$; training implicit feedback set $\mathcal{T}$.
 1: while the iterative relabeling module does not converge do
 2:     Initialize prediction model $p_\theta$ with random parameters $\theta$.
 3:     for $p_\theta$-steps do
 4:         Update the prediction model $p_\theta$ via Eq. (22)
 5:     end for
 6:     Leverage the relabeling model $g_\mu$ to generate label set $\mathcal{T}^{(t)}$
        via Eq. (18)
 7: end while
 8: Initialize the downstream model $d_\phi$ with latent vectors
    $\boldsymbol{v}_u^{(t^*)}, \boldsymbol{v}_i^{(t^*)}$.
 9: for $d_\phi$-steps do
10:     Update the downstream model $d_\phi$ via Eq. (23)
11: end for
```

4.2 Downstream module

The iterative relabeling module so far alleviates the noisy patterns in the original implicit interactions. Based on these refreshed labels, the downstream module aims to make better recommendations. Let $s_{u,i}^d$ denote the prediction score of the downstream model. Then we consider two types of downstream model: (1) Inherit the ranking model obtained in the relabeling module as the predictor with the noisy label corrected:

$$s_{u,i}^d = s_{u,i}^{(t^*)}, \tag{20}$$

where t^* denotes the epoch when the iterative relabeling module reaches convergence. (2) With the noise partially removed from the original implicit interactions, it becomes less likely to overfit. As an alternative solution, we could retrain a finer model d_ϕ to improve the performance of the downstream task:

$$s_{u,i}^d = d_\phi(u,i) = h(h_c(\boldsymbol{v}_u^d, \boldsymbol{v}_i^d)), \tag{21}$$

where $h_c(\cdot)$ is the concatenation operation; the hidden layers $h(\cdot)$ adopt a multi-layer structure as $h(\cdot) = h_{n_h}(\cdots h_1(\cdot)\cdots)$; n_h is the number of hidden layers; $\phi = \{\theta_h, \boldsymbol{v}_u^d, \boldsymbol{v}_i^d\}$ denotes the parameter set.

4.3 Optimization

In this subsection, we introduce the optimization details for the iterative relabeling module and the downstream module.

Iterative relabeling module. In each epoch, we first train a prediction model p_θ based on the current label $\mathcal{T}^{(t-1)}$. In other words, the parameters of the prediction model can be updated as follows:

$$\theta^{(t)} = \arg\min_\theta - \sum_{(u,i,j)\in\mathcal{T}^{(t-1)}} \log\sigma\left(s_{u,i} - s_{u,j}\right), \tag{22}$$

where $\mathcal{T}^{(t-1)}$ is initialized via the original label set $\mathcal{T}$; the prediction model p_θ is initialized with random parameters θ in each epoch t.

With the current prediction $s_{u,i}^{(t)}$ and the learned latent vectors of items $\boldsymbol{v}_i^{(t)}$, the relabeling model g_μ could generate new labels of the implicit interactions via Eq. (18).

Table 1: Statistics of the datasets

	MoiveLens100K	MoiveLens1M	Netflix
#User	943	6,040	8,143
#Item	1,683	3,706	17,770
#Rating	100,000	1,000,209	5,394,409
Sparsity	93.70%	95.53%	96.27%
Positive rate	56.43%	59.19%	54.79%

Downstream module. We train the downstream model d_ϕ based on the refreshed labels and learned latent vectors. Specifically, the downstream model can be updated as follows:

$$\min_{\phi} - \sum_{(u,i,j)\in\mathcal{T}^{(t_*-1)}} \mathcal{L}^d(\sigma(s^d_{u,i} - s^d_{u,j})), \tag{23}$$

where $\mathcal{L}^d : \mathbb{R} \mapsto \mathbb{R}^+$ is a monotonically non-decreasing loss function; the latent vectors $\boldsymbol{v}^d_u$ and $\boldsymbol{v}^d_i$ are initialized via $\boldsymbol{v}^{(t^*)}_u$ and $\boldsymbol{v}^{(t^*)}_i$.

In summary, the overall solution of the entire iterative relabeling framework is shown in Algorithm 1.

5 EXPERIMENTS

In this section, we focus on the following research questions:

- **RQ1**: Are the generated labels more likely to reflect true preference than those of the original interactions?
- **RQ2**: Do the generated labels and learned latent vectors help improve the ranking performance of the downstream module?
- **RQ3**: Does our proposed method outperform existing denoising methods in OCCF?

We perform evaluations on three popular benchmark datasets: MoiveLens100K[8], MovieLens1M[8] and Netflix[21]. The detailed statistics of the datasets are summarized in Table 1.

5.1 MovieLens100K

Dataset description. The full dataset is collected from the MovieLens website and contains 26,000,000 integer movie ratings ranging from 1 to 5. We first use its subset MovieLens100K (i.e., including 100,000 ratings) to evaluate our method. To generate implicit training data, we randomly select n_r ratings for each user as observed interactions, no matter whether the user likes the item or not. Meanwhile, we binarize the rest ratings according to a threshold of 4 to evaluate the performance in predicting true user preference, which is unavailable in implicit datasets. Note that users with less than n_r ratings and items not appearing in the training set are filtered out.
Evaluation Metrics. We adopt three standard ranking metrics [4, 20, 40, 44]: Precision (P@K), Recall(R@K), and Normalized Discounted Cumulative Gain (NDCG@K). Precision calculates the average possibility for a top-K predicted interaction to be positive. NDCG assigns a monotonically increasing discount to the predicted ranking list to emphasize the importance of the top hits. Recall estimates the average likelihood that a positive interaction appears in the top-K ranking list.
Competitors. To answer **RQ3**, we implement the following competitors that develop various strategies to handle the noise in implicit interactions:

- DNS [44]: This method designs a reject sampling strategy on unobserved interactions based on the current prediction.
- NA [20]: This model explicitly quantifies the uncertainty of unobserved interactions by weighing interactions with a pre-defined negative-aware matrix.
- WRMF [25]: This MF-based method proposes a weighed negative sampling strategy based on the assumption that the more popular an item is, the less likely it is a negative sample.
- iCD [2]: This method proposes an implicit regularizer to set a prior estimate on the unobserved interactions.
- SRRMF [4]: This method develops a ranking-based regularizer making the scores of unobserved interactions close to each other.
- T_CE [39]: Observed interactions with large loss values are filtered out based on an adaptive threshold.

Implementation details. We implement our framework with the popular Pytorch package [28]. For the relabeling model g_μ, we set $n_+, n_-, n^{\text{unobserved}}_+, \alpha$ as 9, 9, 6, 1, respectively. Meanwhile, we set n_r to 50 and K to 10 for training and test. To answer **RQ1** and **RQ2**, we adopt the following models as the downstream model d_ϕ:

- IR$_{\text{BPR}}$ [30]: BPR is a popular MF-based method that explicitly optimizes a pairwise ranking loss. We set the latent factor d to 16 and the regularization parameter λ to 0.000025. For a faster convergence, we use Adam [13] as the batch gradient descent optimizer, where the learning rate and the size of a batch are set to 0.0001 and 128, respectively.
- IR$_{\text{MLP}}$ [9]: MLP adopts neural networks to learn a non-linear interaction function. We replace the original loss function with BPR loss to enhance its ranking performance. Meanwhile, we set the latent factor d to 16 and the architecture is condensed as $32 \rightarrow 16 \rightarrow 8 \rightarrow 4$. The corresponding learning rate, regularization parameter and batch size are set to 0.01, 0.005 and 256, respectively.

To answer **RQ3**, the competitors share the same public parameters with BPR. We tune the private parameters and report the best results.
Results. Table 2 and Table 3 show the significant improvements of our proposed framework (**RQ2**). IR$_{\text{BPR}}$ outperforms BPR by 4.21%, 3.05% and 5.11% in terms of P@5, R@5 and NDCG@5, respectively; IR$_{\text{MLP}}$ also gets an improvement of 4.83%, 2.93% and 5.17%, respectively. Figure 3 and 4 illustrate the reason behind (**RQ1**). As the iterative relabeling process goes on, we observe that: (1) The rate of positive samples in $\mathcal{I}^{\text{top}(t)}_{u+}$ and $\mathcal{I}^{\text{potential}(t)}_{u+}$ increases gradually from the original 54% to 64%; (2) the preference improvement is consistent with the number of relabeled samples. In other words, the relabeling process eliminates the noise in implicit interactions and further brings significant performance improvement.

We further compare our proposed method with the competitors (**RQ3**). As shown in Table 4, our method significantly outperforms all the competitors on all the metrics. After the quantitative comparison, we visualize the predicted results of IR$_{\text{BPR}}$, SRRMF, iCD and WRMF in Figure 5 to further illustrate the performance. For each heatmap, the horizontal axis represents 50 randomly selected users, and the vertical axis denotes the top-10 test items of each user from top to bottom. The grid with a deeper color represents a higher ranking position. In these heatmaps, the grids of IR$_{\text{BPR}}$

Table 2: Experimental results of our proposed method on three datasets. The downstream model is BPR.

Dataset	Relabeled	P@5	R@5	NDCG@5
ML100K	✓	**0.7481**	**0.1214**	**0.7635**
	×	0.7179	0.1178	0.7264
ML1M	✓	**0.7861**	**0.0979**	**0.7969**
	×	0.7762	0.0966	0.7791
Netflix	✓	**0.7607**	**0.1368**	**0.7754**
	×	0.7343	0.1315	0.7425

Table 3: Experimental results of our proposed method on three datasets. The downstream model is MLP.

Dataset	Relabeled	P@5	R@5	NDCG@5
ML100K	✓	**0.7513**	**0.1230**	**0.7727**
	×	0.7167	0.1195	0.7347
ML1M	✓	**0.7885**	**0.0977**	**0.7974**
	×	0.7782	0.0970	0.7870
Netflix	✓	**0.7301**	**0.1315**	**0.7448**
	×	0.7180	0.1295	0.7306

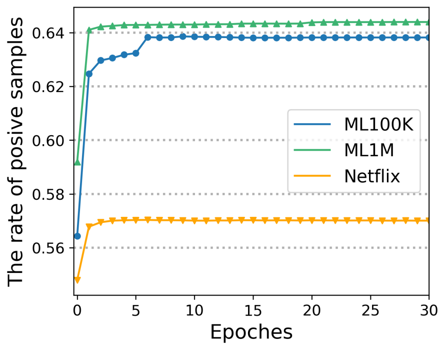


Figure 3: The rate of positive samples increases gradually as the iterative relabeling process goes on. In other words, the pseudo labels are more likely to reflect ture preference than the original ones.

are deeper and more likely to hit the top half of the figure, which represents a better ranking performance.

Finally, we record the results as the number of positive and negative samples change. As shown in Figure 6, our method obtains the best performance when the rate of negative samples ranges from 0.15 to 0.20. Meanwhile, the performance is not sensitive to the rate of positive samples.

5.2 MovieLens1M

Dataset description. This dataset contains 1,000,209 ratings sampled from the full MovieLens dataset. It is larger and also more

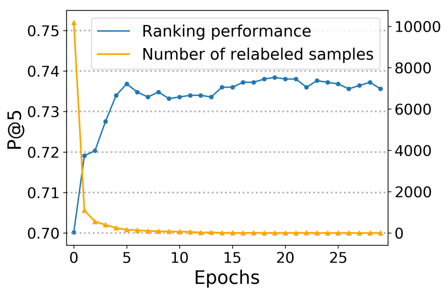


Figure 4: The number of relabeled samples and recommendation performance of our proposed method *w.r.t.* the number of epochs on ML100K. The preference improvement is consistent with the number of relabeled samples.

Table 4: Experimental results on MovieLens100K.

	P@1↑	P@5↑	P@10↑	R@5↑	R@10↑	NDCG@5↑	NDCG@10↑
DNS [44]	0.7606	0.7171	0.6825	0.1175	0.2201	0.7247	0.6980
NA [20]	0.7686	0.7111	0.6870	0.1190	0.2244	0.7225	0.7088
WRMF [25]	0.8048	**0.7284**	0.6855	0.1191	0.2187	**0.7455**	0.7099
iCD [2]	0.7706	0.7171	0.6982	0.1196	0.2245	0.7274	0.7104
SRRMF [4]	**0.8129**	0.7066	0.6730	0.1171	0.2195	0.7281	0.6967
T_CE [39]	0.7968	0.7272	**0.6996**	**0.1212**	**0.2271**	0.7413	**0.7169**
IR_{BPR}(Ours)	**0.8229**	**0.7481**	**0.7040**	**0.1214**	**0.2272**	**0.7635**	**0.7269**

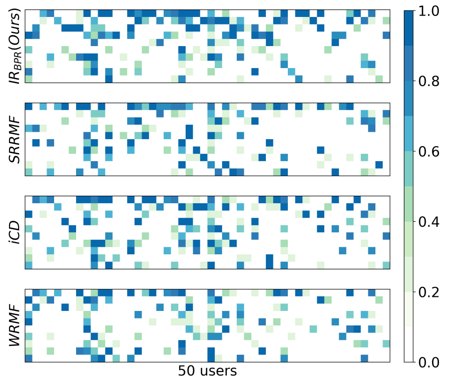
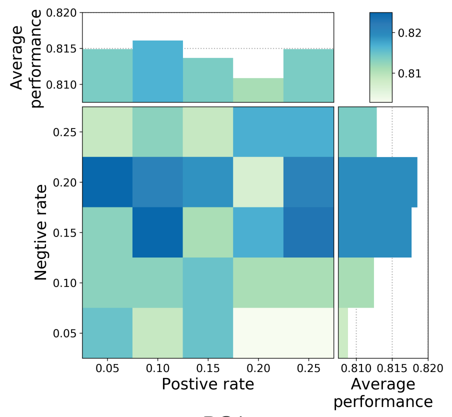


Figure 5: Visualization of ranking results for the top-10 test items on ML100K. The grid with a deeper color represents a higher prediction score. The grids of IR_{BPR} are deeper and more likely to hit the top half of the figure.

sparse than MovieLens100K. In MoiveLens1M, only 3839 of the 6040 users have rated more than $n_r + K = 110$ items.

Implementation details. For the relabeling model g_μ, we set $n_+, n_-, n_+^{\text{unobserved}}, \alpha$ as 18, 18, 8, 0.95, respectively. Due to data sparsity, we set n_r to 100. For BPR and the competitors, we set the regularization parameter λ, batch size and learning rate to 0.000001, 256 and 0.0001, respectively. For MLP, the architecture is condensed as 32 $\longrightarrow$ 16, and the learning rate, regularization parameter and



P@1

Figure 6: Experimental results on ML100K as the numbers of positive and negative samples change.

Table 5: Experimental results one MovieLens1M.

	P@1↑	P@5↑	P@10↑	R@5↑	R@10↑	NDCG@5↑	NDCG@10↑
DNS [44]	0.7901	0.7773	0.7565	0.0960	0.1812	0.7805	0.7651
NA [20]	0.7846	0.7703	0.7548	0.0960	0.1804	0.7736	0.7619
WRMF [25]	0.8019	**0.7843**	**0.7665**	**0.0971**	**0.1828**	0.7893	0.7753
iCD [2]	0.8089	0.7838	0.7603	0.0953	0.1801	**0.7897**	0.7714
SRRMF [4]	0.8115	0.7809	0.7639	0.0969	0.1822	0.7933	**0.7780**
T_CE [39]	**0.8167**	0.7823	0.7612	0.0968	0.1820	0.7892	0.7723
$\mathrm{IR}_{\mathrm{BPR}}$(Ours)	**0.8366**	**0.7861**	**0.7686**	**0.0979**	**0.1833**	**0.7969**	**0.7810**

batch size are set to 0.00001, 0.001 and 256, respectively. The other hyperparameters are the same as MoiveLens100K.

Results. As shown in Table 2 and 3, our model obtains a consistent improvement over BPR and MLP in terms of P@5, R@5 and NDCG@5 (**RQ2**). Meanwhile, the rate of positive feedback in $\mathcal{I}_{u+}^{\mathrm{top}(t)}$ and $\mathcal{I}_{u+}^{\mathrm{potential}(t)}$ increases gradually from the original 56% to 64% (**RQ1**). The performance of all the competitors is recorded in Table 2 (**RQ3**). Similar to the results on MovieLens100K, our method consistently outperforms on all the metrics.

5.3 Netflix

Dataset description. This dataset comes from the famous competition Netflix Prize. It includes 100,480,507 ratings from 480,189 users for 17,770 movies without any additional information like actors or summary provided. We randomly sample 10,000 users with their rated movies for the evaluation. After filtering out users with less than $n_r + K = 110$ ratings, there remain 8,143 users.

Implementation details. For the relabeling model g_μ, we set $n_+, n_-, n_+^{\mathrm{unobserved}}, \alpha$ as 55, 10, 10, 0.95, respectively. For BPR and the competitors, we set the regularization parameter λ, batch size and learning rate to 0.01, 4096 and 0.000001, respectively. For MLP, the architecture is condensed as $32 \longrightarrow 16 \longrightarrow 8 \longrightarrow 4$, and the learning rate, regularization parameter and batch size are set to 0.001, 0.001 and 4096, respectively. Due to data sparsity, we set n_r to 100. The other hyperparameters are the same as MoiveLens100K.

Table 6: Experimental results on Netflix.

	P@1↑	P@5↑	R@1↑	R@5↑	NDCG@5↑	NDCG@10↑
DNS [44]	0.7336	0.6935	0.0266	0.1246	0.7019	0.16874
NA [20]	0.7689	0.7187	0.0278	0.1288	0.7300	0.7086
WRMF [25]	0.7812	0.7185	0.0283	0.1295	0.7320	0.7057
iCD [2]	**0.7980**	**0.7580**	0.0290	**0.1362**	**0.7672**	**0.7427**
SRRMF [4]	0.7944	0.7549	**0.0290**	0.1358	0.7633	0.7399
T_CE [39]	0.7917	0.7501	0.0289	0.1349	0.7602	0.7310
$\mathrm{IR}_{\mathrm{BPR}}$(Ours)	**0.8292**	**0.7607**	**0.0301**	**0.1368**	**0.7754**	**0.7433**

Results. As shown in Table 2 and 3, our method obtains an improvement over BPR by 3.60%, 4.03% and 4.43% and MLP by 1.69%, 1.54% and 1.94% in terms of P@5, R@5 and NDCG@5, respectively (**RQ2**). It should be mentioned that MLP performs worse than BPR, which seems counterintuitive. Recall that our test set composes of true user preference, rather than noisy interactions in training set. Since MLP is more expressive, it becomes more likely to overfit the noise during training. Therefore, the inferior performance of MLP is reasonable and validates the necessity of denoising. Meanwhile, the rate of positive samples in $\mathcal{I}_{u+}^{\mathrm{top}(t)}$ and $\mathcal{I}_{u+}^{\mathrm{potential}(t)}$ increases gradually from the original 54% to 57% (**RQ1**). Compared with the competitors, we can observe that our model consistently achieves the highest performance in Table 6 (**RQ3**).

6 CONCLUSION

In this paper, we propose a novel iterative relabeling framework to eliminate the noise in implicit interactions. Taking inspiration from self-supervised learning, the framework consists of two modules: the iterative relabeling module and the downstream module. As the core of the framework, the iterative relabeling module exploits the self-training principle to dig out reliable information hidden behind the implicit interactions. In each epoch, we first train a prediction model based on the current label and then relabel the implicit data with current predictions. As the iterative relabeling process goes on, the generated labels become less noisy. Based on these labels, the downstream module could make more accurate recommendations. Finally, comprehensive experiments are conducted on three datasets. The corresponding results illustrate a significant improvement of our framework and justify the importance of mitigating the noise in both observed and unobserved interactions.

ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0102003, in part by National Natural Science Foundation of China: 61620106009, U1936208, 61971016, 61931008, 61836002 and 61976202, in part by the Fundamental Research Funds for the Central Universities, in part by Youth Innovation Promotion Association CAS, in part by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDB28000000, and in part by MindSpore, which is a new deep learning computing framework[*].

[*]https://www.mindspore.cn/

REFERENCES

[1] Pulkit Agrawal, João Carreira, and Jitendra Malik. 2015. Learning to See by Moving. In *IEEE International Conference on Computer Vision*. 37–45.

[2] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A Generic Coordinate Descent Framework for Learning from Implicit Feedback. In *International Conference on World Wide Web*. 1341–1350.

[3] Christian Borgs, Jennifer T. Chayes, Christina E. Lee, and Devavrat Shah. 2017. Thy Friend is My Friend: Iterative Collaborative Filtering for Sparse Matrix Estimation. In *Advances in Neural Information Processing Systems*. 4715–4726.

[4] Jin Chen, Defu Lian, and Kai Zheng. 2019. Improving One-Class Collaborative Filtering via Ranking-Based Implicit Regularizer. In *AAAI Conference on Artificial Intelligence*. 37–44.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *ACM Conference on Recommender Systems*. 191–198.

[6] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. 2018. Unsupervised Representation Learning by Predicting Image Rotations. In *International Conference on Learning Representations*.

[7] Udit Gupta, Carole-Jean Wu, Xiaodong Wang, Maxim Naumov, Brandon Reagen, David Brooks, Bradford Cottel, Kim M. Hazelwood, Mark Hempstead, Bill Jia, Hsien-Hsin S. Lee, Andrey Malevich, Dheevatsa Mudigere, Mikhail Smelyanskiy, Liang Xiong, and Xuan Zhang. 2020. The Architectural Implications of Facebook's DNN-Based Personalized Recommendation. In *IEEE International Symposium on High Performance Computer Architecture*. 488–501.

[8] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Transactions on Internet and Information Systems* 5, 4 (2016), 19:1–19:19.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *International Conference on World Wide Web*. 173–182.

[10] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *International ACM conference on Research and Development in Information Retrieval*. 549–558.

[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *IEEE International Conference on Data Mining*. 263–272.

[12] Longlong Jing and Yingli Tian. 2019. Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey. *CoRR* abs/1902.06162 (2019). arXiv:1902.06162 http://arxiv.org/abs/1902.06162

[13] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[14] Bruno Korbar, Du Tran, and Lorenzo Torresani. 2018. Cooperative Learning of Audio and Video Models from Self-Supervised Synchronization. In *Advances in Neural Information Processing Systems*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 7774–7785.

[15] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42 (2009), 30–37.

[16] Ksenia Koroleva, Hanna Krasnova, and Oliver Günther. 2010. 'STOP SPAMMING ME!' - Exploring Information Overload on Facebook. In *Americas Conference on Information Systems*, Martin Santana, Jerry N. Luftman, and Ajay S. Vinze (Eds.). Association for Information Systems, 447.

[17] Yeon-Chang Lee, Sang-Wook Kim, and Dongwon Lee. 2018. gOCCF: Graph-Theoretic One-Class Collaborative Filtering Based on Uninteresting Items. In *AAAI Conference on Artificial Intelligence*. 3448–3456.

[18] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M. Blei. 2016. Factorization Meets the Item Embedding: Regularizing Matrix Factorization with Item Co-occurrence. In *ACM Conference on Recommender Systems*. 59–66.

[19] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *International World Wide Web Conference*. 689–698.

[20] Sheng-Chieh Lin, Yu-Neng Chuang, Sheng-Fang Yang, Ming-Feng Tsai, and Chuan-Ju Wang. 2019. Negative-Aware Collaborative Filtering. In *ACM Conference on Recommender Systems*. 41–45.

[21] Netflix. 2009. Netflix Prize Data Set. (2009).

[22] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *IEEE International Conference on Data Mining*. 497–506.

[23] Mehdi Noroozi and Paolo Favaro. 2016. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In *European conference on computer vision*. 69–84.

[24] Shan Ouyang, Lin Li, Weike Pan, and Zhong Ming. 2019. Asymmetric Bayesian personalized ranking for one-class collaborative filtering. In *ACM Conference on Recommender Systems*. 373–377.

[25] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. 2008. One-Class Collaborative Filtering. In *IEEE International Conference on Data Mining*. 502–511.

[26] Weike Pan and Li Chen. 2013. GBPR: Group Preference Based Bayesian Personalized Ranking for One-Class Collaborative Filtering. In *International Joint Conference on Artificial Intelligence*. 2691–2697.

[27] Ulrich Paquet and Noam Koenigstein. 2013. One-class collaborative filtering with random graphs. In *International World Wide Web Conference*. 999–1008.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Conference on Neural Information Processing Systems*. 8024–8035.

[29] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. 2016. Context Encoders: Feature Learning by Inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2536–2544.

[30] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Conference on Uncertainty in Artificial Intelligence*. 452–461.

[31] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*. 1–35.

[32] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. [n.d.]. Item-based collaborative filtering recommendation algorithms. In *International World Wide Web Conference*, Vincent Y. Shen, Nobuo Saito, Michael R. Lyu, and Mary Ellen Zurko (Eds.). 285–295.

[33] Suvash Sedhain, Hung Hai Bui, Jaya Kawale, Nikos Vlassis, Branislav Kveton, Aditya Krishna Menon, Trung Bui, and Scott Sanner. 2016. Practical Linear Models for Large-Scale One-Class Collaborative Filtering. In *International Joint Conference on Artificial Intelligence*. 3854–3860.

[34] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Darius Braziunas. 2016. On the Effectiveness of Linear Models for One-Class Collaborative Filtering. In *AAAI Conference on Artificial Intelligence*. 229–235.

[35] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *International Conference on World Wide Web*. 111–112.

[36] Brent Smith and Greg Linden. 2017. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing* 21, 3 (2017), 12–18.

[37] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Advances in Neural Information Processing Systems*. 926–934.

[38] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Adv. Artificial Intellegence* 2009 (2009), 421425:1–421425:19.

[39] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising Implicit Feedback for Recommendation. In *ACM International Conference on Web Search and Data Mining*. 373–381.

[40] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alexander J. Smola. 2007. COFI RANK - Maximum Margin Matrix Factorization for Collaborative Ranking. In *Conference on Neural Information Processing Systems*. 1593–1600.

[41] Ga Wu, Maksims Volkovs, Chee Loong Soon, Scott Sanner, and Himanshu Rai. 2019. Noise Contrastive Estimation for One-Class Collaborative Filtering. In *ACM Conference on Research and Development in Information Retrieval*. 135–144.

[42] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *ACM International Conference on Web Search and Data Mining*. 153–162.

[43] Richard Zhang, Phillip Isola, and Alexei A. Efros. 2016. Colorful Image Colorization. In *European Conference on Computer Vision*. 649–666.

[44] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *International ACM Conference on Research and Development in Information Retrieval*. 785–788.

[45] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A Neural Autoregressive Approach to Collaborative Filtering. In *International Conference on Machine Learning*. 764–773.

[46] Yunhong Zhou, Dennis M. Wilkinson, Robert Schreiber, and Rong Pan. 2008. Large-Scale Parallel Collaborative Filtering for the Netflix Prize. In *Algorithmic Aspects in Information and Management*. 337–348.