PopDCL: Popularity-aware Debiased Contrastive Loss for Collaborative Filtering

Zhuang Liu
Engineering Research Center of ACAT, Ministry of Education, Beihang University, Beijing, China
liuzhuang@buaa.edu.cn

Haoxuan Li
Engineering Research Center of ACAT, Ministry of Education, Beihang University, Beijing, China
LucasLi@buaa.edu.cn

Guanming Chen
Sino-French Engineer School, Beihang University, Beijing, China
emilien_chen@buaa.edu.cn

Yuanxin Ouyang*
State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
oyyx@buaa.edu.cn

Wenge Rong
State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
w.rong@buaa.edu.cn

Zhang Xiong
Engineering Research Center of ACAT, Ministry of Education, Beihang University, Beijing, China
xiongz@buaa.edu.cn

ABSTRACT

Collaborative filtering (CF) is the basic method for recommendation with implicit feedback. Recently, various state-of-the-art CF integrates graph neural networks. However, they often suffer from popularity bias, causing recommendations to deviate from users' genuine preferences. Additionally, several contrastive learning methods based on the in-batch sample strategy have been proposed to train the CF model effectively, but they are prone to suffering from sample bias. To address this problem, debiased contrastive loss has been employed in the recommendation, but instead of personalized debiasing, it treats each user equally. In this paper, we propose a popularity-aware debiased contrastive loss for CF, which can adaptively correct the positive and negative scores based on the popularity of users and items. Our approach aims to reduce the negative impact of popularity and sample bias simultaneously. We theoretically analyze the effectiveness of the proposed method and reveal the relationship between popularity and gradient, which justifies the correction strategy. We extensively evaluate our method on three public benchmarks over balanced and imbalanced settings. The results demonstrate its superiority over the existing debiased strategies, not only on the entire datasets but also when segmenting the datasets based on item popularity.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

collaborative filtering, debiased contrastive learning, popularity bias, sample bias

*corresponding author

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom.

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3615009

ACM Reference Format:
Zhuang Liu, Haoxuan Li, Guanming Chen, Yuanxin Ouyang, Wenge Rong, and Zhang Xiong. 2023. PopDCL: Popularity-aware Debiased Contrastive Loss for Collaborative Filtering. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583780.3615009

1 INTRODUCTION

To alleviate information overload on the web, recommender systems are widely deployed to perform personalized information filtering for e-commerce platforms (Alibaba, Amazon), social networks (Facebook, Weibo), or lifestyle apps (Yelp, Meituan) [27].

Collaborative filtering (CF) is the most popular recommendation technique, which estimates the likelihood of user-item interactions based on historical interactions like purchases and clicks. To implement the principle, early CF algorithms employ matrix factorizations (MFs) to learn the matching score of each user-item pair via inner products. However, MFs have limitations in capturing the intricate patterns between users and items. To make up for this deficiency, Graph neural networks (GNNs) nowadays are widely employed for recommendation. Thereby, user-item interactions are typically represented via edges in a bipartite graph where vertices correspond to either users or items. Subsequently, GNNs can form expressive representations of users and items, accurately predicting novel interactions and setting new standards on various benchmarks and recommendation tasks [5, 13, 16, 27, 38].

In these graph-based recommendation algorithms, GNNs are employed as an encoder to learn the representation of users and items. However, noisy edges are often unavoidable in this setting. For example, the so-called *popularity bias* describes the phenomenon that users tend to interact with items that are highly popular even though he/she dislikes them. These interactions produce an uninformative signal that may wash out the users' genuine interest. As it is shown in Figure 1(a), the (yellow) user may have different interests other than the computer-related book. However, this book is prevalent, somehow making the user interact with it (e.g., by clicking on the product page or losing interest at the moment the user reads it). Thus, the book is a false positive instance for the

(yellow) user. Effectively mitigating the impact of popularity bias poses the first challenge in this context.

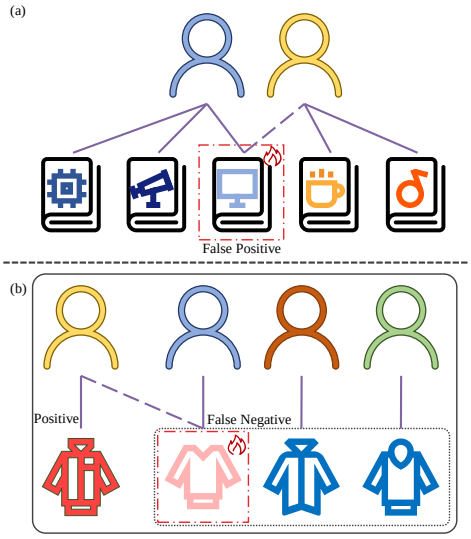

Figure 1: An example of popularity bias and sample bias.

Several contrastive learning-based training schemes [33, 43] have been proposed to effectively train the recommendation model in recent years. They employ contrastive loss with in-batch sample strategies to optimize the recommendation model. However, one drawback of these methods is the potential occurrence of false negatives, which can be attributed to *sample bias*. Empirical evidence suggests that sample bias can lead to a significant performance drop [8]. As shown in Figure 1(b), the other clothes in the same mini-batch are regarded as negatives for the (yellow) user. However, among these negatives, the pink clothes are erroneously classified as negative samples solely due to popularity bias, as most users have interacted with it. Factually, it is consistent with the user's interest. Consequently, the pink clothe will be a false negative instance. To address this problem, [36] proposes a debiased contrastive loss for recommendation. They design a bias correction probability ω^+ to alleviate the sample bias. However, ω^+ is a hyperparameter, which is the same for all users. How to develop a personalized bias correction probability that can adapt to individual users' characteristics is the second challenge.

To tackle the abovementioned challenges, we propose a Popularity-aware Debiased Contrastive Loss with an in-batch sample strategy for collaborative filtering (PopDCL). In particular, we adaptively correct the positive scores and negative scores based on the popularity of users and items. For the positive user-item pair (u, i), we correct its predicted score by cutting $M^+(u, i)$, where $M^+(u, i)$ is the expected score that item i is actually a negative instance for user u. As for the negative user-item pair (u, j), we design a personalized bias correction probability $\omega^+(u)$, and further propose $M^-(u, j)$ to correct negative scores.

To summarize, our main contributions are as follows:

- We propose a novel personalized popularity-aware debiased contrastive loss for collaborative filtering, which can adaptively correct the positive scores and negative scores based on the popularity of users and items.
- We theoretically analyze the effectiveness of PopDCL towards addressing the popularity bias and sample bias and reveal the connection between popularity and gradient.
- We conduct an extensive empirical study on three public datasets. The results demonstrate that our method significantly outperforms several existing state-of-the-art methods while being effective in different evaluations.

2 RELATED WORK

In this section, we briefly review several lines of works closely related to ours, including collaborative filtering, contrastive learning, popularity bias, and sample bias.

Collaborative Filtering (CF) is one of the most fundamental recommendation technique. Early CF models employ MFs to learn the representations of each user and item, and then calculating the predicted score of each user-item pair via inner-product [20, 22]. Recent advancements in neural recommender models, such as DMF[35] and NCF[17] improve the interaction modeling using neural networks. With the success of graph neural networks (GNNs), more and more recommender models are based on GNNs to extract interaction signals on user-item graph. NGCF [27] utilizes the structure of the user-item graph by employing multiple graph convolution layers to propagate embeddings, enabling the capture of high-order connectivity within the graph. LightGCN [16], one of the most popular GNN-based recommenders, is a streamlined version of NGCF. The authors propose an architecture that includes only the most essential component of a GNN - neighborhood aggregation - but still achieves state-of-the-art performance. With the successful application of **Contrastive Learning (CL)** techniques in various fields like computer vision [6, 21], natural language processing [29, 34] and graph learning [14, 26], it has gradually become an important tool for CF [32, 39]. InfoNCE loss [21] is the most fundamental loss function for contrastive learning, which maximizes the similarity between positive samples and minimizes the similarity between negative samples. Designing a pair-wise [23] recommendation task loss based on contrastive loss is also attracting wide attention [36, 41].

Popularity bias [44] is a long-standing challenge in recommender systems: popular items are overly recommended at the expense of less popular items that users may be interested in being under-recommended. To tackle this challenge, several popularity debiased strategies have been proposed. For example, Inverse Propensity Score (IPS) based methods [12, 18, 25] view the item popularity as the propensity score to re-weight loss of each instance. Regularization based frameworks aim to explore the trade-off between recommendation accuracy and coverage by employing regularization. Sam-reg [3] minimizes the biased correction between predicted ratings and item popularity. Reg [45] alleviates the item popularity by the model preference predictions. There are some causal inference methods [2, 30] aim to remove popularity bias, which formulate a causal graph to describe the cause-effect relations in recommendation and mitigate the bias effect on the prediction. BC_loss [41] is the state-of-the art method, which incorporates popularity bias

margins into the contrastive loss for guiding the representation learning.

Sample bias [8] means the common practice of drawing negative samples x_i^- from the data distribution $p(x)$ may result in x_i^- that are actually similar to x or even unlabeled but positive samples to x. One way to correct for sample bias is to devise a novel sampling strategy. RNS [9] optimizes the sampler to select hard and real negative samples based on the Policy-gradient method of reinforcement learning. SRNS [10] proposes a high-variance-based strategy to avoid false-negative instances by preferring high-variance candidates. RHNSR [37] adopts the idea of ensemble learning, combining rule-based, reinforcement learning-based, and KGPolicy-based [28] samplers, and scores the negative samples in different ways to obtain real and hard negative samples respectively. More classic negative sampling strategies in CF can refer to [4]. Another way is to modify the loss function to an unbiased version. DCL [8] and HDCCF [36] develop a debiased contrastive loss that corrects the negative scores by tunning the hyperparameter ω^+, which is denoted as Equation (7). On the basis of DCL, HCL [24] adaptively selects harder negative samples by scoring positive and negative samples.

3 METHODOLOGY

We first formally define the debiased recommendation problem and then give an overview of the proposed method. Subsequently, we delve into the details of each model component and finally discuss the training of model parameters.

3.1 Problem Setup

We denote the user set and item set as $\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$ and $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$, where n and m is the number of users and items, respectively. $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the undirected user-item interaction graph. The vertex set is given by $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$. If user u has interacted with item i, we construct an edge $\{u, i\} \in \mathcal{E}$, where $\mathcal{E}$ is the edge set in $\mathcal{G}$. An example of a user-item interaction graph is shown in the upper left of Figure 2. Let $\mathbf{e}_u \in \mathbb{R}^d$ denote the embedding of a generic user $u \in \mathcal{U}$ and $\mathbf{e}_i \in \mathbb{R}^d$ represent the embedding of an item $i \in \mathcal{I}$, where d is the dimension of embedding vector. Moreover, the popularity of a user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ is defined as $pop(u)$ and $pop(i)$, respectively, and the predicted score between (u, i) is $f(u, i)$. Finally, the debiased recommendation task can be described as follows: Given a user-item interaction pair (u, i) in graph $\mathcal{G}$, we first predict the rating score $f(u, i)$ through CF model (*e.g.*, LightGCN), and then correct the score based on $pop(u), pop(i)$ to alleviate the popularity bias and sample bias. At the end, we predict the users' preferences towards new items with the corrected score $f'(u, i)$. The mathematical notations used in this paper are summarized in Table 1.

3.2 An Overview of the Proposed Method

Motivated by the goal of addressing the challenge of popularity bias and hard negative sampling, we propose Popularity-aware Debiased Contrastive Loss for Collaborative Filtering. Our approach leverages the popularity of users and items to adaptively adjust the scores of positive and negative sample pairs.

Table 1: Notations.

Notation	Description
$\mathcal{U}, \mathcal{I}$	user set, item set
n, m	the number of users, items
$f(u, i)$	the rating value of item i by user u
$\mathbf{e}_u, \mathbf{e}_i$	the embedding of user u, item i
$\mathcal{N}_u, \mathcal{N}_i$	the first-hop neighbors of user u, item i
$M^+(u, i)$	Modified score of positive pair
$M^-(u, j)$	Modified score of negative pair
$pop(u), pop(i)$	degree of user u, item i
N	number of interactions
$\mathcal{D}$	the training set

The architecture of the proposed PopDCL is illustrated in Figure 2. First, we construct an undirected bipartite user-item graph based on the historical interactions of the users. Then, we use GNNs (*e.g.*, LightGCN) to encode the users' behaviors and items. Finally, we propose a popularity-aware module based on contrastive learning with the in-batch sample strategy to alleviate the popularity bias and sample bias mentioned above. More concretely, we correct the positive scores and negative scores according to the popularity of users and items. The loss function can be uniformly defined as:

$$\mathcal{L}(u, i) = -\log \frac{e^{\frac{1}{\tau}[f(u,i) - M^+(u,i)]}}{e^{\frac{1}{\tau}[f(u,i) - M^+(u,i)]} + \sum_{j \in \mathcal{B} \setminus \{i\}} e^{\frac{1}{\tau}[f(u,j) - M^-(u,j)]}}, \tag{1}$$

where item i is a positive sample of user u in a minibatch $\mathcal{B}$, while item j is a negative sample of user u. τ is the temperature parameter. The score of a positive sample (u, i) is adjusted by cutting the expected score $M^+(u, i)$ of pair (u, i), where i is in fact a negative item. Conversely, for negative samples, the adjustment is done in the opposite direction. This process effectively removes unreliable scores associated with each interaction pair. Intuitively, unreliable positive samples for a user are the items he does not actually need but he bought because he followed the general trend or was influenced by popularity bias.

Our proposed loss function is structurally consistent with the form of the softmax loss function, which can theoretically attenuate the negative impact of popularity bias on the recommendation performance[33]. Besides, our method explore hard negatives by adaptively sharpening the gradients of harder instances. To the best of our knowledge, the proposed work outperforms the existing state-of-the-art methods.

3.3 Popularity-aware Positive Scores Correction

Contrastive learning aims to learn an embedding space where positive pairs are close to each other while negative ones are far apart. In recommender systems, noise is often unavoidable when sampling positive pairs from user-item interactions. Figure 1(a) illustrates a scenario where a user's interaction with a computer-related book is driven by its popularity rather than the user's genuine interest. These interactions produce an uninformative signal that may wash out the users' genuine interest. To alleviate the popularity bias, it is necessary to reasonably correct the scores of positive pairs.

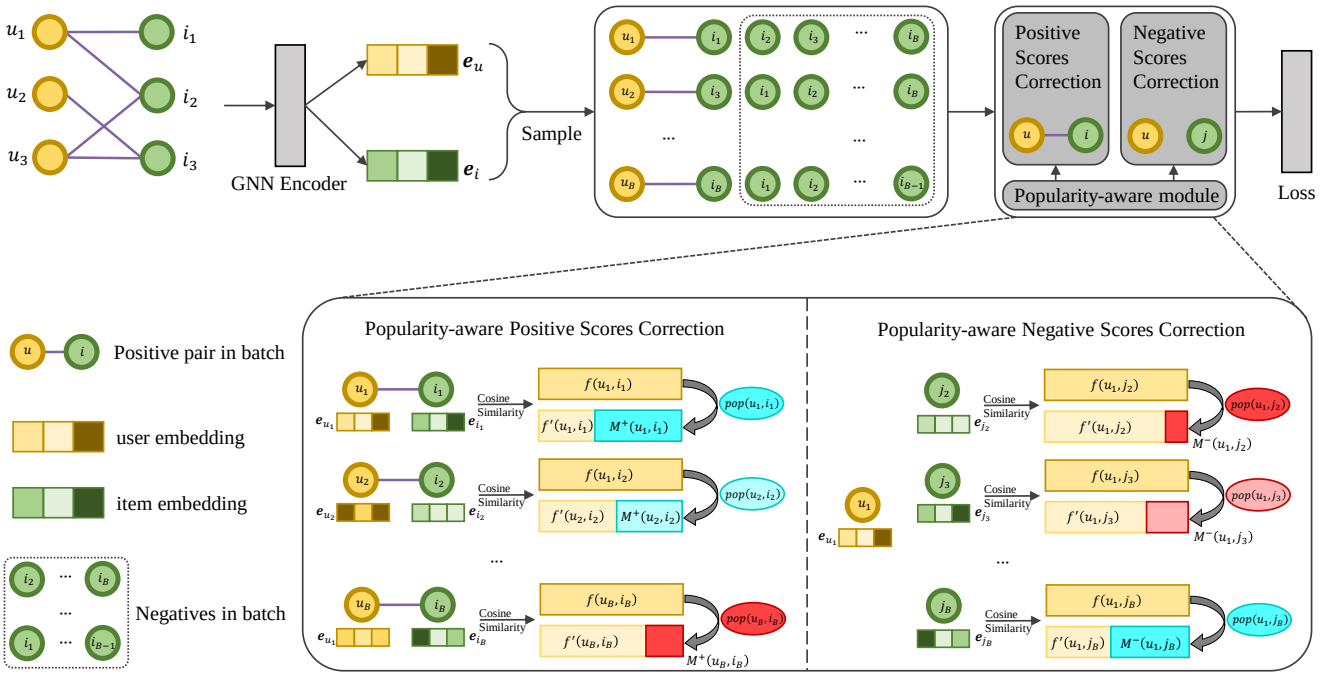

Figure 2: The architecture of the proposed framework.$f^{'}$ denotes the scores after correction. *pop* with colors represents the popularity of u and i, where a deeper red means high popularity and a deeper blue means the opposition.

Given an arbitrary user-item pair (u, i), we can obtain their embedding $\mathbf{e}_u$ and $\mathbf{e}_i$ based on GNN models. The recommender system is supposed to retrieve the top k items relevant to the user u by retrieving the top k candidate $\{\mathbf{e}_i\}_{i \in \mathcal{I}}$ similar to $\mathbf{e}_u$. In this context, we utilize the cosine similarity $f(u, i) =< \mathbf{e}_u, \mathbf{e}_i >$ as the similarity score. However, this score does not reveal the natural preference for user u since the pair is likely be a noise. To this end, we carefully design $M^+(u, i)$ for each (u, i) pair to correct positive scores. In particular, we correct $f(u, i)$ by cutting $M^+(u, i)$, where $M^+(u, i)$ is the expected score that i actually is a negative sample for user u. It can be defined as

$$M^+(u, i) = \mathbb{E}_{i \sim p^-} f(u, i), \tag{2}$$

where p^- is the distribution for negative samples.

Considering the popularity bias, the more popular item i is, the more likely i is to be a false positive sample. Since the precise distribution p^- of the real negative samples is unknown, we fit p^- of each item i using the popularity of u and i. The probability that i becomes a false positive sample for user u can be expressed as

$$P(i \notin \mathcal{N}_u) = \frac{pop(i)}{\sum_{i' \in \mathcal{N}_u} pop(i')}, \tag{3}$$

where $\mathcal{N}_u$ is the item set that user u has interacted, $pop(i)$ is the popularity of item i. Therefore, $M^+(u, i)$ can be defined as

$$M^+(u, i) = P(i \notin \mathcal{N}_u) f^-(u, i) \tag{4}$$

In order to obtain the score of a false positive pair, we resort to the in-batch sampling strategy, which regards the items within the same batch except i as negative samples. We fit the negative score by calculating the average similarity score between user u and each in-batch negative sample j, which leads to

$$f^-(u, i) = \frac{1}{|\mathcal{B}| - 1} \sum_{j \in \mathcal{B} \setminus \{i\}} f(u, j), \tag{5}$$

where $|\mathcal{B}|$ is batch size.

Similar to [43], we use normalization to stabilize contrastive learning, meaning that the user embedding $\mathbf{e}_u$ and item embedding $\mathbf{e}_i$ are both $l2$-normalized. Hence, the negative score $f^-(u, i)$ falls within the range $[-1, 1]$. In addition, we impose the constraint $M^+(u, i) > 0$ to reduce the original positive score $f(u, i)$. That is, to alleviate the impact of the popularity bias. Thus, we use the sigmoid function further to constrain $M^+(u, i)$, which leads to

$$M^+(u, i) = \sigma(P(i \notin \mathcal{N}_u) f^-(u, i)), \tag{6}$$

where $\sigma(x) = 1/(1 + e^{-x})$.

3.4 Popularity-aware Negative Scores Correction

As previously mentioned, the in-batch sample strategy may introduce sample bias, meaning that a negative user-item pair may potentially be a positve one (*e.g.*, false negative) and consequently lead to sub-optimal results. To address this problem, [36] proposed a debiased contrastive loss for CF model:

$$\mathcal{L}_{dcl} = \frac{e^{\frac{1}{\tau} f(u,i)}}{e^{\frac{1}{\tau} f(u,i)} + \frac{1}{\omega^-} \left(\frac{1}{|\mathcal{B}|-1} \sum_{j=1}^{|\mathcal{B}|} \mathbb{I}(j \neq i) e^{\frac{1}{\tau} f(u,j)} - \omega^+ e^{\frac{1}{\tau} f(u,i)} \right)}, \tag{7}$$

where $\mathbb{I}(j \neq i)$ is an indicator function whose value is 1 when $j \neq i$ and 0 otherwise. ω^+ is the probability of false negatives. ω^- is the probability of true negatives and $\omega^- = 1 - \omega^+$. However, ω^+ is regarded as a hyperparameter in implementation, implying that it treats each user equally rather than personalized debiasing for individual users. Generally speaking, in the set of items that user u has interacted with, the likelihood of an item i being a false negative sample increases as its popularity grows.

Based on the above motivation, we design a personalized sample bias correction probability $\omega^+(u)$:

$$\omega^+(u) = \frac{\sum_{i \in \mathcal{N}_u} pop(i)}{N}, \tag{8}$$

where N is the total interactions in the training set, and $\omega^-(u) = 1 - \omega^+(u)$. In this way, we can obtain a personalized sample bias correction probability for each user based on the popularity of items. Moreover, if a user u interacts with an item i that is highly popular, there is a higher chance of item i appearing in the same batch, thereby increasing the possibility of it being wrongly categorized as a false negative for user u. In addition, this observation establishes a connection between sampling bias and popularity bias, enabling adaptive correction of negative scores according to item popularity.

To maintain structural consistency with positive scores correction, we correct the score of each negative instance (u, j) by cutting $M^-(u, j)$. To this end, we set

$$e^{\frac{1}{\tau}[f(u,j) - M^-(u,j)]} = \frac{1}{\omega^-(u)} \left(e^{\frac{1}{\tau} f(u,j)} - \omega^+(u) e^{\frac{1}{\tau} f(u,i)} \right). \tag{9}$$

According to the Maclaurin's expansion $e^x = 1 + x + O(x)$, the expression of $M^-(u, j)$ can be described as

$$M^-(u, j) = \frac{\omega^+(u)}{\omega^-(u)} e^{\frac{1}{\tau}[f(u,i) - f(u,j)]} \tag{10}$$

3.5 Theoretical Analysis

Proposition 1. *The correction strength of positive scores $M^+(u, i)$ is mainly depends on users' popularity. Specifically, $M^+(u, i)$ decreases as the popularity of u increases.*

Given a user-item pair (u, i), $f^-(u, i)$ is fixed, we mainly focus on $P(i \notin \mathcal{N}_u)$ to analyze the change of $M^+(u, i)$. Assuming that the degrees of u and i are increased by k, then $pop'(i) = pop(i) + k$ and $pop'(u) = pop(u) + k$, namely, $\mathcal{N}'_u \geq \mathcal{N}_u + k$, and further leading to:

$$P'(i \notin \mathcal{N}_u) = \frac{pop'(i)}{\sum_{i \in \mathcal{N}'_u} pop(i)}. \tag{11}$$

According to Equation (11), it can be observed that the denominator experiences a higher or significantly higher growth rate compared to the numerator. As a result, the variation in $M^+(u, i)$ is mainly depends on $pop(u)$. Obviously, $M^+(u, i)$ decreases as the popularity of u increases. For more popular users, $M^+(u, i)$ tends to be smaller. In addition, the loss function aims to learn larger scores for positive instances, for popular user, the smaller $M^+(u, i)$ further making the model to learn smaller $f(u, i)$, which alleviating popularity bias to some extent. Moreover, a smaller $M^+(u, i)$ will provide the loss with a less significant gradient, resulting the model to pay less attention to such false positive samples. Concretely, the partial derivative of the loss function L with respect to positive sample i scores with fixed user u and negative j is as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}(u,i)}{\partial f(u,i)} &= -\frac{\hat{\sum}}{1+\hat{\sum}} \frac{1}{\tau} \left(\hat{\sum} + \frac{1}{\tau} \frac{\omega^+(u)}{\omega^-(u)} \sum_{j \in \mathcal{B} \setminus \{i\}} e^{\frac{1}{\tau} - \Delta M} \right) \\ &\overset{r}{=} -\frac{1}{\tau} e^{-\Delta M} = -\frac{1}{\tau} e^{M^+(u,i) - M^-(u,j)}, \end{aligned} \tag{12}$$

where $\Delta M = M^-(u, j) - M^+(u, i)$, $A \overset{r}{=} B$ represents A is positive correlated with B. To make the equation clearer, we abbreviate $\sum_{j \in \mathcal{B} \setminus \{i\}} e^{\frac{1}{\tau}[\Delta f - \Delta M]}$ as $\hat{\sum}$. With the decrease of $M^+(u, i)$, the gradient is less significant, which indicates the model will pay less attention to those samples with high popularity.

Proposition 2. *For any user-item pair (u, i), the more popular item j in the same batch, the smaller $M^-(u, j)$ will be, resulting more significant gradient of $\mathcal{L}$ on $f(u, j)$, for*

$$\frac{\partial \mathcal{L}(u,i)}{\partial f(u,j)} \overset{r}{=} \frac{1}{\tau} e^{M^+(u,i) - M^-(u,j)}. \tag{13}$$

Proof. The loss function can be rewritten as follows:

$$\begin{aligned} \mathcal{L}(u,i) &= \log \left(1 + \sum_{j \in \mathcal{B} \setminus \{i\}} \frac{e^{\frac{1}{\tau}[f(u,j) - M^-(u,j)]}}{e^{\frac{1}{\tau}[f(u,i) - M^+(u,i)]}} \right) \\ &= \log \left(1 + \sum_{j \in \mathcal{B} \setminus \{i\}} e^{\frac{1}{\tau}[\Delta f - \Delta M]} \right) \\ &= \log(1 + \hat{\sum}) \end{aligned} \tag{14}$$

where $\Delta f = f(u, j) - f(u, i)$, $\hat{\sum}$ is positive correlated with $e^{-\Delta M}$

When user u and positive sample i are given, the partial derivative of the loss function L with respect to negative sample j scores is as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}(u,j)}{\partial f(u,j)} &= \frac{\hat{\sum}}{1+\hat{\sum}} \sum_{j' \neq j, j' \in \mathcal{B}} \frac{1}{\tau} Q e^{\frac{1}{\tau}[\Delta f - \Delta M]} \\ &\quad + \frac{\hat{\sum}}{1+\hat{\sum}} \frac{1}{\tau} \left(1 + \frac{\omega^+(u)}{\omega^-(u)} \frac{1}{\tau} e^{\frac{1}{\tau}[-\Delta f]} + Q \right) e^{\frac{1}{\tau}[\Delta f - \Delta M]} \\ &= \frac{\hat{\sum}}{1+\hat{\sum}} \frac{1}{\tau} \left(Q \hat{\sum} + e^{\frac{1}{\tau}[\Delta f - \Delta M]} + \frac{\omega^+(u)}{\omega^-(u)} e^{-\Delta M} \right) \\ &\overset{r}{=} \frac{1}{\tau} e^{-\Delta M} = \frac{1}{\tau} e^{M^+(u,i) - M^-(u,j)}, \end{aligned} \tag{15}$$

where $Q = P(i \notin \mathcal{N}_u) \frac{1}{|\mathcal{B}| - 1}$ is irrelevant with $f(u, j)$.

According to Proposition 2, the loss function can automatically mine hard negatives. Generally speaking, negative item j with high popularity tend to have a larger probability of becoming a false negative instance, indicating the model needs more significant gradient to distinguish it. Specifically, a harder negative instance has a similar predicted score to the positive, *e.g.*, $f(u, j)$ is relatively close to $f(u, i)$, leading to a smaller $M^-(u, j)$, and further brings a larger magnitude of gradients. It indicates that the loss function can automatically focus on optimizing harder negative instances. The hardness level for each negative instance is relevant to its popularity and consequently it can alleviate sample bias to some extent. In addition, we can also control the hard level of negatives

by tuning temperature τ. The greater the τ, the harder the negative instance is, and vice versa, which is consistent with [36].

3.6 Model Training

To effectively learn the parameters of our method, we optimize the following loss function that allows our model to be trained in an end-to-end fashion.

$$\mathcal{L} = \sum_{(u,i)\in\mathcal{D}} \mathcal{L}(u,i) + \lambda \left(||\mathbf{e}_u||_2^2 + ||\mathbf{e}_i||_2^2 \right), \tag{16}$$

where λ controls the L_2 regularization strength. We employ Adam [19] as the optimizer.

4 EXPERIMENTS

In this section, we conduct experiments to evaluate the performance of our proposed method on three public datasets. Specifically, we aim to answer the following four research questions:

- **RQ1**: Does our proposed method outperform different debiased strategies in the recommendation, such as BC_loss [41], DCL [36], HCL [24], etc.?
- **RQ2**: How does our proposed method perform on different item groups, *e.g.*, popular items, unpopular items, etc.?
- **RQ3**: What is the contribution of various components in our framework (e.g., positive or negative correction) to the overall performance?
- **RQ4**: How robust is PopDCL with respect to the temperature hyperparameter τ? What is the relationship between the correction strength of positive and negative samples ($M^+(u,i)$, $M^-(u,j)$) and the popularity of users and items?

4.1 Experimental Settings

4.1.1 Datasets. To evaluate the performance of our proposed method, we conduct experiments on three real-world benchmark datasets: Tencent [40], Amazon-Book [16], and Alibaba-iFashion [7]. These datasets vary significantly in their domains, size, and sparsity.

- **Tencent** is a short video dataset created by the Weishi Team at Tencent Inc. Here, we only focus on the interaction between users and videos, *e.g.*, whether a user has interacted with a video.
- **Amazon-Book** consists of book review data crawled from Amazon.com, which is widely used for product recommendation tasks [15].
- **Alibaba-iFashion** is a fashion-related dataset collected by [7] from Alibaba, a large online consumer-to-consumer platform in China. We randomly sample 100000 users and their interactions as our dataset.

We apply the preprocessing to these datasets detailed in [41]. Concretely, we retain users and items with at least ten interactions. The statistics of the processed datasets are summarized in Table 2. To verify the performance on both imbalanced and balanced distribution, following [41], we first randomly sample 15% of interactions with equal probability *w.r.t.* items and assign them to the balanced test set. The remaining collection is randomly split into training, validation, and imbalanced test sets in a ratio of 60%, 10%, and 15%, respectively.

Table 2: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Sparsity
Tencent	95,709	41,602	2,937,228	0.00074
Amazon-Book	52,643	91,599	2,984,108	0.00062
Alibaba-iFashion	100,000	61,607	322,830	0.00005

4.1.2 Evaluation Metrics. We adopt two widely-used ranking-based metrics to evaluate the quality of different recommendation algorithms: Recall@K and NDCG@K. Furthermore, we assume the all-ranking protocol: all items except those with which the user has interacted in the training set are considered for the ranking. Specifically, Recall@K measures the average number of items that the users interact with that are ranked among the top-K candidates. Moreover, NDCG@K is a precision-based metric that accounts for the predicted position of the ground truth instance. Concerning both metrics, larger values indicate better performances. Following [16, 27, 41], we set $K = 20$ and report the average metrics for all users in the test set.

4.1.3 Baselines. To prove the effectiveness of our framework, we compare it with the following representative baselines:

Popularity debiased methods

- **IPS-CN** [12] is a sample re-weighting method based on items' popularity. Here, we employ the inverse of an item's popularity as the propensity score to re-weight each user-item pair.
- **CausE** [2] is a domain adaptation-inspired method that alleviates the popularity bias by causal inference.
- **sam+reg** [3] is a regularization-based popularity debiased method, which minimizes the biased correction between user-item score and item popularity.
- **MACR** [30] is model-agnostic counterfactual reasoning for eliminating popularity bias, which formulates a causal graph to describe the essential cause-effect relations in the recommendation and perform counterfactual inference to eliminate the effect of item popularity.
- **BC_loss** [41] is a popularity-aware collaborative filtering method that incorporates popularity bias margins into the contrastive loss for guiding the head and tail representation learning.

Sample debiased methods

- **DCL** [36] aims to circumvent false negatives in contrastive collaborative filtering and improves the reliability of negative instances.
- **HCL** [24] designs a "concentration parameter" β based on DCL to control the hard level for negative instances.

Here, we choose LightGCN as the backbone model and optimize the BPR loss to encode the user behaviors and items. For a fair comparison, we retain the backbone model and change the loss function to each baseline to learn the representation of users and items. For example, +IPS-CN means we train the LightGCN model by employing IPS-CN as the loss function.

4.1.4 Parameter Settings. We optimize all models using Adam [11] and employ Xavier initialization. We fix the embedding size to 64

Table 3: A comparison of the overall performance among all considered baseline methods in balanced and imbalanced test sets.

Methods	Dataset											
	Tencent				Amazon-Book				Alibaba-iFashion			
	Balanced		Imbalanced		Balanced		Imbalanced		Balanced		Imbalanced	
	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
LightGCN	0.0055	0.0042	0.1065	0.0712	0.0123	0.0116	0.0941	0.0724	0.0018	0.0009	0.0442	0.0213
+IPS-CN	0.0072	0.0054	0.0900	0.0599	0.0148	0.0136	0.0836	0.0639	0.0026	0.0013	0.0405	0.0192
+CausE	0.0055	0.0040	0.0966	0.0665	0.0134	0.0121	0.0926	0.0717	0.0013	0.0006	0.0274	0.0130
+sam+reg	0.0076	0.0056	0.0653	0.0436	0.0157	0.0149	0.0773	0.0600	0.0016	0.0008	0.0428	0.0198
+MACR	0.0075	0.0050	0.0731	0.0532	0.0183	0.0153	0.0767	0.0600	0.0010	0.0004	0.0379	0.0176
+BC_loss	0.0095	0.0073	0.1194	0.0832	0.0257	0.0227	0.1123	0.0903	0.0037	0.0017	0.0724	0.0364
+DCL	0.0082	0.0062	**0.1243**	**0.0870**	0.0203	0.0180	0.1135	0.0903	0.0041	0.0019	0.0695	0.0357
+HCL	0.0083	0.0063	0.1191	0.0830	0.0219	0.0192	0.1125	0.0889	0.0042	0.0019	0.0647	0.0334
+PopDCL	**0.0104**	**0.0078**	0.1206	0.0845	**0.0287**	**0.0250**	**0.1154**	**0.0944**	**0.0045**	**0.0020**	**0.0852**	**0.0432**
Imp.%	9.47%	6.85%	-	-	11.67%	10.13%	1.64%	4.54%	7.14%	5.26%	17.68%	18.68%

and the batch size to 2048 for all baseline models and our algorithm. The learning rate and the L_2 regularization coefficient λ are set to $1e-3$ and $1e-5$, respectively. The number of layers for LightGCN is assigned to 2. GridSearch is applied to choose the best temperature τ over $\{0.07, 0.08, \cdots, 0.27\}$. All other hyperparameters are specified according to the suggestions from the settings specified in the original publications. All models are trained on a single NVIDIA GeForce GTX 3090 GPU.

4.2 Overall Performance Comparison (RQ1)

Table 3 summarizes the best results of all considered methods across the three benchmark datasets. The percentages in the last row represent the relative improvements of the proposed method compared to the best baseline. The best performance in each column is highlighted in bold, and the second-best scores are underlined.

On the one hand, PopDCL consistently outperforms all the other baselines in balanced test sets amongst the three datasets. Especially on Amazon-Book, PopDCL exhibits an impressive increase of 11.67% in Recall@20 and 10.13% in NDCG@20. It demonstrates that PopDCL not only effectively mitigates the popularity bias but also enhances the recommendation performance by correcting both positive and negative scores. On the other hand, PopDCL also outperforms all different considered baselines in imbalanced test sets on Amazon-Book and Alibaba-iFashion datasets. Particularly on Alibaba-iFashion, PopDCL achieves a 17.68% improvement in Recall@20 and an 18.68% improvement in NDCG@20. While PopDCL slightly underperforms DCL in imbalanced test sets on the Tencent dataset. This discrepancy may be attributed to the denser nature of the Tencent dataset compared to the other two, where correcting negative samples alone leads to better performance, while correcting positive samples will have a negative impact.

Further empirical observations have been identified. First, some popularity debiased methods, such as IPS-CN, CausE, sam+reg, and MACR, have demonstrated superior performance on the Tencent and Amazon-Book datasets in balanced evaluations. However, these methods tend to struggle with imbalanced data, compromising their effectiveness. Second, BC_loss, DCL, and HCL have better performance under balanced and imbalanced evaluations. Yet, BC_loss only focuses on correcting the popularity bias for positives, while DCL and HCL focus solely on correcting sample bias for negatives. In comparison, our proposed PopDCL outperforms these methods by simultaneously correcting positive and negative scores. Third, CausE, sam+reg, and MACR underperform the original LightGCN model on the sparser Alibaba-iFashion dataset, indicating limitations in their ability to handle popularity bias effectively. Our proposed PopDCL, in contrast, exhibits the best performance, highlighting its effectiveness in addressing popularity bias.

4.3 Head, Mid, Tail Performance (RQ2)

To further verify the effectiveness of PopDCL at a finer granularity, we conducted experiments to measure the performance of the head, mid, and tail segments on the Amazon-Book dataset. Due to the limited space and the similar results in the other two datasets, we do not present them in this paper. The results of the evaluations in balanced and imbalanced settings are presented in Table 4. The percentages displayed in the upper right corner represent the relative improvements or declines of each method compared to the LightGCN baseline.

Specifically, we split the balanced and imbalanced test set of the Amazon-Book dataset into three subgroups according to the descending order of item popularity: head, mid, and tail, while ensuring that the total number of interactions in each subgroup remains the same. It can be seen from Table 4 that IPS-CN, CauseE, sam+reg, and MACR fail to achieve better performance under balanced evaluation and imbalanced evaluation simultaneously. Taking MACR as an example, its tail performance in both balanced and imbalanced evaluations has improved by 92% and 57% in NDCG@20, respectively. However, the improvement comes at the cost of the declined mid performance (-6.4% in both balanced and imbalanced evaluation) and head performance (-16% in balanced evaluation and -6.6% in imbalanced evaluation).

Furthermore, although BC_loss, DCL, and HCL achieve better head, mid and tail performance in both balanced and imbalanced evaluations, the relative improvement in each subgroup falls short of those achieved by PopDCL, for which they only provide a partial correction for either positive or negative samples. While PopDCL

Table 4: A comparison of the head, mid, and tail performance on Amazon-Book dataset.

Methods	Balanced NDCG@20				Imbalanced NDCG@20			
	Tail	Mid	Head	Overall	Tail	Mid	Head	Overall
LightGCN	0.00072	0.00515	0.02817	0.01163	0.00148	0.00765	0.07680	0.07240
+IPS-CN	$0.00098^{+36\%}$	$0.00711^{+38\%}$	$0.03291^{+17\%}$	$0.01361^{+17\%}$	$0.00200^{+35\%}$	$0.01068^{+40\%}$	$0.06898^{-10\%}$	$0.06393^{-12\%}$
+CausE	$0.00060^{-17\%}$	$0.00453^{-12\%}$	$0.02899^{+2.9\%}$	$0.01210^{+4.0\%}$	$0.00124^{-16\%}$	$0.00666^{-13\%}$	$0.07452^{-3.0\%}$	$0.07173^{-0.9\%}$
+sam+reg	$0.00089^{+24\%}$	$0.00518^{+0.6\%}$	$0.02830^{+0.5\%}$	$0.01492^{+28\%}$	$0.00159^{+7.4\%}$	$0.00771^{+0.8\%}$	$0.07482^{-2.6\%}$	$0.06001^{-17\%}$
+MACR	$0.00138^{+92\%}$	$0.00482^{-6.4\%}$	$0.02364^{-16\%}$	$0.01534^{+32\%}$	$0.00232^{+57\%}$	$0.00716^{-6.4\%}$	$0.07170^{-6.6\%}$	$0.06004^{-17\%}$
+BC_loss	$0.00521^{+624\%}$	$0.01187^{+130\%}$	$0.03952^{+40\%}$	$0.02274^{+96\%}$	$0.00612^{+314\%}$	$0.01486^{+94\%}$	$0.08835^{+15\%}$	$0.09030^{+25\%}$
+DCL	$0.00297^{+313\%}$	$0.00852^{+65\%}$	$0.03705^{+32\%}$	$0.01802^{+55\%}$	$0.00379^{+156\%}$	$0.01099^{+44\%}$	$0.09855^{+28\%}$	$0.09029^{+25\%}$
+HCL	$0.00375^{+421\%}$	$0.00963^{+87\%}$	$0.03863^{+37\%}$	$0.01919^{+65\%}$	$0.00468^{+216\%}$	$0.01276^{+67\%}$	$0.09692^{+26\%}$	$0.08890^{+23\%}$
+PopDCL	$0.00535^{+643\%}$	$0.01429^{+177\%}$	$0.04541^{+61\%}$	$0.02504^{+115\%}$	$0.00717^{+384\%}$	$0.01820^{+138\%}$	$0.09878^{+29\%}$	$0.09436^{+37\%}$

corrects the prediction score in both positive and negative samples, resulting in superior performance. In particular, PopDCL exhibits remarkable improvements in tail performance, achieving 643% and 384% enhancement in balanced and imbalanced evaluations, respectively. These findings further verify the effectiveness of PopDCL in enhancing both head and tail performance.

4.4 Ablation Study (RQ3)

Since our method corrects for both positive and negative sample scores, we conduct an ablation study to investigate the effect of positive and negative sample score corrections on overall recommendation performance. The ablation experiment setup is as follows:

- **w/o P&N** denotes that neither positive nor negative sample scores are corrected in the loss function, in which case the loss function is equivalent to SSM [33].
- **w/o P** denotes that the loss function does not correct the scores of positive samples but only negative samples.
- **w/o N** denotes that the loss function does not correct the scores of negative samples but only positive samples.
- **PopDCL** denotes that scores of both positive and negative samples are corrected.

Figure 3 shows the performance of the four variants on the Amazon-Book dataset for the balanced and imbalanced test sets.

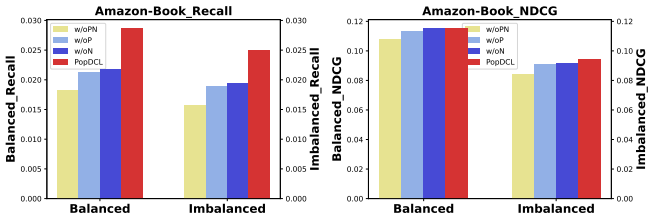

Figure 3: Ablation study (Recall@20 and NDCG@20) on the Amazon-Book dataset.

We can see that PopDCL achieves the best performance with respect to Recall@20 and NDCG@20 on both the balanced and imbalanced test sets. On the other hand, not correcting any scores leads to the worst performance, indicating the presence of false samples in both positives and negatives. In addition, correcting only a part (positives or negatives) gives a partial improvement compared to the no-correction case. However, its Recall@20 is much lower than that of PopDCL, where all scores are corrected. This highlights the necessity of correcting both positive and negative samples. The Recall@20 performance of the model with only the negatives corrected is slightly worse than that with only the positives corrected, and such a difference is almost negligible in NDCG@20. The comparable figures between the two suggest that the positive and negative sample corrections have almost the same magnitude of improvement in performance. We interpret this phenomenon as the positive and negative sample corrections share equal importance.

4.5 Study of PopDCL (RQ4)

4.5.1 Parameter Sensitivity Analysis. In this subsection, we examine the robustness with respect to the most influential temperature hyperparameter: τ. We analyze τ over the ranges $\{0.07, 0.08, \cdots, 0.21\}$. Specially, due to the sparsity of iFashion, we set τ between the ranges $\{0.07, 0.08, \cdots, 0.27\}$ to better illustrate its impact on evaluation metircs. Figure 4 depicts the Recall@20 for LightGCN+PopDCL obtained with different temperature τ on Tencent and iFashion.

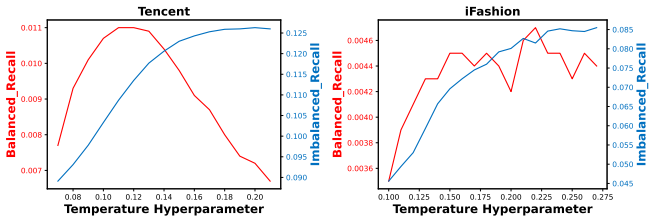

Figure 4: Recall@20 of LightGCN+PopDCL obtained with different temperature τ.

The obvious observation is that Imbalanced_Recall@20 shows the same trend on different datasets. It first increases steadily and stabilizes when the temperature coefficient reaches a certain threshold. The value τ at which Imbalanced_Recall@20 stabilizes varies among datasets, likely due to differences in dataset properties.

Balanced_Recall@20 shows a cresting trend on Tencent. With the increase of τ, Balanced_Recall@20 rises to the peak and then drops sharply. The increase of τ will smoothen the relative gradients of scores for different interaction pairs, making it more difficult for the model to distinguish hard samples. Consequently, it gradually loses the ability of hard negative mining. Since hard items are typically associated with high popularity, the model will, in turn, tend to recommend popular items, resulting in a decrease in performance on the balanced test set and an increase in performance on the imbalanced test set. This trend is consistent with the effect of τ on mining hard samples, demonstrated in Section 3.5. Since the tendency of the evaluation metrics on Amazon-Book determined by the temperature parameter τ is similar to that of Tencent, we do not show its Figure in the paper.

It is worth noting that Balanced_Recall@20 oscillates in the range $[0.0042, 0.0047]$ on iFashion when τ falls in a specific interval. We believe such oscillations are due to iFashion being too sparse and the sparsity makes the nuances of recommending the cold items affect the metrics to a large extent. Considering that τ does not overlap between Balanced and Imbalanced metrics when they reach their peak, we choose τ of 0.12, 0.10 and 0.24 as the optimal value for the Tencent, Amazon-Book and iFashion datasets respectively.

4.5.2 Effect of M. To study the relationship between the correction strength of positive and negative instances $(M^+(u,i), M^-(u,j))$ and the popularity of users and items, we randomly sample 1000 user-item pairs from the Amazon-Book dataset, Figure 5 shows the results about the relationship among $M^+(u,i), M^-(u,j)$, user popularity $pop(u)$ and item popularity $pop(i)$ or $pop(j)$. The scatter plot on the left side of Figure 5 indicates the correction strength $M^+(u,i)$ for the 1000 sampled positive pairs (u,i). The color indicates the correction strength. A redder dot indicates a higher value of $M^+(u,i)$, reflecting a larger correction strength for positive pairs. Similarly, the right plot depicts the score correction strength $M^-(u,j)$ for the negative pairs formed by the sampled users and items. To get a better view, we set the maximum value of popularity and negative correction strength $M^-(u,j)$, to 100.

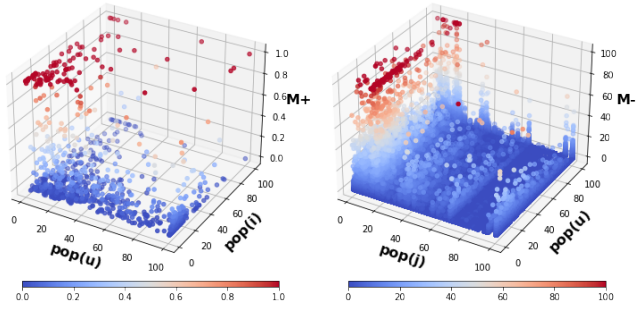

Figure 5: The relationship among $M^+(u,i), M^-(u,j)$, user popularity $pop(u)$ and item popularity $pop(i)$ or $pop(j)$ on the Amazon-Book dataset.

From the left side of Figure 5, it can be found that the red dots are mainly concentrated in the upper left corner, which indicates that for positive instances, as the popularity of the users decreases, the strength for positive scores correction is stronger. It means that if a user u has fewer interactions, the probability $P(i \notin \mathcal{N}_u)$ of item i becoming false positive for user u increases, further leading to a greater $M^+(u,i)$ according to equation (4). Theoretical analysis in Section 3.5 demonstrates if an item is likely to be a false positive sample, the score of the corresponding interaction pair (u,i) should provide the model with a less significant gradient to pay less attention to such false positive sample. The experimental results validate this claim and illustrates the effectiveness of the method.

As for $M^-(u,i)$, the results are showed in the right side of Figure 5. It is obvious that red dots are concentrated in the left region, which indicates the value of $M^-(u,j)$ increases as the popularity of item j decreases. Besides, it is worth noting that for the same item, the value of $M^-(u,j)$ does not vary significantly with the change of $pop(u)$. It can be concluded that the value of $M^+(u,i)$ mainly depends on the popularity of items j. Items with high popularity tend to have closer embedding due to the nature of GNN encoder [13, 31, 38]. It means for a certain user u, if an item j is popular, its embedding in the space will be relatively similar to those of other popular items, closing the gap between $f(u,i)$ and $f(u,j)$ and resulting a smaller $M^-(u,j)$ according to equation (10). Theoretical analysis in Section 3.5 also demonstrates that the more popular a negative sample is, the more likely it is to be a false negative sample, implying that the model needs a larger gradient to distinguish it, which requires the decrease of $M^-(u,j)$. This experimental results is also consistent with the findings of the theoretical analysis.

5 CONCLUSION AND FUTURE WORK

We proposed PopDCL, a popularity-aware debiased contrastive loss with in-batch sample strategy for collaborative filtering. Specifically, we adaptively correct the positive scores and negative scores based on the popularity of users and items. For positive instance, we designed $M^+(u,i)$ to alleviate popularity bias, where $M^+(u,i)$ is the expected score that i actually is a negative instance for user u. As for negative instances, we designed a personalized bias correction probability $\omega^+(u)$ to tackle the sample bias and further proposed $M^-(u,i)$ to correct negative scores. Theoretical analysis shows the effectiveness of PopDCL in addressing the popularity bias and sample bias. We conducted extensive empirical studies on three public datasets to verify that PopDCL significantly outperforms several existing debiased strategies in different evaluations. While the ablation study establishes the relative contributions of the respective components, the parameter sensitivity analysis shows the robustness of temperature τ for the overall performance. Finally, we reveal the relationship between the correction strength of positive and negative instances and the popularity of users and items.

In future works, we have interests in the following two aspects: 1) we use the mean of the negative sample scores to fit the expectation score of the positive pair where the positive item turns out to be negative for the user. More accurate ways can be explored in the future. 2) we plan to explore other bias problems in the recommendation, such as position bias [1] and exposure bias [42].

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China (61977002) and the State Key Laboratory of Software Development Environment of China (No. SKLSDE-2022ZX-14).

REFERENCES

[1] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating position bias without intrusive interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 474–482.

[2] Stephen Bonner and Flavian Vasile. 2018. Causal embeddings for recommendation. In *Proceedings of the 12th ACM conference on recommender systems*. 104–112.

[3] Ludovico Boratto, Gianni Fenu, and Mirko Marras. 2021. Connecting user and item perspectives in popularity debiasing for collaborative recommendation. *Information Processing & Management* 58, 1 (2021), 102387.

[4] Chong Chen, Weizhi Ma, Min Zhang, Chenyang Wang, Yiqun Liu, and Shaoping Ma. 2023. Revisiting Negative Sampling vs. Non-Sampling in Implicit Recommendation. *ACM Trans. Inf. Syst.* 41, 1, Article 12 (feb 2023), 25 pages. https://doi.org/10.1145/3522672

[5] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI*. AAAI Press, 27–34.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 1597–1607. https://proceedings.mlr.press/v119/chen20j.html

[7] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: personalized outfit generation for fashion recommendation at Alibaba iFashion. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2662–2670.

[8] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. 2020. Debiased contrastive learning. *Advances in neural information processing systems* 33 (2020), 8765–8775.

[9] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 2230–2236. https://doi.org/10.24963/ijcai.2019/309

[10] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. 2020. Simplify and Robustify Negative Sampling for Implicit Collaborative Filtering. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1094–1105. https://proceedings.neurips.cc/paper_files/paper/2020/file/0c7119e3a6a2209da6a5b90e5b5b75bd-Paper.pdf

[11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS (JMLR Proceedings, Vol. 9)*. JMLR.org, 249–256.

[12] Alois Gruson, Praveen Chandar, Christophe Charbuillet, James McInerney, Samantha Hansen, Damien Tardieu, and Ben Carterette. 2019. Offline evaluation to make decisions about playlistrecommendation algorithms. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 420–428.

[13] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* 40, 3 (2017), 52–74.

[14] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 4116–4126. https://proceedings.mlr.press/v119/hassani20a.html

[15] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

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

[17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *CoRR* abs/1708.05031 (2017).

[18] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the tenth ACM international conference on web search and data mining*. 781–789.

[19] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

[20] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

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

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian Personalized Ranking from Implicit Feedback. *CoRR* abs/1205.2618 (2012).

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

[24] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2021. CONTRASTIVE LEARNING WITH HARD NEGATIVE SAMPLES. In *International Conference on Learning Representations (ICLR)*.

[25] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased recommender learning from missing-not-at-random implicit feedback. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 501–509.

[26] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1lfF2NYvH

[27] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat Seng Chua. 2019. Neural Graph Collaborative Filtering. In *the 42nd International ACM SIGIR Conference*.

[28] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced Negative Sampling over Knowledge Graph for Recommendation. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 99–109. https://doi.org/10.1145/3366423.3380098

[29] Jason Wei and Kai Zou. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 6382–6388. https://doi.org/10.18653/v1/D19-1670

[30] Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. 2021. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1791–1800.

[31] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[32] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-Supervised Graph Learning for Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 726–735. https://doi.org/10.1145/3404835.3462862

[33] Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, Tianyu Qiu, and Xiangnan He. 2022. On the Effectiveness of Sampled Softmax Loss for Item Recommendation. *CoRR* abs/2201.02327 (2022).

[34] Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. CLEAR: Contrastive Learning for Sentence Representation. *ArXiv* abs/2012.15466 (2020).

[35] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 3203–3209. https://doi.org/10.24963/ijcai.2017/447

[36] Chenxiao Yang, Qitian Wu, Jipeng Jin, Xiaofeng Gao, Junwei Pan, and Guihai Chen. 2022. Trading Hard Negatives and True Negatives: A Debiased Contrastive Collaborative Filtering Approach. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 2355–2361. https://doi.org/10.24963/ijcai.2022/327 Main Track.

[37] Wun-Ting Yang, Chiao-Ting Chen, Chuan-Yun Sang, and Szu-Hao Huang. 2023. Reinforced PU-Learning with Hybrid Negative Sampling Strategies for Recommendation. *ACM Trans. Intell. Syst. Technol.* (feb 2023). https://doi.org/10.1145/3582562 Just Accepted.

[38] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*. ACM, 974–983.

[39] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are Graph Augmentations Necessary? Simple Graph Contrastive Learning for Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 1294–1303. https://doi.org/10.1145/3477495.3531937

[40] Fajie Yuan, Xiangnan He, Haochuan Jiang, Guibing Guo, Jian Xiong, Zhezhao Xu, and Yilin Xiong. 2020. Future data helps training: Modeling future contexts for session-based recommendation. In *Proceedings of The Web Conference 2020*. 303–313.

[41] An Zhang, Wenchang Ma, Xiang Wang, and Tat-Seng Chua. 2022. Incorporating Bias-aware Margins into Contrastive Loss for Collaborative Filtering. *arXiv preprint arXiv:2210.11054* (2022).

[42] Wenhao Zhang, Wentian Bao, Xiao-Yang Liu, Keping Yang, Quan Lin, Hong Wen, and Ramin Ramezani. 2020. Large-scale causal approaches to debiasing post-click conversion rate estimation with multi-task learning. In *Proceedings of The Web Conference 2020*. 2775–2781.

[43] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive learning for debiased candidate generation in large-scale recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3985–3995.

[44] Ziwei Zhu, Yun He, Xing Zhao, and James Caverlee. 2021. Popularity Bias in Dynamic Recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 2439–2449. https://doi.org/10.1145/3447548.3467376

[45] Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee. 2021. Popularity-opportunity bias in collaborative filtering. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 85–93.