TOWARD RESPONSIBLE RECOMMENDER SYSTEMS

A Dissertation

by

ZIWEI ZHU

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2022

Major Subject: Computer Science

ABSTRACT

Recommender systems have become essential conduits: they can shape the media we consume, the jobs we seek, and even the friendships and professional contacts that form our social circles. With such a wide usage and impact, recommender systems can exert strong, but often unforeseen, and sometimes even detrimental influence on the social processes connected to culture, lifestyles, politics, education, ethics, economic well-being, and even social justice. Hence, in this dissertation research, we aim to identify, analyze, and alleviate potential risks and harms on users, item providers, the platforms, and ultimately the society, and to lay the foundation for new responsible recommender systems. In particular, we make three unique contributions toward responsible recommender systems:

- First, we study how to counteract the exposure bias in user-item interaction data. To overcome the challenge that the user-item exposure information is hard to be estimated when aiming to produce unbiased recommendations, we develop a novel combinational joint learning framework to learn unbiased user-item relevance and unbiased user-item exposure information simultaneously. Then, we push the problem to an extreme where we aim to predict relevance for items with zero exposure in the interaction data. For this, we propose a neural network utilizing a randomized training mechanism and a Mixture-of-Experts Transformation structure. Experiments validate the effective performance by the proposed methods.

- Second, we study what bias the machine learning based recommendation algorithms can bring and how to alleviate these bias. We uncover the popularity-opportunity bias on items and the mainstream bias on users. We conduct extensive data-driven study to show the existence of these bias in fundamental recommendation algorithms. Then, we explore and propose potential solutions to relieve these two types of bias, which empirically demonstrate outstanding performance for debiasing.

- At last, we move our attention to the problem of how to measure and enhance fairness in recommendation results. We study the recommendation fairness in three different recommendation scenarios – the multi-dimension recommendation scenario, the personalized ranking recommendation scenario, and the cold-start recommendation scenario. With respect to different recommendation scenarios, we develop different algorithms to enhance the recommendation fairness. We also conduct extensive experiments to empirically show the effectiveness of the proposed solutions.

DEDICATION

To mom, dad, Yue and all the people who love me and I love.

ACKNOWLEDGMENTS

Without the advice, support, and love from a few important people who I am fortunate to have in my life, I would never have the chance to complete this dissertation.

First, I want to express my sincerest gratitude to my advisor, mentor, and good friend Dr. James Caverlee, who changed and shaped me into a mature researcher and a better person with his selfless and tirelessly advice and support. It was still fresh in memory that in August 2017, at the moment when I run into his homepage the first time, I realized immediately he is the advisor I want to work with. It turns out I was right, and it was one of the best and most important decisions in my life to reach out to him and join his InfoLab. His influence on students is far beyond research. During the 10-th year reunion of the InfoLab in October 2017, a formal Ph.D. student Kyumin, who was an Associate Professor at Worcester Polytechnic Institute, told me that the reason why he chose to get a faculty job is that he wanted to become a cool educator just like Dr. Caverlee. And after a few years, I finally understand what he said and cannot agree more. So, I decided to follow the same path and set the goal for myself to become a cool educator like Dr. Caverlee and to pass his influence to other people. Five years fly so fast, and it's time to say goodbye and start my new journey. But with no doubt, he will be my lifelong role model, with all his guidance and advice, as well as his kindness, generousness, and patience kept in my mind.

In addition, I would also like to thank the rest of my dissertation committee: Dr. Xia Hu, Dr. Xiaoning Qian, and Dr. Ricardo Gutierrez-Osuna for their continuous advice and support. I would also like to thank Dr. Alex Beutel for his help and essential inspirations to my research. Besides, all the members in the InfoLab provide me support and help everyday in the past five years. Many thanks to them – Xing Zhao, Yun He, Jianling Wang, Yin Zhang, Majid Alfifi, Parisa Kaghazgaran, and Zhuoer Wang. I enjoyed all the time spending with them. They helped me get through countless failures and difficulties, and it would not be possible for me to complete this dissertation without them. And I also want to express my appreciation to Dr. Shahin Sefati and Dr. Jingu Kim for offering great internship opportunities to work with them. They are great people and great researchers, from whom I learned things that are hard to learn in the academia.

At the end, I am deeply indebted to my parents Jianhong Cai and Libo Zhu for their endless love and support. Studying abroad and being far away from home is not only a pain for me, but a greater pain for them. But they encourage me to pursue my dream all the time. Without their love and sacrifice for me, I cannot become who I am right now. Finally, I would like to thank my wife Yue He. With no doubt, without her, it would be impossible for me to pursue the Ph.D. degree in the U.S. and survive from all these difficulties. It is still like a miracle to me that seven years before, we were working so hard together in China for applying for graduate schools in the U.S., and now, we are married and are having a good life here. Deep in my heart, I know she makes me stronger, and with her, I am not afraid of any difficulties.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor James Caverlee [advisor], Professor Ricardo Gutierrez-Osuna, and Professor Xia Hu of the Department of Computer Science and Engineering, and Professor Xiaoning Qian of the Department of Electrical and Computer Engineering. Section 3.3 is conducted based on the work majorly done while I was interning at Comcast Applied AI Lab. Section 5.4 is conducted based on the work majorly done while I was interning at Netflix. All work conducted for the dissertation was completed by the student independently.

Funding Sources

This work is, in part, supported by DARPA (#W911NF-16-1-0565), NSF (#IIS-1939716 and #IIS-1841138), and Amazon Research Awards. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Background and Motivation

In the past few years, tremendous efforts have been dedicated to creating intelligent recommendation algorithms and systems, such as with the help of machine learning and deep learning techniques [14, 15, 16, 17, 18]. The success of these powerful recommender systems is evident across many aspects of our lives: nowadays, we rely on recommender systems to discover media to consume (like YouTube and Netflix), news to read (like Apple News and Google News), places to visit (like Airbnb and Expedia), courses to take (like Coursera and Udacity), even the jobs to apply to (like LinkedIn), and friendships to build (like Facebook).

With such a wide usage and impact, these recommender systems can exert strong but often unforeseen influence on the social processes connected to culture, lifestyles, politics, education, ethics, economic well-being, and even social justice. This influence can sometimes have sinister repercussions: for example, machine learning based recommender systems have been shown to exhibit discrimination against women and people of color [19]; produce bias against women in job hunting [20] and content creation [21]; expose children to inappropriate content [22]; and intensify political polarization [23]. Hence, advancing the development of recommender systems while being blind to these potential risks and adverse impacts is dangerous.

Therefore, this dissertation aims to lay the foundation for new *responsible recommender systems*. That is, rather than myopically focusing on maximizing a single utility metric (e.g., revenue, views, or user engagement), we should design recommender systems with an eye on the potential risks and adverse impacts on users, item providers, the platforms, and ultimately the society. These risks exist in every step in the pipeline of a recommender system, including bias in its many forms, unfairness for users or items, filter bubbles, polarization, rabbit holes, and so on.

In particular, as shown in Figure 1.1, the pipeline of a recommender system can be viewed as a closed feedback loop involving three key components: users, interaction data from users, and

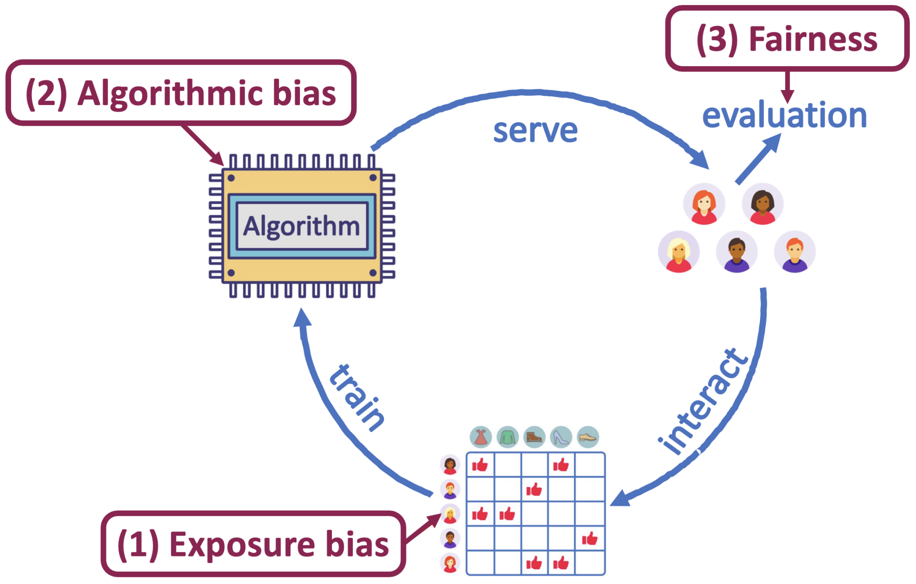


Figure 1.1: A closed feedback loop as the pipeline of a recommender system, in which we study the problems at each of the three key components in the pipeline.

the machine learning algorithms. The machine learning algorithm aims to recommend potential interesting items (items can be anything on the Internet, e.g., web pages, job positions, or people as potential friends) to users; users interact with the recommendations through a set of actions, like clicks, views or purchases; the collected interaction data is then used to further train the machine learning algorithm; the performance of the system is evaluated based on the feedback on the recommendation result provided by users; and then, this loop continues.

In such a pipeline, risks and harms can happen at every step. So, in this dissertation, we target three different problems in each of the three key components in this pipeline – the exposure bias in the interaction data, algorithmic bias in the machine learning based recommendation algorithms, and unfairness in the recommendation result – and aim to identify, analyze, and alleviate these risks and harms to realizing responsible recommender systems.

Exposure Bias

One core problem that can incur risks in a recommender system is the inherent *exposure bias* in user-item interaction data. Modern recommender systems based on machine learning highly rely on user-item interaction data to learn user-item relevance. However, this user-item interaction data is usually generated in a biased environment, that is, users are exposed to different items with different probabilities, and only if a user is exposed to an item, can an interaction between them happen. So, the user-item interaction data with such a bias cannot correctly reflect the true user-item relevance. For example, if the user-item interaction data is collected from an existing recommender system, items more frequently recommended in this logging system potentially have more interactions in the collected data. Then, a new recommendation model trained on this user-item interaction data cannot precisely learn the true user-item relevance and would be biased toward those items previously preferred by the logging system.

Building recommender systems unaware of this exposure bias can result in inferior recommendation quality because a recommendation model trained on this biased data cannot accurately learn the true user-item relevance but learns a mixture of user-item relevance and the behavior patterns of the logging system. More importantly, because the newly built systems can inherit the behavior patterns of logging systems due to the exposure bias, the more crucial harm of such an exposure bias is that it can contribute to accumulating and exaggerating other detrimental issues, like filter bubble, unfairness, polarization, and so on.

Hence, techniques to counteract the exposure bias is in need. However, it is non-trivial to achieve this. One major challenge is that to parse the true user-item relevance information from the user-item interaction data, we need to know the user-item exposure information beforehand, which is in fact unknown and hard to be estimated. Besides, another vital challenge (also known as the well-known *cold-start* problem) is how can we deal with situations with extreme exposure bias. That is, when there are some items with zero exposure before and without any historical interaction data, how do we build a recommender system being capable to effectively recommend these items?

Algorithmic Bias

In addition to the bias raised by data, it is also crucial that we are aware of and are able to address the bias intrinsic in the machine learning algorithms. One long-standing algorithmic bias is the **popularity bias** that algorithms have the tendency to more frequently recommend popular items to users at the expense of less exposure for less popular ones, even if the recommendation models are trained based on unbiased interaction data. This can potentially incur the ‘rich get richer’ problem, which will decrease user satisfactions and is unfair toward less popular but not necessarily lower-quality items. However, the conventional measurement of popularity bias considers the difference of the recommendation frequency to all users across items with different levels of popularity, which does not take the ground truth of user-item matching into account. Yet, only recommending items to matched users, who will interact the item once recommended, can influence user satisfactions and engagement items receive. Hence, measuring the popularity bias over recommendations to all users without conditioned on the ground truth of user-item matching cannot really indicate an undesired harm and the ‘rich get richer’ problem. Challenges remain unsolved: How can we measure the popularity bias with the ground truth of user-item matching taken into account? Whether this new type of bias does exist in recommender systems? And how can we effectively eliminate such a bias?

Besides bias on items, we also wonder what kind of bias the machine learning algorithms can engender to users. We want to study the bias on users that: Whether users with different preferences are treated differently by the recommender algorithms? And if the majority of users sharing similar preferences receive recommendations of higher quality than niche users who prefers items that are out of the mainstream do? Such a bias can be problematic because it is unfair to consistently provide low-quality recommendations to niche users and can eventually drive niche users away from the system. More importantly, this bias could reduce the inclusiveness with respect to niche opinions and views in essential platforms related to social information and opinions, such as social media or news platform, leading to intellectual segregation and societal polarization. Toward thoroughly understanding and solving this *mainstream bias*, we need to tackle several research questions: How do we identify mainstream users vs. niche ones? What impact does the degree of mainstream-ness have on recommendation utility? And can we develop methods to ameliorate this mainstream bias? Can we improve the recommendation utility for users of low mainstream levels while preserving or even increasing the utility for mainstream users at the same time?

Recommendation Unfairness

At last, we are interested in how the bias in interaction data and the bias in the algorithms lead to unfair recommendation. In other words, we aim to study are different items or item groups treated fairly in a recommender system? For example, in a job recommender that recommends job openings, are high-paying jobs and non-profit jobs treated fairly? In a news recommender, are news of different political ideologies recommended at similar rate? And even for product recommenders, are products from big companies favored over products from new entrants? The harm of recommendation unfairness among items has been recognized in the literature [24, 25, 26], with potential adverse impacts on item providers, user satisfaction, the recommendation platform itself, and ultimately social good.

Although many existing work have put efforts to develop new recommendation algorithms that can enhance fairness, it is still challenging to measure and enhance fairness in some special but pervasive recommendation scenarios. How can we enhance recommendation fairness in a multi-dimension recommender system, where items are recommended to users given different conditions? How can we measure and enhance fairness directly on ranking results in a personalized ranking system? And how can we enhance recommendation fairness among new items in a cold-start recommender system, where there is no historical interaction data for these new items?

1.2 Dissertation Contributions

To answer these research questions and address the challenges, we make three unique contributions investigating and tackling three different types of problems in recommender systems.

- The first contribution of this dissertation research lies in counteracting *exposure bias* in user-item interaction data used for training recommendation models. Toward tackling this problem, we propose a novel combinational joint learning framework to learn user-item relevance and exposure information simultaneously, which can effectively remove the exposure bias in the training data and provide unbiased recommendations. Moreover, we also consider the more challenging situation with extreme exposure bias, where some items have zero exposure probability. This is also known as the cold-start problem in recommender systems. We further develop a novel and effective cold-start recommendation algorithm – Heater – to provide high-quality recommendations for these cold-start items. Extensive experiments validate the effective performance by the two proposed methods.

- Second, we investigate how do the machine learning based recommendation algorithms introduce additional bias even if the training data is free of exposure bias. We first conduct both theoretical and empirical studies on fundamental recommendation models to uncover *popularity-opportunity bias*, an intrinsic problem for recommender algorithms. Popularity-opportunity bias refers to the phenomenon that recommendation models tend to more frequently recommend popular items to matched users (who are willing to interact with these items once recommended) than less popular items. To mitigate this popularity-opportunity bias, we propose a simple but effective post-processing popularity compensation algorithm. Besides the bias on the item side, we also study the bias produced by algorithms on the user side. We conduct data-driven study to reveal the *mainstream bias* among users that users with mainstream preference (the majority group of users who share similar preference) receive more accurate recommendations than users with niche preference. To alleviate this mainstream bias among users, we propose a local fine tuning method to improve the recommendation quality for every user by delivering a customized model for each user. We also conduct extensive experiments to empirically show the outstanding performance of the proposed algorithms for debiasing.

- Third, we investigate the *fairness* in the recommendation result, and aim to propose methods to measure and enhance recommendation fairness. In particular, we explore recommendation fairness in three different scenarios: the multi-dimension recommendation scenario, where besides the two dimensions of users and items, there are other dimensions as additional conditions determining how items are recommended to users; the personalized ranking recommendation scenario, where recommendations are presented as ranking lists of items and all evaluations including recommendation quality and fairness should be conducted based on ranking results; and the cold-start recommendation scenario, where cold-start items with no historical interaction are to be recommended. For each of these different recommendation scenarios, we conduct empirical analysis on recommendation fairness and propose novel solutions to enhance fairness which empirically show effective performance.

1.3 Dissertation Overview

The remainder of this dissertation is organized as follows:

- **Chapter 2:** In this chapter, we introduce the preliminaries of this dissertation. We first briefly introduce the formalization of the recommendation task and also some widely-used metrics for evaluating the recommendation result. Then, we introduce the fundamental recommendation algorithm – matrix factorization, which serves as the foundation for the research in this dissertation.

- **Chapter 3:** In this chapter, we introduce the first contribution in this dissertation – to counteract the exposure bias in the user-item interaction data in a recommender system. Specifically, we first propose a novel and effective combinational joint learning framework, which is able to learn user-item exposure and user-item relevance jointly to deliver unbiased recommendation results. Then, we explore further to investigate how to produce high-accuracy recommendations in the scenario with extreme exposure bias, where there are items without any historical interaction data. We introduce our proposed algorithm Heater to tackle this cold-start problem.

- **Chapter 4:** In this chapter, we move our focus from bias in data to the bias in the machine learning algorithms. We first introduce the popularity-opportunity bias among items produced by the recommendation algorithm. We conduct both empirical and theoretical study to show the existence of this bias and propose simple but effective post-processing algorithms to mitigate this bias. Then, we explore the bias produced by recommendation algorithms among users and introduce the mainstream bias. We explore the potential ways to identify niche users and introduce two global methods and one novel local method to alleviate this mainstream bias.

- **Chapter 5:** In this chapter, we study how to measure and enhance fairness in recommender systems. In particular, we study recommendation fairness in three different scenarios. First, we study how to enhance fairness in a multi-dimension recommender system and develop a fairness-aware tensor-based model to achieve the goal. Then, we switch our attention to personalized ranking systems. We introduce two new fairness measurements directly on ranking results and propose an adversarial learning based algorithm for enhancing fairness in the personalized ranking recommender system. At last, we investigate fairness in the cold-start recommender systems and develop a novel learnable post-processing framework and a joint-learning generative model to improve fairness among cold-start items without any historical interaction data.
- **Chapter 6:** At last, we conclude this dissertation by a summary of contributions and a discussion of future research opportunities.

2. PRELIMINARIES

In this chapter, we introduce the preliminaries of this dissertation, including the formalization of the recommendation problem, the evaluation metrics, and the fundamental matrix factorization algorithm for recommendation.

2.1 Formalization of Recommender Systems

In a recommender system as demonstrated in Figure 1.1, we have a set of N users denoted as $\mathcal{U} = \{1, 2, \ldots, N\}$ and a set of M items denoted as $\mathcal{I} = \{1, 2, \ldots, M\}$. At a certain moment in the life cycle of such a system, we can have a set of user-item interactions (like clicks, views, or purchase records) collected up to now. We denote this set of interactions from users to items as $\mathcal{O} = \{(u, i)\}$ where $u \in \mathcal{U}$ indexes one user, and $i \in \mathcal{I}$ indexes one item. We use this data as the training data to train/update a core recommendation model. And the newly trained/updated recommendation model will provide the next-round recommendations to users. Specifically, by the recommendation model, we need to provide a list $\mathbf{R}^u$ containing k items to every user. And for a user u, the recommendation is based on her historical interaction record $\mathcal{O}_u = \{i, j, \ldots\}$, where $i, j, \ldots$ are the items u has interacted with before. With the new recommendations, users provide new interactions to items, resulting in new training data, and such a loop in Figure 1.1 continues.

In a real-world online recommender system, we can directly evaluate the system based on the user engagement in the system over a period of time. However, when we conduct offline experiments to evaluate a recommender algorithm, we need to have another item set $\widetilde{\mathcal{O}}_u$ to represent the items that user u will like during evaluation, which serve as the ground-truth test data for evaluating recommendation quality. And most research works [27, 17, 16, 12] adopt this offline experiment paradigm to evaluate and compare different regimentation algorithms.

2.2 Evaluation Metrics

Many ranking evaluation metrics can be used to evaluate the recommendation result in an offline experiment. Here we introduce four most widely used ones:

Recall@k: calculates the true positive rate of the provided recommendations. In other words, it shows the expectation of how many items a user likes can be recommended to the user. The higher the metric is, the more items the user likes can be exposed to the user, indicating the better the provided recommendations are. The metric can be formulated as:

$$Recall@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\mathbf{R}^u \cap \widetilde{\mathcal{O}}_u|}{|\widetilde{\mathcal{O}}_u|}. \tag{2.1}$$

Precision@k: calculates the positive predictive value of the provided recommendations. It shows the expectation of how many items in the recommendations are useful for the user. the higher the metrics is, the more items in the recommendations will be liked by the user, the better the recommendations are. The metric is formulated as:

$$Precision@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\mathbf{R}^u \cap \widetilde{\mathcal{O}}_u|}{k}. \tag{2.2}$$

F1@k: aims to combine Recall@k and Precision@k because Recall@k and Precision@k evaluate different perspectives of the recommendations. F1@k calculates the harmonic mean of Recall@k and Precision@K , formulated as:

$$F1@k = 2\frac{Recall@k \cdot Precision@k}{Recall@k + Precision@k}. \tag{2.3}$$

DCG@k: evaluates the quality of a ranking system and takes the importance of ranking position into account. Unlike Recall@k, Precision@k, and F1@k, which consider all the top k positions equally important, DCG@k applies a logarithmic decay to estimate the position importance. It can be formulated as:

$$DCG@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \sum_{0 < i \leq k} \frac{\delta(\mathbf{R}^u[i] \in \widetilde{\mathcal{O}}_u)}{log_2(i+1)}, \tag{2.4}$$

where $\delta(x)$ returns 1 if x is true, otherwise 0; and $\mathbf{R}^u[i]$ fetches the item at the i-th position in the recommended list $\mathbf{R}^u$.

NDCG@k: further normalizes the DCG@k to avoid the problem that different users may have DCG@k at different scales. So, NDCG@k normalizes the DCG@k by the best possible DCG@k value for each user, making the resulted metric to be in the range of $[0, 1]$. It is formulated as:

$$NDCG@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \Big(\sum_{0<i\leq k} \frac{\delta(\mathbf{R}^u[i] \in \widetilde{\mathcal{O}}_u)}{log_2(i+1)} \cdot \sum_{0<j\leq|\widetilde{\mathcal{O}}_u|} log_2(j+1) \Big). \tag{2.5}$$

2.3 Matrix Factorization

Collaborative filtering [28] is the core of machine learning based recommendation algorithms. The main idea is to find items liked by users who are similar to the target user and recommend them to the target user, or to recommend a target item to users who like items that are similar to the target item. Matrix factorization is one of the most widely used collaborative filtering algorithms [29, 27], which considers both user-user similarity and item-item similarity simultaneously. Matrix factorization has become the foundation for many state-of-the-art recommendation models [30, 31], as well as recent neural-network based models [17, 32, 33] that use matrix factorization as the final layer for predicting preference scores. The main idea of matrix factorization is to learn low-dimensional latent representations for users and items based on existing user-item interactions, and then to predict preference scores for unobserved user-item pairs by the dot-product of latent representations:

$$\widehat{y}_{u,i} = \mathbf{P}_u^\top \mathbf{Q}_i, \tag{2.6}$$

where $\mathbf{P}_u \in \mathbb{R}^{H\times 1}$ is the latent representation of user u, $\mathbf{Q}_i \in \mathbb{R}^{H\times 1}$ is the latent representation of item i, and H is the latent dimension. With the predicted user-item relevance scores $\widehat{y}_{u,i}$, we rank items for each user by sorting them in the descending order of their relevance scores.

There are two main categories of objective functions for matrix factorization models: point-wise objective functions (including Root Mean Square Error (RMSE) [29], Cross-Entropy [17], among others) and pair-wise objective functions (including Bayesian Personalized Ranking loss (BPR) [27], Hinge loss [34], and others). Since RMSE and BPR are two of the most widely applied objective functions, we mainly focus on these two in the rest of this dissertation. We denote the matrix factorization model with RMSE as **RMSE**, and the one with BPR loss as **BPR**. The formulations are shown below:

$$\min_{\mathbf{\Theta}} \mathcal{L}_{RMSE} = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{O}_u \cup \mathcal{O}_u^-} \sqrt{(\widehat{y}_{u,i} - y_{u,i})^2}, \tag{2.7}$$

$$\min_{\mathbf{\Theta}} \mathcal{L}_{BPR} = -\sum_{u \in \mathcal{U}} \sum_{\substack{i \in \mathcal{O}_u \\ j \in \mathcal{O}_u^-}} ln\, \sigma(\widehat{y}_{u,i} - \widehat{y}_{u,j}), \tag{2.8}$$

where $\mathcal{O}_u^-$ is the randomly sampled negative item set for u; $y_{u,i}$ indicates the relevance of the user-item pair (u, i) in training data – $y_{u,i} = 1$ if $i \in \mathcal{O}_u$, otherwise 0; $\sigma(\cdot)$ is the Sigmoid function; and $\mathbf{\Theta}$ represents the model parameters, i.e., the latent representations for users and items $\mathbf{P}$ and $\mathbf{Q}$. Using the point-wise objective function like RMSE means that we treat the recommendation task as a user-item relevance regression problem. The advantage of using the point-wise objective function is that the training process can be simple and flexible because we treat each user-item pair independently during training. However, the drawback of the point-wise objective function is that there is a big gap between the objective function and the final ranking-based evaluation of the recommendations. Hence, the pair-wise objective function like BPR, which is directly aligned with ranking-based evaluation, can often deliver better recommendation quality.

3. COUNTERACTING EXPOSURE BIAS IN USER-ITEM INTERACTION DATA[1]

In this chapter, we introduce the first contribution of this dissertation research – a set of techniques for counteracting the exposure bias in user-item interaction data. In particular, we first target one of the major challenges for addressing the exposure bias that the user-item exposure information is unknown and hard to be estimated. To tackle this challenge, we propose a combinational joint learning framework, which learns the user-item relevance and user-item exposure simultaneously. Then, we also develop a new model with a randomized training mechanism and a Mixture-of-Experts Transformation structure to attack the big challenge of delivering recommendation in the situation with extreme exposure bias. In this extreme exposure bias situation, some items have zero exposure in the user-item interaction data, in other words, they do not have any historical interaction data. This recommendation problem with extreme exposure bias is also known as the cold-start problem. Our developed methods can effectively alleviate the exposure bias and produce recommendation of high accuracy.

3.1 Related Work

In this section, we will introduce the related work about how to address the exposure bias in the warm-start recommendation task and how to provide recommendation in the cold-start scenario.

3.1.1 Addressing Exposure Bias

Prior works have studied feedback bias in explicit ratings in recommender systems [35, 36, 37, 38, 39, 40, 41, 42], especially since people can be selective for which items to provide ratings [35, 43, 36, 44, 37]. As a result, many approaches are inherently biased, with more accurate predictions for high ratings than for low ratings. To address this, previous works propose to alleviate bias in

[1]This chapter is reprinted with permission from "Unbiased Implicit Recommendation and Propensity Estimation via Combinational Joint Learning" by Ziwei Zhu, Yun He, Yin Zhang, and James Caverlee, 2020, Proceedings of the 14th ACM Conference on Recommender Systems, Copyright 2020 by ACM; and "Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation" by Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah and James Caverlee, 2020, Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Copyright 2020 by ACM.

terms of both model learning [35, 36, 38, 39, 40, 41, 42, 45] and evaluation metrics [36, 37, 35].

With increasing impact made by implicit recommender systems, investigating the bias in implicit feedback is in high demand. Yang et al. [46] studied the influence of bias in implicit feedback in term of evaluation, showing that conventional evaluation metrics are biased toward high-exposure items and proposing unbiased metrics based on IPS method. Based on the IPS method as well, Saito et al. [47] proposed the first unbiased recommendation model to learn unbiased user-item relevance from biased implicit data. Later, Saito [48] extended the point-wise model in [47] to a pair-wise version, which delivers improved performance. However, although these existing unbiased methods theoretically guarantee to generate unbiased recommendation, their reliance on naïve item popularity based estimation of propensity in the IPS method can still lead to inaccurate and biased recommendation. In this work, we address this issue by the proposed combinational joint learning method to learn both unbiased relevance and unbiased propensity simultaneously.

3.1.2 Addressing Cold-start Problem

One inherent drawback of the recommendation algorithms mentioned above is that they can not recommend in the extreme exposure bias scenario, where there are users or items without any historical interaction data. However, this cold-start scenario is common in many real-world applications. There are two main categories of modern approaches: separate-training and joint-training methods.

Separate-training methods separate the learning of the CF model and the transformation from auxiliary representations to CF representations. One typical group of methods is to first pretrain a CF model on the user-item interaction data and generate pretrained CF representations, then learn the transformation functions based on the pretrained CF representations. One example is LinMap proposed by Gantner et al. [6], which learns a linear transformation matrix to map auxiliary representations to pretrained CF representations. Another example is DeepMusic [49], which use deep neural networks to transform music audio signals to pretrained CF representations. Another group of methods [50, 51, 52] combines the learning of CF and transformation functions into one model, but they have their own loss components in the objective function and a trade off of learning strength between them must be achieved. For example, CMF proposed by Singh et al. [50] learns a matrix factorization and a linear transformation function in one model, whose core idea is similar to LinMap.

Joint-training methods learn CF and transformation functions jointly to minimize the recommendation error on the user-item interaction data in one model. A typical example is DropoutNet proposed by Volkovs et al. [5], which transforms auxiliary representations to CF representations by multi-layer perceptions, and learns the model parameters by minimizing a recommendation error. Another thread of methods directly learns a transformation from auxiliary representations to preference scores [3, 4] without generating any intermediate CF representations, which is inspired by Autoencoder based CF [53].

Beside these two directions, there are some works adapting the idea of meta-learning to the cold-start recommendation field [2, 54]. Vartak ei al. [2] propose to generate different models for different users based on their historical preferences, then apply these user-specific models to auxiliary representations of cold-start tweets to predict user preference towards tweets. Lee et al. [54] build a recommender that is able to adapt to users or items with very few interaction records (instead of no feedback as in our work) so that it can perform well on new users or items.

3.2 Counteracting Exposure Bias via a Combinational Joint Learning Framework

3.2.1 Introduction

Widespread implicit user-item interactions such as user purchases, views, and clicks, have been widely used in recommender systems with far-reaching impact [55, 56, 27]. However, recent studies [47, 57, 48, 46] show that these implicit interactions are not necessarily aligned with user preferences. Since the observed interactions are determined by both *user-item relevance* and *item exposure*, learning a model directly from implicit interactions results in biased recommendations.

To address this issue, recent works have proposed unbiased recommendation models by applying principles from Inverse Propensity Scoring (IPS) [47, 48]. These unbiased algorithms can theoretically guarantee to produce unbiased user-item relevance prediction conditioned on having an unbiased estimation of the propensity (i.e., the user-item exposure probability). However, these works model this item exposure probability by a power-law function of item popularity (e.g., using the number of interactions received by an item), which *is not an unbiased estimate of the propensity*. The item exposure probability of an item depends on the total number of users who have seen the item, which is a function of both the observed positive feedback (e.g., clicks) and the *unobserved negative feedback*. That is, some users may see an item but not interact with it. This unobserved negative feedback is missing from existing item exposure methods based on item popularity, and so bias may still be introduced into seemingly unbiased methods.

Therefore, we propose a combinational joint learning framework that is designed to simultaneously learn unbiased user-item relevance and unbiased propensity. More specifically, we first introduce an unbiased propensity estimation method that aims to learn the unbiased user-item exposure probability directly from observed user-item interaction records, rather than assuming a power law distribution. Such an approach has the benefit of learning propensity directly from data, sidestepping the disadvantages of heuristic methods used in previous works. Because learning unbiased relevance and learning unbiased propensity are conditioned on each other, a straightforward way for unbiased recommendation is to learn both of them via a joint learning model. However, we show how a naïve joint learning method that iteratively train an unbiased relevance model and an unbiased propensity model can still lead to a special *estimation-training overlap* problem, wherein the learning of the relevance and propensity models shares the same training data, leading to biased results. Hence, we propose a new combinational joint learning framework that jointly learns multiple unbiased relevance and propensity sub-models from different parts of the training dataset to avoid this estimation-training overlap problem. We further show how to incorporate residual components trained by the complete training data to complement these relevance and propensity sub-models, leading to unbiased prediction of user-item relevance and propensity. By experiments on two real-world datasets, we show how the proposed model improves existing unbiased recommendation methods with an improvement of 4% on average over the best alternatives.

3.2.2 Proposed Combinational Joint Learning Framework

In this section, we introduce a combinational joint learning framework that jointly learns unbiased relevance and propensity simultaneously. We begin by formalizing the implicit recommendation problem and introducing an unbiased objective function from previous work to model the unbiased user-item relevance. We then show how to estimate unbiased propensity – to overcome the hidden bias prevalent in many previous approaches to estimate propensity – and then provide the details of the combinational joint learning framework.

3.2.2.1 Preliminaries

Problem Statement

Suppose we have a user set $\mathcal{U} = \{1, 2, \dots, N\}$, an item set $\mathcal{I} = \{1, 2, \dots, M\}$, and a user-item interaction variable $Y_{u,i} \in \{0, 1\}$ where $u \in \mathcal{U}$ and $i \in \mathcal{I}$ recording observed interactions ($Y_{u,i} = 1$) or unknown interactions ($Y_{u,i} = 0$). We use $\mathcal{D}$ to denote the training data with all observed user-item interactions and some unknown interactions (by random negative sampling). To model the observed interaction variable, previous works [48, 47, 46] introduce two hidden variables: the relevance variable $R_{u,i} \in \{0, 1\}$ indicating whether user u likes item i ($R_{u,i} = 1$) or not ($R_{u,i} = 0$); and the exposure variable $O_{u,i} \in \{0, 1\}$ indicating whether item i is exposed to user u ($O_{u,i} = 1$) or not ($O_{u,i} = 0$). Then, the interaction variable is modeled as: $Y_{u,i} = R_{u,i} \cdot O_{u,i}$, i.e., only if user u likes item i ($R_{u,i} = 1$) and i is exposed to u ($O_{u,i} = 1$), can we observe $Y_{u,i} = 1$. Hence, the task of unbiased implicit recommender systems is to infer user-item relevance $R_{u,i}$ and provide ranked lists of items to users based on the observed interaction variable $Y_{u,i}$.

Unbiased Objective Function via IPS

To model the relevance variable $R_{u,i}$, we can have the ideal objective function [47] :

$$\mathcal{L}_{ideal} = \sum_{(u,i)\in\mathcal{D}} R_{u,i}(log(\widehat{R}_{u,i})) + (1 - R_{u,i})(log(1 - \widehat{R}_{u,i})), \tag{3.1}$$

where $\widehat{R}_{u,i}$ is the predicted relevance probability for user u to item i, which can be formulated as a matrix factorization model: $\widehat{R}_{u,i} = \sigma(\mathbf{P}_u^\top \cdot \mathbf{Q}_i)$ with $\mathbf{P}_u$ as the user latent factors, $\mathbf{Q}_i$ as the item latent factors, and $\sigma(\cdot)$ as the Sigmoid function. Note that here we adopt cross entropy, but other loss functions can be selected as well.

However, in practice, $R_{u,i}$ is unobservable. That is, we can only observe the interaction variable $Y_{u,i}$ that conflates both relevance and exposure. Conventional algorithms [17, 58, 27, 29] directly replace $R_{u,i}$ in $\mathcal{L}_{ideal}$ by $Y_{u,i}$, which is problematic because it will lead to the learned $\widehat{R}_{u,i}$ combining both relevance $R_{u,i}$ and exposure $O_{u,i}$ (because $Y_{u,i} = R_{u,i} \cdot O_{u,i}$) to generate biased recommendations. Hence, to address this problem, previous work [47] adopts Inverse Propensity Scoring (IPS), leading to the following unbiased objective function:

$$\mathcal{L}_{IPS} = \sum_{(u,i)\in\mathcal{D}} \frac{Y_{u,i}}{\theta_{u,i}}(log(\widehat{R}_{u,i})) + (1 - \frac{Y_{u,i}}{\theta_{u,i}})(log(1 - \widehat{R}_{u,i})), \tag{3.2}$$

where all variables $Y_{u,i}$, $R_{u,i}$, $O_{u,i}$ are assumed to be Bernoulli variables as $P(Y_{u,i} = 1) = \gamma_{u,i} \cdot \theta_{u,i}$, $\gamma_{u,i} = P(R_{u,i} = 1)$ and $\theta_{u,i} = P(O_{u,i} = 1)$. It is straightforward to show $\mathbb{E}[\mathcal{L}_{IPS}] = \mathbb{E}[\mathcal{L}_{ideal}]$ (for details, please refer to [47]). Thus, by minimizing $\mathcal{L}_{IPS}$, we can have unbiased recommendation.

3.2.2.2 Unbiased Propensity Estimation

We call the parameter $\theta_{u,i}$ (the probability of exposing item i to user u) the propensity in the IPS method. This propensity is estimated by a power-law function of item popularity (the number of interactions received by each item) in [47, 48, 46]:

$$\theta_{*,i} = (\sum_{u\in\mathcal{U}} Y_{u,i}/max_{i\in\mathcal{I}}(\sum_{u\in\mathcal{U}} Y_{u,i}))^{\eta}. \tag{3.3}$$

However, the power-law function of item popularity is itself not an unbiased estimation of the exposure probability: item popularity only considers the observed positive user-item interactions, but item exposure is determined by both observed positive interactions and unobserved negative feedback. That is, users may see an item but not interact with it. As a result, bias may still be introduced into seemingly unbiased methods such as in Equation 3.2.

Hence, we propose an unbiased propensity estimation method that aims to learn (i) the trade-off between the item popularity of positive and negative interactions; and (ii) the relative popularity for unobserved negative interactions. Such an approach has the added side benefit of learning propensity directly from data, sidestepping the challenge of tuning the exponent hyper-parameter η accurately for every new dataset.

Because all of $Y_{u,i}$, $R_{u,i}$, and $O_{u,i}$ are Bernoulli variables, the unbiased objective function for modeling $O_{u,i}$ is symmetric to the unbiased objective function for $R_{u,i}$ in Equation 3.2. Thus, by replacing $\theta_{u,i}$ with $\gamma_{u,i}$ and replacing $\widehat{R}_{u,i}$ with $\widehat{O}_{u,i}$ in Equation 3.2, we have the Inverse Relevance Scoring objective function:

$$\mathcal{L}_{IRS} = \sum_{(u,i)\in\mathcal{D}} \frac{Y_{u,i}}{\gamma_{u,i}}(log(\widehat{O}_{u,i})) + (1 - \frac{Y_{u,i}}{\gamma_{u,i}})(log(1 - \widehat{O}_{u,i})), \tag{3.4}$$

where $\widehat{O}_{u,i}$ is the predicted exposure probability, i.e., the estimation of the propensity $\theta_{u,i}$. Concretely, we model $\widehat{O}_{u,i}$ by an item-based model: $\widehat{O}_{u,i} = (w \cdot a + (1 - w) \cdot K_i)^e$, where $w = f_w(\mathbf{Q}_i)$, $a = f_a(\mathbf{Q}_i)$, $e = f_e(\mathbf{Q}_i)$; $f_w(\cdot)$, $f_a(\cdot)$, $f_e(\cdot)$ are three one layer perceptrons activated by a Sigmoid function with the item latent vector as input and a probability scalar as output; $K_i = \sum_{u\in\mathcal{U}} Y_{u,i}/max_{i\in\mathcal{I}}(\sum_{u\in\mathcal{U}} Y_{u,i})$ is the relative item popularity. We adopt the same power-law function as previous works [47, 48, 46] do in Equation 3.3, but set the exponent as a learnable parameter by $e = f_e(\mathbf{Q}_i)$. In essence, $a = f_a(\mathbf{Q}_i)$ aims to learn the relative popularity for unobserved negative interactions for items and $w = f_w(\mathbf{Q}_i)$ learns the trade-off between the popularity of positive and negative interactions. Moreover, we can view the propensity estimation in Equation 3.3 as a special case of our item-based propensity model when $w = 0$ and $e = \eta$.

3.2.2.3 Combinational Joint Learning Framework

Up to now, we have objective functions in Equation 3.2 for learning unbiased user-item relevance probability given we know the propensity (user-item exposure probability), and we also have the objective function in Equation 3.4 for learning unbiased propensity given we know the user-item relevance probability.

Algorithm 1: Training algorithm.

```
1 repeat
2     for $\mathcal{D}_c$ in $\{\mathcal{D}_1, \ldots, \mathcal{D}_C\}$ do
3         for $(u, i)$ in $\mathcal{D}_c$ do
4             Calculate $\gamma_{u,i}$ and $\theta_{u,i}$ by $\Psi_c$ and $\Phi_c$;
5             Update $\{\Psi_1, \ldots, \Psi_C\} \setminus \Psi_c$ by Equation 3.2, and update $\{\Phi_1, \ldots, \Phi_C\} \setminus \Phi_c$ by
               Equation 3.4;
6             with $\{\Psi_1, \ldots, \Psi_C\}$ and $\{\Phi_1, \ldots, \Phi_C\}$ fixed:
7                 Update $\{\overline{\Psi}_1, \ldots, \overline{\Psi}_C\}$ by Equation 3.2 with $\widehat{R}_{u,i}$ calculated by
                   $\{\Psi_1 + \overline{\Psi}_1, \ldots \Psi_C + \overline{\Psi}_C\}$;
8                 Update $\{\overline{\Phi}_1, \ldots, \overline{\Phi}_C\}$ by Equation 3.4 with $\widehat{O}_{u,i}$ calculated by
                   $\{\Phi_1 + \overline{\Phi}_1, \ldots, \Phi_C + \overline{\Phi}_C\}$;
9 until converge;
```

Naïve Joint Learning Method and Estimation-training Overlap Problem

Therefore, a straightforward idea is to combine them together in one model and jointly learn both of them. Concretely, assume we have a relevance model $\Psi = \{\mathbf{P}, \mathbf{Q}\}$ and a propensity model $\Phi = \{f_w, f_a, f_e\}$. For one observed user-item interaction $(u, i) \in \mathcal{D}$, we can first fix the propensity model Φ, and use the prediction $\widehat{O}_{u,i}$ of Φ as the propensity in Equation 3.2 to update the relevance model Ψ; then fix Ψ and update Φ based on Equation 3.4 with the prediction $\widehat{R}_{u,i}$ of Ψ as the relevance probability.

However, such a naïve method faces the *estimation-training overlap* problem. That is, the user-item pairs for training and propensity estimation (or relevance estimation) overlap with each other. More specifically, for a random user-item pair (u, i) in $\mathcal{D}$, we can use it to train the relevance model Ψ, and then use $\widehat{R}_{u,i}$ by Ψ as the relevance probability $\gamma_{u,i}$ in Equation 3.4 to update the propensity model Φ. This is problematic because Ψ has been trained by (u, i), so that $\gamma_{u,i} = \widehat{R}_{u,i}$ becomes the probability of u liking i given u has already provided positive feedback to i. Hence, $\gamma_{u,i}$ will be predicted as 1 by Ψ, which violates the definition of $\gamma_{u,i}$ as the probability parameter for the Bernoulli variable $R_{u,i}$. Similarly, using $\widehat{O}_{u,i}$ by Φ, which is trained by (u, i), as the propensity $\theta_{u,i}$ for updating Ψ brings the same problem.

Combinational Joint Learning

To address this problem, we propose a combinational joint learning framework, which separately learns unbiased relevance model Ψ and unbiased propensity model Φ by different data samples. The key idea is to split the training data into multiple chunks, and have multiple relevance sub-models and propensity sub-models so that the data chunks used for training any one of them and the chunks they predict relevance and propensity for do not overlap. Formally, we randomly divide the original training data $\mathcal{D}$ into C chunks with the same size: $\{\mathcal{D}_1, \ldots, \mathcal{D}_C\}$, C is a predefined combination hyper-parameter. Then, we define C relevance and propensity sub-models: $\{\Psi_1, \ldots, \Psi_C\}$ and $\{\Phi_1, \ldots, \Phi_C\}$. Each of the relevance sub-models and the propensity sub-models has the same structure as the conventional relevance and propensity model $\Psi = \{\mathbf{P}, \mathbf{Q}\}$ and $\Phi = \{f_w, f_a, f_e\}$. During training, for the c-th relevance and propensity sub-models Ψ_c and Φ_c, we will use all data chunks except $\mathcal{D}_c$ to update them, and output $\widehat{R}^c_{u,i}$ by Ψ_c and $\widehat{O}^c_{u,i}$ by Φ_c as the relevance probability $\gamma_{u,i}$ and propensity $\theta_{u,i}$ for user-item pairs in $\mathcal{D}_c$ for training other sub-models. And $\gamma_{u,i}$ and $\theta_{u,i}$ for chunks except $\mathcal{D}_c$ for training Ψ_c and Φ_c are provided by the other $C - 1$ relevance and propensity sub-models. For example, Φ_1 and Ψ_1 are trained by $\{\mathcal{D}_2, \ldots, \mathcal{D}_C\}$ with $\gamma_{u,i}$ and $\theta_{u,i}$ provided by $\{\Phi_2, \ldots, \Phi_C\}$ and $\{\Psi_2, \ldots, \Psi_C\}$ correspondingly, and Φ_1 and Ψ_1 output $\theta_{u,i}$ and $\gamma_{u,i}$ for $\mathcal{D}_1$ for the training process of other sub-models. In this way, data for training and propensity estimation (or relevance estimation) does not overlap for all of the sub-models.

Yet, there is another issue. Each of the sub-models is only trained by partial training data ($C - 1$ chunks), leading to information loss and compromised performance even if we average all sub-models as the final output. Hence, we further introduce a residual component to complement each sub-model. For example, for sub-models $\Psi_c = \{\mathbf{P}_c, \mathbf{Q}_c\}$ and $\Phi_c = \{f^c_w, f^c_a, f^c_e\}$ we have the residual components $\overline{\Psi}_c = \{\overline{\mathbf{P}}_c, \overline{\mathbf{Q}}_c\}$ and $\overline{\Phi}_c = \{\overline{f^c_w}, \overline{f^c_a}, \overline{f^c_e}\}$, and add the residual components to the original sub-models as the final models for output: $\Psi'_c = \{\mathbf{P}_c + \overline{\mathbf{P}}_c, \mathbf{Q}_c + \overline{\mathbf{Q}}_c\}$ and $\Phi'_c = \{f^c_w + \overline{f^c_w}, f^c_a + \overline{f^c_a}, f^c_e + \overline{f^c_e}\}$. The residual component is trained by the complete $\mathcal{D}$ with sub-models fixed, and the relevance and propensity are provided by all the sub-models. The training algorithm is shown in Algorithm 1. Last, after training, by averaging the output of $\{\Psi'_1 \ldots \Psi'_C\}$

and $\{\Phi'_1 \dots \Phi'_C\}$, we have the final unbiased relevance and propensity predictions.

3.2.3 Experiments

We conduct experiments on two real-world datasets to answer three research questions: **RQ1** How does the proposed method perform compared with state-of-the-art alternatives? **RQ2** How effective is the estimated propensity? and **RQ3** What is the impact of the combination hyper-parameter C and of the residual components in the proposed model?

3.2.3.1 Experimental Settings

Datasets

To evaluate unbiased recommendation, we need datasets with items uniformly exposed to users so that we can directly evaluate user-item relevance without influence of exposure. Thus, we use the Yahoo and Coat datasets, which are the only two publicly available datasets containing separate test sets where users provide feedback to uniformly drawn samples of items. **Yahoo! R3** (`https://webscope.sandbox.yahoo.com/`) contains over 300K ratings (1 to 5) from 15.4K users to 1K songs in the training set (a biased training set). Besides, an unbiased test set is collected by sampling a subset of 5.4K users, each of whom is randomly assigned 10 songs, and asked to provide ratings to these random items. Following the preprocessing procedure in [46], we regard ratings ≥ 4 as positive feedback, and we randomly split 10% of the training set to be a biased validation set. **Coat Shopping** [35] contains around 7K ratings (1 to 5) from 290 users to 300 coats in the training set (a biased training set). Similar to the Yahoo dataset, the Coat dataset also has an unbiased random test set by asking all 290 users to rate 16 randomly selected coats.

Metrics

Following [47, 48], we adopt two widely used implicit recommendation evaluation metrics – *DCG@k* (Discounted Cumulative Gain) and *MAP@k* (Mean Average Precision). The detailed formulations can be found in [47, 48]. We report results with $k = 1, 2, 3$ since the number of candidate items for ranking is small in the test set. We rank the 10 exposed items for each user in the Yahoo dataset, and 16 items in the Coat dataset.

Combinational Approaches

We evaluate combinational joint learning for both a point-wise version (**CJMF**) and a pair-wise variation (**CJBPR**). CJBPR uses the unbiased Bayesian Personalized Ranking (BPR) loss proposed by [48] for the relevance model, and uses the same propensity loss in Equation 3.2 for the propensity estimation. Since the output of the relevance model $\widehat{R}_{u,i}$ in CJBPR is not a probability, for a user u, we further use a softmax function on $\widehat{R}_{u,*}$ to have a multinomial distribution for predicted scores and then divide the scores by the maximum score in the multinomial distribution to transform scores to relevance probabilities $\gamma_{u,*}$ in Equation 3.4.

Baselines

For fair comparison, we compare CJMF with point-wise baselines, and compare CJBPR with pair-wise baselines. For point-wise baselines, we use the following biased models: **MF-RMSE** [29], the most commonly used matrix factorization model for implicit recommender systems with RMSE loss; **MF-CE**, a variation of MF-RMSE adopting cross entropy loss (to have a fair comparison with CJMF that also uses cross entropy loss). We also consider the following unbiased baselines: **RelMF-RMSE** [47], an unbiased model adopting the IPS approach and the RMSE loss; **RelMF-CE** is a variation of RelMF-RMSE which adopts cross entropy loss to have a fair comparison with the proposed CJMF. For pairwise baselines, we adopt the biased **BPR** [27] model and **UBPR** [48], an unbiased version of BPR which also uses the IPS approach.

All point-wise and pair-wise unbiased baselines [47, 48] use the power-law function of item popularity introduced in Equation 3.3 with $\eta = 0.5$ (the same as in the original papers [47, 48]) as propensity. We also adopt the propensity clipping approach [47, 48] to reduce the variance for all unbiased baselines and also our proposed models.

Reproducibility

We implement the proposed models based on Tensorflow [59] and use Adam [60] optimizer. We set the learning rate as 0.001, the batch size as 1024, the latent dimension as 100, and the negative sampling rate as 5 for all models and datasets. For other hyper-parameters, we tune them

Table 3.1: Performance comparison, where best baselines are marked by underlines. Reprinted with permission from [1].

			Point-wise models							Pair-wise models			
			MF -RMSE	MF -CE	RelMF -RMSE	RelMF -CE	NJMF	CJMF	Δ	BPR	UBPR	CJBPR	Δ
Yahoo	DCG	@1	.531	.528	.536	.534	.540	**.561**	4.58%	.541	.543	**.565**	3.96%
		@2	.730	.739	.735	.740	.743	**.775**	4.71%	.745	.749	**.775**	3.42%
		@3	.852	.858	.860	.862	.868	**.896**	4.00%	.867	.878	**.897**	2.22%
	MAP	@1	.531	.528	.536	.534	.540	**.561**	4.58%	.542	.543	**.565**	3.96%
		@2	.619	.618	.620	.622	.626	**.648**	4.09%	.626	.630	**.650**	3.19%
		@3	.642	.642	.643	.647	.649	**.669**	3.54%	.649	.653	**.672**	2.88%
Coat	DCG	@1	.531	.549	.5490	.561	.570	**.591**	5.26%	.532	.574	**.591**	2.94%
		@2	.761	.770	.788	.785	.795	**.822**	4.34%	.774	.787	**.822**	4.51%
		@3	.919	.930	.934	.937	.943	**.968**	3.33%	.930	.939	**.960**	2.21%
	MAP	@1	.531	.549	.549	.561	.570	**.591**	5.26%	.532	.574	**.591**	2.94%
		@2	.612	.620	.637	.642	.648	**.671**	4.26%	.618	.639	**.671**	4.95%
		@3	.626	.640	.650	.649	.657	**.674**	3.73%	.638	.660	**.682**	3.36%

by grid search on the biased validation sets with the self-normalized inverse propensity scoring (SNIPS) estimator [46] as the performance indicator. More specifically, we set $C = 8$ for both Yahoo and Coat datasets for CJMF and set $C = 6$ for CJBPR. Since the proposed CJMF and CJBPR adopt the combinational joint learning method leading to more model parameters than baselines, to have a fair comparison, we run point-wise baselines for 16 times (run pair-wise baselines for 12 times) and average their outputs as final predictions. All code and data are at `https://github.com/Zziwei/Unbiased-Propensity-and-Recommendation`.

3.2.3.2 RQ1: Comparing Recommendation Performance

We begin by investigating the recommendation performance of the proposed CJMF and CJBPR approaches compared with corresponding point-wise and pair-wise baselines. Detailed results for all models are shown in Table 3.1. First, comparing among point-wise models and among pair-wise models, we observe that the proposed CJMF and CJBPR can significantly outperform corresponding baselines (the best baselines are marked by underlines in Table3.1), which indicates that the proposed combinational joint learning approach is effective for unbiased implicit recommendation. Second, we also implement the naïve joint learning model (denoted as NJMF in Table 3.1)

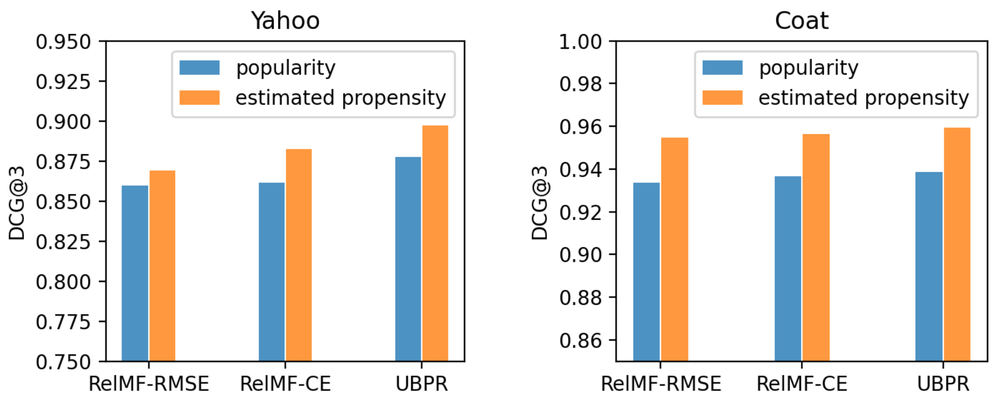


Figure 3.1: Comparing unbiased models with item popularity as propensity and with estimated propensity from proposed models. Reprinted with permission from [1].

which has the estimation-training overlap issue. From the table we observe that CJMF produces better recommendation results than NJMF, demonstrating the effectiveness of the proposed combinational joint strategy over the naïve approach. Last, by comparing CJMF and CJBPR, we find that results of CJBPR are better than CJMF but the difference is small, which is not as obvious as the difference between conventional MF and BPR. Since the propensity and the relevance are modeled differently in CJBPR (the propensity is modeled as probabilities while relevance is modeled as real-number scores), we see some limits to the effectiveness of the joint learning. We leave further study along this line to future work.

3.2.3.3 RQ2: Investigating the Effectiveness of Estimated Propensity

Next, we study the effectiveness of the estimated propensity by comparing the performance of three unbiased baselines using Equation 3.3 as propensity estimation versus using $\widehat{O}_{u,i}$ predicted by CJMF and CJBPR as propensity estimation. Results presented in Figure 3.1 show that for both datasets, the two variations of RelMF can perform better if use the estimated propensity from CJMF. The same conclusion can be drawn that with the estimated propensity from CJBPR, UBPR can perform better.

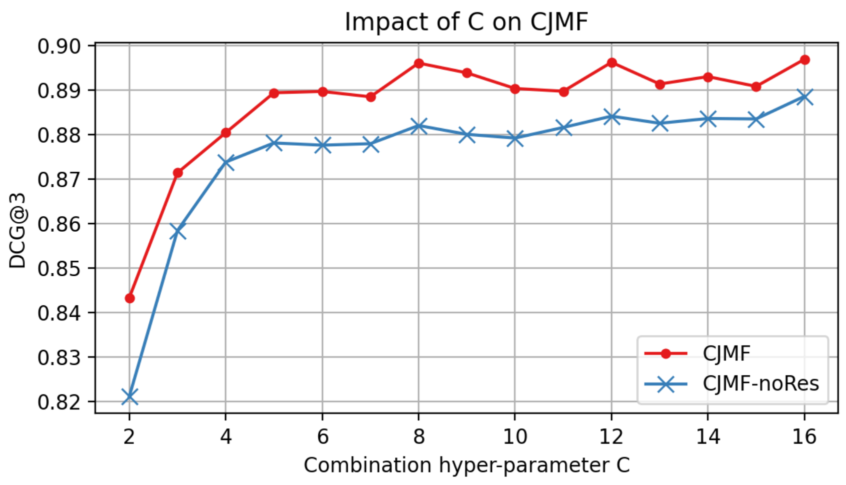


Figure 3.2: *DCG@3* of CJMF and CJMF without residual components on the Yahoo dataset, with varying C. Reprinted with permission from [1].

3.2.3.4 RQ3: Investigating the Impact of Hyper-parameter C and Residual Component

Finally, we investigate the impact of the combination hyper-parameter C on CJMF. We vary C from 2 to 16 and show the results of CJMF as the red line in Figure 3.2. We see that the performance of CJMF improves rapidly then converges as C increases, reaching a peak level when $C \geq 5$.

Then, we study the effect of the residual components. We denote the variation of CJMF without the residual component as CJMF-noRes, which directly averages the output of all sub-models as the final prediction of the complete model. The $DCG@3$ results of CJMF-noRes are plotted in Figure 3.2 as the blue lines. We observe the effectiveness of the residual components by comparing with the complete model (the red lines in the figure). Note that CJBPR has similar patterns as CJMF demonstrated in Figure 3.2.

3.2.4 Summary

In this section, we propose a combinational joint learning framework, which jointly learns unbiased relevance and propensity simultaneously, to produce unbiased recommendations based on biased implicit data. Extensive experiments on two public datasets show the effectiveness of the proposed method.

3.3 Tackling Cold-start Recommendation via Randomized Training and Mixture-of-Experts Transformation

Although the proposed combinational joint learning framework can effectively counteract the exposure bias and learn unbiased user-item relevance for recommendation, a strong assumption of it and other alternatives is that all items have non-zero exposure probabilities in the user-item interactive data. Yet, this assumption cannot hold in real world. For example, 500 hours of new videos are uploaded to YouTube every minute,[2] 500,000 new users register in Facebook every day.[3] These new items have no exposure in the logged user-item interactive data, in other word, it is the extreme exposure bias scenario. How to provide effective recommendations for them is a long-standing challenge, which is also known as the cold-start recommendation problem and these new items are called cold-start items. To solve this problem, we develop a novel cold-start recommendation algorithm – Heater, which can not only provide high-quality recommendation for cold-start items, but can also work for cold-start users at the same time.

3.3.1 Introduction

To provide recommendations for these cold-start users and items, many content-based methods [61, 62, 63] and heuristic methods have been deployed, e.g., recommending popular items or geographically near items. However, recent research efforts [2, 3, 4, 5, 6] that tackle the cold-start problem from the perspective of machine learning have made promising strides. As illustrated in Figure 3.3a, these ML-based efforts combine user-item interactions from existing warm start users and items (as in CF-based methods) with auxiliary information from both warm and cold users and items (as in content-based methods). This auxiliary information – be it from user profiles, item descriptions, reviews, and other sources – is often readily available even in the absence of user-item interactions.

Conventional CF models provide recommendations for warm users and items by finding sim-

[2]https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/
[3]https://www.brandwatch.com/blog/facebook-statistics/

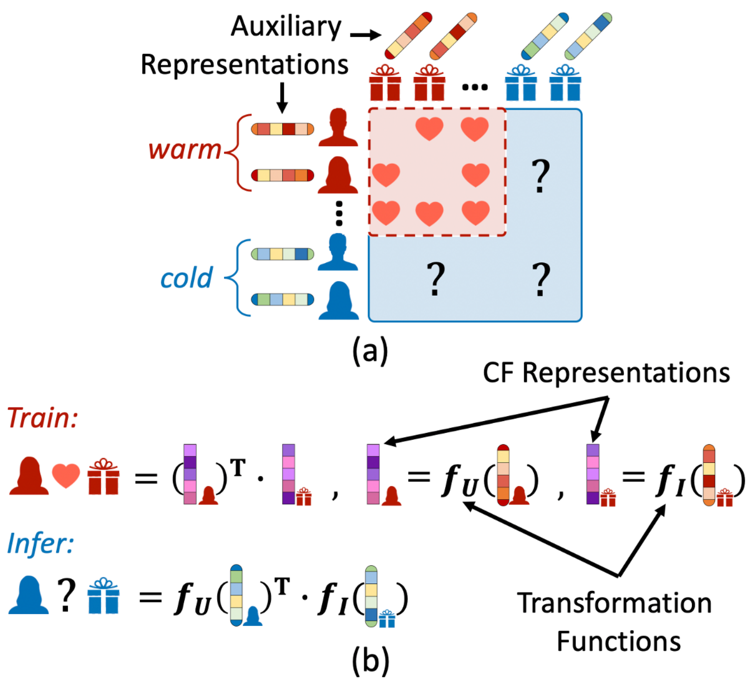


Figure 3.3: (a) setup of cold-start recommendation problem, where both warm and cold users and items have auxiliary representations (such as user profiles and item content); and (b) the main idea of existing cold-start recommendation algorithms [2, 3, 4, 5, 6]: learn transformation functions to transform auxiliary representations to CF representations. Reprinted with permission from [7].

ilarities between the CF representations of users and items, which are learned based on existing user-item interactions. In contrast, these ML-based cold-start recommendation approaches aim to learn CF representations for cold-start users and items lacking historical interactions. The key insight is to learn two transformation functions – one for users and one for items – to transform the auxiliary representations of these new users and items into the CF space. As illustrated in Figure 3.3b, the two transformation functions (f_U and f_I) are learned from interactions and auxiliary representations of warm users and items; the learned transformation functions are then applied on auxiliary representations of cold-start users and items to predict preference scores at inference time. Hence, the fundamental challenge is *how to generate effective transformation functions based on the given auxiliary information and user-item interactions*.

In general, there exist two major categories of algorithms to learn these transformation functions – separate-training methods and joint-training methods. Here, we identify three major issues with these existing methods that can impact the quality of the learned transformation functions. These issues motivate our proposed **Heater** framework.

To begin with, separate-training methods [6, 49, 50, 51, 52] separately learn a CF model (by minimizing the Collaborative Filtering error $\mathcal{L}_{CF}$ on the user-item interactions as in conventional CF models for warm start recommendation) and transformation functions to transform auxiliary representations (by minimizing the difference $\mathcal{L}_{trans}$ between transformed auxiliary representations and CF representations from a CF model), either in an end-to-end way or non-end-to-end two-step way [50, 51, 52, 6, 49]. Separate-training methods can fully utilize the user-item interaction data because they apply sophisticated CF models directly on the interaction data. However, they leave one challenge unsolved – the *error superimposition problem*. Due to the separation of learning CF and learning transformation functions in the model objective function, the final cold-start recommendation error during inference is the summation of the CF error $\mathcal{L}_{CF}$ and the transformation error $\mathcal{L}_{trans}$. Hence, an increase in either of the two errors will decrease the overall cold-start recommendation performance.

On the other hand, joint-training methods [3, 4, 64, 5] fuse CF and transformation functions together (i.e., models input auxiliary representations and output recommendations with CF representations as hidden layers), and train models with the only aim of minimizing the recommendation error $\mathcal{L}_{CF}$ on the warm interaction data. Since joint-training methods directly minimize $\mathcal{L}_{CF}$, and learning of transformation functions is also guided by $\mathcal{L}_{CF}$, there is no error superimposition issue. Nevertheless, there is another challenge for joint-training methods – the *ineffective learning problem* – that is, because learning transformation functions is only guided by the recommendation error which is based on the final model output layer, the long distance from transformation layers to the model output layer from the perspective of backpropagation leads to ineffective model learning.

Moreover, a common issue with both separate-training and joint-training methods is the *unified transformation problem*. Concretely, almost all existing separate-training and joint-training methods adopt unified transformation functions (either a linear transformation or a neural network based non-linear transformation) for all users or items under the assumption that users (or items) keep the same relative relationships in both the auxiliary representation space and CF space. This assumption seldom holds because auxiliary information is usually noisy and complex, and cannot be directly aligned with the CF space. Thus, a unified transformation process is not effective and can generate low-quality CF representations.

To address these three challenges, we propose a new model **Heater**, which is designed to keep the advantages of both separate-training and joint-training methods, while overcoming the drawbacks identified above. To deal with the problem of error superimposition and the problem of ineffective learning, we combine the structures of separate-training and joint-training methods together as the basic framework of Heater. The main procedure of training is that Heater first transforms auxiliary representations to intermediate representations, then further refines intermediate representations to final CF representations to minimize recommendation error. Meanwhile, we also require the intermediate representations to be as close as possible to high-quality pretrained CF representations to improve the model effectiveness. To further address the ineffective learning problem, we propose a **Randomized Training** mechanism in which we randomly feed pretrained CF representations or intermediate representations alternatively to the refining component of Heater. By doing this, the effectiveness of model training is protected by the high-quality pretrained CF representations even when the intermediate representations are of poor quality. Last, we propose a **Mixture-of-Experts Transformation**, which adopts the Mixture-of-Expert [65] structure as transformation functions so that Heater can provide 'personalized' transformations for different users and items to tackle the unified transformation issue. Furthermore, unlike most existing methods [64, 50, 3, 2, 4], the proposed Heater can simultaneously recommend for both cold-start users and items, rather than requiring separate user-based and item-based models. Last, we conduct extensive experiments on three real-world datasets to show the effectiveness of Heater over state-of-the-art alternatives, and the effectiveness of each proposed component.

3.3.2 Proposed Cold-start Recommendation Model – Heater

In this section, we formalize the cold-start recommendation problem, then introduce the fundamental framework of Heater, the Randomized Training and Mixture-of-Experts Transformation mechanisms.

3.3.2.1 Cold-start Recommendation Formalization

Assume we have N_w warm users $\mathcal{U}_w = \{1, 2, \ldots, N_w\}$ and M_w warm items $\mathcal{I}_w = \{1, 2, \ldots, M_w\}$, each of which has at least one historical interaction record. We denote the set of all historical records as $\mathcal{O} = \{(u, i)\}$, where u indexes one user, and i indexes one item. We also have N_c cold-start users $\mathcal{U}_c = \{1, 2, \ldots, N_c\}$ and M_c cold-start items $\mathcal{I}_c = \{1, 2, \ldots, M_c\}$, all of which have zero historical interaction record. For these cold-start users and items, there are three tasks:

- **Task 1**: recommend warm items from $\mathcal{I}_w$ to cold users in $\mathcal{U}_c$;
- **Task 2**: recommend cold items from $\mathcal{I}_c$ to warm users in $\mathcal{U}_w$;
- **Task 3**: recommend cold items from $\mathcal{I}_c$ to cold users in $\mathcal{U}_c$.

Note that most previous works only consider the one-sided cold-start situation [64, 50, 3, 2, 4], i.e., there are only cold-start users or cold-start items in the system. Here, we consider the more complex situation where there are cold-start users and items simultaneously in the system. Furthermore, we assume we have access to *auxiliary information* such as an external user profile and item content information for both warm and cold users and items, denoted as $\mathbf{U} \in \mathbb{R}^{(N_w+N_c)\times E_u}$ and $\mathbf{I} \in \mathbb{R}^{(M_w+M_c)\times E_i}$, where E_u and E_i are the auxiliary representation dimensions for users and items.

3.3.2.2 Heater Framework

As we have discussed, there are two main categories of cold-start recommendation approaches – separate-training and joint-training methods, which are determined by the relationship between CF and the transformation functions in the model. Both of these have obvious advantages and

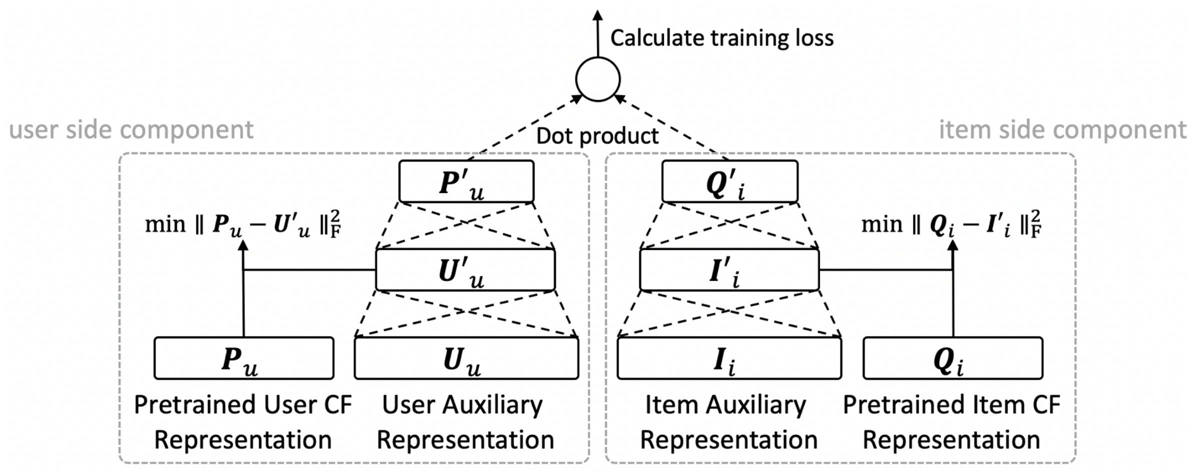

Calculate training loss
Dot product
user side component
item side component
P'_u
Q'_i
$\min \| P_u - U'_u \|_F^2$
U'_u
I'_i
$\min \| Q_i - I'_i \|_F^2$
P_u
U_u
I_i
Q_i
Pretrained User CF Representation
User Auxiliary Representation
Item Auxiliary Representation
Pretrained Item CF Representation

Figure 3.4: The framework of Heater, which incorporate structures of the separate-training and joint-training methods to solve the error superimposition and ineffective learning problems. Reprinted with permission from [7].

disadvantages: separate-training methods make full use of the user-item interaction data while suffering from the error superimposition problem; joint-training methods are free of the error superimposition issue but face the ineffective learning problem. Thus, to overcome these problems, we propose a framework to integrate the two distinct structures.

Due to the special characteristic that users or items involved during inference time are never seen before during training, the execution of Heater is different for training and inference. In the following, we first focus on the training process, then describe how to do inference by a trained model for cold-start situations.

Training

Separate-training approaches learn CF representations and auxiliary-to-CF transformation separately by two independent losses $\mathcal{L}_{CF}$ and $\mathcal{L}_{trans}$, thus, increase of each of them leads to worse cold-start recommendations (the error superimposition issue). To avoid this error superimposition issue, we adopt the joint-training structure as the foundation for Heater basic framework as shown in Figure 3.4. Generally, Heater takes one warm user auxiliary representation $\mathbf{U}_u \in \mathbb{R}^{1\times E_u}$ and one warm item auxiliary representation $\mathbf{I}_i \in \mathbb{R}^{1\times E_i}$ as inputs, and by multiple layers of transformation

$\mathbf{U}_u$ and $\mathbf{I}_i$ are transformed to the CF representations $\mathbf{P}'_u$ and $\mathbf{Q}'_i$, of which the dot product is the predicted preference score $\hat{R}_{u,i} = (\mathbf{P}'_u)^T\mathbf{Q}'_i$ for the given user-item pair. More specifically, we first transform $\mathbf{U}_u$ and $\mathbf{I}_i$ to intermediate representations $\mathbf{U}'_u$ and $\mathbf{I}'_i$ by two transformation functions f_U and f_I as $\mathbf{U}'_u = f_U(\mathbf{U}_u)$ and $\mathbf{I}'_i = f_I(\mathbf{I}_i)$. Then, we further refine the representations by a multi-layer perceptron (MLP) (ϕ_U for user side, ϕ_I for item side) to get user and item CF representations: $\mathbf{P}'_u = \phi_U(\mathbf{U}'_u)$, $\mathbf{Q}'_i = \phi_I(\mathbf{I}'_i)$.

However, because the transformation layers f_U and f_I are trained based on the recommendation error $\mathcal{L}_{CF}$ which is calculated by the model output layer $\hat{R}$, the long distance from f_U and f_I to the output layer from the view of backpropagation makes it difficult to learn effective parameters for f_U and f_I, which can further impact the learning effectiveness of the whole model (the ineffective learning issue). Hence, to address this, we input high-quality user and item pretrained CF representations (denoted as $\mathbf{P}_u \in \mathbb{R}^{1\times D}$ and $\mathbf{Q}_i \in \mathbb{R}^{1\times D}$) from a pretrained CF model to help guide the learning processing for f_U and f_I by setting a *similarity constraint* to minimize the difference between the intermediate representations $\mathbf{U}'_u$ and $\mathbf{I}'_i$ and pretrained CF representations $\mathbf{P}_u$ and $\mathbf{Q}_i$:

$$\min_{\mathbf{P}_u,\mathbf{Q}_i} \|\mathbf{U}'_u - \mathbf{P}_u\|_{\mathrm{F}}^2 + \|\mathbf{I}'_i - \mathbf{Q}_i\|_{\mathrm{F}}^2. \tag{3.5}$$

With the similarity constraint, f_U and f_I are guided by the final recommendation error as well as the high-quality pretrained CF representations, leading to more effective f_U and f_I.

Heater can be trained by any popular top-k recommendation loss function, such as Sum Squared Error (SSE) loss, cross-entropy loss, or Bayesian Personalized Ranking (BPR) loss [27]. Here, we use SSE loss because most of existing baselines [64, 50, 3, 4, 5] adopt this loss, and it shows good empirical performance. The objective function of Heater can be written as:

$$\begin{aligned}\min_{\mathbf{\Theta}} \mathcal{L} = \sum_{(u,i)\in\mathcal{O}\cup\mathcal{O}^-} & \|\hat{R}_{u,i} - R_{u,i}\|_{\mathrm{F}}^2 \\ & + \frac{\alpha}{2}(\|\mathbf{U}'_u - \mathbf{P}_u\|_{\mathrm{F}}^2 + \|\mathbf{I}'_i - \mathbf{Q}_i\|_{\mathrm{F}}^2) + \frac{\lambda}{2}\|\mathbf{\Theta}\|_{\mathrm{F}}^2,\end{aligned} \tag{3.6}$$

where $\mathcal{O}^-$ is the negative samples randomly generated based on $\mathcal{O}$; $R_{u,i}$ is the ground-truth preference with value 1 if $(u, i) \in \mathcal{O}$, 0 otherwise; α is the trade-off weight for similarity constraint; and λ is the trade-off weight for regularization.

Inference

With a trained Heater, it is straightforward to provide recommendations, which is similar to the training process. The only difference is that we do not need the similarity constraint and pretrained CF representations shown in Equation 3.5, and we only input the auxiliary representations of cold users and items into the model. For Task 1 mentioned in Section 3.3.2.1 – recommending warm items to cold users – assume we want to provide recommendation for the cold user u. All we need is to input all the $(\mathbf{U}_u, \mathbf{I}_i)$ pairs into Heater to calculate $\hat{R}_{u,i}$, where $i \in \mathcal{I}_w$, and show items with top scores in descending order to user u. In the same way, we can generate recommendations for Task 2 and Task 3.

The Heater framework can also be applied to cases where there is only auxiliary information for users or items, and the side (user side or item side) without auxiliary information is warm side (all users in the side or all items in the side are warm). During training, the only modifications needed are first removing the corresponding similarity constraint for the warm side, and then directly input the pretrained CF representations $\mathbf{P}_u$ (or $\mathbf{Q}_i$) as $\mathbf{U}'_u$ (or $\mathbf{I}'_i$) to calculate the final CF representations $\mathbf{P}'_u$ (or $\mathbf{Q}'_i$) for the warm side. During inference, similarly, we directly use the pretrained CF representations as input for the warm side.

3.3.2.3 Randomized Training

The introduced similarity constraint in the Heater framework (shown in Equation 3.5) is capable to guide the learning of transformation functions f_U and f_I, and thus improves the quality of $\mathbf{U}'$ and $\mathbf{I}'$ because the pretrained CF representations are known to be of high quality. As a result, the Heater framework should have higher model learning effectiveness than conventional joint-training methods.

However, even if we have the similarity constraint, there is always information loss between $\mathbf{P}$ (or $\mathbf{Q}$) and $\mathbf{U}'$ (or $\mathbf{I}'$) due to the low quality of auxiliary representations and the structural

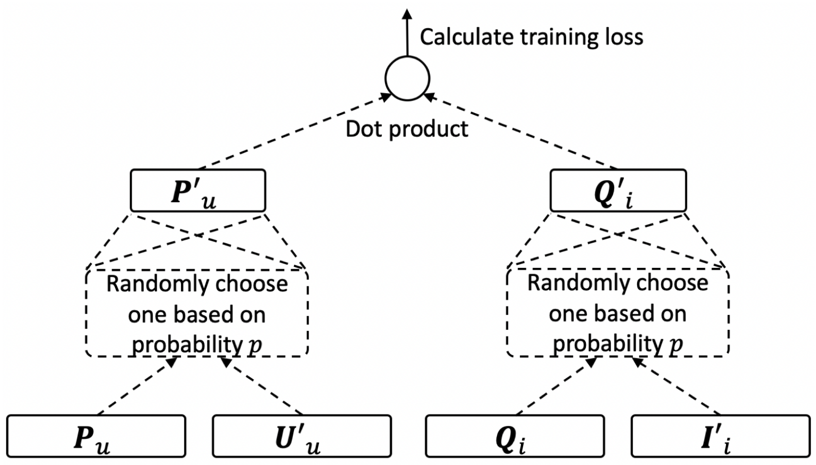

Calculate training loss
Dot product
P'_u
Q'_i
Randomly choose one based on probability p
Randomly choose one based on probability p
P_u
U'_u
Q_i
I'_i

Figure 3.5: Randomized Training: during training, randomly feed pretrained CF representations or transformed auxiliary representations alternatively to generate final representations $\mathbf{P}'_u$ and $\mathbf{Q}'_i$. Reprinted with permission from [7].

limitation of the transformation functions. This will lead to ineffective learning for ϕ_U and ϕ_I due to the low quality of $\mathbf{U}'$ and $\mathbf{I}'$, and further decrease the quality of final CF representations $\mathbf{P}'$ and $\mathbf{Q}'$. To address this, we propose a Randomized Training strategy, which does not only feed user and item transformed auxiliary representations $\mathbf{U}'$ and $\mathbf{I}'$ to generate final representations $\mathbf{P}'$ and $\mathbf{Q}'$ during training, but also uses the pretrained CF representations $\mathbf{P}$ and $\mathbf{Q}$ in a stochastic way as demonstrated in Figure 3.5. Note that the proposed Randomized Training is only for the training and does not influence the inference process.

Concretely, first, we need to pre-define a hyper-parameter $p \in [0, 1]$ representing the probability of using pretrained CF representations $\mathbf{P}$ and $\mathbf{Q}$ for training. Then, during the training process, for a given training sample (u, i), based on p, we randomly choose whether to use $\mathbf{U}'_u$ or $\mathbf{P}_u$ to generate $\mathbf{P}'_u$, and whether to use $\mathbf{I}'_i$ or $\mathbf{Q}_i$ to generate $\mathbf{Q}'_i$. Note that the random processes of user and item sides are independent, and they can even have different values of probability p.

By using both auxiliary representations and pretrained CF representations alternatively, we can alleviate the ineffective learning problem because the high-quality pretrained CF representations can help guide ϕ_U and ϕ_I to learn effective parameters, especially when $\mathbf{U}'$ and $\mathbf{I}'$ are not of high quality. The choice of p depends on the quality of the auxiliary information. If the auxiliary

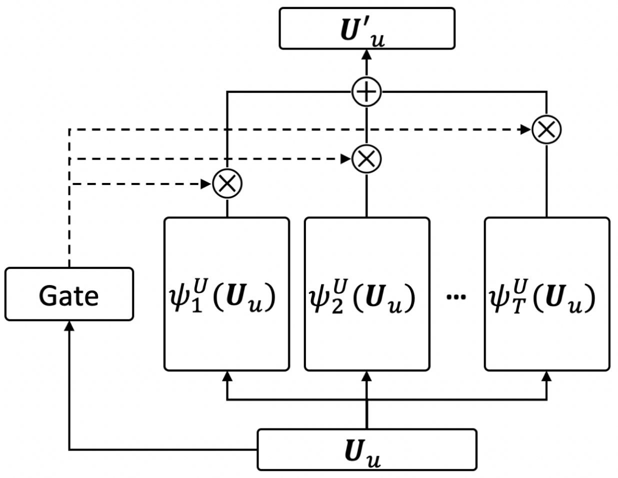


Figure 3.6: Mixture-of-Experts Transformation: apply T multi-layer perceptrons as experts to transform auxiliary representations, weighted sum outputs of experts as final output. Reprinted with permission from [7].

representations contain rich information about the users and items with limited noise, then we can use a small p, otherwise, we need a large p to ensure the effectiveness of the model.

3.3.2.4 Mixture-of-Experts Transformation

Last, we turn to address the unified transformation problem of existing methods. Recall that previous works apply the same transformation process f_U (or f_I) to map auxiliary representations into the CF space for all the users (or all items). In other words, they assume the relationships between users (or items) in the auxiliary representation space are the same as the relationships in the CF space. But this assumption can seldom hold because the auxiliary information is usually noisy and complex. For example, two users may have a large distance in the auxiliary representation space because they have little common information in their profiles, but they could have similar preference, i.e., small distance in CF space. A unified transformation function cannot handle this situation. Thus, an algorithm which is able to assign 'personalized' transformations to different users (or items) is required. As a result, we propose to adopt Mixture-of-Experts [65] to implement f_U and f_I for every single user and item.

The Mixture-of-Experts Transformation (as shown in Figure 3.6) consists of T experts, where each is a MLP, denoted as $\psi_1^U, \ldots, \psi_T^U$ for user side, $\psi_1^I, \ldots, \psi_T^I$ for item side. All of the experts take the same input $\mathbf{U}_u$ (or $\mathbf{I}_i$), and the final output of the Mixture-of-Experts Transformation is a weighted sum of the outputs of all experts. The formulation of the user side is:

$$MoE(\mathbf{U}_u) = g_1\psi_1^U(\mathbf{U}_u) + g_2\psi_2^U(\mathbf{U}_u) + \ldots + g_T\psi_T^U(\mathbf{U}_u), \tag{3.7}$$

where $g_1, g_2, \ldots, g_T$ are the weights for experts, which are calculated by another one-layer perceptron: $\mathbf{g} = \varphi(\mathbf{W}^T\mathbf{U}_u + \mathbf{b})$, where $\mathbf{g} \in \mathbb{R}^T$ is the vector consisted of $g_1, g_2, \ldots, g_T$; and $\varphi(\cdot)$ is the activation function, we use *tanh* in this work. The reason why we do not use *softmax* for the weights is that experts here work in a collaborative way rather than an exclusive way, thus every expert should have an independent weight. And *tanh* empirically outperforms *softmax*. Note that the formulation for the item side is similar.

To better explain the effect of Mixture-of-Experts Transformation, we assume all experts are linear transformation matrices, denoted as $\mathbf{V}_1^U, \mathbf{V}_2^U, \ldots, \mathbf{V}_T^U$ for user side, $\mathbf{V}_1^I, \mathbf{V}_2^I, \ldots, \mathbf{V}_T^I$ for item side. Then, the output of Mixture-of-Experts Transformation for user u is:

$$MoE(\mathbf{U}_u) = (g_1\mathbf{V}_1^U + g_2\mathbf{V}_2^U + \ldots + g_T\mathbf{V}_T^U)^T\mathbf{U}_u, \tag{3.8}$$

where $(g_1\mathbf{V}_1^U + g_2\mathbf{V}_2^U + \ldots + g_T\mathbf{V}_T^U)$ generates a new transformation matrix specifically for user u based on her own auxiliary representation, which achieves our goal that assigning different transformations to different users (or items). By doing this, the auxiliary representations $\mathbf{U}$ and $\mathbf{I}$ can be transformed into the CF space more effectively than using a unified transformation function. It is also similar to the idea of applying meta-learning to solve cold-start recommendation [2], which generates a unique logistic regression model for each user based on her historical interactions and then applies the logistic regression model on cold-start items. Therefore, the Mixture-of-Experts Transformation can also be viewed as a meta-learning based method.

3.3.3 Experiments

In this section, we empirically evaluate the proposed model over three cold-start recommendation tasks and three datasets from different domains. We aim to answer four key research questions: **RQ1** How does Heater perform compared with state-of-the-art cold-start recommendation models? **RQ2** How effective are the proposed similarity constraint, Randomized Training, and Mixture-of-Experts Transformation mechanisms? **RQ3** What are the impact of three key hyper-parameters: similarity constraint weight α, Randomized Training probability p, and number of experts T in Mixture-of-Experts Transformation? **RQ4** What is the impact of the quality of pretrained CF representations on Heater compared with other models taking pretrained representations as input?

3.3.3.1 Experimental Settings

Datasets

Datasets for evaluating cold-start recommendation need rich auxiliary information for users and items, thus we use three real-world datasets commonly used to evaluate the three different cold-start recommendation tasks introduced in Section 3.3.2.1:

- **CiteULike** [66] is a dataset recording user preferences towards scientific articles. There are 5,551 users, 16,980 articles and 204,986 user-like-article interactions in the dataset. Besides, we have the abstracts of the articles as the auxiliary information for the item side, but there is no auxiliary information for the user side, hence we *evaluate Task 2 on CiteULike*. Following the processing of [5], we first generate an 8,000 dimension feature vector by calculating tf-idf of top 8,000 words for each item, and then keep the top 300 dimensions after dimensionality reduction by SVD. As a result, we have a $16,980 \times 300$ item auxiliary representation $\mathbf{I}$.

- **LastFM** [67] consists of 1,892 users, 17,632 music artists as items to be recommended, and *92,834 user-listen-to-artist interactions* (rather than the user-tag-artist interactions as used in previous work [3], because user-listen-to-artist interaction is more general and the data is sparser). In this dataset, we have the social relationships between all the users, thus we have

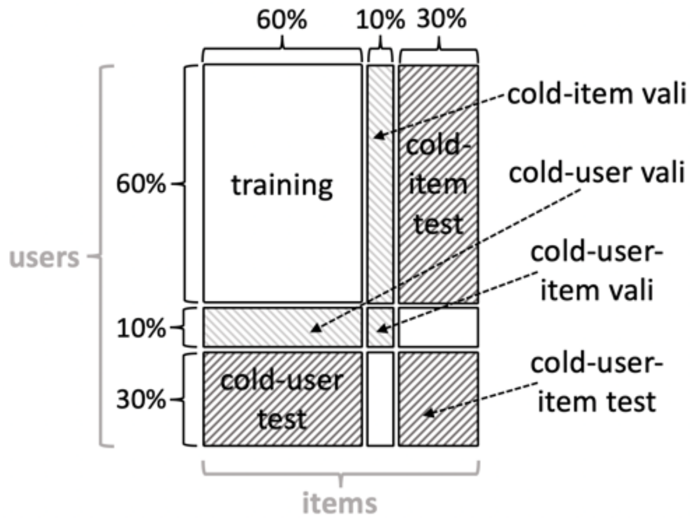


Figure 3.7: Validation and test set splitting for XING dataset. Reprinted with permission from [7].

a $1,892 \times 1,892$ user auxiliary representation $\mathbf{U}$, but have no auxiliary information for the item side. Therefore, we *evaluate Task 1 on LastFM*.

- **XING** [68] is a subset of the ACM RecSys 2017 Challenge dataset, which contains 106,881 users, 20,519 jobs as items to be recommended to users, and 4,306,183 user-view-job interactions. We have user profile information such as current job, location and education level. For items, we have career level, tags, and other related information. Following the processing of [5], we have a $106,881 \times 831$ user auxiliary representation $\mathbf{U}$ and a $20,519 \times 2,738$ item auxiliary representation $\mathbf{I}$. We *evaluate all three tasks on XING*.

For CiteULike, we use the same training and test splitting of [5], but further select 30% of items and all records of them from the test set in [5] as validation set and the remaining part as our test set. For LastFM, we randomly select 10% of users and all their records as the validation set and 30% of items and all their records as the test set. For XING, as shown in Figure 3.7, we select cold-start users and cold-start items randomly for validation set and test set in the same way, and generate cold-user, cold-item, and cold-user-item validation sets and test sets (6 sets in total). The detailed statistics of the datasets are shown in Table 3.2.

Table 3.2: Statistics of training, validation and test sets in the three datasets. XING-U: XING dataset with cold-start users (for Task 1); XING-I: XING with cold-start items (for Task 2); XING-UI: XING with both cold statr users and items (for Task 3). Reprinted with permission from [7].

	Training				Validation			Test		
	#user	#item	#record	density	#user	#item	#record	#user	#item	#record
CiteULike	5,551	13,584	164,210	0.22%	5,551	1,018	13,037	5,551	2,378	27,739
LastFM	1,136	12,850	55,810	0.38%	189	12,850	9,209	567	12,850	27,815
XING-U	64,129	12,312	1,549,242	0.20%	10,688	12,312	258,497	32,064	12,312	775,837
XING-I	64,129	12,312	1,549,242	0.20%	64,129	2,051	275,782	64,129	6,156	756,638
XING-UI	64,129	12,312	1,549,242	0.20%	10,688	2,051	45,807	32,064	6,156	379,730

Metrics

We adopt three different ranking evaluation metrics to evaluate model performance: *Precision@k* (*P@k*), *Recall@k* (*R@k*) and *NDCG@k*. The detailed definitions of these metrics can be found in [17, 16]. We report $k = \{20, 50, 100\}$ here.

Baselines

We consider eight state-of-the-art cold-start recommendation algorithms to compare with the proposed model:

- **KNN** [64] generates recommendations by conventional nearest neighbor algorithm. The user-user or item-item similarity is computed by the given auxiliary representations. This method works for Task 1 and 2 but not Task 3.

- **CMF** [50] combines matrix factorization and auxiliary representations to CF representations transformation together into one objective function and trains these two parts simultaneously. CMF works for Task 1 and Task 2 but not Task 3.

- **LinMap** [6] inputs pretrained CF representations and learns a matrix to transform auxiliary representations to pretrained CF representations. LinMap can work for all three tasks.

- **NLinMap** is similar to [49], which applies deep neural networks to extract features from auxiliary representation to transform the auxiliary representations to CF space. We use a MLP of architecture $300 \rightarrow 300 \rightarrow 200$ for CiteULike, a MLP of architecture $800 \rightarrow 400 \rightarrow 200$ for LastFM and XING. All hidden layers have *ReLU* as the activation function. NLinMap can work for all three tasks.

- **LoCo** [3] is a linear low-rank regression method, which learns a low-rank transformation matrix to directly transform auxiliary representations to final predicted preference scores. It can only work for Task 1 and 2 but not Task 3.

- **LWA** [2] is a meta-learning based algorithm which constructs different logistic regression classifiers for different users based on their historical records. The user-specific logistic regression takes the auxiliary representation of one cold-start item as input and predicts whether the user will like input item or not. LWA can only work for Task 2.

- **DropoutNet** [5] inputs both pretrained CF representations and auxiliary representations into a MLP and randomly dropouts pretrained CF representations during training. It works for all tasks.

- **LLAE** [4] applies the idea of zero-shot learning to solve cold-start recommendation problems. Similar to LoCo, LLAE also learns a transformation matrix to directly transform auxiliary representation to predicted preference scores.

Reproducibility

Code and data, and experimental settings can be found at `https://github.com/Zziwei/Heater--Cold-Start-Recommendation`. We implement the proposed model by Tensorflow [59] and Adam [60] optimizer. For the hyper-parameters, we fix the CF latent factor dimension as 200, and set the learning rate as 0.005, the mini-batch size as 1024 for all models. Besides, we re-sample negative samples in each epoch and set the negative sampling rate 5 for all models. Then we tune other hyper-parameters by grid search on validation sets. More specifically, for

Heater, we set the regularization weight $\lambda = 0.0001$ for CiteULike and XING, and $\lambda = 0.001$ for LastFM. We set the similarity constraint weight $\alpha = 0.0001$, set the Randomized Training probability $p = 0.5$, set the number of experts in Mixture-of-Experts Transformation as 5, have one hidden layer activated by *tanh* of dimension 200 as the expert , and have one hidden layer of dimension 200 activated by *tanh* as ϕ_U and ϕ_I for all 3 datasets.

Heater and some of the baselines require pretrained CF representations as input. Hence, we train a Bayesian Personalized Ranking (BPR) [27] model with latent factors of 200 dimensions, *L2* regularization weight 0.001, and learning rate as 0.005 for the three datasets, and use the learned latent factors of BPR as $\mathbf{P}$ and $\mathbf{Q}$.

All experiments are performed on a desktop machine with 32GB memory, an 8 core Intel i7-4820k 3.7GHz CPU and an Nvidia GeForce GTX Titan X GPU with 12 GB memory. The runtime of one epoch for Heater is 7s for CiteUlike, 6s for LastFM, and 4 minutes and 58 seconds for XING. Heater can converge within 100 epochs.

3.3.3.2 RQ1: Heater vs. Baselines

We begin by comparing the performance of Heater with eight state-of-the-art alternatives on three datasets. $Recall@k$, $Precision@k$ and $NDCG@k$ ($k = \{20, 50, 100\}$) are shown in Table 3.3. XING-U represents recommending warm items to cold-start users (Task 1) in XING dataset. Similarly, XING-I represents recommending cold-start items to warm users (Task 2), and XING-UI represents recommending cold-start items to cold-start users (Task 3). The best baselines are marked in bold, and the relative improvement (denoted as Δ) of Heater over the best baselines are also calculated. As we can see from the table, for both metrics and all datasets, Heater is able to outperform other models for different cold-start recommendation tasks. We also calculate the p-value of paired t-test for the relative improvement rates, showing the improvements are statistically significant. Note that LWA cannot work for situations with cold-start users, and there is no sufficient memory to run LLAE on XING. Hence, we do not report results of LWA for LastFM, XING-U, and XING-UI, and do not report results of LLAE for all three XING cases. Besides, KNN, CMF, LoCo, LWA and LLAE cannot work for Task 3, thus we do not report their results for

Table 3.3: $Recall@k$ (R), $Precision@k$ (P), and $NDCG@k$ (N) of all methods. '-' represents unavailable result: KNN, CMF, LoCo, LWA and LLAE cannot work for Task 3, thus there is no result for them on XING-UI; LWA cannot work for Task 1 thus there is no result for LWA on LastFM and XING-U; LLAE run into out-of-memory error on XING dataset thus there is no result of LLAE on XING-U and XING-I. Reprinted with permission from [7].

		CiteULike (Task 2)			LastFM (Task 1)			XING-U (Task 1)			XING-I (Task 2)			XING-UI (Task 3)		
		@20	@50	@100	@20	@50	@100	@20	@50	@100	@20	@50	@100	@20	@50	@100
	R	.219	.385	.521	.135	.234	.341	.122	.219	.301	.073	.105	.142	-	-	-
KNN	P	.048	.036	.025	.307	.211	.147	.150	.107	.074	.041	.024	.017	-	-	-
	N	.150	.231	.291	.354	.296	.363	.172	.215	.258	.074	.091	.106	-	-	-
	R	.235	.420	.574	.115	.204	.295	.290	.448	.556	.163	.299	.436	.112	.218	.331
LinMap	P	.059	.042	.030	.250	.178	.129	.350	.218	.135	.098	.073	.053	.068	.053	.041.
	N	.215	.305	.374	.288	.255	.315	.393	.456	.410	.161	.230	.287	.110	.164	.212
	R	.266	.445	.577	.132	.220	.298	.247	.393	.490	.074	.195	.326	-	-	-
CMF	P	.064	.045	.030	.288	.192	.130	.300	.193	.120	.043	.047	.040	-	-	-
	N	.229	.322	.382	.333	.283	.335	.349	.408	.457	.063	.125	.180	-	-	-
	R	.292	.500	.641	.137	.235	.321	.246	.395	.507	.207	.362	.483	-	-	-
LoCo	P	.070	.049	.033	.305	.208	.142	.301	.194	.124	.126	.087	.058	-	-	-
	N	.250	.354	.419	.359	.307	.365	.354	.413	.470	.223	.298	.350	-	-	-
	R	.275	.461	.625	.140	**.247**	**.346**	**.297**	**.450**	.553	.211	.367	.500	.141	.255	.370
NLinMap	P	.069	.047	.032	.307	**.216**	**.152**	**.358**	**.219**	.134	.127	.088	.060	.086	.062	.045
	N	.264	.359	.430	.354	**.311**	**.377**	**.400**	**.458**	**.515**	.212	.289	.346	.142	.199	.247
	R	.322	.497	.626	-	-	-	-	-	-	.198	.345	.481	-	-	-
LWA	P	.077	.051	.033	-	-	-	-	-	-	.120	.083	.058	-	-	-
	N	.296	.392	.452	-	-	-	-	-	-	.201	.275	.332	-	-	-
	R	.328	.509	**.652**	.135	.237	.338	.242	.422	**.564**	**.222**	**.371**	**.509**	**.144**	**.260**	**.378**
DropoutNet	P	.077	.050	.033	.300	.210	.150	.292	.206	**.137**	**.133**	**.090**	**.062**	**.088**	**.064**	**.046**
	N	.309	.403	.467	.344	.301	.369	.276	.392	.465	**.224**	**.301**	**.358**	**.145**	**.205**	**.255**
	R	**.362**	**.531**	.643	**.140**	.234	.322	-	-	-	-	-	-	-	-	-
LLAE	P	**.084**	**.054**	**.034**	**.311**	.207	.142	-	-	-	-	-	-	-	-	-
	N	**.325**	**.422**	**.476**	**.366**	.309	.367	-	-	-	-	-	-	-	-	-
	R	**.373**	**.55**	**.685**	**.145**	**.258**	**.369**	**.307**	**.473**	**.581**	**.242**	**.398**	**.537**	**.161**	**.290**	**.411**
Heater	P	**.089**	**.055**	**.035**	**.322**	**.228**	**.162**	**.371**	**.231**	**.141**	**.143**	**.096**	**.065**	**.097**	**.070**	**.050**
	N	**.373**	**.467**	**.528**	**.371**	**.327**	**.399**	**.415**	**.480**	**.535**	**.237**	**.317**	**.376**	**.157**	**.221**	**.272**
	R	2.9%	4.1%	5.1%	3.4%	4.3%	6.5%	3.6%	5.1%	3.1%	9.3%	7.3%	5.4%	12.0%	11.4%	8.8%
Δ	P	6.3%	3.2%	5.1%	3.7%	5.7%	6.6%	3.8%	5.3%	3.1%	7.3%	6.2%	4.7%	10.7%	9.7%	7.3%
	N	14.8%	10.8%	10.9%	1.3%	5.3%	5.9%	3.7%	4.7%	3.8%	6.1%	5.5%	4.7%	7.7%	7.7%	6.5%

XING-UI.

In addition to the outstanding cold-start recommendation performance, another advantage of

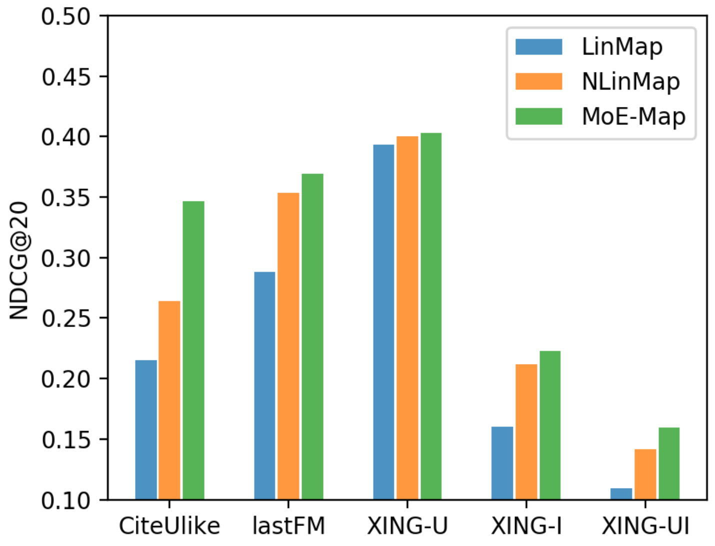


Figure 3.8: $NDCG@20$ comparison between MoE-Map model, LinMap and NLinMap. Reprinted with permission from [7].

Heater is that it is able to address all three cold-start tasks simultaneously by one time of training. Unlike LinMap and NLinMap, which have to first solve Task 1 and Task 2 one by one, then generate recommendations for Task 3 based on the trained models from Task 1 and Task 2.

3.3.3.3 RQ2: Ablation Study

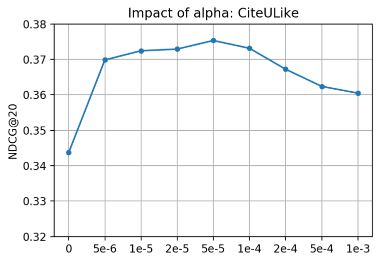
Next, we turn to investigate the effects of different components of Heater. We compare the complete version of Heater with three variations: (i) Heater without the similarity constraint (noted as w/o SC), which puts no constraint on $\mathbf{U}'_u$ and on $\mathbf{I}'_i$; (ii) Heater without Mixture-of-Experts Transformation (noted as w/o MoET), which just adopts a linear transformation as the transformation functions f_U and f_I; and (iii) Heater without Randomized Training (noted as w/o RT), which adopts the Heater basic framework as introduced in Section 3.3.2.2 with Mixture-of-Experts Transformation. Table 3.4 shows $Recall@20$, $Precision@20$, and $NDCG@20$ of the three models over all three datasets. Generally, Heater outperforms all variations for all metrics and all datasets, which indicates that the proposed similarity constraint, Mixture-of-Experts Transformation and Randomized Training mechanisms are effective and help to improve the cold-start recommendation quality.

Table 3.4: $Recall@20$, $Precision@20$ and $NDCG@20$ of proposed Heater, Heater w/o similarity constraint, Heater w/o Mixture-of-Experts Transformation, and Heater w/o Randomized Training. MoET represents Mixture-of-Experts Transformation, RT represents Randomized Training. Reprinted with permission from [7].

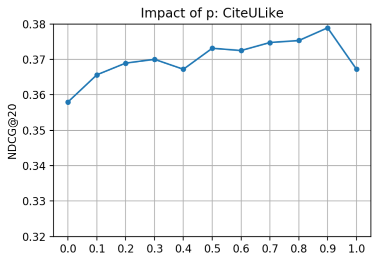
		CiteULike	LastFM	XING-U	XING-I	XING-UI
Heater	*R@20*	**.3727**	**.1451**	**.3074**	**.2420**	**.1609**
	P@20	**.0894**	**.3221**	**.3714**	**.1431**	**.0973**
	NDCG@20	**.3731**	**.3705**	**.4150**	**.2372**	**.1566**
w/o SC	*R@20*	.3273	.0944	.2723	.2092	.1252
	P@20	.0818	.2112	.3307	.1259	.0781
	NDCG@20	.3437	.2387	.3595	.2053	.1263
w/o MoET	*R@20*	.3406	.1431	.2856	.2160	.1454
	P@20	.0827	.3185	.3449	.1291	.0890
	NDCG@20	.3382	.3689	.3753	.2132	.1434
w/o RT	*R@20*	.3654	.1415	.2407	.1843	.1534
	P@20	.0887	.3095	.2946	.1099	.0929
	NDCG@20	.3672	.3532	.3145	.1833	.1511

Another observation is that for different datasets and different cold-start recommendation tasks, the performance improvement brought by the similarity constraint, Mixture-of-Experts Transformation or Randomized Training are different. For instance, for CiteULike, Randomized Training improves the $NDCG@20$ by only 1.61% but Mixture-of-Experts Transformation improves by 10.32% and similarity constraint improves by 8.55%, however, for LastFM, Mixture-of-Experts Transformation improves only by 0.43% while Randomized Training improves by 4.90% and similarity constraint improves by 55.22%.

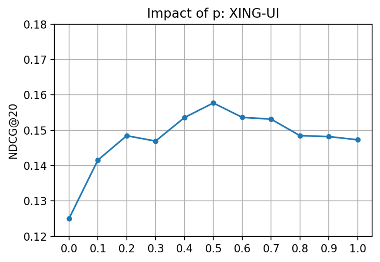
Moreover, to further study the effectiveness of Mixture-of-Experts Transformation, we also implement a variation of NLinMap with the original MLP replaced by a Mixture-of-Expert Transformation (with the same structure as it in Heater described in section 3.3.3.1). The variation is denoted as MoE-Map. Comparisons of $NDCG@20$ between MoE-Map and LinMap and NLinMap are present in Figure 3.8. From the figure we can observe that MoE-Map outperforms LinMap and NLinMap for all cases. As a result, we can conclude that Mixture-of-Experts Transformation is effective for auxiliary representation transformation.



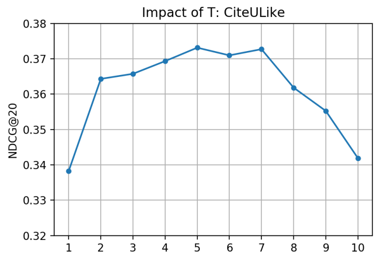
(a) Varying α in CiteULike.



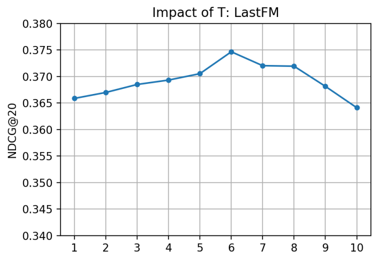
(b) Varying p in CiteULike.



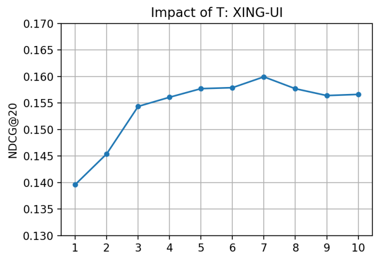
(c) Varying p in XING-UI.



(d) Varying T in CiteULike.



(e) Varying T in LastFM.



(f) Varying T in XING-UI.

Figure 3.9: $NDCG@20$ results of Heater with different hyper-parameters. Reprinted with permission from [7].

3.3.3.4 RQ3: Impact of Hyper-parameters

Next, we study the impact of three hyper-parameters: the similarity constraint weight α, the Randomized Training probability p, and the number of experts in Mixture-of-Experts Transformation T.

Similarity constraint weight α

We first vary the similarity constraint weight α in $\{0, 5e-6, 1e-5, 2e-5, 5e-5, 1e-4, 2e-4, 5e-4, 1e-3\}$ and set the other hyper-parameters as the same as described in Section 3.3.3.1. α controls the strength of similarity constraint and is directly connected with model effectiveness. For conciseness, we only report results on CiteULike because the patterns on other datasets show similar results as CiteULike. $NDCG@20$ results of Heater on CiteULike are shown in Figure 3.9a, where we can observe that with α increasing, the cold-start recommendation quality first improves and then decreases. This is reasonable because small α causes underfitting for the auxiliary representation transformation, while large α does not only give rise to overfitting for the transformation, but also decreases the ratio of parameters updating due to recommendation loss. For CiteULike, the best performance is achieved when $\alpha = 5e - 5$, and for other datasets the best performances are achieved around the same value.

Randomized Training probability p

Then, we vary the Randomized Training probability p in the range of $[0.0, 1.0]$ with step 0.1. Our goal is to know how p influences the randomized training process and how to set a reasonable p to maximize the effect of Randomized Training. Because experiments on LastFM, XING-U and XING-I show similar patterns as XING-UI, thus we only plot $NDCG@20$ results of CiteULike in Figure 3.9b and XING-UI in Figure 3.9c. Generally, with p growing from 0.0 to 1.0, $NDCG@100$ first increases then decreases, but the peak performances are different for different datasets: the best metrics are achieved when $p = 0.9$ in CiteULike and $p = 0.5$ in XING-UI (around 0.5 for other datasets). We can draw two conclusions: (i) Randomized Training is effective for improving cold-start recommendation: when p is within a reasonable range, the performance improves with p increasing; and (ii) large p is not always helpful because f_U and f_I are trained less in this case.

Number of experts in Mixture-of-Experts Transformation T

Last, we investigate the impact of the number of experts T in the Mixture-of-Experts Transformation. T controls the model complexity of the transformation component, and theoretically larger T (within reasonable range) is supposed to produce better performance. We experiment with T varying from 1 to 10 with step 1, and the empirical results on three datasets are shown in Figure 3.9d, 3.9e and 3.9f (XING-U, XING-I and XING-UI show similar patterns, thus we only show results of XING-UI here). Generally, on all datasets, with T increasing from 1, the performance improves first and arrives at a peak. However, the best T for different datasets are different, which is reasonable because the auxiliary representations in different datasets have distinct characteristics and requires different degrees of complexity for Mixture-of-Experts Transformation. The best choice of T is 5 for CiteULike, 6 for LastFM, and 7 for XING-UI.

Another direction of changing the complexity of the Mixture-of-Experts Transformation is to

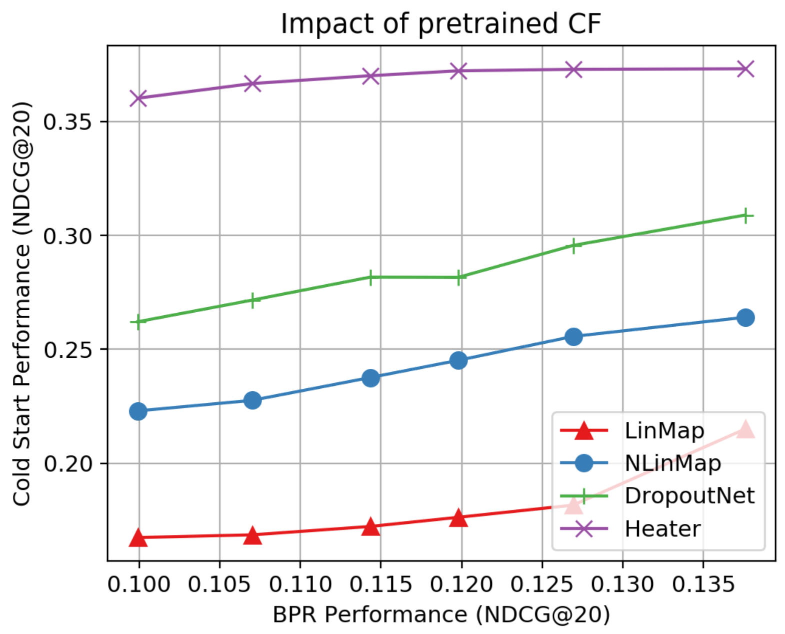


Figure 3.10: $NDCG@20$ results on CiteULike of LinMap, NLinMap, DropoutNet and Heater with different pretrained CF representations of varying quality. Reprinted with permission from [7].

increase the number of layers in each expert. But adding more layers dramatically increases the computational cost, and does not bring much performance improvement, hence we do not show experimental results here. Besides, we only use one layer for one expert, thus, we cannot change the model complexity by changing the hidden layer dimension because it must be the same as the dimension of pretrained CF representations.

3.3.3.5 RQ4: Impact of Pretrained CF Quality

We last study the impact of the quality of pretrained CF representations on Heater compared with other models which also take pretrained CF representations as input. We want to know with better pretrained CF representations, does Heater perform better? And is Heater relatively robust to the change of pretrained CF model compared with other models? For conciseness, we only show results on CiteULike. Experiments on other datasets show similar patterns. We generate user and item CF representations by training BPR on user-item interactions of warm start users and items, and we fix all hyper-parameters of BPR except the number of training epochs to generate representations of different qualities. Because we can only use the training set to train and evaluate BPR, thus we further split the training set shown in Table 3.2 into 90% for training, and 10% for evaluating. Then, with different quality of pretrained representations, we compare the cold-start recommendation performance of DropoutNet, LinMap, NLinMap, and Heater. $NDCG@20$ results on CiteULike are shown in Figure 3.10, where we involve five sets of pretrained representations of different qualities, x-axis represents the $NDCG@20$ of BPR trained by different numbers of epochs (6, 8, 12, 16 and 30 epochs respectively, and the epoch number of experiment in Section 3.3.3.2 is 30), y-axis represents the cold-start recommendation performances of models. From this we can observe that for all pretrained representations of different qualities, Heater always outperforms alternatives, and as the quality of BPR increases, the performance of all models improves. Moreover, the performance of Heater is much more consistent when the quality of BPR representations differs, while other models have a larger range of performance changes. Note that $NDCG@20$ of BPR is lower than that of cold-start recommendation performance because they are under different experiment setups: BPR is evaluated by recommending $13,584$ items to $5,551$ users, cold-start models are evaluated by recommending $2,378$ items to $5,551$ users.

3.3.4 Summary

In this section, we propose a novel model called Heater to address the cold-start recommendation problem. There are three innovative features of Heater, which are designed to address three key challenges in existing cold-start recommendation algorithms: (i) a combined framework incorporating the structures of separate-training and joint-training methods to avoid the error superimposition issue and improve the model effectiveness; (ii) the Randomized Training strategy to further promote quality of model learning; and (iii) the Mixture-of-Experts Transformation mechanism to provide 'personalized' transformations for users and items. Empirical results on three public datasets show the performance improvement of Heater over state-of-the-art alternatives.

3.4 Conclusions

In this chapter, we investigate the exposure bias in user-item interaction data. To address the exposure bias, we first propose a combinational joint learning framework, which jointly learns unbiased relevance and propensity simultaneously, to produce unbiased recommendations based on biased implicit data. Extensive experiments on two public datasets show the effectiveness of the proposed method. Then, we also further explore the scenario with extreme exposure bias, where there are users and items without any historical interactions. This is also known as the cold-start recommendation problem. To tackle this cold-start problem, we propose a novel model called Heater. There are three innovative features of Heater, which are designed to address three key challenges in existing cold-start recommendation algorithms: i) a combined framework incorporating the structures of separate-training and joint-training methods to avoid the error superimposition issue and improve the model effectiveness; ii) the Randomized Training strategy to further promote quality of model learning; and iii) the Mixture-of-Experts Transformation mechanism to provide 'personalized' transformations for users and items. Empirical results on three public datasets show the performance improvement of Heater over state-of-the-art alternatives.

4. IDENTIFYING AND MITIGATING BIAS IN RECOMMENDATION ALGORITHMS[1]

Next, we move our attention from studying the bias raised by data to investigate the bias intrinsic in recommendation algorithms. Along this line, we focused on how Collaborative Filtering (CF) based recommendation algorithms amplify the imbalance in the training data to produce biased recommendations. Here, we show that even when the training data is free of exposure bias and can perfectly reflect user-item relevance, the machine learning algorithms trained by unbiased data can still introduce bias! Specifically, in this chapter, we first introduce our work about the *popularity-opportunity bias*, which investigates the inequality of recommendation probability to matched users across items with different popularity. We conduct both empirical and theoretical studies on the new popularity-opportunity bias for two fundamental recommendation algorithms – matrix factorization (MF) [29] and Bayesian personalized ranking (BPR) [27], showing the prevalence of such a bias in recommendation algorithms. Correspondingly, we design a simple but powerful post-processing approach to mitigate the popularity-opportunity bias. Then, we turn to investigate the *mainstream bias* among users with different preferences, which refers to the problem that mainstream users receive recommendations of much higher accuracy than niche users. To tackle this, we design a local fine tuning algorithm which trains a customized model for each user.

4.1 Related Work

In this section, we introduce previous work about the bias introduced by recommendation algorithms in terms of both items and users. Specifically, we introduce related work on conventional popularity bias and recommendation bias on users.

[1]This chapter is reprinted with permission from “Popularity-Opportunity Bias in Collaborative Filtering" by Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee, 2021, Proceedings of the 14th ACM International Conference on Web Search and Data Mining, Copyright 2021 by ACM; and “Fighting Mainstream Bias in Recommender Systems via Local Fine Tuning" by Ziwei Zhu and James Caverlee, 2022, Proceedings of the 15th ACM International Conference on Web Search and Data Mining, Copyright 2022 by ACM.

4.1.1 Popularity Bias

Conventional Popularity Bias refers to the phenomenon that recommenders tend to assign high rankings for popular items at the expense of lower recommendation opportunities for less popular items [69, 70, 71, 72, 73, 74]. This concept and its influence on recommendations has been studied in [72, 73, 74], and later, Jannach et al. [75] empirically showed that different recommendation algorithms have different vulnerabilities to popularity bias. Long-tail items are considered valuable because they often represent novelty and serendipity [72, 76, 77], thus, they are important in terms of promoting user satisfaction and preventing the monopoly by big brands [71]. To mitigate the harmful effects of popularity bias, many debiasing approaches have been proposed [71, 70, 78, 79, 44, 80].

However, existing works [69, 70, 71, 79] mainly study the effects of item popularity on the ranking results themselves – e.g., are popular items recommended more often or ranked higher than less popular ones? – without considering what are the user preferences toward them (aligned with the concept of statistical parity). This is problematic because without conditioning on user preferences, recommendation difference is not necessarily evidence of bias. Thus, we propose popularity-opportunity bias, which studies the impact of item popularity conditioned on user preferences (which is aligned with the concept of equal opportunity). Furthermore, most prior works study the group-level impact of popularity on recommendations by grouping items based on their popularity [75, 70, 71, 81, 79, 78, 69]. These studies often consider two groups – popular items vs. long-tail items – which ignores the subtle distinction between individual items at different ranking positions. In contrast, we directly investigate rankings and popularity of individual items.

4.1.2 Bias on users

These prior works mainly focus on the item perspective. Yet, how users are treated is an equally important topic. The majority of research works investigating the bias on users aim to analyze the utility difference among different user groups determined by user demographic attributes, such as age or gender [82, 83, 84, 85, 86]. For example, Schedl et al. [82] study the music preference difference among different user age groups and shows that the recommendation performance for these age groups are also different. Ekstrand et al. [83] empirically study multiple types of recommendation models and demonstrate that all of the investigated models produce a utility difference across user demographic groups. To address this problem, Fu et al. [85] propose to take advantage of the rich information from knowledge graphs, and Li et al. [84] create a re-ranking algorithm to reduce the utility gap among user groups.

Different from the aforementioned works studying bias based on demographic groups, we aim to recognize the mainstream and niche users and study the utility difference between them. Also note that user demographic attributes may not necessarily explain the interests and behaviors of a user. A similar work to our work is [87], whose goal is to improve the utility for niche users. Nevertheless, the major differences are: in [87], the mainstream and niche users are determined purely based on recommendation utility they receive rather than based on the true user preference reflected by historical feedback; and the algorithm proposed in [87] requires additional auxiliary information of users and items (such as the review text users give to items), while we aim to debias relying merely on feedback from users.

Here, we find that local recommendation models [88, 89, 90, 91], although not designed for this purpose, can mitigate the bias to some degree by improving the utility for niche users. The main idea of these methods is to use different local models to serve different types of users. Among existing methods, the recently proposed local collaborative autoencoder (LOCA) [88] produces the state-of-the-art performance, which uses multiple variational autoencoders (VAE) [92] as local models to capture the special patterns of different sub-communities. Hence, in our experiments, we follow LOCA to consider VAE as the base model for our proposed local method, and we empirically compare our proposed methods with LOCA.

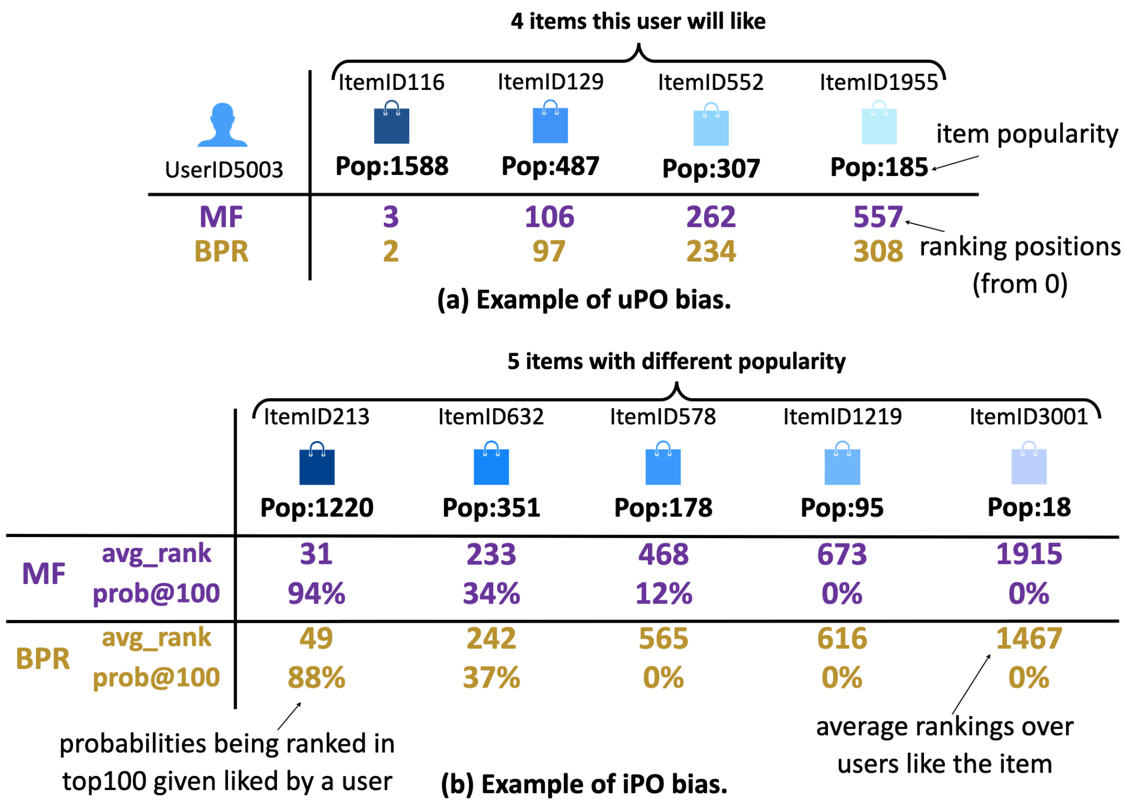
4.2 Analyzing and Mitigating Popularity-opportunity Bias

4.2.1 Introduction

Statistical parity and equal opportunity are two important concepts for studying fairness and bias in classification and recommendation tasks [93, 94, 95, 24, 12]. Statistical parity requires the same *positive rate* over individuals or groups [11, 10]. On the other hand, equal opportunity requires the same *true positive rate* [24, 12]. Because statistical parity investigates algorithmic bias without conditioning on the ground truth, the bias identified and removed based on statistical parity is not necessarily an undesired harmful bias [24, 12].

In this chapter, we re-examine popularity bias from the perspective of equal opportunity. We observe that previous studies of popularity bias [69, 70, 71, 72, 73, 74] are mainly governed by statistical parity, and so inherit its limitations. We then connect the concept of equal opportunity to this conventional popularity bias to introduce the new problem of *popularity-opportunity bias* in implicit recommenders.

Suppose we consider the popularity of items as the number of feedback actions toward each item (clicks or views). Conventional popularity bias [69, 70, 71, 72, 73, 74] refers to the phenomenon that high rankings are tend to assigned for popular items at the expense of lower rankings for less popular items. These studies of conventional popularity bias examine the impact of item popularity on recommendation results alone, without taking user preferences into account. That is, the positive rate difference over items of different popularity is calculated for measuring the conventional popularity bias, which is essentially aligned with the concept of statistical parity [93, 94, 95]. However, such a bias definition is problematic because without conditioning on user preferences, the recommendation result (or positive rate) alone is not necessarily evidence of bias. For example, for a user u, one popular item i and one less popular item j, better ranking for the popular item i than the less popular item j is a biased recommendation defined by conventional popularity bias. Yet, if we know that u likes i but dislikes j, then this ranking result is in fact reasonable and not a harmful bias. Moreover, forcing similar rankings for i and j as in previous



4 items this user will like

UserID5003	ItemID116 Pop:1588	ItemID129 Pop:487	ItemID552 Pop:307	ItemID1955 Pop:185
MF	3	106	262	557
BPR	2	97	234	308

item popularity

ranking positions (from 0)

(a) Example of uPO bias.

5 items with different popularity

		ItemID213 Pop:1220	ItemID632 Pop:351	ItemID578 Pop:178	ItemID1219 Pop:95	ItemID3001 Pop:18
MF	avg_rank	31	233	468	673	1915
	prob@100	94%	34%	12%	0%	0%
BPR	avg_rank	49	242	565	616	1467
	prob@100	88%	37%	0%	0%	0%

probabilities being ranked in top100 given liked by a user

average rankings over users like the item

(b) Example of iPO bias.

Figure 4.1: Examples of (a) uPO bias and (b) iPO bias in ML1M. Reprinted with permission from [8].

works [80, 44] to remove conventional popularity bias could actually hurt user satisfaction and engagement of the popular item i.

Thus, inspired by equal opportunity, we propose to investigate the **popularity-opportunity bias**: *conditioned on user preferences that a user likes both items*, is the more popular item more likely to be recommended (or ranked higher) to the user than the less popular one? That is, we calculate the true positive rate difference over items of different popularity for measuring the bias during testing, and require the true positive rate to be the same for items of different popularity to achieve equal opportunity. To our best knowledge, this is the first work which studies popularity bias from the view of equal opportunity for recommender systems.

To identify popularity-opportunity bias during testing, one critical question is how do we know user preferences to measure the bias? That is, how do we know whether u likes i or j? In practice, the utility of a recommender system is typically evaluated through a train-test split, where a learned model (based on the training data) is evaluated over the testing data, where the testing data contains held-out evidence of user preferences (e.g., by likes, views, or clicks). Similarly, we can leverage the same testing data as indicators of user preferences to identify popularity-opportunity bias.

User-side popularity-opportunity bias

More specifically, we investigate the proposed popularity-opportunity bias from the views of users and items separately. To illustrate, let's first consider the example in Figure 4.1a. Here we show four items from the MovieLens 1M dataset [96] that user ID5003 likes during testing. That is, these items are not seen during training but are in the test set of this user, and the user will interact with these items once recommended (i.e., they are true positives). Item ID116 is the most popular one with 1588 feedback actions, while item ID1955 is the least popular with only 185 feedback records. Then, we show the ranking positions of these four items for user ID5003 according to two fundamental collaborative filtering models – matrix factorization with Root Mean Square Error loss (denoted as MF) [29] and Bayesian Personalized Ranking loss (denoted as BPR) [27]. We observe that popular items are ranked higher than less popular items by both models, *even though we know the user likes all of them*. We refer to this as *user-side popularity-opportunity bias* or uPO bias for short.

Item-side popularity-opportunity bias

Complementary to this user-side perspective, we show an example of five items in Figure 4.1b. Item ID213 is the most popular, while item ID3001 is the least popular. If we consider only the matched users who like each item in testing data (i.e., for item i, only the ranking positions for matched users who have i in their test set are considered), we observe that more popular items will have better rankings and higher probabilities of being ranked in the top-100. For example, item ID213 is ranked by MF in the top-100 for 94% of all matched users, whereas item ID3001 is never ranked in the top-100 for its matched users. This reveals a systematic low recommendation opportunity for low-popularity items. We refer to this as *item-side popularity-opportunity bias* or iPO bias for short.

Both this user-side and item-side bias raise critical issues. User-side (uPO) bias is harmful because a user's need corresponding to these low-popularity items is not acknowledged and not satisfied by the recommender. Moreover, low-popularity items sometimes are more important than popular items because they can be serendipitous and novel for users, crucial for extending the area of users' interests and promoting user engagement [72, 76]. Item-side (iPO) bias brings damaging outcomes that long-tail items may not have any chance to become popular or even known, and providers of these items will receive less engagement in the system. In the long-term, iPO bias could accumulate, leading to a recommender dominated by well-known popular items.

Our contributions

Hence, we propose a three-part study of the popularity-opportunity bias:

- Figure 4.1 shows cases of the bias, but is it prevalent beyond these examples? To answer this, we conduct a comprehensive data-driven study over four datasets to investigate the presence of popularity-opportunity bias. We focus on two fundamental collaborative filtering approaches (MF and BPR) that serve as foundations of many recommenders including recent neural ones [17]. We empirically demonstrate both models produce user-side and item-side bias.

- While this data-driven study showcases the prevalence of the bias, is it truly inherent to these models or an artifact of these datasets? To answer this, we theoretically analyze the impact of item popularity on ranking by MF and BPR to confirm the existence of the bias in both methods.

- Last, we investigate the potential of two approaches to reduce this bias: a post-processing approach to compensate for popularity in recommendation; and an in-processing approach that regularizes predicted scores and item popularity. Through experiments on four datasets, we explore the trade-offs between debiasing effectiveness and recommendation utility, showing the more effective debiasing performance of the two proposed methods over existing debiasing baselines designed for conventional popularity bias.

4.2.2 Preliminaries

In this section, we first describe the implicit recommendation problem, then introduce matrix factorization based collaborative filtering models with two different objective functions.

Implicit Recommendation

Suppose we have a user set $\mathcal{U} = \{1, 2, \ldots, N\}$ and an item set $\mathcal{I} = \{1, 2, \ldots, M\}$. We need to recommend a list of k items to every user u based on her implicit feedback record $\mathcal{O}_u^+ = \{i, j, \ldots\}$, where $i, j, \ldots$ are the items u has provided positive feedback to before, which are used as training data for model learning. Besides, we have another item set $\widetilde{\mathcal{O}}_u^+$ to represent the items that user will like during testing, which are the test data for evaluating recommendation utility and recommendation bias.

Matrix Factorization

Matrix factorization based collaborative filtering [29, 27] is the foundation of many state-of-the-art recommendation models [30, 31], as well as recent neural-network based models [17, 32, 33] that use matrix factorization as the final layer for predicting preference scores. The main idea is to learn low-dimensional latent representations for users and items based on existing user-item interactions, and then to predict preference scores for unobserved user-item pairs by the dot-product of latent representations: $\widehat{\mathbf{R}}_{u,i} = \mathbf{P}_u^\top \mathbf{Q}_i$, where $\mathbf{P}_u \in \mathbb{R}^{H \times 1}$ is the latent representation of user u, $\mathbf{Q}_i \in \mathbb{R}^{H \times 1}$ is the latent representation of item i, and H is the latent dimension.

There are two main categories of objective functions for matrix factorization models: point-wise objective functions (include Root Mean Square Error (RMSE) [29], Cross-Entropy [17], among others) and pair-wise objective functions (include Bayesian Personalized Ranking loss (BPR) [27], Hinge loss [34], and others). Since RMSE and BPR are two of the most widely applied objective functions, we focus on these two in the rest of the chapter. We denote the matrix factorization model with RMSE as MF, and the one with BPR loss as BPR. The formulations are

Table 4.1: Characteristics of the four public datasets. Reprinted with permission from [8].

	#users	#items	density	pop_avg	pop_std
ML1M	6,040	3,260	3.55%	214.41	276.85
Ciao	5,047	8,102	0.21%	10.82	19.13
Epinions	12,168	11,283	0.18%	21.88	33.07
App	16,201	4,869	0.23%	37.96	66.34

shown below:

$$\min_{\boldsymbol{\Theta}} \mathcal{L}_{MF} = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{O}_u^+ \cup \mathcal{O}_u^-} \sqrt{(\widehat{\mathbf{R}}_{u,i} - \mathbf{R}_{u,i})^2}, \tag{4.1}$$

$$\min_{\boldsymbol{\Theta}} \mathcal{L}_{BPR} = -\sum_{u \in \mathcal{U}} \sum_{\substack{i \in \mathcal{O}_u^+ \\ j \in \mathcal{O}_u^-}} ln\, \sigma(\widehat{\mathbf{R}}_{u,i} - \widehat{\mathbf{R}}_{u,j}), \tag{4.2}$$

where $\mathcal{O}_u^-$ is the randomly sampled negative item set for u; $\sigma(\cdot)$ is the Sigmoid function; and $\boldsymbol{\Theta}$ represents the model parameters, i.e., the latent representations for users and items $\mathbf{P}$ and $\mathbf{Q}$.

4.2.3 Data-driven Study

In this section, we conduct a data-driven study of popularity-opportunity bias over four datasets, and show how MF and BPR are vulnerable to this bias on both user (uPO bias) and item (iPO bias) sides. While many previous studies have identified conventional popularity bias, this is the first to identify popularity-opportunity bias.

We adopt four widely used datasets from different domains: ML1M [96], Ciao [97], Epinions [97], Amazon-App [98]. For all datasets, we consider the rating or reviewing behaviors as positive feedback from users to items, and regard the number of feedback actions an item receives as its popularity. We first filter out users and items with interactions fewer than 10, and then randomly split them into 60%, 20%, and 20% for training, validation, and testing. The details of these datasets are presented in Table 4.1, where pop_avg shows the average popularity of the items and pop_std shows the standard deviation of item popularity.

We train MF and BPR models by the training sets of these datasets; tune hyper-parameters by grid search on validation sets; and report the results on test sets. Further details of the experimental setup can be found in Section 4.2.6.1.

4.2.3.1 Measuring uPO and iPO Bias

First, we introduce two metrics to measure uPO and iPO bias. Similar to recommendation utility metrics, such as *NDCG*, the two introduced bias metrics are calculated based on the test item set $\widetilde{\mathcal{O}}_u^+$ for each user u.

Measuring uPO bias

For uPO bias, we want to know for each user u, among all items u will like during testing (items in $\widetilde{\mathcal{O}}_u^+$), whether less popular items are ranked lower than more popular ones, i.e., whether the rankings are correlated with popularity given items are liked by the user. Thus, for each user u, we calculate the *Spearman's rank correlation coefficient* between the popularity of items in $\widetilde{\mathcal{O}}_u^+$ and their ranking positions, then average all users to have the *popularity-rank correlation for users* (denoted as ***PRU***):

$$PRU = -\frac{1}{N}\sum_{u\in\mathcal{U}} SRC(pop(\widetilde{\mathcal{O}}_u^+), rank_u(\widetilde{\mathcal{O}}_u^+)), \tag{4.3}$$

where $SRC(\cdot,\cdot)$ calculates Spearman's rank correlation; $pop(\cdot)$ returns item popularity (it counts the number of feedback actions for each item) for given items; and $rank_u(\widetilde{\mathcal{O}}_u^+)$ returns the rankings (from 0 to $M-1$, 0 represents the top-most ranking) of given items for user u by a specific model. Spearman's rank correlation coefficient assesses the monotonic relationship between two variables and has values in the range $[-1,1]$. Hence, a large positive value (note that we add a negative sign before $SRC(\cdot,\cdot)$ to flip the sign) of ***PRU*** means that low popularity leads to low rankings for items a user likes during testing, which violates the requirement of equal opportunity for items of different popularity, i.e., high uPO bias.

Measuring iPO bias

For iPO bias, we want to know whether the expected rankings of low-popularity items for matched users are lower than the expected rankings of high-popularity items, i.e., whether the expected ranking position of an item for a matched user is correlated with its popularity. Hence, we calculate the Spearman's rank correlation coefficient between the popularity of all items and their average ranking positions over matched users (for each item i, fetch all the users who have i in test set $\widetilde{\mathcal{O}}_u^+$, and then average the ranking positions in the ranking lists of these users) to have the *popularity-rank correlation for items* (denoted as ***PRI***):

$$PRI = -SRC(pop(\mathcal{I}), avg\_rank(\mathcal{I})), \tag{4.4}$$

where $avg\_rank(i) = \frac{1}{|\widetilde{\mathcal{U}}_i|} \sum_{u \in \widetilde{\mathcal{U}}_i} rank_u(i)$ returns the average ranking for item i over the set of matched user $\widetilde{\mathcal{U}}_i$ (i.e., for each $u \in \widetilde{\mathcal{U}}_i$, i is in $\widetilde{\mathcal{O}}_u^+$)). A large positive value of ***PRI*** means that lower popularity leads to worse rankings, violating the requirement of equal opportunity, i.e., high iPO bias. In our experiments, we also evaluate the iPO bias by calculating the probability of being ranked in top-k for a matched user (as examples in the Figure 4.1b), which shows similar pattern as the introduced metric PRI. Thus, here, we will only report results based on ***PRI***.

Compare PRU and PRI

Both PRU and PRI measure popularity-opportunity bias. The main difference is how they calculate the popularity-ranking correlation and aggregate across users. Due to this calculation difference, PRU and PRI measure different aspects of popularity-opportunity bias. PRU represents the expectation of popularity-ranking correlation of matched items a random user will get from a model, which is to say, it quantifies the bias from the view of users. On the other hand, PRI measures the correlation between item popularity and the expectation of ranking position from matched users for items, which is to say, it quantifies the bias from the view of items. Although in practice, these two metrics usually show similar patterns, they are essentially not the same. It is possible that a model generates high uPO bias measured by PRU while low iPO bias measured by PRI, or vice versa. Hence, it is necessary to study the proposed popularity-opportunity bias from both PRU and PRI perspectives.

Table 4.2: Measuring uPO bias (*PRU*) and iPO bias (*PRI*) for MF and BPR on four datasets. * indicates that the Spearman's rank correlation coefficients are statistically significant for $p < 0.01$ judged by t-test. Reprinted with permission from [8].

	ML1M		Ciao		Epinions		App	
	MF	BPR	MF	BPR	MF	BPR	MF	BPR
PRU	0.835	0.779	0.542	0.591	0.684	0.708	0.567	0.636
PRI	0.980*	0.969*	0.363*	0.433*	0.535*	0.573*	0.609*	0.692*

4.2.3.2 Observations

In the following, we report our observations of uPO and iPO bias for MF and BPR over the four datasets.

Observations of uPO bias

First, we show *PRU* for both MF and BPR across all four datasets in Table 4.2. We can see that for both MF and BPR on all datasets, *PRU* values are large positive numbers, indicating both MF and BPR produce uPO bias. More precisely, for a user, even if we know that two items are equally liked by the user, the more popular one will have better ranking position than the less popular one. Note that we do not show the significance test results for *PRU* because the size of $\widetilde{\mathcal{O}}_u^+$ in Equation 4.3 is small for most of the users which makes the significance test uninformative (because the p-value is always large when only few instances are included). An example of such uPO bias in ML1M dataset is shown in Figure 4.1a, which is consistent with our observations from Table 4.2.

Observations of iPO bias

Next, we focus on the metric *PRI* to evaluate the iPO bias in Table 4.2. For all four datasets and both models, *PRI* are large positive values, which means in the recommendations by MF and BPR, items with high popularity have better expected rankings for their matched users, while the opposite holds for low-popularity items. Thus, we can confirm that MF and BPR produce the iPO bias.

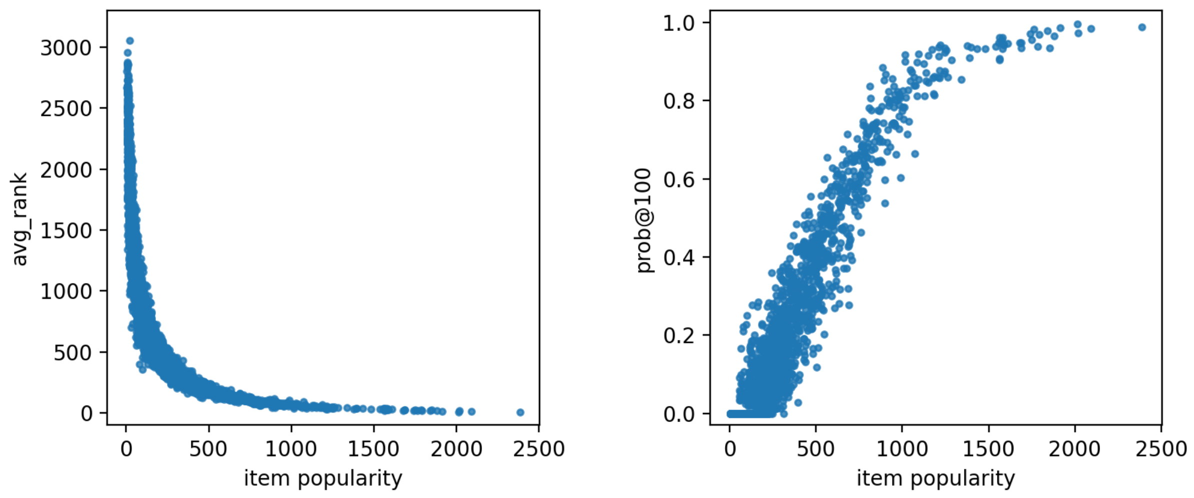


Figure 4.2: Scatter plots of ranking results by MF on ML1M. Reprinted with permission from [8].

To better show the effects of iPO bias, we present two scatter plots in Figure 4.2 for ranking results of MF on ML1M data (BPR and other datasets have similar patterns). Each dot represents one item. In the left figure, we plot the average rankings of items over matched users (y-axis) vs. popularity (x-axis), from which we can observe a monotonic decreasing trend for the average rankings as the popularity increases. In the right figure, for each item, we plot the probability of being ranked in the top-100 for matched users (y-axis) vs. popularity (x-axis), where we see a monotonic increasing trend for the recommendation probabilities when the popularity increases. These observations are consistent with the conclusions drawn from the bias metric shown in Table 4.2 that more popular items have better rankings for matched users than less popular items do. Real examples of such iPO bias in ML1M dataset are presented in Figure 4.1b.

4.2.4 Theoretical Study

After empirically confirming the existence of bias in MF and BPR, we turn in this section to theoretically analyze the relationship between item popularity and ranking results generated by MF and BPR under two simplifying assumptions, to confirm the existence of uPO and iPO bias in MF and BPR.

4.2.4.1 Existence of Bias in MF

We first formulate the input and output of the MF model. Given a training user-item interaction matrix $\mathbf{R} \in \{0,1\}^{N \times M}$ with N users, M items, 1 represents a known user-item interaction, and 0 represents an unknown user-item relationship. If we train an MF model on $\mathbf{R}$, we can get a user latent representation matrix $\mathbf{P} \in \mathbb{R}^{H \times N}$ and an item latent representation matrix $\mathbf{Q} \in \mathbb{R}^{H \times M}$. Now we have **Assumption 1**: we assume the model is trained in an ideal condition where the loss function in Equation 4.1 is minimized close to 0. Then the dot product of the latent matrices will reconstruct $\mathbf{R}$ with very minor error: $\mathbf{P}^\top\mathbf{Q} = \widehat{\mathbf{R}}$ and $||\widehat{\mathbf{R}} - \mathbf{R}||_\mathrm{F}^2 < \epsilon$. This is to say that $\widehat{\mathbf{R}}_{u,i} \approx 1$ if $\mathbf{R}_{u,i} = 1$, and $\widehat{\mathbf{R}}_{u,i} \approx 0$ if $\mathbf{R}_{u,i} = 0$. We represent the reconstructed interaction matrix as $\widehat{\mathbf{R}} \in \{\sim 0, \sim 1\}^{N \times M}$, where ~ 0 and ~ 1 are numbers very close to 0 and 1. Without loss of generality, we assume values in $\widehat{\mathbf{R}}$ are non-negative because we can always add a positive constant to $\widehat{\mathbf{R}}$ to make all elements positive without changing the ranking results.

Because the number of ~ 1 values in columns of $\widehat{\mathbf{R}}$ can indicate the item popularity, we introduce the item popularity information to the formulations of $\mathbf{P}$ and $\mathbf{Q}$ by $\widehat{\mathbf{R}}$. Given a user u, the predicted preference scores for her toward all items can be calculated by $\mathbf{P}_u^\top\mathbf{Q} = \widehat{\mathbf{R}}_{u,:}$, where $\widehat{\mathbf{R}}_{u,:} \in \{\sim 0, \sim 1\}^{1 \times M}$ is the u-th row in $\widehat{\mathbf{R}}$. Moving $\mathbf{Q}$ to the right-hand side by pseudo-inverse, we can have $\mathbf{P}_u^\top = \widehat{\mathbf{R}}_{u,:}\mathbf{Q}^\top(\mathbf{Q}\mathbf{Q}^\top)^{-1}$. Similarly, we have $\mathbf{Q}_i = (\mathbf{P}\mathbf{P}^\top)^{-1}\mathbf{P}\widehat{\mathbf{R}}_{:,i}$, where $\widehat{\mathbf{R}}_{:,i} \in \{\sim 0, \sim 1\}^{N \times 1}$ is the i-th column in $\widehat{\mathbf{R}}$.

Based on the new formulations of $\mathbf{P}_u$ and $\mathbf{Q}_i$, we define several new matrices for the analysis. First, we define the *normalized user latent representation*: $\mathbf{A} = (\mathbf{P}\mathbf{P}^\top)^{-1}\mathbf{P}$, which normalizes $\mathbf{P}$ by the variances of its principal components over the principal component directions. The explanation for $\mathbf{A}$ is that $\mathbf{P}\mathbf{P}^\top$ can be factorized as $\mathbf{P}\mathbf{P}^\top = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ by Eigen-Decomposition, where $\mathbf{U}$ is an orthogonal matrix ($\mathbf{U}^\top = \mathbf{U}^{-1}$) with eigenvectors of $\mathbf{P}\mathbf{P}^\top$ as columns, and $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues of $\mathbf{P}\mathbf{P}^\top$ as diagonal elements. Then based on the definition of Principal Component Analysis [99], $\mathbf{U}^\top\mathbf{P}$ are the principal components of $\mathbf{P}$, $\mathbf{\Lambda}$ are the variances of these principal components. As a result, $\mathbf{A} = (\mathbf{P}\mathbf{P}^\top)^{-1}\mathbf{P} = \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^\top\mathbf{P}$, i.e., $\mathbf{P}$ is first transformed to the principal component space by $\mathbf{U}^\top$, then normalized by the variances of principal components by $\mathbf{\Lambda}^{-1}$, and last, transformed back to the original space by $\mathbf{U}$. In the same way, we can have the *normalized item latent representation*: $\mathbf{B} = (\mathbf{Q}\mathbf{Q}^\top)^{-1}\mathbf{Q}$, and the *normalized preference matrix*: $\mathbf{Z} = \mathbf{A}^\top\mathbf{B} \in \mathbb{R}^{N\times M}$ (values in $\mathbf{Z}$ are non-negative because all calculations do not change sign).

Now we can derive the predicted score for a user-item pair. Given user u will like item i during testing (i is in $\widetilde{\mathcal{O}}_u^+$):

$$\widehat{\mathbf{R}}_{u,i}^+ = \mathbf{P}_u^\top\mathbf{Q}_i = \widehat{\mathbf{R}}_{u,:}\mathbf{B}^\top\mathbf{A}\widehat{\mathbf{R}}_{:,i} = \widehat{\mathbf{R}}_{u,:}\mathbf{Z}^\top\widehat{\mathbf{R}}_{:,i} = \sum\widehat{\mathbf{R}}_{u,:}^\top\widehat{\mathbf{R}}_{:,i}\odot\mathbf{Z}^\top, \tag{4.5}$$

where $\widehat{\mathbf{R}}_{u,i}^+$ represents the predicted preference score from u to i given the ground truth for this user-item pair is positive; $\odot$ is the Hadamard product; and $\sum\mathbf{D}$ ($\mathbf{D}$ is a matrix) is to sum up all elements of $\mathbf{D}$. The intuitive way to interpret Equation 4.5 needs two steps: i) First, $\widehat{\mathbf{R}}_{u,:}^\top\widehat{\mathbf{R}}_{:,i} \in \{\sim 0, \sim 1\}^{M\times N}$ is the process to select *key user-item pairs* from a user candidate set $\mathcal{U}_i$ and an item candidate set $\mathcal{I}_u$ that help to indicate preference from u to i, where $\mathcal{U}_i$ are the users who like i in the training set, and $\mathcal{I}_u$ are the items u likes in the training set. Because $\mathcal{U}_i$ reveals characteristics of i and $\mathcal{I}_u$ reveals preferences of u, we can infer $\widehat{\mathbf{R}}_{u,i}$ based on the preferences of $\mathcal{U}_i$ toward $\mathcal{I}_u$, and elements with value ~ 1 in $\widehat{\mathbf{R}}_{u,:}^\top\widehat{\mathbf{R}}_{:,i}$ indicates these key user-item pairs. ii) Then, $\sum\widehat{\mathbf{R}}_{u,:}^\top\widehat{\mathbf{R}}_{:,i}\odot\mathbf{Z}^\top$ retrieves the preference scores of the selected key user-item pairs in $\mathbf{Z}$ and sums them up as $\widehat{\mathbf{R}}_{u,i}^+$.

To simplify Equation 4.5, we have **Assumption 2**: we assume the preference scores in $\mathbf{Z}$ for key user-item pairs follow the same distribution. The intuitive interpretation of this assumption is that similar users (and similar items) share similar feedback patterns. Or from another aspect, any positive user-item interaction can be inferred by other user-item relationships. Based on this assumption, we denote the expectation of the preference score in $\mathbf{Z}$ for a key user-item pair as $\mathbf{E}[\mathbf{Z}_+]$ ($\mathbf{E}[\mathbf{Z}_+]$ is non-negative). We can further derive Equation 4.5 as:

$$\widehat{\mathbf{R}}_{u,i}^+ = \sum\widehat{\mathbf{R}}_{u,:}^\top\widehat{\mathbf{R}}_{:,i}\odot\mathbf{Z} = (\sum\widehat{\mathbf{R}}_{u,:})(\sum\widehat{\mathbf{R}}_{:,i})\mathbf{E}[\mathbf{Z}_+]. \tag{4.6}$$

Theorem 1. *Given Assumption 1 and 2, MF produces uPO bias.*

Proof. Suppose user u will like items i and j during testing, and i is more popular than j, i.e., $(\sum \mathbf{R}_{:,i}) > (\sum \mathbf{R}_{:j})$, which is also equivalent to $(\sum \widehat{\mathbf{R}}_{:,i}) > (\sum \widehat{\mathbf{R}}_{:j})$, the difference between predicted preference scores of the two is:

$$\widehat{\mathbf{R}}^{+}_{u,i} - \widehat{\mathbf{R}}^{+}_{u,j} = (\sum \widehat{\mathbf{R}}_{u,:})((\sum \widehat{\mathbf{R}}_{:,i}) - (\sum \widehat{\mathbf{R}}_{:j}))\mathbf{E}[\mathbf{Z}_{+}] > 0, \tag{4.7}$$

which is to say for user u, even though both items are liked by u, the lower popularity of j makes it have a worse ranking than i in the recommendation list for u, i.e., MF produces uPO bias. □

Theorem 2. *Given Assumption 1 and 2, MF produces iPO bias.*

Proof. First, we formulate the expectation of the preference score of item i from matched users as:

$$\mathbf{E}[\widehat{\mathbf{R}}^{+}_{:,i}] = \mathbf{E}[(\sum \widehat{\mathbf{R}}_{u,:})](\sum \widehat{\mathbf{R}}_{:,i})\mathbf{E}[\mathbf{Z}_{+}], \tag{4.8}$$

where $\mathbf{E}[(\sum \widehat{\mathbf{R}}_{u,:})]$ is the expectation of the sum of predicted scores for a user, which is independent with items. Hence, given two items i, j, where i is more popular than j, we calculate the difference between expected scores of i and j:

$$\mathbf{E}[\widehat{\mathbf{R}}^{+}_{:,i}] - \mathbf{E}[\widehat{\mathbf{R}}^{+}_{:,j}] = \mathbf{E}[(\sum \widehat{\mathbf{R}}_{u,:})]((\sum \widehat{\mathbf{R}}_{:,i}) - (\sum \widehat{\mathbf{R}}_{:j}))\mathbf{E}[\mathbf{Z}_{+}] > 0, \tag{4.9}$$

which is to say that the lower popularity of j brings worse expected ranking for users who like j than i, i.e., MF produces iPO bias. □

4.2.4.2 *Existence of Bias in BPR*

In a similar fashion, we analyze the bias in BPR. Due to the pair-wise BPR loss, we cannot directly apply the same process in Section 4.2.4.1 to BPR. Thus, we need to first transform a BPR model to an MF one.

Because the pair-wise objective function in BPR is calculated by fixing a user and then computing the difference of predicted scores between one pair of positive and negative items, the output matrix $\widehat{\mathbf{R}}$ is not an approximated version of $\mathbf{R}$ as in MF. Instead, a well trained BPR model will have $\widehat{\mathbf{R}}$ where $\sigma(\widehat{\mathbf{R}}_{u,i} - \widehat{\mathbf{R}}_{u,j}) \approx 1$ given $\mathbf{R}_{u,i} = 1$ and $\mathbf{R}_{u,j} = 0$. Without loss of generality, we can remove the Sigmoid function, and assume that $\widehat{\mathbf{R}}_{u,i} - \widehat{\mathbf{R}}_{u,j} \approx a$ (a is a large positive number) for $\mathbf{R}_{u,i} = 1$ and $\mathbf{R}_{u,j} = 0$. Besides, we define a vector $\mathbf{x} \in \mathbb{R}^{N \times 1}$ to record the expectations of predicted scores for items not in the training set (i.e., $\mathcal{I} \setminus \mathcal{O}_u^+$) for each user as $\mathbf{x}_u = \mathbf{E}[\widehat{\mathbf{R}}_{u,\mathcal{I} \setminus \mathcal{O}_u^+}]$. Now, for user u, $\widehat{\mathbf{R}}_{u,:}$ is a vector consisting of values close to $\mathbf{x}_u$ and $\mathbf{x}_u + a$, denoted as $\sim \mathbf{x}_u$ values and $\sim (\mathbf{x}_u + a)$ values, where $\sim \mathbf{x}_u$ are for items in $\mathcal{I} \setminus \mathcal{O}_u^+$ and $\sim (\mathbf{x}_u + a)$ are for items in $\mathcal{O}_u^+$.

Next, we define a *centralized preference matrix* $\widetilde{\mathbf{R}} \in \{\sim 0, \sim 1\}^{N \times M}$ by subtracting $\mathbf{x}_u$ and dividing a for each user: $\widetilde{\mathbf{R}} = \frac{1}{a}(\widehat{\mathbf{R}} - \mathbf{J} \circ \mathbf{x})$, where $\mathbf{J} = \{1\}^{N \times M}$; and $\circ$ times elements of $\mathbf{x}$ to corresponding rows of $\mathbf{J}$. $\widetilde{\mathbf{R}}$ contains ~ 0 and ~ 1 values, which is exactly the same as the $\widehat{\mathbf{R}}$ in for MF. Meanwhile, $\widetilde{\mathbf{R}}$ maintains the item ranking orders for all users compared with $\widehat{\mathbf{R}}$ generated by BPR because the ranking is executed for each row of $\widehat{\mathbf{R}}$, thus, subtracting and dividing constants will not change the order of the elements in one row. Then, we have a new user latent representation matrix:

$$\widetilde{\mathbf{P}} = \frac{1}{a}(\mathbf{P} - \mathbf{J} \circ \mathbf{x}\mathbf{Q}^\top(\mathbf{Q}\mathbf{Q}^\top)^{-1}), \tag{4.10}$$

so that $\widetilde{\mathbf{P}}^\top \mathbf{Q} = \widetilde{\mathbf{R}}$. Now, we transform the original BPR model with latent matrices $\mathbf{P}$ and $\mathbf{Q}$ to a new model with $\widetilde{\mathbf{P}}$ and $\mathbf{Q}$, where the two models have the same recommendation results. Last, we can easily apply the same analysis process for MF to the new model to prove the existence of uPO and iPO bias in BPR.

4.2.5 Proposed Debiasing Methods

After empirically and theoretically studying popularity-opportunity bias in matrix factorization models, we next explore several approaches to alleviate this bias. Many methods [71, 70, 78, 79, 44, 80] have been studied for alleviating conventional popularity bias, which aim to promote the rankings of low-popularity items in the recommendations. These methods can also help promote the rankings of low-popularity items for matched users, which may mitigate the popularity-opportunity bias. However, this could also promote the rankings of low-popularity items for unmatched user, which could significantly degrade the overall recommendation utility. Hence, we explore debiasing methods that are designed explicitly for the popularity-opportunity bias.

Typically, there are three categories of methods: *pre-processing* [100], *post-processing* [101, 70], and *in-processing* [102, 24, 71] methods. Pre-processing approaches modify the training data so that models trained on the purified data are free of undesired issues (like bias). However, these kinds of algorithms are usually hard to design and may be ineffective since they cannot remove the algorithmic bias inherent in model architectures.

Hence, we focus here on the potential of post-processing and in-processing approaches to alleviate the bias. Concretely, we propose a simple but effective post-processing algorithm – Popularity Compensation (**PC** for short) and a regularization-based in-processing debiasing model (**Reg** for short).

4.2.5.1 Post-processing: Popularity Compensation

We begin by investigating a post-processing approach that modifies the predicted user-item preference matrix $\widehat{\mathbf{R}}$ by adding compensation to items with small popularity so that they have higher preference scores and thus higher ranking positions. We propose such a *popularity compensation* that follows three key guidelines:

- **Guideline 1**: Compensation should follow item popularity: items with lower popularity should be compensated more.
- **Guideline 2**: Compensation should follow user preferences: items with higher probabilities of being liked by a user should be compensated more.
- **Guideline 3**: Compensation should follow the value scale of each user: for a user who has a larger value scale for $\widehat{\mathbf{R}}_u$, item candidates for her should be compensated more.

Guideline 1 promotes low-popularity items to mitigate the bias. Guideline 2 ensures that items a user does not like but with low popularity will not be mistakenly promoted by the algorithm.

Guideline 3 makes sure that users with large value scales of predicted preference scores will have large compensation to items so that the algorithm is effective to all users.

Based on these guidelines, we propose the Popularity Compensation (PC) debiasing algorithm. Given a user u, we have the user-item interaction records in the training data $\mathbf{R}_{u,:} \in \{0,1\}^{1\times M}$, the interacted item set in the training data $\mathcal{O}_u^+$, and the predicted preference scores from u to items generated by MF or BPR $\widehat{\mathbf{R}}_{u,:} \in \mathbb{R}^{1\times M}$. The PC algorithm has three steps. First, we calculate the norm of predicted scores for user u by:

$$\mathbf{n}_u = \|(\widehat{\mathbf{R}}_{u,:} \odot (1 - \mathbf{R}_{u,:}))/(M - |\mathcal{O}_u^+|)\|_{\mathrm{F}}, \tag{4.11}$$

where we only consider the predicted preference scores to items that are not in the training data (by $\widehat{\mathbf{R}}_{u,:} \odot (1 - \mathbf{R}_{u,:})$) because the ranking is executed only on these un-interacted items and we should exclude the influence of items in the training set. Second, we calculate the popularity compensation score for one item i given u:

$$\mathbf{C}_{u,i} = \frac{1}{pop(i)} \cdot (\widehat{\mathbf{R}}_{u,i} \cdot \beta + 1 - \beta), \tag{4.12}$$

where there are two parts: $1/pop(i)$ is to achieve Guideline 1, and $(\widehat{\mathbf{R}}_{u,i} \cdot \beta + 1 - \beta)$ is to achieve Guideline 2 by using the predicted score as the indicator of user preference to i. $\beta \in [0,1]$ is a trade-off weight to control the ratio of predicted preference score in the compensation: larger β means higher ratio for predicted scores. Last, following Guideline 3, we need to scale the compensation to match the user preference score scale and add it to $\widehat{\mathbf{R}}_{u,i}$:

$$\widehat{\mathbf{R}}^*_{u,i} = \widehat{\mathbf{R}}_{u,i} + \alpha \cdot \mathbf{C}_{u,i} \cdot \mathbf{n}_u/\mathbf{m}_u, \tag{4.13}$$

where $\widehat{\mathbf{R}}^*_{u,i}$ is the new preference score from u to i; $\mathbf{m}_u = \|(\mathbf{C}_u \odot (1 - \mathbf{R}_u))/(M - |\mathcal{O}_u^+|)\|_{\mathrm{F}}$ is the norm of compensation scores of u excluding those for items in $\mathcal{O}_u^+$; $\mathbf{n}_u/\mathbf{m}_u$ is to normalize the compensation scores based on Guideline 3; and α is the trade-off weight for the whole PC algorithm. With new preference scores for all candidate items, we can provide a debiased ranking list for u.

4.2.5.2 In-processing: Regularization

In this section, we introduce a regularization-based in-processing way to debias. The proposed method is inspired by previous work enhancing equal opportunity based recommendation fairness for different item groups [24], which try to decrease the correlation between item group variable and model output scores to achieve fairness. We adapt this idea to the context of alleviating the popularity-opportunity bias by decreasing the correlation between item popularity and model output scores.

We adopt the square of the *Pearson correlation coefficient* between predicted preference scores for positive user-item pairs and corresponding item popularity as a regularization term, and mitigate the bias by minimizing this regularization term together with the recommendation error:

$$\min_{\mathbf{\Theta}} \mathcal{L}_{Rec} + \gamma PCC(\widehat{\mathbf{R}}_{+}, pop(\mathcal{I}))^2, \tag{4.14}$$

where $\mathcal{L}_{Rec}$ is the loss of recommendation models as shown in Section 4.2.2; $PCC(\widehat{\mathbf{R}}_{+}, pop(\mathcal{I}))$ computes Pearson correlation coefficient between predicted scores for positive user-item pairs and the popularity of corresponding items; and γ is the trade-off weight.

The proposed Reg is designed to decouple the item popularity with the model preference predictions to alleviate the popularity-opportunity bias. However, minimizing the correlation between item popularity and the predicted score is a challenging task because item popularity is continuous and unevenly distributed. Thus, a decrease in recommendation utility is expected when we aim to reduce the bias significantly by Reg.

4.2.6 Experiments

In this section, we investigate the impact of the proposed debiasing methods w.r.t. recommendation utility and debiasing performance, compared with biased base models and baselines of removing conventional popularity bias. Then, we illustrate these impacts over the same examples from Figure 4.1 to better understand their effects. Last, we study the impact of hyper-parameters on the two proposed debiasing algorithms.

4.2.6.1 Experimental Settings

Data

We use the same four datasets introduced in Section 4.2.3. We compare the biased models MF and BPR with their debiased versions: **MF-PC** and **BPR-PC** denote the debiased versions based on the Popularity Compensation algorithm, while **MF-Reg** and **BPR-Reg** denote the debiased versions based on the regularization-based model. Besides, we also include two baselines which are designed to remove the conventional popularity bias for comparison, in other words, models forcing items of different popularity to receive similar rankings for all users.

Baselines

The first baseline removes the conventional popularity bias by weighted matrix factorization [44], which assigns weights to training samples in the recommendation loss in Equation 4.1 and Equation 4.2 based on the popularity of involved items – items of low popularity will be assigned with high weights to promote the predicted scores for them. The weight for item i is chosen as $w_i \propto 1/pop(i)^e$, where e is an exponent to control the strength of the debiasing effect. We denote the corresponding versions with MF and BPR as base models as **MF-weight** and **BPR-weight**.

The second baseline removes the conventional popularity bias by rescaling the training data [80], which multiplies rescaling values to the binary training samples based on the popularity of involved items to uniformly promote the scores of low-popularity items. Then, it trains the vanilla MF or BPR models on the rescaled training data. The rescaling values are determined by the same way as the weights in the weighted model: $w_i \propto 1/pop(i)^e$ with the exponent e to control the debiasing strength. We denote the corresponding baselines as **MF-rescale** and **BPR-rescale**.

Because the two conventional popularity bias based baselines uniformly promote low-popularity items in recommendations, the popularity-opportunity bias is expected to be reduced as well. However, these baselines modify the recommendations without considering the potential user preferences as the two proposed debiasing models do. Hence, it is also expected that the two baselines will decrease the recommendation utility significantly.

Metrics

We evaluate user-side and item-side bias for all the models using the metrics introduced in Section 4.2.3.1, and compare the recommendation utility based on *NDCG@k* with $k = 20$ and 50.

Reproducibility

All models are implemented in Tensorflow [59] and optimized by Adam [60] algorithm. For all models and all datasets, we fix the latent dimension as 64, set the learning rate as 0.001, the negative sampling rate as 2, and set the mini-batch size as 1024. Then we tune hyper-parameters for all models by grid search over validation sets. More specifically, for post-processing methods MF-PC and BPR-PC, we directly apply the PC algorithm on the outputs from MF and BPR, and tune α in $[0.1, 1.5]$ with step 0.1, tune β in $[0.0, 1.0]$ with step 0.1. For in-processing models, we tune γ in $\{1e2, 1e3, 1e4, 1e5, 1e6, 1e7\}$. Note that for all the debiasing models, there is a trade-off between recommendation utility and debiasing performance. Hence, we explore hyper-parameters that minimize the bias metrics while preserving an acceptable utility.

4.2.6.2 RQ1: Comparing Debiasing Performance

We begin in Table 4.3 with a comprehensive study on four datasets for all MF based models (including original biased model: MF; debiased baselines designed for conventional popularity bias: MF-weight and MF-rescale; and the proposed debiased ones designed for the popularity-opportunity bias: MF-Reg and MF-PC). Here, we walk through the key findings:

First, we investigate the recommendation utility of the two proposed debiasing models and the two baselines compared with the original MF. Typically there is a trade-off between recommendation utility and debiasing effectiveness, and we observe such a trade-off here as well. Focusing on the $NDCG$ columns for different values of k, we see that in all cases there is a drop in recommendation utility between original MF and its debiased versions (proposed MF-PC and MF-reg, and baselines for conventional popularity bias MF-weight and MF-rescale). Then, by comparing

Table 4.3: Evaluation of recommendation utility (*NDCG@k*), uPO bias (*PRU*), and iPO bias (*PRI*) for MF based models on four datasets. * indicates the correlation coefficients are statistically significant for $p < 0.01$. Reprinted with permission from [8].

		$NDCG@k$		PRU	PRI
		@20	@50		
ML1M	MF	0.2726	0.2930	0.8350	0.9799*
	MF-weight	0.1484	0.1793	0.4845	0.6407*
	MF-rescale	0.1361	0.1658	0.4365	0.6936*
	MF-Reg	0.1492	0.1720	0.1910	0.5916*
	MF-PC	0.1435	0.1980	0.4552	0.5594*
Ciao	MF	0.0717	0.0934	0.5420	0.3625*
	MF-weight	0.0447	0.0675	0.3174	0.3293*
	MF-rescale	0.0425	0.0608	0.3219	0.2526*
	MF-Reg	0.0497	0.0639	0.2881	0.1905*
	MF-PC	0.0647	0.0845	0.3073	−0.0150
Epinions	MF	0.0693	0.0938	0.6840	0.5351*
	MF-weight	0.0349	0.0526	0.3453	0.2341*
	MF-rescale	0.0343	0.0509	0.3678	0.2182*
	MF-Reg	0.0386	0.0516	0.2175	0.2251*
	MF-PC	0.0605	0.0848	0.3549	−0.0415
App	MF	0.1026	0.1359	0.5667	0.6089*
	MF-weight	0.0388	0.0596	0.3552	0.2334*
	MF-rescale	0.0384	0.0583	0.3350	0.2147*
	MF-Reg	0.0439	0.0599	-0.0571	0.2207*
	MF-PC	0.0965	0.1280	0.3527	−0.0487

Table 4.4: Evaluation of recommendation utility, uPO bias (*PRU*), and iPO bias (*PRI*) for BPR based models on ML1M datasets. * indicates the correlation coefficients are statistically significant for $p < 0.01$. Reprinted with permission from [8].

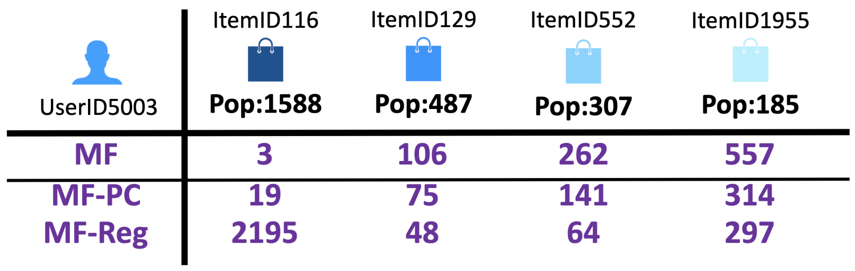
		$NDCG@k$		PRU	PRI
		@20	@50		
ML1M	BPR	0.2983	0.3220	0.7793	0.9688*
	BPR-weight	0.1458	0.1757	0.5121	0.6249*
	BPR-rescale	0.1446	0.1784	0.4349	0.6064*
	BPR-Reg	0.1660	0.1769	0.2862	0.5633*
	BPR-PC	0.2308	0.2711	0.5712	0.5080*

the four debiasing models, we observe that the MF-PC can preserve recommendation utility more effectively than the others, and MF-Reg performs similarly to the two baselines. Given these utility results, if we can observe lower bias by the proposed models, we can conclude that proposed models are able to achieve more effective debiasing performance with recommendation utility preserved.

Hence, we next study the impact different approaches have on reducing user-side (uPO) bias. Let's focus on the PRU column (which measures the popularity-rank correlation for users: high values correspond with high bias). We observe that all debiasing algorithms can significantly reduce PRU compared with the original MF. And these findings hold across all four datasets. Comparing the four debiasing models, in general, MF-Reg is able to improve PRU more significantly, and MF-PC performs similarly to MF-weight and MF-rescale. It may be because MF-Reg reduces the correlation between popularity and model predictions, which can effectively shuffle the rankings of matched items for each user. While the other three debiasing models are to re-rank items based on heuristics, which are expected to keep the original rankings to some degree. Another reason of less effective performance of MF-PC compared with MF-Reg is that MF-PC provide much better recommendation utility than MF-Reg, and a lower PRU is expected if we strengthen the debiasing effect for MF-PC.

Third, we investigate the impact different approaches have on reducing item-side (iPO) bias. Here, we focus on the PRI column (which measures the popularity-rank correlation for items: high values correspond with high bias). All four debiasing methods can improve PRI against original MF. Comparing the four debiasing methods, the PC algorithm is much more effective, which can reduce the PRI to a great extent. Although with a smaller improvement, the proposed Reg algorithm is more effective than the two baselines for removing conventional popularity bias.

Similar results can be observed from experiments on BPR and its debiasing variations (the results on ML1M dataset is shown in table 4.4). Based on these results, we can draw the conclusion that the proposed two debiasing algorithms can indeed mitigate both uPO and iPO bias, with the post-processing PC algorithm preserving recommendation utility better than the in-processing Reg approach. Comparing the two proposed methods with the two baselines, we can conclude that both proposed debiasing methods can alleviate the popularity-opportunity bias and preserve



UserID5003	ItemID116 Pop:1588	ItemID129 Pop:487	ItemID552 Pop:307	ItemID1955 Pop:185
MF	3	106	262	557
MF-PC	19	75	141	314
MF-Reg	2195	48	64	297

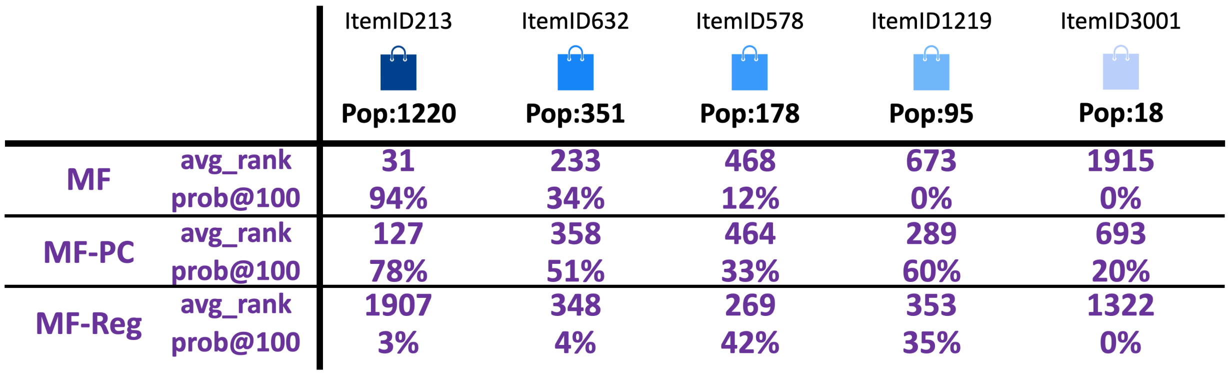
Figure 4.3: Case study: ranking results for items that user 5003 in ML1M will like by different models. Reprinted with permission from [8].

the recommendation utility more effectively than baseline methods designed for removing the conventional popularity bias.

4.2.6.3 RQ2: Case Study

To further understand the effects of the proposed models, we compare the recommendation results of the debiasing algorithms and the base model MF for the same examples shown in Figure 4.1 (results for BPR based models show similar pattern). First, Figure 4.3 shows the ranking results for matched items of user 5003 by different models (recall this is based on the ML1M dataset). By comparing the debiasing models with their original base model, we can see that both debiasing algorithms are able to promote the rankings for unpopular items. The PC algorithm promotes unpopular items and maintains relatively high rankings for popular ones, meaning that it is fairly effective at overcoming popularity-opportunity bias. But the Reg model cannot preserve high rankings for these popular items, giving insight into the challenges Reg faced in Table 4.3.

Next, we show the results from the perspective of iPO bias in Figure 4.3, where we compare the recommendation results for five items by different models. We can see that compared with MF, the debiasing models promote the less popular items to have better ranking results. For example, MF-PC decreases the recommendation probability (assuming 100 items are recommended for each user) for item213 from 94% to 78%, but increases the probabilities for items with lower popularity, especially for item1219 and item3001, which do not have any chance to be exposed to users who like them by MF, but have 60% and 20% probabilities by MF-PC. Similar in spirit to our previous



		ItemID213 Pop:1220	ItemID632 Pop:351	ItemID578 Pop:178	ItemID1219 Pop:95	ItemID3001 Pop:18
MF	avg_rank	31	233	468	673	1915
	prob@100	94%	34%	12%	0%	0%
MF-PC	avg_rank	127	358	464	289	693
	prob@100	78%	51%	33%	60%	20%
MF-Reg	avg_rank	1907	348	269	353	1322
	prob@100	3%	4%	42%	35%	0%

Figure 4.4: Case study: average ranking results of items for matched users in ML1M by different models. Reprinted with permission from [8].

observation, the Reg also increases rankings for unpopular items but cannot preserve rankings for popular items.

4.2.6.4 RQ3: Impact of Hyper-parameters

Finally, we study the impact of the hyper-parameters. Due to the space limitation, we only show the conclusions based on the experiments here but do not show the detailed results.

For the PC algorithm, we have two hyper-parameters: α controls the ratio of popularity compensation, with larger values meaning more weight to the compensation; β controls the strength of predicted preference scores on the popularity compensation, with larger values meaning more weight for predicted preference scores. Based on our experimental results, we observe that as α increases, recommendation utility decreases and the debiasing performance is being improved. This result is because a larger α means a higher ratio of the popularity compensation in the final output, leading to worse recommendation utility but less bias. For β, we observe that the recommendation utility keeps increasing, and the debiasing effect is first improved and then degraded as β increases. The reason behind this is that reasonable β can indicate user preferences and help calculate accurate compensation scores, but higher β makes preference scores dominate the compensation lead to a decrease in the debiasing performance. For the Reg algorithm, as γ increases, the recommendation utility is reduced, and the debiasing performance first improves then decreases due to overfitting.

4.2.7 Summary

In this section, we conduct a three-part study to investigate popularity-opportunity bias in matrix factorization based models: i) we empirically show the vulnerability of two matrix factorization models to the bias by a data-driven study on four datasets; ii) we theoretically show how these two models inherently produce the popularity-opportunity bias on both user and item sides; and iii) we explore the potential of in-processing and post-processing approaches to alleviate the bias. Experiments on four datasets validate the debiasing effectiveness of both proposed methods over debiasing baselines designed for conventional popularity bias.

4.3 Analyzing and Mitigating Mainstream Bias

Next, we switch our focus from items to users, and study how users with different preferences are treated differently by the recommender algorithms. More specifically, we aim to investigate the mainstream bias on users in recommender systems that users with mainstream preference can receive more accurate recommendations at the expense of users with non-mainstream preference receiving low-quality recommendations. And we propose a local fine tuning method to improve the recommendation performance for niche users and mitigate this mainstream bias.

4.3.1 Introduction

Recommender systems play an increasingly important role in connecting users to interesting items to alleviate the information overload issue. Most recommendation systems, including those based on classic linear models [29, 103, 27] and recent neural-network models [17, 92, 104], predict user preference and provide recommendations based on Collaborative Filtering (CF). The main idea is to estimate the preference from a user to an item depending on the attitudes from other similar users to the item. By finding other users with similar interests as the target user, these CF approaches have demonstrated strong recommendation performance.

Naturally, the quality of recommendations critically relies on how easily the model can find similar users for a target user. A *niche (or "indie") user* who prefers items that are out of the mainstream may have few if any nearby users, resulting in poor recommendations. In contrast, a

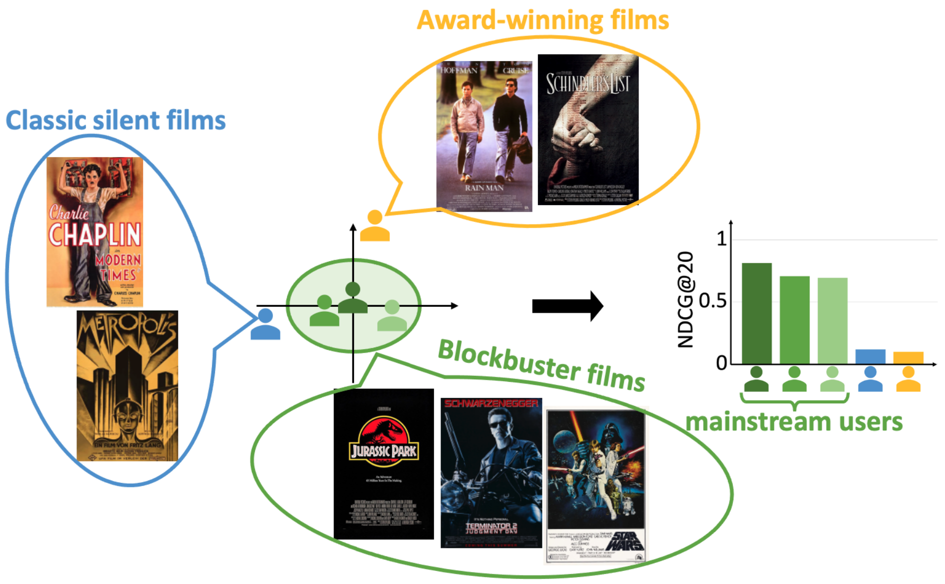


Figure 4.5: Mainstream vs. niche users in MovieLens data. Reprinted with permission from [9].

mainstream user who shares interests with many other users will likely receive many high-quality recommendations. To illustrate, Figure 4.5 shows three mainstream users and two niche users from the MovieLens dataset [96], who are identified based on a method we propose. All three mainstream users share similar preferences for blockbuster films: the recommendations from a recent variational autoencoder (VAE) model [92] result in high NDCG@20. In contrast, we see that for the two niche users – one of whom prefers classic silent films, while one prefers award-winning films of the late 80s/early 90s – the resulting recommendation quality is quite poor. Indeed, we find similar patterns for mainstream vs. niche users across multiple datasets (including Yelp [105] and Epinions [97]) and for different models (including matrix factorization [29], BPR [27], and local collaborative autoencoders [88]).

This *mainstream bias* – **the tendency for recommendation models to favor mainstream users over niche users** – is a critical challenge for the ongoing success of recommendation systems. But how do we identify mainstream users vs. niche ones? What impact does the degree of mainstream-ness have on recommendation utility? And can we develop methods to ameliorate this mainstream bias? Can we improve the recommendation utility for users of low mainstream levels while preserving or even increasing the utility for mainstream users at the same time? Toward answering these research questions, our work is organized around three key thrusts:

First, to understand the impact of mainstream bias on recommendation, we first propose to identify a mainstream score to indicate the mainstream level for each user. We explore four different methods based on outlier detection techniques to compute the mainstream scores for users, including similarity-based, density-based, distribution-based, and DeepSVDD-based approaches. While all provide good ability to assess mainstream-ness, we empirically find that the DeepSVDD-based method is most effective for distinguishing mainstream and niche users.

Second, based on this assessment of each user's mainstream level, we empirically show that conventional recommendation models do indeed produce severe mainstream bias. We find that after grouping users based on their mainstream scores, the users with highest mainstream level receive recommendation utility more than twice larger than users with the lowest mainstream level.

Finally, we explore how to mitigate such mainstream bias. We introduce both global methods and local methods to improve the recommendation quality for niche users. Global methods achieve this by learning a single model that promotes the importance of niche users during model training. Concretely, we propose i) a Distribution Calibration method (DC) to debias by data augmentation; and ii) a Weighted Loss model (WL) to debias by adding weights to the loss function. On the other hand, local methods aim to customize specialized models for different users so that niche users receive better recommendations from their customized models. For this, we propose the Local Fine Tuning algorithm (LFT) that improves the model utility for every user by fine tuning a global base model with partial data that is most informative for this user. Unlike global methods and other local baselines which maintain a trade-off between the utility for mainstream and niche users, we find that LFT improves the utility for both of them.

In sum, we make the following contributions: i) To analyze the impact of mainstream bias, we explore four different methods to calculate mainstream scores for users based on outlier detection techniques, followed by empirical studies comparing the effectiveness of these approaches and further showing the severe bias produced by conventional recommendation models; ii) We introduce global and local methods for bias mitigation, where for global method, we propose the Distribution Calibration model (DC) and the Weighted Loss model (WL), for local method, we propose the Local Fine Tuning algorithm (LFT); iii) Extensive experiments show that all proposed solutions are able to improve utility for niche users, while LFT is more effective and can preserve or even improve the utility for mainstream users at the same time.

4.3.2 Analyzing Mainstream Bias

In this section, we begin with the first research question: what is the impact of the mainstream bias on recommendation? To answer this, we first formalize the problem and introduce four approaches based on outlier detection techniques for identifying mainstream and niche users to analyze the mainstream bias. We then conduct experiments to investigate the impact the degree of mainstream-ness has on the quality of recommendations.

4.3.2.1 Preliminaries

Formally, we have a set of N users as $\mathcal{U} = \{1, 2, \ldots, N\}$ and a set of M items as $\mathcal{I} = \{1, 2, \ldots, M\}$. We denote the set of implicit feedback from users to items as $\mathcal{O} = \{(u, i)\}$ where $u \in \mathcal{U}$ indexes one user, and $i \in \mathcal{I}$ indexes one item. We use this feedback set as the training data to train a recommendation model and provide recommendations to users. For a user u, we use a binary vector of size M, denoted as $\mathbf{O}_u \in \{0, 1\}^M$, to represent the feedback record vector of user u, with 1 representing u likes the corresponding item. During evaluation, the trained model provides a ranked list of items for each user as recommendations, and we evaluate the recommendation list based on the ranking positions of positive items in a testing set for every user. Many ranking evaluation metrics can be used, such as NDCG@k and recall@k [92], which are typically averaged over all users.

4.3.2.2 Evaluating Mainstream Level of Users

Since the typical way to evaluate a recommender system is to average the recommendation utility over all users, the performance difference among users is ignored. So, to analyze mainstream bias, we first need to divide users into subgroups based on their mainstream levels, and then compare the recommendation utility across these subgroups. Therefore, we aim to calculate a mainstream score for each user to indicate the mainstream level of the user. A large mainstream score means that the user is more likely to be a mainstream user. Then, we can analyze the mainstream bias by dividing users into subgroups based on their mainstream scores and comparing the utility across subgroups.

The problem of assessing a user's mainstream level can be easily turned to an outlier detection problem: we consider niche users who have different preferences from the majority as the outlier samples to detect. So, inspired by various outlier detection techniques [106], we explore four approaches and want to determine which one performs the best for assessing mainstream level.

Similarity-based approach

First, we propose a similarity-based approach to evaluate a user's mainstream level. The main intuition is that mainstream users should have more similar users sharing similar feedback records, while niche users have fewer similar users. Thus, we first calculate the user-user similarity by Jaccard similarity for all user-user pairs. The similarity between users u and v is denoted as $J_{u,v}$. Then, we use the average similarity between a user u and other users as the mainstream score of u:

$$MS_u^{sim} = \sum_{v \in \mathcal{U} \backslash u} J_{u,v}/(N-1). \tag{4.15}$$

Density-based approach

The next approach we propose is based on the density-based outlier detection method, which determines whether one sample is an outlier by investigating the density of the sample's neighbors. Here, we propose to directly apply the well-known local outlier factor (LOF) algorithm [107] to the user feedback records to identify niche users. The LOF algorithm outputs the local outlier factor value for each user, which indicates an outlier if its value is large. Thus, we add a negative sign to the local outlier factor value as the mainstream score for a user u:

$$MS_u^{den} = -LOF(u). \tag{4.16}$$

Distribution-based approach

In the third method, we first generate a distribution vector $\mathbf{d}$ that captures the probability of each item being liked by users. We assume the probability is based on a binomial distribution and the distribution vector is calculated by averaging the feedback records of all users. Then, we calculate the mainstream score for u by the Cosine similarity between the feedback record vector $\mathbf{O}_u$ of u and the distribution vector $\mathbf{d}$. Given a function $cos(\cdot,\cdot)$ to compute Cosine similarity between two vectors, we calculate the mainstream score:

$$MS_u^{dis} = cos(\mathbf{O}_u, \mathbf{d}). \tag{4.17}$$

DeepSVDD-based approach

Last, we apply the recent deep learning based outlier detection algorithm – deep support vector data description (DeepSVDD) [108] – to identify niche users. DeepSVDD attempts to map most of the data samples (belonging to one class) into a hypersphere by neural networks and considers the samples far from the center of the hypersphere as outliers. Moreover, since in a recommender system, there can be more than one mainstream preference, resulting in more than one user class in terms of preference. Hence, we further replace the multi-layer perceptron in the mapping component in the original DeepSVDD to a mixture-of-experts structure [65], so that the model can have different mapping functions for different classes to handle the multi-class situation more effectively. In our experiments, we set a 2-layer perceptron of size (400, 300) as one expert component and adopt 10 experts in total. After the mapping, we have a vector $\mathbf{c}$ in the new hyper-space representing the center of the hypersphere covering the majority of users, and we also have a vector $DeepSVDD(\mathbf{O}_u)$ to represent user u in the mapped hyper-space. For a user u, we use the negative distance from $DeepSVDD(\mathbf{O}_u)$ to center $\mathbf{c}$ as the score:

$$MS_u^{deep} = -\|DeepSVDD(\mathbf{O}_u) - \mathbf{c}\|_{\mathrm{F}}. \tag{4.18}$$

Table 4.5: NDCG@20 of different subgroups determined by different mainstream level evaluation approaches. Reprinted with permission from [9].

	User subgroups of different mainstream levels				
	low	med-low	medium	med-high	high
Similarity	0.2056	0.2666	0.2915	0.3563	0.4566
Density	0.2219	0.2658	0.2789	0.3431	0.4669
Distribution	0.2059	0.2666	0.2862	0.3408	0.4771
DeepSVDD	0.2092	0.2642	0.2832	0.3368	0.4831

Table 4.6: Niche users classifying accuracy of four approaches. Reprinted with permission from [9].

	Similarity	Density	Distribution	DeepSVDD
Accuracy	0.66	0.31	0.68	0.73

After calculating the mainstream scores for all users, we sort users by the scores and divide them into subgroups. We then can compare the average utility across subgroups: if the subgroups with high mainstream scores have higher utility than subgroups with lower scores, then severe mainstream bias is observed.

4.3.2.3 Empirical Studies

Given these four approaches to evaluate mainstream-ness of users, we conduct experiments to answer two questions: i) do commonly used recommendation models produce mainstream bias? and ii) how effective are proposed approaches to identify niche users?

Recommendation models produce mainstream bias

To answer the first question, we conduct experiments with real-world datasets and state-of-the-art recommendation models. More specifically, we first run a VAE [92] on the MovieLens 1M dataset [96]. Then, we apply the introduced four approaches to calculate mainstream scores for all users. Last, we sort users based on calculated mainstream scores in non-descending order and divide them into five subgroups with equal size. Note that we also run experiments with other models including MF [29], BPR [27], and LOCA [88], and other datasets including Yelp [105] and Epinions [97]. These experiments show similar patterns. Code and data can be found at `https://github.com/Zziwei/Measuring-Mitigating-Mainstream-Bias`.

The average NDCG@20 for subgroups corresponding to different mainstream-ness measuring ways are shown in Table 4.5, where we denote the first 20% of users with lowest mainstream scores as users of 'low' mainstream level, the subgroup of 20%-40% users as users of 'med-low' mainstream level, and so on for 40%-60% ('medium'), 60%-80% ('med-high'), and 80%-100% ('high'). From the table, we can observe that all four proposed approaches show a similar pattern – users with larger mainstream scores receive higher NDCG@20. For example, for all four bias measuring cases, the average NDCG@20 of 'high' mainstream level users is more than twice larger than those of 'low' mainstream level users. This result reveals that all proposed approaches are able to identify niche users who are under-served by the recommendation model and severe mainstream bias is produced by the recommendation model.

Proposed approaches effectively identify niche users

Next, we aim to quantitatively evaluate the effectiveness of identifying niche users of these four introduced approaches. To do this, we need to have a dataset with ground-truth labels of niche users, which is not easy to get from real-world systems. Hence, we use synthetic data to compare the proposed approaches. To generate the synthetic data, we assume we have four item groups, each of which includes 250 items. For each item group, we randomly generate 100 sets of Gaussian distribution parameters. Based on the Gaussian distribution parameters, we randomly generate a 100-dimension embedding for each of the 250 items in this group. We consider the first two item groups as mainstream items and the other two groups as non-mainstream items. Then, we create two user groups of size 800 as the mainstream users, where the first user group likes the first item group and the second user group likes the second item group. We also create two user groups of size 200 as niche users, where each of them likes one of the non-mainstream item groups. We use the Gaussian distribution parameters of corresponding item groups to generate user embeddings for these user groups. Last, we generate the user-item interaction data by randomly sampling from the completed user-item relevance matrix, which is from the dot product of the generated user and item embeddings.

Given this setup, we run the four proposed approaches to identify niche users in this dataset. Here, we can formalize a binary classification task, where we consider the 400 users with lowest mainstream scores from each approach as the predicted niche users, and the 400 users from the last two generated user groups are the ground-truth labels. The classification accuracy is shown in Table 4.6, from which we can observe that with the help of deep learning techniques, the DeepSVDD-based approach performs the best. The next best approaches are similarity-based and distribution-based approaches, which perform similarly because both of them rely on the similarity calculation between users by their feedback records. Density-based approach performs the worst, which may be because the LOF algorithm cannot work effectively for high-dimensional and sparse data. As a result, we adopt the DeepSVDD-based approach as the best choice to analyze mainstream bias.

4.3.3 Mitigating Mainstream Bias

In the previous section, we observed a significant utility gap between mainstream and niche users. The question then is: can we mitigate this mainstream bias by increasing the utility for niche users? In this section, we explore both global and local methods to mitigate mainstream bias. Global methods learn a single model with the importance of niche users being promoted during model training. Local methods, on the other hand, train customized local models for different users. In the following, we first detail these two different solution directions and then empirically test them.

4.3.3.1 Global Methods

One of the reasons mainstream bias is induced is that a model trained based on a loss function averaging all users tends to focus more on how to accurately predict for mainstream users while overlooking niche users so that it can minimize the loss function more effectively. Therefore, a straightforward way to debias is to keep the model structure the same but increase the importance of niche users in the model training process. Because this type of method uses one model globally for all users, we call this a global method. In the following, we introduce two different methods belonging to this category: a Distribution Calibration method and a Weighted Loss method.

Distribution Calibration Method (DC)

This first method is a data augmentation based approach, whose main intuition is to generate synthetic users similar to existing niche users so that these niche users become mainstream in the training dataset. To achieve this, we adapt the Distribution Calibration method [109] for few-shot learning to the recommendation task. In the original paper [109], the distributions of few-shot classes are calibrated by transferring statistics from similar classes with abundant data. Then, synthetic examples of few-shot classes can be sampled based on the calibrated distributions to augment the training data. In a recommendation task, we can consider each niche user as a single class. Then, in a similar way, we can calibrate the distribution for each niche user by transferring statistics from other similar users and generate synthetic users based on the calibrated distribution.

Specifically, we first identify niche users by any of the proposed approaches in Section 4.3.2.2. For example, we consider the 50% users with lowest mainstream scores from DeepSVDD-based approach as niche users. Then, for one niche user u, we fetch similar users to u, and have the calibrated distribution vector $\mathbf{p}_u$ of u:

$$\mathbf{p}_u = \alpha \mathbf{O}_u + (1-\alpha)\frac{1}{|\mathcal{N}_u|}\sum_{v\in\mathcal{N}_u}\mathbf{O}_v, \tag{4.19}$$

where $\mathcal{N}_u$ is the set of similar users to u in terms of Jaccard similarity on feedback records; and $0 \leq \alpha \leq 1$ is a hyper-parameter to control the importance of original feedback of u in the resulting distribution. Last, we sample synthetic users based on $\mathbf{p}_u$ for u. Given a budget for synthetic users (we use the total number of real users in this chapter), the number of synthetic users for each niche user is proportional to the reciprocal mainstream score of the user. At the end, a model trained by such an augmented dataset can mitigate the mainstream bias and improve utility for niche users.

Weighted Loss Method (WL)

Instead of expanding the training data by synthetic users, another way to promote the importance of niche users during model training is to directly increase the weights of niche users in the loss function. Take the VAE model as an example, we can have a weighted loss for the model:

$$\mathcal{L}_{WL} = \sum_{u \in \mathcal{U}} w_u \cdot \mathcal{L}_{VAE}(u), \quad w_u \propto (\frac{1}{MS_u})^{\beta}, \tag{4.20}$$

where $\mathcal{L}_{VAE}(u)$ is the original VAE loss for user u; w_u is the weight for user u, which is proportional to $(\frac{1}{MS_u})^{\beta}$; and $\beta \geq 0$ is a hyper-parameter to control the strength of debiasing: larger β means stronger debiasing strength, and 0 means no debiasing at all. By this weighted loss, we can promote the importance of niche users: a user with lower mainstream score can be promoted more in the loss function and thus receive better utility after debiasing.

4.3.3.2 Local Method

Although the two introduced global methods are able to improve the utility for niche users, one major drawback is that there can be a trade-off between the utility of mainstream users and niche users in these global methods. In other words, the global methods increase utility for niche users but decrease utility for mainstream users at the same time. Due to the limited expression capability of one single recommendation model, these global methods cannot support high utility for so many users with different or even opposite preferences. Hence, another direction to tackle the mainstream bias problem is to customize local models for different types of users instead of applying the same global model to all users.

Local recommendation methods have been studied in prior works [88, 89, 90, 91]. The main idea is to first select anchor users and train specialized anchor models for each of the anchor users. Then, during inference, given a user, we can customize a local model for this user by ensembling anchor models based on the relationship between the target user and anchor users. Although these local recommendation algorithms are not specifically designed for addressing mainstream bias, we empirically find that they can improve utility for niche users. Hence, in this section, we move

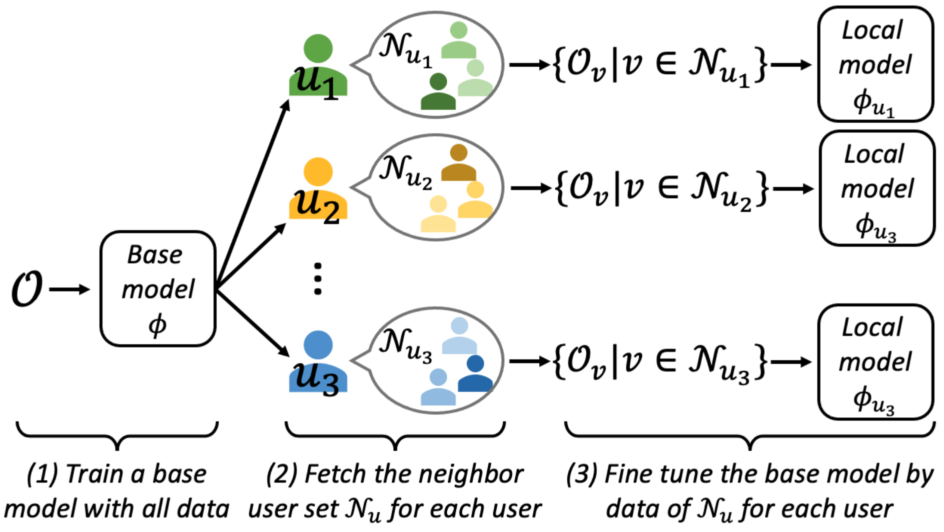

$\mathcal{O}$
Base model ϕ
u_1
u_2
u_3
$\mathcal{N}_{u_1}$
$\mathcal{N}_{u_2}$
$\mathcal{N}_{u_3}$
$\{\mathcal{O}_v | v \in \mathcal{N}_{u_1}\}$
$\{\mathcal{O}_v | v \in \mathcal{N}_{u_2}\}$
$\{\mathcal{O}_v | v \in \mathcal{N}_{u_3}\}$
Local model ϕ_{u_1}
Local model ϕ_{u_3}
Local model ϕ_{u_3}
(1) Train a base model with all data
(2) Fetch the neighbor user set $\mathcal{N}_u$ for each user
(3) Fine tune the base model by data of $\mathcal{N}_u$ for each user

Figure 4.6: The proposed Local Fine Tuning method. Reprinted with permission from [9].

further based on these local recommendation models to propose a Local Fine Tuning (LFT) method to effectively mitigate the mainstream bias, whose goal is to increase the utility for niche users with the utility for mainstream users preserved or even increased.

Local Fine Tuning

The fundamental motivation is that feedback data from very different users may not be helpful or can even play negative roles when learning a model for one or a small group of similar users. Moreover, niche users can be very different from the majority incurring poor utility. Thus, we consider recommending for each user as an independent task requiring a unique local model. And for each user, we propose to learn the local model with partial data that is selected to be most useful for serving this user.

The proposed LFT is illustrated in Figure 4.6. Concretely, we first assume we have a global base model ϕ which is trained by the entire dataset as the step (1) in Figure 4.6, such as an ordinary VAE model. Then, demonstrated as the step (2): for a target user u, we fetch the neighbor users $\mathcal{N}_u$ (including u herself) that are similar to u in terms of preference, and create a sub-dataset $\mathcal{O}_{\mathcal{N}_u} = \{\mathcal{O}_v | v \in \mathcal{N}_u\}$ only containing feedback data of neighbor users. Last, during inference, for a target user u, we further train the base model ϕ by the sub-dataset $\mathcal{O}_{\mathcal{N}_u}$ to fine tune the model as step (3) in Figure 4.6, so that it can provide accurate prediction for u without influence from irrelevant users. We denote the local model after fine tuning for u as ϕ_u. In this way, niche users can receive better utility by the local models since influence from mainstream users and other niche users with different preference is eliminated. Furthermore, mainstream users also benefit from their local models since they also suffer from the influence of niche users and other mainstream users with different preferences.

Now, the key question is: how to find the neighbor users $\mathcal{N}_u$ of a user u so that the feedback data of them $\mathcal{O}_{\mathcal{N}_u}$ can help improve the fine tuning effectiveness and eliminate the influence from irrelevant users? A naive way is to fetch the users with highest similarity (e.g., Jaccard or Cosine similarity) based on feedback records. However, the limitation is that the similarity between users based on discrete and sparse feedback records does not consider the latent relationship between users. For example, if a user only likes item A and another user only likes item B, the similarity between them will be 0 based on their feedback records. However, if item A and B are very similar, then the ground truth similarity between them should be high. Therefore, the naive neighbor user searching method could omit important neighbor users. Instead, we propose to fetch the neighbor users based on the similarity between the calibrated distributions of users introduced in Section 4.3.3.1. More specifically, we calculate the Cosine similarity between users by their calibrated distributions $\mathbf{p}_u$ calculated by Equation 4.19. For a target user u, we regard users with similarity over a threshold t as her neighbor users: $\mathcal{N}_u = \{v | cos(\mathbf{p}_u, \mathbf{p}_v) > t\}$. Because the calibrated distribution of a user tries to approximate the preference probability of the user toward items, it can help to capture the latent relationship between users that naive methods cannot.[2]

Choice of Base Model

Another key factor that can influence the performance of the proposed LFT is the choice of base model ϕ. To allow the base model to be fine tuned effectively, the base model should not be overly optimized for specific users and neglect other users, i.e., the base model should produce

[2]Because conventional recommendation models, such as MF [29], BPR [27], or VAE [92], are vulnerable to various bias including the mainstream bias, it is not appropriate to use the generated embeddings from these regular models to fetch neighbor users.

low mainstream bias. Otherwise, even if local fine tuning is applied, the final prediction will still be biased and in low quality for users overlooked by the base model. Hence, we propose to use the global debiasing model DC or WL in Section 4.3.3.1 as the base model. Besides, another desirable property of the base model is to adapt quickly to a specific user to provide accurate prediction for this user after few epochs of fine tuning training. Thus, we also consider meta-learning techniques [110, 111] to train a base model that can be easily fine tuned to serve specific users. For these meta-learning approaches, we regard every user as an independent learning task and use the sub-dataset $\mathcal{O}_u$ as the training data for each user u. In Section 4.3.4, we will show the empirical comparison of these difference choices of base model, where we find that with WL as the base model, LFT performs the best. Hence, in the rest of this chapter, we consider WL as the default choice of the base model.

Ensemble Model

Since the proposed LFT requires additional fine-tuning training every time a user visits the recommendation platform, it requires more computational resources than conventional inference paradigm without additional fine-tuning training. Although we can control the consuming of time and computational resources by choosing appropriate fine-tuning epoch number and the size of neighbor user set, it may still not be feasible for platforms with limited computational resources and high concurrency of user visits. Hence, we also provide an ensemble version of the propose LFT, which finishes all the model training and stores the model during the training phase, and provides predictions without additional training during inference. Similar to existing local recommendation models [88, 90, 91], during training, we select anchor users and train anchor models for them by the proposed LFT. During inference, for each target user, we ensemble anchor models based on the relationship between the target user and anchor users. So, the key is: how to select anchor users so that they can cover as diverse user preference as possible?

Prior local recommendation models either randomly select anchor users [90] or select mainstream users to maximize the neighbor user coverage by anchor users [88], which tends to select mainstream users as anchor users. Both of these are not ideal for addressing mainstream bias.

Hence, in this chapter, we propose a *similar-dissimilar anchor user selection algorithm* to adequately cover mainstream and non-mainstream preference. The proposed similar-dissimilar algorithm is a greedy algorithm, whose core idea is to select the user who is most similar to unselected users and dissimilar to already selected users in each iteration. Concretely, we define an anchor user set $\mathcal{A}$ beginning as an empty set. Then, we iteratively add users into $\mathcal{A}$ until reach a pre-defined set size. In each iteration, we select the user by:

$$\underset{u \in \mathcal{U}-\mathcal{A}}{\arg\min} \frac{1}{|\mathcal{U}-\mathcal{A}|} \sum_{v \in \mathcal{U}-\mathcal{A}} cos(\mathbf{p}_u, \mathbf{p}_v) - \lambda \frac{1}{|\mathcal{A}|} \sum_{v \in \mathcal{A}} cos(\mathbf{p}_u, \mathbf{p}_v), \tag{4.21}$$

where we calculate the Cosine similar between users by their calibrated distributions from Section 4.3.3.1; and λ controls the balance between similarity to unselected users and dissimilarity to selected users during the current anchor selection.

After selecting anchor users and training anchor models for them by proposed LFT, during inference, we ensemble results of anchor models weighted by similarity to anchor users for a target user u:

$$\widehat{\mathbf{R}}_u = \frac{\sum_{v \in \mathcal{A}} cos(\mathbf{p}_u, \mathbf{p}_v) \phi_v(u)}{\sum_{v \in \mathcal{A}} cos(\mathbf{p}_u, \mathbf{p}_v)}. \tag{4.22}$$

We denote the ensemble version of LFT as EnLFT. Compared with the state-of-the-art local recommendation model LOCA [88], EnLFT has two major improvements. First, EnLFT adopts the more effective similar-dissimilar algorithm to maximize the coverage of different user preference for anchor user selection. Second, EnLFT can train more effective anchor models by fine tuning a global base model with precise neighbor users following LFT.

4.3.4 Experiments

In this section, we conduct extensive experiments to investigate the effectiveness of the proposed debiasing methods and the impact of model design and hyper-parameters.

Table 4.7: Characteristics of three datasets. Reprinted with permission from [9].

	#users	#items	density
ML1M	6,040	3,472	4,75%
Yelp	12,171	9,252	0.38%
Epinions	10,507	9,552	0.32%

4.3.4.1 Experimental Settings

Data

We use three public datasets for the experiments: **ML1M** [96], **Yelp** [105], and **Epinions** [97]. For all datasets, we consider the ratings or reviews as positive feedback from users to items. Then, we randomly split each dataset into 70%, 10%, and 20% for training, validation, and testing. The details of these datasets are shown in Table 4.7.

Metrics

Since in Section 4.3.2.3 we show that DeepSVDD-based bias measuring approach is more effective than other approaches, in this section, we only report the results based on DeepSVDD-based approach for evaluating the mainstream bias. Concretely, we apply DeepSVDD-based approach to all three datasets to compute mainstream scores for users. Then, we sort users in non-descending order based on mainstream scores and divide them into five subgroups evenly. Last we report the average NDCG@20 for each subgroup to show the mainstream bias. *The goal of debiasing is to improve the average NDCG@20 for subgroups with low mainstream scores while preserving or even increasing the utility for subgroups with high mainstream scores at the same time.*

Methods

In the experiments, we adopt the variational autoencoder (**VAE**) [92] as the base and develop different debiasing models based on the VAE. By focusing on the same base model, we can isolate and directly analyze the effects of different debiasing algorithms. VAE is also a baseline method representing the vanilla recommendation model without any debiasing. For debiasing, we consider both global and local methods as introduced in Section 4.3.3. For global methods, we have our proposed Distribution Calibration (**DC**) and Weighted Loss (**WL**) methods. For local methods, we include the state-of-the-art Local Collaborative Autoencoders (**LOCA**) model [88] as a strong local method baseline that has demonstrated superior performance over other local recommendation models like LLORMA [90, 91] and GLSVD [89]. LOCA adopts VAE as its local component. Further, we consider our proposed Local Fine Tuning method with WL as the base model (**LFT**), and we also have the ensemble version of the LFT model (**EnLFT**).

Reproducibility

All models are implemented in PyTorch [112] and optimized by Adam algorithm [60]. For the baseline VAE and the VAE component in other models, we set one hidden layer of size 100. For all methods involving the Distribution Calibration step, including the DC model, LFT, and EnLFT, we set $\alpha = 0.7$. For the WL model, we set $\beta = 1.5$ for ML1M, $\beta = 3$ for Yelp and Epinions. For both LFT and EnLFT: we set the fine tuning epoch number as 30 for ML1M and Yelp, as 5 for Epinions; and we set the similarity threshold t for fetching neighbor users as 0.2 for ML1M, 0.01 for Yelp, and 0.05 for Epinions. For both ensemble models LOCA and EnLFT, we set the number of anchor models as 100. And we set $\lambda = 1.5$ in EnLFT. All code and data can be found at `https://github.com/Zziwei/Measuring-Mitigating-Mainstream-Bias`.

4.3.4.2 RQ1: Compare Debiasing Performance

First, we compare different methods and answer three research questions: i) which method performs the best in terms of overall utility and bias mitigating? ii) how do the two proposed global methods perform compared to each other? and iii) how do the proposed local method and its ensemble version perform compared with the state-of-the-art local recommendation baseline? To answer these, we present the overall NDCG@20 and average NDCG@20 for 5 user subgroups of different mainstream levels for all methods and datasets in Table 4.8, where the best results of all metrics for each dataset are marked in bold, and the improvement rate from proposed LFT over the best baseline LOCA is exhibited as well (results are significant judged by paired t-test). The

Table 4.8: Compare overall utility and utility for different subgroups. Reprinted with permission from [9].

	ML1M						Yelp						Epinion					
	NDCG	Subgroups					NDCG	Subgroups					NDCG	Subgroups				
	@20	l	m-l	m	m-h	h	@20	l	m-l	m	m-h	h	@20	l	m-l	m	m-h	h
VAE	.315	.209	.264	.283	.337	.483	.089	.059	.071	.081	.097	.139	.082	.060	.071	.076	.091	.114
DC	.317	.222	.266	.282	.335	.480	.090	.063	.074	.083	.094	.137	.082	.063	.071	.075	.089	.110
WL	.319	.232	.272	.284	.332	.476	.090	.063	.076	.082	.094	.137	.082	.066	.074	.075	.088	.108
LOCA	.323	.242	.276	.286	.335	.476	.092	.065	.079	.083	.096	.137	.084	.065	.073	.078	.092	.113
EnLFT	.333	.252	.285	.296	.343	.488	.096	.069	.084	.087	.099	.140	.086	.067	.076	.080	.093	.114
LFT	**.337**	**.255**	**.287**	**.298**	**.349**	**.496**	**.098**	**.071**	**.086**	**.092**	**.104**	**.140**	**.088**	**.070**	**.078**	**.083**	**.093**	**.115**
$\Delta_{LOCA}(\%)$	4.4	5.6	4.1	4.2	4.2	4.2	6.8	9.1	9.4	10.1	7.4	2.1	4.0	6.6	7.2	5.2	1.4	2.1

‘l’ column represents the 20% users with lowest mainstream score, ‘m-l’ represents 20% to 40% users in the sorted user sequence, and so on for 40%-60% (‘m’), 60%-80% (‘m-h’), and 80%-100% (‘h’).

From the table we see that for all datasets, the proposed LFT produces the best overall NDCG@20 and also provides the best NDCG@20 for each subgroup of different mainstream levels. Compared with the original VAE, utility for niche users belonging to ‘l’, ‘m-l’, and ’m’ subgroups is greatly promoted (improvement rate is 13.8% in average). At the same time, LFT also improves the utility for mainstream users belonging to ‘med-high’ and ‘h’ subgroups (improvement rate is 3.1%). Hence, we can conclude that the proposed LFT is able to mitigate the mainstream bias by significantly improving the utility for niche users and can improve the utility for mainstream users at the same time.

Second, we compare the two global methods. We can observe that the two proposed global methods – DC and WL – improve the overall recommendation utility and improve the utility for niche users compared with VAE. However, when they mitigate the mainstream bias by improving the utility for niche users, they decrease the utility for mainstream users. It is because global methods keep a single model and have to mitigate the bias by balancing between mainstream and niche users during training instead of improving all of users as proposed LFT does. And comparing DC and WL, we find that WL performs slightly better than DC, which may be because the data

Table 4.9: Compare different neighbor user selection methods. Reprinted with permission from [9].

			Subgroups of different mainstream levels				
		NDCG@20	low	med-low	medium	med-high	high
ML1M	Cosine	0.3302	0.2442	0.2780	0.2896	0.3433	0.4959
	Jaccard	0.3293	0.2417	0.2789	0.2903	0.3409	0.4949
	DC	**0.3372**	**0.2549**	**0.2876**	**0.2982**	**0.3492**	**0.4963**
Yelp	Cosine	0.0949	0.0683	0.0822	0.0883	0.0987	0.1370
	Jaccard	0.0956	0.0688	0.0811	0.0892	0.0995	0.1392
	DC	**0.0984**	**0.0706**	**0.0860**	**0.0917**	**0.1035**	**0.1403**
Epinions	Cosine	0.0867	0.0688	0.0763	0.0813	0.0928	0.1140
	Jaccard	0.0864	0.0690	0.0763	0.0811	0.0924	0.1133
	DC	**0.0876**	**0.0697**	**0.0779**	**0.0825**	**0.0930**	**0.1150**

augmentation method is more challenging to tune and to find an effective setup.

Last, we compare LFT with the local recommendation baseline LOCA. We observe that LOCA can also improve the utility for niche users but with a lower rate compared with LFT. However, utility for mainstream users is decreased compared with VAE. It can be because that LOCA trains anchor models from scratch and the neighbor users to train an anchor model are naively identified based on raw feedback record similarity. This leads to the result that no anchor model in LOCA can capture full information for mainstream users. To be fair, we also propose an ensemble version of LFT with a similar setup as LOCA. Due to the more effective anchor model training (based on LFT) and proposed similar-dissimilar anchor users selection algorithm in EnLFT, we can see from the table that EnLFT performs better than baseline LOCA in terms of overall recommendation utility and utility for different user subgroups. However, EnLFT is slightly less effective than LFT, which is expected because LFT trains a specialized model for each user while a limited number of anchor models are shared in EnLFT.

4.3.4.3 RQ2: Ablation Study

Next, we study how different choices of model component influence the performance, including the way to select neighbor users, the choice of base model, and the way to select anchor users in EnLFT.

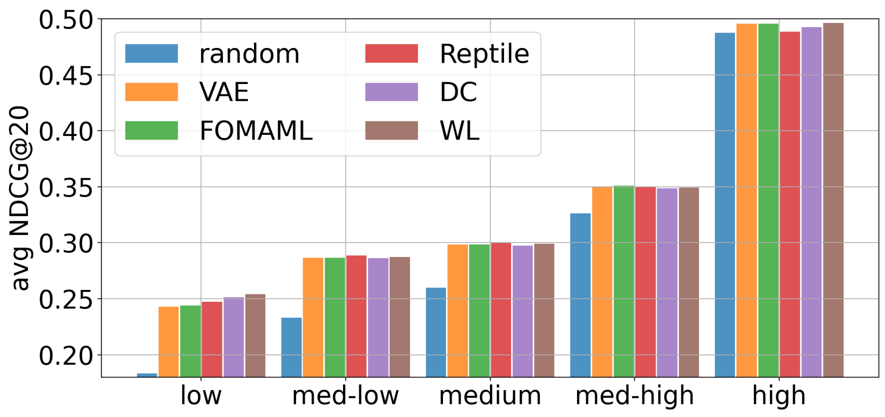


Figure 4.7: Compare different base model choices. Reprinted with permission from [9].

Selecting Neighbor users

In the proposed LFT, when we want to fine tune a local model for one user u, we need to fetch the neighbor users for u. The naive way is to directly calculate similarity on raw feedback records of users, such as Jaccard and Cosine similarity. We compare the performance of LFT by the naive way (by Jaccard and Cosine similarity) and the proposed method of calculating similarity on calibrated distribution of users (denoted as DC), and show the results in Table 4.9. For each dataset, the best results are marked in bold. We can see that LFT with the proposed neighbor user selecting method performs the best for all methods in terms of both overall utility and utility for each subgroup. The superior performance of the proposed method is because that calibrated distribution of users can help to capture the latent relationship between users. Due to this, the improvement for niche users is larger than for mainstream users because identifying neighbor users for niche users heavily relies on latent relationships.

Choosing base model

Next, we study the impact of different choices of base model in LFT. As discussed in Section 4.3.3.2, there are many different choices of base model, including: an ordinary recommendation model without debiasing, such as a VAE; a global debiasing model, such as the proposed DC or WL; and a meta-learning model which is supposed to be easily adapted to a specialized local

Table 4.10: Compare different anchor user selection methods. Reprinted with permission from [9].

			Subgroups of different mainstream levels				
		NDCG@20	low	med-low	medium	med-high	high
ML1M	random	0.3257	0.2376	0.2773	0.2914	0.3387	0.4834
	LOCA	0.3291	0.2428	0.2814	0.2943	0.3403	0.4865
	EnLFT	**0.3328**	**0.2521**	**0.2847**	**0.2963**	**0.3434**	**0.4876**
Yelp	random	0.0922	0.0632	0.0781	0.0839	0.0983	0.1375
	LOCA	0.0933	0.0651	0.0792	0.0853	0.0977	0.1393
	EnLFT	**0.0959**	**0.0688**	**0.0840**	**0.0872**	**0.0994**	**0.1400**
Epinions	random	0.0838	0.0647	0.0736	0.0782	0.0895	0.1130
	LOCA	0.0850	0.0656	0.0743	0.0793	0.0924	0.1136
	EnLFT	**0.0859**	**0.0671**	**0.0758**	**0.0802**	**0.0926**	**0.1139**

model to predict accurately for a specific user, for which we adopt the FOMAML model from [110] and Reptile model from [111]. To train these meta-learning models, we consider every user as an independent task and use the same way in Section 4.3.3.2 to get the sub-dataset $\mathcal{O}_u$ for each user u as the training data for this task. Besides, we also include a random model as a baseline to show the importance of having a good base model. For ML1M dataset, the overall NDCG@20 for choices of random, VAE, FOMAML, Reptile, DC, and WL are 0.2979, 0.3347, 0.3351, 0.3349, 0.3353, and 0.3372, where base model of WL performs the best. Then, we show the average NDCG@20 for user subgroups by different choices of base model in Figure 4.7, from which we observe that base model of WL performs the best for niche users, while FOMAML and Reptile perform similarly as WL for subgroups with higher mainstream level. Hence, we can conclude that choosing WL as the base model produces the best result, and adopting meta-learning techniques does not significantly help to improve the performance. Moreover, we find salient difference between results of random model and other choices, showing that choosing a well-trained base model is important, especially for niche users.

Selecting anchor users in ensemble model

Last, we investigate the impact of different anchor user selection methods in EnLFT. In the baseline LOCA [88], mainstream users are selected as anchor users to maximize the neighbor user coverage, which has limited coverage for niche users. Instead, we propose the similar-dissimilar method to cover both niche and mainstream users to maximize the coverage of user preference. Besides, we also include the random method to randomly select anchor users. We compare these methods with other settings the same for EnLFT, and results are listed in Table 4.10, where 'LOCA' represents EnLFT with anchor user selection method from LOCA, 'EnLFT' represents EnLFT with proposed similar-dissimilar method, and 'random' represents EnLFT with random selection method. This table shows that with the proposed similar-dissimilar method, EnLFT performs the best. Compared with LOCA method, we find that the major improvements are from the niche users (the first three subgroups with lowest mainstream levels), and utility for mainstream users are very similar for these two methods. This is because the proposed similar-dissimilar method improve the coverage for niche users compared with the method from LOCA. Besides, we see that utility for mainstream users by EnLFT with anchor user selection method from LOCA (the 'LOCA' rows in Table 4.10) is higher than the original LOCA (the 'LOCA' columns in Table 4.8), which validates the effectiveness of proposed LFT to learn powerful anchor models.

4.3.4.4 RQ3: Hyper-parameter Study

Here, we study how two hyper-parameters in LFT – the number of training epochs for local fine tuning; and the similarity threshold for fetching neighbor users – influence the performance.

Number of epochs

Here, we run LFT on ML1M dataset with the number of epochs for local fine tuning (denoted as $\#epoch$) varying in {10, 20, 30, 40, 60, 70, 80, 90} and other settings the same as in Section 4.3.4.1. The overall NDCG@20 are shown as the red line in Figure 4.8a, where v0 to v8 represent 10 to 90. We can observe that the utility first increases then decreases with increasing epochs. Then, we also show how the average NDCG@20 changes for the user subgroup of 'low' mainstream level in Figure 4.8b and for the subgroup of 'high' mainstream level in Figure 4.8c. It shows that with increasing training epochs, utility for niche users first increases and then converges, which may decrease with more epochs. However, the utility for mainstream users first

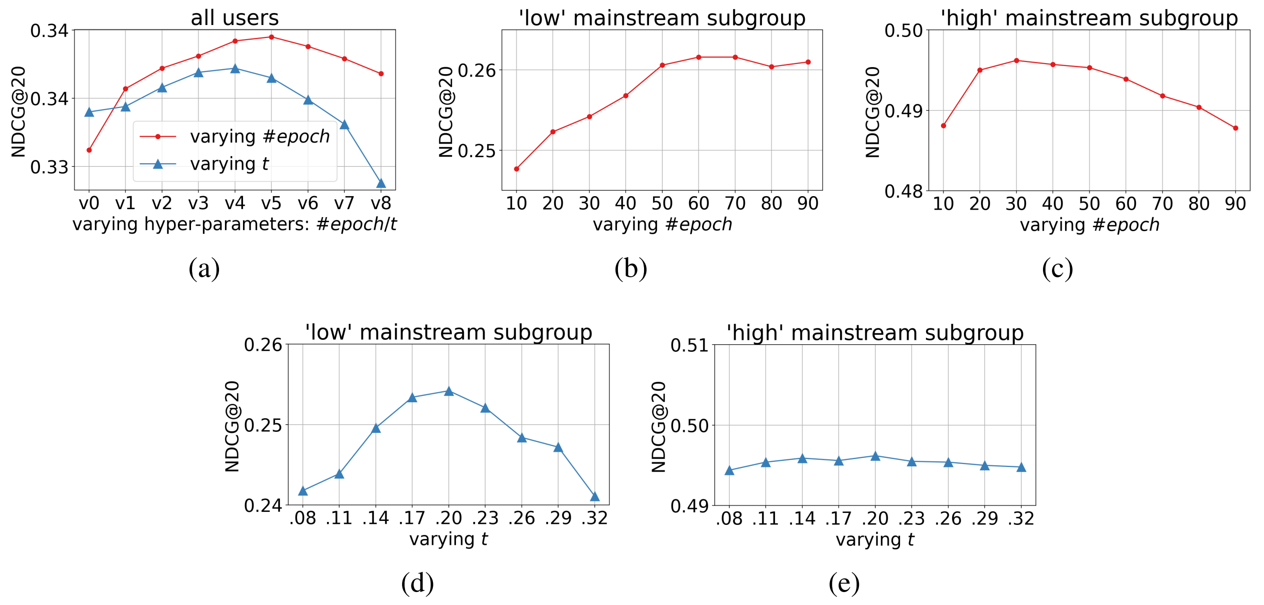


Figure 4.8: Hyper-parameter study: (a) how NDCG@20 changes with varying $\#epoch$ and t; (b) and (c) how NDCG@20 for users of ‘low’ and ‘high’ mainstream levels changes with varying $\#epoch$; (d) and (e) how NDCG@20 for users of ‘low’ and ‘high’ mainstream levels changes with varying t. Reprinted with permission from [9].

increases and turns to decrease quickly, which is because the base model already provides high accuracy and fewer local fine tuning epochs are needed for mainstream users. Hence, the next step of the local fine tuning research is to personalize the training epochs for different users.

Similarity threshold

Then, we study the impact of the similarity threshold t when we fetch neighbor users. Lower t leads to more neighbor users are selected in LFT. We vary t from 0.08 to 0.32 with step 0.03 and show how the overall NDCG@20 changes as the blue line in Figure 4.8a. In this figure, v0 to v8 represent 0.08 to 0.32. We can see that with increasing t, NDCG@20 first increases then decreases and reaches peak at 0.20. We also show how the average NDCG@20 changes for the user subgroup of ‘low’ mainstream level in Figure 4.8d and for the subgroup of ‘high’ mainstream level in Figure 4.8e. From these two figures, we find that with increasing t, NDCG@20 for niche users first increases then decreases, while for mainstream users, the utility does not change notably. It is because niche users have small numbers of neighbor users, and it is more challenging to find these neighbor users. So, utility for niche users is more sensitive to the choice of t. But for mainstream users, there are a large number of similar users with high similarity, thus LFT produces strong performance for mainstream users and is not sensitive to t within certain value range.

4.3.5 Summary

In this work, we study the mainstream bias centering around three thrusts. First, to identify mainstream and niche users, we propose and compare four approaches to calculate a mainstream score indicating mainstream levels of each user. Second, we empirically show the severe mainstream bias produced by conventional recommendation models. Then, we explore both global and local methods to mitigate such a bias. We propose two global models: Distribution Calibration and Weighted Loss; and a local algorithm: Local Fine Tuning. Extensive experiments show the effectiveness of these proposed methods to improve utility for niche users.

4.4 Conclusions

In this chapter, we focus on how the machine learning based recommendation algorithms introduce bias on both users and items even if the data is considered to be free of exposure bias. Specifically, we first we conduct a three-part study to investigate popularity-opportunity bias on items: i) we empirically show the vulnerability of two matrix factorization models to the bias by a data-driven study on four datasets; ii) we theoretically show how these two models inherently produce the popularity-opportunity bias on both user and item sides; and iii) we explore the potential of in-processing and post-processing approaches to alleviate the bias. Experiments on four datasets validate the debiasing effectiveness of both proposed methods over debiasing baselines designed for conventional popularity bias. Then, we switch our attention to users and we study the mainstream bias among users centering around three thrusts. First, to identify mainstream and niche users, we propose and compare four approaches to calculate a mainstream score indicating mainstream levels of each user. Second, we empirically show the severe mainstream bias produced by conventional recommendation models. Then, we explore both global and local methods to mitigate such a bias. We propose two global models: Distribution Calibration and Weighted Loss;

and a local algorithm: Local Fine Tuning. Extensive experiments show the effectiveness of these proposed methods to improve utility for niche users.

5. MEASURING AND ENHANCING FAIRNESS IN RECOMMENDATIONS[1]

The third contribution of this dissertation is to investigate how bias raised by both data and algorithms can incur unfairness issues in the outputs of a recommender. That is, are different items or item groups recommended fairly in a recommender system? For example, in a job recommender that recommends job openings to people, are high-paying jobs and non-profit jobs treated fairly? In a news recommender, are news of different political stances recommended at a similar rate? And even for product recommenders, are products from big companies favored over products from new entrants?

Concretely, we investigate how to measure and enhance recommendation fairness in three different recommendation scenarios. First, we study recommendation fairness among item groups in the multi-dimension recommendation scenario, where instead of typical two-dimension case with only users and items, the multi-dimension scenario involves other dimensions to specify the conditions of recommending items, such as specify the genre for recommending movies or songs. Second, we study recommendation fairness among item groups in personalized ranking recommendation scenario, where the crucial question is how to measure and enhance fairness directly on ranking results. And last, we investigate recommendation fairness for cold-start recommendation scenario, where we consider the recommendations for cold-start items with no historical interaction in training data and study recommendation fairness among them.

5.1 Related Work

In this section, we introduce previous work on the topic of recommendation fairness. Besides, we also introduce some other topics related to fairness, and also some background about cold-start

[1]This chapter is reprinted with permission from “Fairness-Aware Tensor-Based Recommendation" by Ziwei Zhu, Xia Hu, and James Caverlee, 2018, Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Copyright 2018 by ACM; “Measuring and Mitigating Item Under-Recommendation Bias in Personalized Ranking Systems" by Ziwei Zhu, Jianling Wang and James Caverlee, 2020, Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Copyright 2020 by ACM; and “Fairness among New Items in Cold Start Recommender Systems" by Ziwei Zhu, Jingu Kim, Trung Nguyen, Aish Fenton, and James Caverlee, 2021, 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Copyright 2021 by ACM.

recommendation.

5.1.1 Recommendation Fairness

Friedman [113] defined that a computer system is biased "if it systematically and unfairly discriminates against certain individuals or groups of individuals in favor of others." As we have mentioned, considerable efforts have focused on classification tasks (e.g., recidivism prediction, loan approval) [114, 115, 116, 117, 118]. In the context of recommenders, early studies mainly focus on rating prediction tasks and investigate item fairness by measuring the difference of predicted rating distributions across item groups [119, 120, 121, 10, 11]. To improve this score/rating based concept of item fairness, regularization based [119, 120, 121, 11] and latent factor manipulation based [10] methods have been proposed. Later, with the prevalence of ranking based recommender systems, new formulations [12, 24, 26, 101, 122] to directly study item fairness on ranking results instead of on the intermediate predicted scores/ratings have been proposed. Another line of research [123, 124, 125] studies fairness in terms of equality of exposure, which investigates whether the amortized exposure of items in recommendations are proportional to the amortized relevance of items. Further, inspired by fairness-aware classification [94], equal opportunity based fairness that requires an equal true positive rate across item groups has been proposed [12, 24, 122, 26]. To improve such ranking based equal opportunity fairness, many new algorithms utilizing regularization [122, 24], re-ranking [26], and adversarial learning [12] have also been proposed.

5.1.2 Topics Related to Fairness

There are some recent efforts investigating topics related to recommendation fairness. For example, Beutel et al. [126] and Krishnan et al. [127] explored approaches to improve the fairness w.r.t. recommendation accuracy for niche items. Recommendation diversity [128], which requires as many groups as possible appearing in the recommendation list for each user, is related to the metric RSP we propose, but fundamentally different. Another similar concept to RSP called calibrated recommendations is proposed by Steck [129], which encourages the same group proportions as the historical record for each user and can be regarded as a special fairness for individual users.

The main differences of our work and these previous works are that research of recommendation diversity and recommendation calibration investigate the distribution skews for each individual user rather than for the whole system, and they only consider the recommendation distributions without taking into account the ground truth of user preference and item quality.

5.1.3 Cold-start Recommender Systems

In almost all real-world applications, recommending new items without any historical feedback from users (formally called the cold-start recommendation task), is heavily in demand and challenging. Over the years, many approaches have been proposed including heuristic non-parametric algorithms like KNN [64], though now there is an emphasis on optimization based machine learning algorithms. These methods can be categorized into two types [7]: separate-training methods and joint-training methods.

Separate-training methods [6, 52, 49, 50, 51] separately learn two models: a collaborative filtering (CF) model learns CF embeddings of warm start items; and a content model learns how to transform the item content features to the learned collaborative embeddings using warm start items. During inference, the content model is first applied on content features of new items to generate CF embeddings and then recommendations for these new items are provided by these embeddings. A typical example is DeepMusic [49], which utilizes a multi-layer perceptron (MLP) to transform item content features to CF embeddings learned by a pretrained matrix factorization model. Joint-training methods [5, 3, 4] combine the CF model and content model together and train both of them through a single backpropagation process. A typical example is DropoutNet [5], which has an MLP based content component to first transform item content features, followed by a dot-product based neural CF component to provide recommendations. Moreover, Heater in [7] mixes separate-training and joint-training methods, which delivers the state-of-the-art performance. Yet, there is no notion of fairness in these cold-start recommenders. Hence, we aim to investigate the fairness of these representative models and propose novel methods to enhance fairness in the cold-start scenario. Specifically, we investigate four typical cold-start recommenders: Heater, DropoutNet, DeepMusic, and KNN.

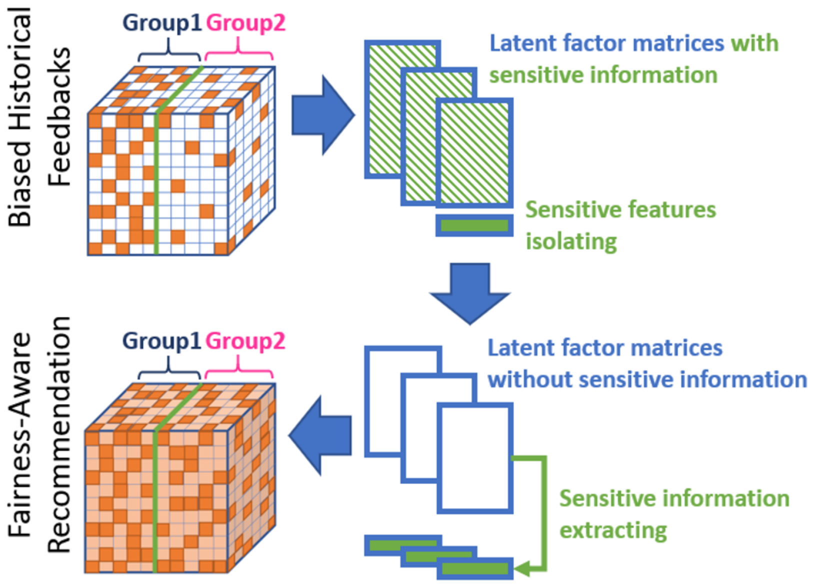

Biased Historical Feedbacks
Group1
Group2
Latent factor matrices with sensitive information
Sensitive features isolating
Latent factor matrices without sensitive information
Sensitive information extracting
Fairness-Aware Recommendation
Group1
Group2

Figure 5.1: Overview of FATR: sensitive features are isolated (top right), then sensitive information is extracted (bottom right), resulting in fairness-aware recommendation. Reprinted with permission from [10].

5.2 Enhancing Fairness in Multi-dimension Recommender Systems

5.2.1 Introduction

Fairness in recommendation has attracted increasing attention in the research community. Many different approaches have been developed [130, 11, 131, 102]. While encouraging, most existing approaches make a number of limiting assumptions: (i) focusing on two-dimensional matrix factorization that has been the cornerstone of recommender research in the past ten years [132, 133, 134]; (ii) assuming there is only a single binary case (e.g., male vs. female); and (iii) trading-off considerable recommendation quality for improving the fairness characteristics of the recommender.

In contrast, we aim to create a new *tensor-based* framework that can overcome these limitations. Tensors, as n-dimensional generalizations of matrices, have shown great promise across a variety of data mining and analytics tasks – e.g., [135, 136, 137, 138] – where their multi-aspect models naturally fit domains that go beyond two dimensions. Recommenders, in particular, are well-suited for tensors that can capture multi-way interactions among users, items, and contexts (e.g., time, location). But there are key challenges: How can we model sensitive attributes (e.g., age, gender) in a tensor-based recommender? How can we minimize the impact of these sensitive attributes on recommendations, which can be correlated with non-sensitive attributes [139, 118])? How can we build an optimization model for this problem and efficiently solve it? And can such efforts maintain recommendation quality while improving fairness?

To tackle these challenges, we propose a novel **F**airness-**A**ware **T**ensor-based **R**ecommendation framework called **FATR**. The overview is illustrated in Figure 5.1. The intuition of the proposed framework is that the latent factor matrices of the tensor completion model contain latent information related to the sensitive attributes, which introduces the unfairness. Therefore, by *isolating* and then *extracting* the sensitive information from the latent factor matrices, we may be able to improve the fairness of the recommender itself. Concretely, we propose (i) a new *sensitive latent factor matrix* for isolating sensitive features; (ii) a sensitive information regularizer that extracts sensitive information which can taint other latent factors; and (iii) an effective algorithm to solve the proposed optimization model.

In sum, our contributions are as follows.

- First, FATR is built on a tensor foundation that can analyze multiple aspects simultaneously, promising potentially better recommendation quality than matrix-based approaches, while also supporting traditional two-dimensional data (since tensors are generalizations of matrices).

- Second, moving beyond binary sensitive features, FATR supports multi-feature cases with multisided features (e.g., recommendation where both age of items and gender of users are considered sensitive) and multi-category cases (e.g., where the sensitive attribute can take on multiple values like Low, Medium, and High) which are challenging for traditional regularization-based approaches [120, 131].

- Finally, we empirically show that FATR can provide recommendation quality on par with traditional (unfair) recommenders while significantly improving the fairness of recommen-

Table 5.1: Main symbols and operations. Reprinted with permission from [10].

Notations	Definitions
$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$	N^{th}-order tensor
$\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (\Pi_{i \neq n}^{N} I_i)}$	Mode-n unfolding matrix of tensor $\mathcal{X}$
$[\![\cdot]\!]$	Kruskal operator, e.g., $\mathcal{X} \approx [\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!]$
$\odot$	Khatri-Rao product
$\circledast$	Hadamard product
$\circ$	Vector outer product
$(\mathbf{A}_k)^{\odot k \neq n}$	$\mathbf{A}_N \odot \ldots \odot \mathbf{A}_{n+1} \odot \mathbf{A}_{n-1} \odot \ldots \odot \mathbf{A}_1$
$\mathbf{A}[:, i : j]$	Matrix slicing operation (index starts from 1)
$[\mathbf{A} \ \ \mathbf{B}]$	Matrices concatenating operation (horizontal)

dations, and does so better than state-of-the-art alternatives.

5.2.2 Preliminaries

In this section, we first introduce the notations we use here and the basics of tensor-based recommendation, then we discuss fairness in recommendation.

5.2.2.1 Notations

Notations and definitions are presented as follows. Tensors are denoted by Euler script letters like $\mathcal{X}$, matrices are denoted by boldface uppercase letters like $\mathbf{A}$, and vectors are denoted by boldface lowercase letters like $\mathbf{a}$. The $[i_1, \ldots, i_N]$ entry of the tensor $\mathcal{X}$ is denoted as $\mathcal{X}[i_1, \ldots, i_N]$. We denote the pseudo inverse, transpose, and Frobenius norm of a matrix $\mathbf{A}$ respectively by $\mathbf{A}^{\dagger}$, $\mathbf{A}^{\top}$, and $\|\mathbf{A}\|_{\mathrm{F}}$. Notation $[\![\cdot]\!]$ represents the Kruskal operator. Notations $\odot$, $\circledast$, and $\circ$ denote the Khatri-Rao product, Hadamard product, and vector outer product, respectively. Besides, we use the syntax similar to Python to denote the matrix slicing operation (the index starts from 1), for example $\mathbf{A}[:, 2 :]$ denotes the matrix $\mathbf{A}$ without the first column. And we use $[\mathbf{A} \ \ \mathbf{B}]$ to present the horizontal matrices concatenating operation. The main symbols and operations are listed in Table 5.1. More details about tensor calculations can be found in [140].

5.2.2.2 Tensor-Based Recommendation

Matrix factorization is the foundation of many modern recommenders [29]. These matrix factorization methods estimate missing ratings by uncovering latent features of users and items. Building on these user-item interactions, *tensor-based* methods have been growing in appeal recently since they can naturally model multi-way (or multi-aspect) interactions [135, 136, 137, 138]. For example, a 3-order tensor could represent users, items, and time of day. Additional contexts can lead to an N-way tensor. And, of course, the classic user-item problem can be viewed as a 2-way tensor.

Formally, given an N-order tensor $\boldsymbol{\mathcal{T}}$ representing the users, items, and multiple aspects related to the items, the basic tensor-based recommendation model can be defined as:

$$\begin{aligned} \underset{\boldsymbol{\mathcal{X}},\mathbf{A}_1,\mathbf{A}_2,\ldots,\mathbf{A}_N}{\text{minimize}} \quad & \boldsymbol{\mathcal{L}} = \|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]\|_{\mathrm{F}}^2 \\ \text{subject to} \quad & \boldsymbol{\Omega} \circledast \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{T}}, \end{aligned} \tag{5.1}$$

where $\boldsymbol{\mathcal{X}}$ denotes the complete preferences of users, $\boldsymbol{\mathcal{T}}$ denotes the observations, $\boldsymbol{\Omega}$ is a non-negative indicator tensor with the same size as $\boldsymbol{\mathcal{X}}$ with $\boldsymbol{\Omega}[i_1, \ldots, i_N] = 1$ indicating that we observe the preference, otherwise $\boldsymbol{\Omega}[i_1, \ldots, i_N] = 0$, $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$ are the latent factor matrices of all the modes of the tensor.

The objective function can be written in the unfolding form so that it can be solved by optimization algorithms, as follows:

$$\begin{aligned} \underset{\boldsymbol{\mathcal{X}},\mathbf{A}_1,\ldots,\mathbf{A}_N}{\text{minimize}} \quad & \boldsymbol{\mathcal{L}} = \|\mathbf{X}_{(n)} - \mathbf{A}_n[(\mathbf{A}_k)^{\odot_{k\neq n}}]^\top\|_{\mathrm{F}}^2 \\ \text{subject to} \quad & \boldsymbol{\Omega}_{(n)} \circledast \mathbf{X}_{(n)} = \mathbf{T}_{(n)}, \end{aligned} \tag{5.2}$$

where $\boldsymbol{\Omega}_{(n)}$ is the mode-n unfolding of the indicator tensor $\boldsymbol{\Omega}$, $\mathbf{T}_{(n)}$ is the mode-n unfolding of the tensor $\boldsymbol{\mathcal{T}}$, and $\mathbf{X}_{(n)}$ is the mode-n unfolding of the tensor $\boldsymbol{\mathcal{X}}$. To solve this basic recommendation by tensor completion, we can use Alternating Least Squares (ALS), which optimizes every latent factor matrix by linear least squares in each iteration. The update rule is:

$$\widehat{\mathbf{A}_n} \leftarrow \mathbf{X}_{(n)}[[(\mathbf{A}_k)^{\odot_{k\neq n}}]^\top]^\dagger, \tag{5.3}$$

where $\widehat{\mathbf{A}_n}$ is the updated latent factor matrix of $\mathbf{A}_n$.

5.2.2.3 Fairness in Recommendation

Such a tensor-based approach has no notion of fairness. Here, we assume that there exists a *sensitive attribute* for one mode of the tensor, and this mode is a *sensitive mode*. For example, the sensitive attribute could correspond to gender, age, ethnicity, location, or other domain-specific attributes of users or items in the recommenders. The feature vectors of the sensitive attributes are called the *sensitive features*. Further, we call all the information related to the sensitive attributes as *sensitive information*, and note that attributes other than the sensitive attributes can also contain sensitive information [139, 118]. While there are emerging debates about what constitutes algorithmic fairness [114], we adopt the commonly used notion of *statistical parity*. Statistical parity encourages a recommender to ensure similar probability distributions for both the dominant group and the protected group as defined by the sensitive attributes. Formally, we denote the sensitive attribute as a random variable S, and the preference rating in the recommender system as a random variable R. Then we can formulate fairness as $\mathbf{P}[R] = \mathbf{P}[R|S]$, i.e. the preference rating is independent of the sensitive attribute. This statistical parity means that the recommendation result should be unrelated to the sensitive attributes. For example, a job recommender should recommend similar jobs to men and women with similar profiles. Note that some recent works [94, 102, 131] have argued that statistical parity may be overly strict, resulting in poor utility to end users. Our work here aims to achieve comparable utility to non-fair approaches, while providing stronger fairness.

5.2.3 Fairness-Aware Tensor-Based Recommendation

Given this notion of fairness, we turn in this section to the design of a novel **F**airness-**A**ware **T**ensor-based **R**ecommendation framework (**FATR**) – as illustrated in Figure 5.2. The intuition of

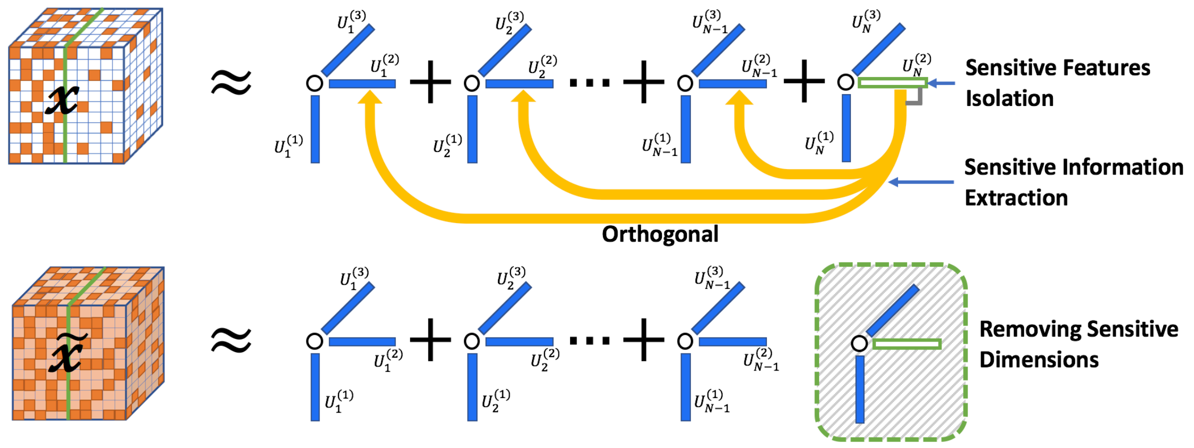


Figure 5.2: FATR isolates sensitive features in the latent matrix with non-sensitive dimensions orthogonal to them and eliminates the sensitive information by removing the sensitive dimensions. $\mathcal{X}$ is the tensor with bias, and $\widetilde{\mathcal{X}}$ is the fairness-enhanced recommendation tensor. Reprinted with permission from [10].

the proposed framework is that the latent factor matrices of the tensor completion model contain latent information related to the sensitive attributes, which introduces the unfairness. Therefore, by *isolating* and then *extracting* the sensitive information from the latent factor matrices, we may be able to improve the fairness of the recommender itself.

In the rest of this section, we aim to address four key questions: (i) How can we represent (and ultimately isolate) the sensitive attributes in the tensor completion model? (ii) How do we extract all the sensitive information into the isolated explicit representation? (iii) How can we eliminate the extracted sensitive information from the tensor completion model? and (iv) How do we solve the new fairness-aware recommendation model? In the following, we address these questions in turn. We focus in this section on a single binary sensitive attribute for mode-n (e.g., gender). In Section 5.2.4, we will generalize to consider multi-feature and multi-category cases.

5.2.3.1 Isolating Sensitive Features

In conventional tensor completion, the sensitive features will mingle with other features and distribute over different dimensions in the latent factor matrices, which makes it difficult to extract them. For example, a 3-way tensor of user-expert-topic can be factorized into three latent factor matrices [135], where the feature vector of a sensitive attribute for the experts like gender is mixed with other features and represented by the latent factors, which means that the sensitive information hides in the expert latent factor matrix.

We propose to first isolate the impact of the sensitive attribute by plugging the sensitive features into the latent factor matrix. For instance, in our user-expert-topic example, we can create one vector $\mathbf{s}_0$ with 1 representing male and 0 representing female, and another vector $\mathbf{s}_1$ with 1 indicating female and 0 indicating male. $\mathbf{s}_0$ and $\mathbf{s}_1$ together form a matrix, denoted as *Sensitive Features* $\mathbf{S}$. We put $\mathbf{S}$ to the last two columns of the latent factor matrix of sensitive mode mode-n. Then we construct a new *sensitive latent factor matrix*:

DEFINITION *(Sensitive Latent Factor Matrix).* Given the latent factor matrix $\mathbf{A}_n \in \mathbb{R}^{d_n \times r}$ of the sensitive mode mode-n, where r is the dimension of the latent factors and d_n is the number of entities of the mode-n. We split $\mathbf{A}_n$ horizontally into two parts: matrix $\mathbf{A}'_n \in \mathbb{R}^{d_n \times (r-2)}$ and $\mathbf{A}''_n \in \mathbb{R}^{d_n \times 2}$. If $\mathbf{A}''_n$ is forced to take the same values as the sensitive features $\mathbf{S} \in \mathbb{R}^{d_n \times 2}$, then the new matrix $\widetilde{\mathbf{A}_n} = [\mathbf{A}'_n \;\; \mathbf{S}]$ is called sensitive latent factor matrix.

The matrix $\mathbf{A}'_n$ represents the *non-sensitive dimensions*, while $\mathbf{A}''_n$ represents the *sensitive dimensions* (where the corresponding dimensions in other non-sensitive factor latent matrices are also called sensitive dimensions). Thus, sensitive dimensions of the sensitive latent factor matrix will take the same values of the sensitive features. In this way, we can explicitly represent the sensitive attributes and isolate them from non-sensitive attributes in the latent factor matrix. Hence, we can update the tensor-based recommender in Section 5.2.2.2 with the following objective function:

$$\begin{aligned}
\underset{\boldsymbol{\mathcal{X}},\mathbf{A}_1,\ldots,\widetilde{\mathbf{A}_n},\ldots,\mathbf{A}_N}{\text{minimize}} \quad & \boldsymbol{\mathcal{L}} = \|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}_1,\ldots,\widetilde{\mathbf{A}_n},\ldots,\mathbf{A}_N]\!]\|_{\mathrm{F}}^2 \\
\text{subject to} \quad & \boldsymbol{\Omega} \circledast \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{T}}, \\
& \widetilde{\mathbf{A}_n} = [\mathbf{A}'_n \;\; \mathbf{A}''_n], \\
& \mathbf{A}''_n = \mathbf{S}.
\end{aligned} \tag{5.4}$$

5.2.3.2 Extracting Sensitive Information

By isolating the sensitive features, we provide a first step toward improving the fairness of the recommender. But there may still be sensitive information that resides in non-sensitive dimensions. To extract this remaining sensitive information, we propose an additional constraint that the non-sensitive dimensions should be orthogonal to the sensitive dimensions in the sensitive latent factor matrix based on the following theorem.

Theorem 3. *If one non-sensitive dimension is not perpendicular to all the sensitive dimensions, then this dimension is related to the sensitive attribute.*

Proof. Regarding all dimensions in the sensitive latent factor matrix as vectors in a high dimensional space. If the angle between one non-sensitive dimension vector $\mathbf{v}$ and the plane p_{s_1,s_2} decided by sensitive features $\mathbf{s}_1$ and $\mathbf{s}_2$ is not $90°$, then $\mathbf{v}$ can be resolved into two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ on the same directions as $\mathbf{s}_1$ and $\mathbf{s}_2$, and another vector $\mathbf{v}_3$ perpendicular to p_{s_1,s_2}. Therefore, $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3$, and $\mathbf{v}_1$ and $\mathbf{v}_2$ can be merged into $\mathbf{s}_1$ and $\mathbf{s}_2$ when reconstructing the tensor as shown in Equation (5.5), which changes the values of sensitive dimensions, i.e., this latent factor represented by dimension $\mathbf{v}$ is related to the sensitive attribute.

$$
\begin{aligned}
\mathcal{X} \approx\ & \mathbf{a}_1^{(1)} \circ \mathbf{a}_2^{(1)} \circ \mathbf{a}_3^{(1)} + \ldots + \mathbf{a}_1^{(r-2)} \circ \mathbf{a}_2^{(r-2)} \circ \mathbf{v} \\
& + \mathbf{a}_1^{(r-1)} \circ \mathbf{a}_3^{(r-1)} \circ \mathbf{s}_1 + \mathbf{a}_1^{(r)} \circ \mathbf{a}_2^{(r)} \circ \mathbf{s}_2 \\
=\ & \mathbf{a}_1^{(1)} \circ \mathbf{a}_2^{(1)} \circ \mathbf{a}_3^{(1)} + \ldots + \mathbf{a}_1^{(r-2)} \circ \mathbf{a}_2^{(r-2)} \circ \mathbf{v}_3 \\
& + l_1 \cdot \mathbf{a}_1^{(r-2)} \circ \mathbf{a}_2^{(r-2)} \circ \mathbf{s}_1 + l_1 \cdot \mathbf{a}_1^{(r-2)} \circ \mathbf{a}_2^{(r-2)} \circ \mathbf{s}_2 \\
& + \mathbf{a}_1^{(r-1)} \circ \mathbf{a}_3^{(r-1)} \circ \mathbf{s}_1 + \mathbf{a}_1^{(r)} \circ \mathbf{a}_2^{(r)} \circ \mathbf{s}_2 \\
=\ & \mathbf{a}_1^{(1)} \circ \mathbf{a}_2^{(1)} \circ \mathbf{a}_3^{(1)} + \ldots + \mathbf{a}_1^{(r-2)} \circ \mathbf{a}_2^{(r-2)} \circ \mathbf{v}_3 \\
& + (\mathbf{a}_1^{(r-1)} \circ \mathbf{a}_2^{(r-1)} + l_1 \cdot \mathbf{a}_1^{(r-2)} \circ \mathbf{a}_2^{(r-2)}) \circ \mathbf{s}_1 \\
& + (\mathbf{a}_1^{(r)} \circ \mathbf{a}_2^{(r)} + l_2 \cdot \mathbf{a}_1^{(r-2)} \circ \mathbf{a}_2^{(r-2)}) \circ \mathbf{s}_2,
\end{aligned}
\tag{5.5}
$$

where $\mathbf{a}_1^{(1\ldots r)}$, $\mathbf{a}_2^{(1\ldots r)}$, and $\mathbf{a}_3^{(1\ldots r)}$ are the columns in the three latent factor matrices, $l_1$is the scale coefficient between $\mathbf{s}_1$ and $\mathbf{v}_1$ so that $l_1 \cdot \mathbf{s}_1 = \mathbf{v}_1$, and l_2 is from $l_2 \cdot \mathbf{s}_2 = \mathbf{v}_2$.

□

After extracting the sensitive information, all the sensitive information is gathered in the isolated sensitive dimensions. Then we have a new objective function for the tensor completion as:

$$
\begin{aligned}
\underset{\mathcal{X},\mathbf{A}_1,\ldots,\mathbf{A}'_n,\ldots,\mathbf{A}_N}{\text{minimize}} \quad & \mathcal{L} = \|\mathcal{X} - [\![\mathbf{A}_1,\ldots,\widetilde{\mathbf{A}_n},\ldots,\mathbf{A}_N]\!]\|_{\mathrm{F}}^2 \\
& \quad + \frac{\lambda}{2}\|\mathbf{A}''^{\top}_n \mathbf{A}'_n\|_{\mathrm{F}}^2 + \frac{\gamma}{2}\sum_{i=1}^{N}\|\mathbf{A}_i\|_{\mathrm{F}}^2 \\
\text{subject to} \quad & \boldsymbol{\Omega} \circledast \mathcal{X} = \mathcal{T}, \\
& \widetilde{\mathbf{A}_n} = [\mathbf{A}'_n \ \ \mathbf{A}''_n], \\
& \mathbf{A}''_n = \mathbf{S},
\end{aligned}
\tag{5.6}
$$

where $\frac{\lambda}{2}\|\mathbf{A}''^{\top}_n \mathbf{A}'_n\|_{\mathrm{F}}^2$ is the orthogonal constraint term, λ is the trade-off parameter, $\frac{\gamma}{2}\sum_{i=1}^{N}\|\mathbf{A}_i\|_{\mathrm{F}}^2$ is the L2-norm constraint to the norms of the latent factor matrices so that the minimizing of $\frac{\lambda}{2}\|\mathbf{A}''^{\top}_n \mathbf{A}'_n\|_{\mathrm{F}}^2$ is because the cosine angles are close to zero rather than because the norms of columns in $\mathbf{A}''_n$ or $\mathbf{A}'_n$ are small (if it is this case, norms of other latent factor matrices will get larger, which will increase the value of the term $\frac{\gamma}{2}\sum_{i=1}^{N}\|\mathbf{A}_i\|_{\mathrm{F}}^2$), γ is the trade-off parameter of this L2-norm term.

5.2.3.3 Fairness-Aware Recommendation

After the above two steps, we can get the new latent factor matrices $\mathbf{A}_1,\ldots,\widetilde{\mathbf{A}_n},\ldots,\mathbf{A}_N$, whose sensitive dimensions hold features exclusively related to the sensitive attributes. And their non-sensitive dimensions are decoupled from the sensitive attributes. Thus, we can derive the fairness-enhanced recommendation by combining these matrices after removing their sensitive dimensions as:

$$
\widetilde{\mathcal{X}} \leftarrow [\![\mathbf{A}'_1,\ldots,\mathbf{A}'_n,\ldots,\mathbf{A}'_N]\!], \tag{5.7}
$$

Algorithm 2: FATR Solver

```
   Input: $\mathcal{T}$, $\boldsymbol{\Omega}$, $r$, $\mathbf{S}$, $n$, $\alpha$, $\lambda$, $\gamma$, $tol$;
   Output: $\widetilde{\mathcal{X}}$, $\{\mathbf{A}_i\}_{i=1}^N$
1  Randomly Initialize $\{\mathbf{A}_i \in \mathbb{R}^{I_i \times r}\}_{i=1}^N$;
2  repeat
3      for $i = 1 : N$ do
4          if $i == n$ then
5              Update $\mathbf{A}'_n$ using (5.11);
6          else
7              Update $\mathbf{A}_i$ using (5.10);
8      Update $\mathbf{A}'_n \leftarrow \mathbf{A}'_n - \alpha \frac{\partial \mathcal{F}}{\partial \mathbf{A}'_n}$;
9      Form $\widetilde{\mathbf{A}_n} \leftarrow [\mathbf{A}'_n \;\; \mathbf{S}]$;
10     Update $\mathcal{X} \leftarrow \mathcal{T} + \boldsymbol{\Omega} \circledast [\![\mathbf{A}_1, \ldots, \widetilde{\mathbf{A}_n}, \ldots, \mathbf{A}_N]\!]$;
11 until $\|\mathcal{X}_{pre} - \mathcal{X}\|_F / \|\mathcal{X}_{pre}\|_F < tol$;
12 Update $\widetilde{\mathcal{X}} \leftarrow [\![\mathbf{A}'_1, \ldots, \mathbf{A}'_n, \ldots, \mathbf{A}'_N]\!]$;
```

where $\widetilde{\mathcal{X}}$ is the fairness-enhanced tensor completion result, and $\mathbf{A}'_1, \ldots, \mathbf{A}'_n, \ldots, \mathbf{A}'_N$ are the non-sensitive dimensions of the latent factor matrices (i.e. the first $r-2$ columns in $\mathbf{A}_1, \ldots, \widetilde{\mathbf{A}_n}, \ldots, \mathbf{A}_N$).

5.2.3.4 Optimization Algorithms

To solve the optimization problem in Equation (5.6), we need to first rewrite the objective function to be the unfolding matrix form. For the sensitive mode mode-n, the unfolding form is :

$$\begin{aligned}
\underset{\mathcal{X}, \mathbf{A}_1, \ldots, \mathbf{A}'_n, \ldots, \mathbf{A}_N}{\text{minimize}} \quad & \mathcal{L} = \|\mathbf{X}_{(n)} - \mathbf{A}''_n(\mathbf{B}''_n)^\top - \mathbf{A}'_n(\mathbf{B}'_n)^\top\|_\text{F}^2 \\
& \quad + \frac{\lambda}{2}\|\mathbf{A}''^\top_n \mathbf{A}'_n\|_\text{F}^2 + \frac{\gamma}{2}\sum_{i=1}^N \|\mathbf{A}_i\|_\text{F}^2 \\
\text{subject to} \quad & \boldsymbol{\Omega}_{(n)} \circledast \mathbf{X}_{(n)} = \mathbf{T}_{(n)}, \\
& \mathbf{B}_n = [(\mathbf{A}_k)^{\odot_{k \neq n}}], \\
& \mathbf{B}'_n = \mathbf{B}_n[:, : r-2], \\
& \mathbf{B}''_n = \mathbf{B}_n[:, r-1 :], \\
& \mathbf{A}''_n = \mathbf{S},
\end{aligned} \tag{5.8}$$

where $\mathbf{B}_n$ is the result of the Khatri-Rao product of all the latent factor matrices without mode-n, $\mathbf{B}'_n$ is the first $r-2$ dimensions of $\mathbf{B}_n$, and $\mathbf{B}''_n$ is the last 2 dimensions of $\mathbf{B}_n$.

For non-sensitive modes (denoted as m), the unfolding objective function is:

$$\begin{aligned} \underset{\mathcal{X},\mathbf{A}_1,\ldots,\mathbf{A}_N}{\text{minimize}} \quad & \boldsymbol{\mathcal{L}} = \|\mathbf{X}_{(m)} - \mathbf{A}_m[(\mathbf{A}_k)^{\odot_{k\neq m}}]^\top\|_\text{F}^2 \\ & \qquad + \frac{\lambda}{2}\|\mathbf{A}''^\top_n \mathbf{A}'_n\|_\text{F}^2 + \frac{\gamma}{2}\sum_{i=1}^{N}\|\mathbf{A}_i\|_\text{F}^2 \\ \text{subject to} \quad & \mathbf{\Omega}_{(m)} \circledast \mathbf{X}_{(m)} = \mathbf{T}_{(m)}, \ \ \mathbf{A}''_n = \mathbf{S}. \end{aligned} \tag{5.9}$$

Equation (5.8) cannot be solved by ALS because of $\frac{\lambda}{2}\|\mathbf{A}''^\top_n \mathbf{A}'_n\|_\text{F}^2$, but Equation (5.9) can be solved by ALS because $\frac{\lambda}{2}\|\mathbf{A}''^\top_n \mathbf{A}'_n\|_\text{F}^2$ is a constant term for non-sensitive modes. We can use Gradient Descent to solve them together, but its performance is not as good as ALS for tensor completion task. However, if we can separate $\frac{\lambda}{2}\|\mathbf{A}''^\top_n \mathbf{A}'_n\|_\text{F}^2$ from the objective function and optimize it alone, we can efficiently and effectively solve the problem. Thus, we propose a hybrid optimization algorithm which treats the sensitive and non-sensitive modes differently. It follows the ALS rule to update the non-sensitive modes in each iteration. For the sensitive mode mode-n, we first use ALS to update $\mathbf{A}'_n$ with $\frac{\lambda}{2}\|\mathbf{A}''^\top_n \mathbf{A}'_n\|_\text{F}^2$ being considered as a constant term, and then use Gradient Descent to update $\mathbf{A}'_n$ again only to minimize $\frac{\lambda}{2}\|\mathbf{A}''^\top_n \mathbf{A}'_n\|_\text{F}^2$. The update rule for the non-sensitive modes is defined in rule (5.10), and the first ALS step for the sensitive mode mode-n uses update rule (5.11).

$$\widehat{\mathbf{A}_m} \leftarrow \mathbf{X}_{(m)}(\mathbf{A}_k)^{\odot_{k\neq m}}[\gamma\mathbf{I} + [(\mathbf{A}_k)^{\odot_{k\neq m}}]^\top(\mathbf{A}_k)^{\odot_{k\neq m}}]^\dagger, \tag{5.10}$$

$$\widehat{\mathbf{A}_n}' \leftarrow [\mathbf{X}_{(n)} - \mathbf{A}''_n(\mathbf{B}''_n)^\top]\mathbf{B}'_n[\gamma\mathbf{I} + (\mathbf{B}'_n)^\top\mathbf{B}'_n]^\dagger, \tag{5.11}$$

where $\widehat{\mathbf{A}_m}$ is the updated non-sensitive latent factor matrix, $\widehat{\mathbf{A}_n}'$ is the updated non-sensitive dimensions of the sensitive latent factor matrix, $\mathbf{I}$ is an identity matrix.

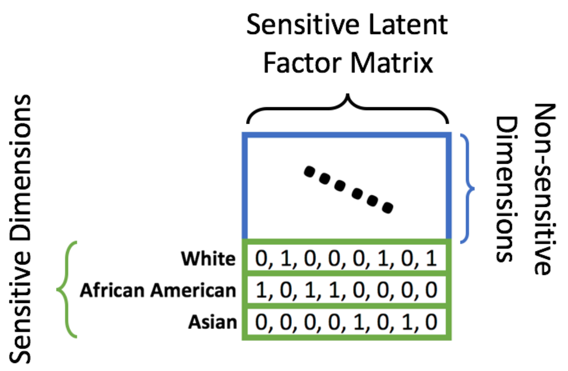


Figure 5.3: In the case of multi-category sensitive dimensions (e.g., by ethnicity), this example shows how to generate the sensitive latent factor matrix. Reprinted with permission from [10].

In the second optimization step for the sensitive mode, we need the gradient of $\mathfrak{F} = \frac{\lambda}{2}\|\mathbf{A}_n''^{\top}\mathbf{A}_n'\|_{\mathrm{F}}^2$, which is calculated by $\frac{\partial \mathfrak{F}}{\partial \mathbf{A}_n'} = \lambda \mathbf{A}_n''(\mathbf{A}_n'')^{\top}\mathbf{A}_n'$.

The entire optimization process is described in Algorithm 2. We can also use Newton's method to replace gradient descent, which has the advantages of fast convergence speed and less effort of tedious learning rate tuning. Newton's method requires the second-order derivative of $\mathfrak{F}$, which is calculated by: $\frac{\partial^2 \mathfrak{F}}{\partial \mathbf{A}_n' \partial \mathbf{A}_n'^{\top}} = \lambda \mathbf{A}_n'' \mathbf{A}_n''^{\top}$.

Finally, line 8 of Algorithm 2 should be modified to be "Update $\mathbf{A}_n' \leftarrow \mathbf{A}_n' - (\frac{\partial^2 \mathfrak{F}}{\partial \mathbf{A}_n' \partial \mathbf{A}_n'^{\top}})^{\dagger} \frac{\partial \mathfrak{F}}{\partial \mathbf{A}_n'}$ ".

5.2.4 Generalizing FATR

So far, we have focused on a single binary sensitive attribute. We show here how to handle multi-feature cases (i.e., there are more than one sensitive attributes) and multi-category cases (i.e., the attribute can take more than two values). We also consider multisided attributes (i.e., more than one mode is considered sensitive), which is important in real-world applications [25]. Such multi-feature and multi-category cases are challenging for traditional regularization-based approaches [120, 131] since a regularization term can only account for fairness between two groups defined by one binary attribute. By missing the multi-way interactions among multiple categorical sensitive attributes, such a regularization-based approach may lead to less effective (and less fair) recommendation. However, the multi-feature and multi-category problems fit naturally into the proposed FATR framework.

For the multi-feature case, we need to put all the sensitive features into the corresponding sensitive latent factor matrices, and add the orthogonal constraints to all the sensitive modes to isolate and extract all the sensitive information. For the multi-category case, we need to have c columns in the sensitive dimensions if the attribute can take c distinct values. Hence, the binary-feature case is just a special multi-category case where $c = 2$. Every dimension only indicates one specific category, for example, dimension i has value 1 for the entities who belong to category c_i and has value 0 for other instances. One example is shown in Figure 5.3.

For ease of presentation, we assume there are three sensitive attributes, one is denoted as S_1 belonging to the mode-n_1, another two are denoted as S_2 and S_3 belonging to the mode-n_2. And all of them have three available categories to take. For example, in the Twitter experts recommender, we want to enhance the fairness for experts with different genders (Female, Male, and Unspecified) and with different ethnicities (African-American, Asian, and White), and at the same time we also want to augment the fairness for the topics with different numbers of experts (small, medium, and large). The sensitive features of S_1 is $\mathbf{S}_1$ which has 3 columns. The sensitive features of S_2 and S_3 are $\mathbf{S}_2$ and $\mathbf{S}_3$, and concatenate them together to be $\mathbf{S}_{2,3}$ which has 6 columns. Then the objective function is:

$$\begin{aligned}
\underset{\boldsymbol{\mathcal{X}},\mathbf{A}_1\ldots\widetilde{\mathbf{A}_{n_1}}\ldots\widetilde{\mathbf{A}_{n_2}}\ldots\mathbf{A}_N}{\text{minimize}} \quad & \boldsymbol{\mathcal{L}} = \|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}_1 \ldots \widetilde{\mathbf{A}_{n_1}} \ldots \widetilde{\mathbf{A}_{n_2}} \ldots \mathbf{A}_N]\!]\|_{\mathrm{F}}^2 \\
& \quad + \frac{\lambda}{2}(\|\mathbf{A}''^{\top}_{n_1}\mathbf{A}'_{n_1}\|_{\mathrm{F}}^2 + \|\mathbf{A}''^{\top}_{n_2}\mathbf{A}'_{n_2}\|_{\mathrm{F}}^2) \\
& \quad + \frac{\gamma}{2}\sum_{i=1}^{N}\|\mathbf{A}_i\|_{\mathrm{F}}^2 \\
\text{subject to} \quad & \boldsymbol{\Omega} \circledast \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{T}}, \\
& \widetilde{\mathbf{A}_{n_1}} = [\mathbf{A}'_{n_1} \;\; \mathbf{A}''_{n_1}], \\
& \widetilde{\mathbf{A}_{n_2}} = [\mathbf{A}'_{n_2} \;\; \mathbf{A}''_{n_2}], \\
& \mathbf{A}''_{n_1} = \mathbf{S}_1, \;\; \mathbf{A}''_{n_2} = \mathbf{S}_{2,3},
\end{aligned} \tag{5.12}$$

where $\widetilde{\mathbf{A}_{n_1}}$ and $\widetilde{\mathbf{A}_{n_2}}$ are the sensitive latent factor matrices, $\mathbf{A}'_{n_1}$ and $\mathbf{A}'_{n_2}$ are non-sensitive dimensions of $\widetilde{\mathbf{A}_{n_1}}$ and $\widetilde{\mathbf{A}_{n_2}}$, $\mathbf{A}''_{n_1}$ and $\mathbf{A}''_{n_2}$ are the sensitive dimensions of $\widetilde{\mathbf{A}_{n_1}}$ and $\widetilde{\mathbf{A}_{n_2}}$ which have the same values as $\mathbf{S}_1$ and $\mathbf{S}_{2,3}$.

We can still use Algorithm 2 to solve the new objective function with only line 8 and line 9 modified to update both $\widetilde{\mathbf{A}_{n_1}}$ and $\widetilde{\mathbf{A}_{n_2}}$. In the same way, the proposed method can be applied to model the cases with more sensitive features and more categories.

5.2.5 Experiments

In this section, we empirically evaluate the proposed approach w.r.t three aspects – recommendation quality, recommendation fairness, and effectiveness of eliminating sensitive information – over four scenarios: (i) under the traditional matrix scenario; (ii) then by comparing matrix to tensor approaches; (iii) by varying the degrees of bias and sparsity to better explore their impact; and (iv) evaluating FATR's generalizability to the multi-feature and multi-category scenario.

5.2.5.1 Experimental Settings

Dataset

We consider a real-world movie dataset, a real-world social media dataset, and a collection of synthetic datasets for which we can vary degrees of bias and sparsity. We report the average results over three runs for all datasets.

- **MovieLens.** We use the MovieLens 10k dataset [96], keeping all movies with at least 35 ratings. Following previous works [11, 119], we use the year of the movie as a sensitive attribute and consider movies before 1996 as old movies. Those more recent are considered new movies. In total, we have 671 users, 373 old movies, and 323 new movies. The sparsity of the dataset is 11.4%. Since we focus on implicit recommendation, we consider ratings to be 1 if the original ratings are higher than 3.5, otherwise 0. Then we have 15,579 positive ratings for new movies and 20,387 positive ratings for old movies, which reflects the bias in the dataset. We randomly split the dataset into 90% for training and 10% for testing.

- **User-Expert-Topic Twitter Data.** We use a Twitter dataset introduced in [135] that has 589 users, 252 experts, and 10 topics (e.g., news, sports). There are $16,867$ links from users to experts across these topics capturing that a user is interested in a particular expert. The sparsity of this dataset is 1.136%. We consider race as a sensitive attribute and aim to divide experts into two groups: whites and non-whites. We apply the Face++ (https://www.faceplusplus.com/) API to the images of each expert in the dataset to derive ethnicity. In total, we find 126 whites and 126 non-whites, with 11,612 positive ratings for white experts but only 5,255 for non-whites. Since this implicit feedback scenario has no negative observations, we randomly pick unobserved data samples to be negative feedback with probability of 0.113% (one tenth of the sparsity). We randomly split the dataset into 70% training and 30% testing.

- **Synthetic Expert Datasets.** To gauge the impact of degrees of bias and sparsity, we further generate a suite of synthetic expert datasets. We first generate three latent factor matrices by uniform distribution for user, expert, and topic, which are $\mathbf{U} \in \mathbb{R}^{200\times 30}$, $\mathbf{E} \in \mathbb{R}^{100\times 30}$, and $\mathbf{T} \in \mathbb{R}^{5\times 30}$. Second, we set the last dimension of $\mathbf{E}$ to be the binary sensitive features to indicate two groups and make the numbers of the two groups equal. Third, we add constant values v_u and v_t to the sensitive dimensions of $\mathbf{U}$ and $\mathbf{T}$ to increase the bias. Then, we get the preference ratings tensor of size $200 \times 100 \times 5$ by calculating the Khatri-Rao product of $\mathbf{U}$, $\mathbf{E}$, and $\mathbf{T}$. Last, we set 1 to ratings lager than 0.5, meaning the user selects the expert with respect to the topic and set 0 to ratings less than 0.5, meaning the user does not select the expert with respect to the topic. We randomly sample the 1's based on a probability p to produce the final observed dataset. By adjusting the values of v_u and v_t, we generate datasets with varying imbalance of the proportion of the number of the positive ratings for the protected group over the total number of the positive ratings. With a proportion of 0.1, only 10% of positive ratings are for the protected group. We call this an extreme bias case. Similarly, we generate datasets with high bias (0.2), middle bias (0.3), and low bias (0.4). We further generate three levels of sparsity, which are 0.01 (high sparsity), 0.02 (middle sparsity), and 0.03 (low sparsity) by adjusting p. As a result, we have 12 different datasets: High Bias / High Sparsity, High Bias / Middle Sparsity, etc. All datasets are randomly split into 70% for training and 30% for testing.

Metrics

We consider metrics to capture recommendation quality, recommendation fairness, and the impact of eliminating sensitive information.

To measure *recommendation quality*, we adopt **Precision@k** (P@K) and **Recall@k** (R@K), defined as:

$$P@k = \frac{1}{|\mathbb{U}|}\sum_{u\in\mathbb{U}}\frac{|\mathbb{O}_u^k \cap \mathbb{O}_u^+|}{k}, \quad R@k = \frac{1}{|\mathbb{U}|}\sum_{u\in\mathbb{U}}\frac{|\mathbb{O}_u^k \cap \mathbb{O}_u^+|}{\mathbb{O}_u^+}, \tag{5.13}$$

where $\mathbb{O}_u^+$ is the set of items user u gives positive feedback to in test set and $\mathbb{O}_u^k$ is the predicted top-k recommended items. We also consider **F1@k** score, which can be calculated by $F1@k = 2 \cdot (P@k \times R@k)/(P@k + R@k)$. We set $k = 15$ in our experiments.

To measure *recommendation fairness*, we use two complementary metrics. The first one is the absolute difference between mean ratings of different groups (**MAD**):

$$MAD = |\frac{\sum R^{(0)}}{|R^{(0)}|} - \frac{\sum R^{(1)}}{|R^{(1)}|}|, \tag{5.14}$$

where $R^{(0)}$ and $R^{(1)}$ are the predicted ratings for the two groups and $|R^{(i)}|$ is the total number of ratings for group i. Larger values indicate greater differences between the groups, which we interpret as unfairness.

The second measure is the Kolmogorov-Smirnov statistic (**KS**), which is a nonparametric test for the equality of two distributions. The KS statistic is defined as the area difference between two empirical cumulative distributions of the predicted ratings for groups:

$$KS = |\sum_{i=1}^{T} l \times \frac{\mathfrak{G}(R^{(0)}, i)}{|R^{(0)}|} - \sum_{i=1}^{T} l \times \frac{\mathfrak{G}(R^{(1)}, i)}{|R^{(1)}|}|, \tag{5.15}$$

where T is the number of intervals for the empirical cumulative distribution, l is the size of each interval, $\mathfrak{G}(R^{(0)}, i)$ counts how many ratings are inside the i^{th} interval for group 0. In our experiments, we set $T = 50$. Lower values of KS indicate the distributions are more alike, which we interpret as being more fair.

MAD and KS can be directly applied to binary sensitive attributes. For multi-category cases, we need to calculate MAD and KS statistics for every dominant group vs. protected group pair among the categories. For example, for the attribute of ethnicity with three categories – White (W), African-American (AA) and Asian (A), where AA and A are the two groups to be protected – we need to calculate the MAD and KS metrics for two pairs – W vs. AA, and W vs. A.

Note that we measure the fairness in terms of MAD and KS metrics across groups rather than within individuals, since absolute fairness for every individual may be overly strict and in opposition to personalization needs of real-world recommenders.

To evaluate the impact of *eliminating sensitive information*, we use the sum of absolute cosine angles between non-sensitive and sensitive dimensions (**SCos**):

$$SCos = \sum_{i=1}^{r-2} \sum_{j=r-1}^{r} |cos(\mathbf{A}_i, \mathbf{A}_j)|, \tag{5.16}$$

where $\mathbf{A}_i$ and $\mathbf{A}_j$ are one non-sensitive dimension and one sensitive dimension indexed by i and j, and cos calculates the cosine angle between two vectors.

We also use the sum of absolute *Pearson correlation coefficient* between non-sensitive and sensitive dimensions (**SCorr**) to quantify the sensitive information:

$$SCorr = \sum_{i=1}^{r-2} \sum_{j=r-1}^{r} |corr(\mathbf{A}_i, \mathbf{A}_j)|, \tag{5.17}$$

where $corr$ calculates the *Pearson correlation coefficient* between two vectors. The lower the SCos and SCorr are, the better the sensitive information elimination result is.

For multi-category cases, Scos and Scorr should be calculated for every category separately to evaluate whether the impact of the multi-category attribute is eliminated with respect to all categories. Following our ethnicity example from earlier, we need to calculate SCos and SCorr for W, AA, and A separately.

Baselines

To evaluate FATR, we consider two variations – one using Gradient Descent (**FT(G)**) and one using Newton's Method (**FT(N)**) – in comparison with two tensor-based alternatives:

- *Ordinary Tensor Completion (**OTC**)*: The first is the conventional CP-based tensor completion method using ALS optimization algorithm as introduced in Section 5.2.2.2. This baseline incorporates no notion of fairness, so it will provide a good sense of the state-of-the-art recommendation quality we can achieve.

- *Regularization-based Tensor Completion (**RTC**)*: The second one is an extension from the fairness-enhanced matrix completion with regularization method introduced in [11, 120, 102], which adds a bias penalization term to the objective function. For tensor-based recommenders, we can use the regularized objective function (5.18) to enforce the statistical parity.

$$\begin{aligned} \underset{\mathcal{X},\mathbf{A}_1,\ldots,\mathbf{A}_N}{\text{minimize}} \quad & \mathcal{L} = \|\mathcal{X} - [\![\mathbf{A}_1,\ldots,\mathbf{A}_N]\!]\|_{\text{F}}^2 \\ & \quad + \frac{\lambda}{2}(\frac{1}{n_0}\|\boldsymbol{\Omega_0} \circledast [\![\mathbf{A}_1,\ldots,\mathbf{A}_N]\!]\|_{\text{F}}^2 \\ & \qquad - \frac{1}{n_1}\|\boldsymbol{\Omega_1} \circledast [\![\mathbf{A}_1,\ldots,\mathbf{A}_N]\!]\|_{\text{F}}^2)^2 \\ \text{subject to} \quad & \boldsymbol{\Omega} \circledast \mathcal{X} = \mathcal{T}, \end{aligned} \tag{5.18}$$

  where $\lambda > 0$ is the regularization coefficient, $\boldsymbol{\Omega_0}$ and $\boldsymbol{\Omega_1}$ are the indicator tensors to indicate the ratings of the two groups determined by the binary sensitive attribute, n_0 and n_1 are the numbers of ratings to the two groups. We use Gradient Descent to solve this optimization problem.

Since the MovieLens data has only two modes (users and movies), we consider matrix versions of our tensor based methods (named **FM(G)** and **FM(N)**) versus matrix baselines of *Ordinary Matrix Completion (**OMC**)* and *Regularization-based Matrix Completion (**RMC**)* corresponding to RTC.

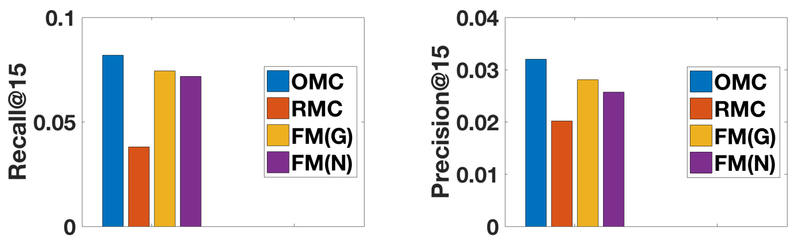


Figure 5.4: Recommendation quality (MovieLens). Reprinted with permission from [10].

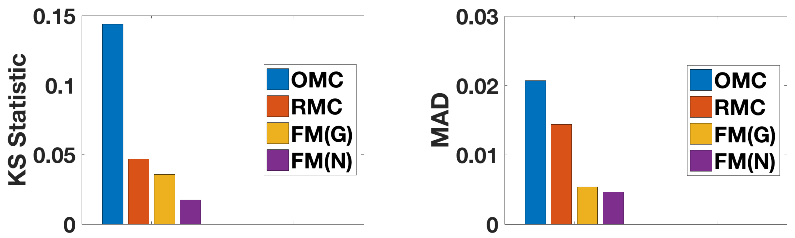


Figure 5.5: Recommendation fairness (MovieLens). Reprinted with permission from [10].

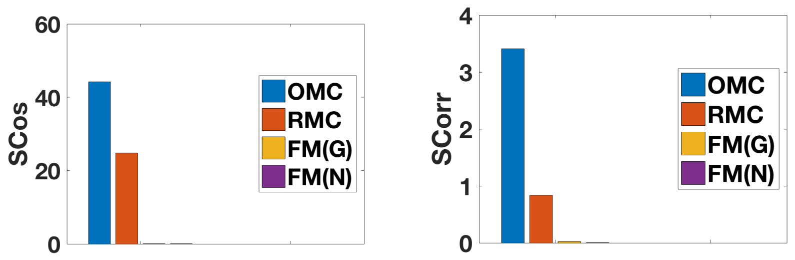


Figure 5.6: Eliminating Sensitive Information (MovieLens). Reprinted with permission from [10].

5.2.5.2 RQ1: Compare Matrix-based Methods

For the first experiment, we evaluate the four matrix-based approaches (OMC, RMC, FM(G) and FM(N)) over the MovieLens dataset. We set 50 as the latent dimension for all the methods and fine tune all other parameters; for our proposed methods we set $\lambda = 1$, $\gamma = 0.05$ and learning rate as 0.001 for FM(G), and $\lambda = 0.00001$ and $\gamma = 0.01$ for FM(N).

We begin by considering the quality of recommendation of the four approaches in Figure 5.4. As expected, the baseline with no notion of fairness – OMC – results in the best overall precision and recall. Of the three fairness-aware approaches, the regularization-based approach – RMC –

Table 5.2: Comparison for recommending Twitter experts. Reprinted with permission from [10].

Methods	R@15	P@15	KS	MAD	SCos	SCorr
OMC	0.3467	0.0842	0.1660	0.0122	7.8035	1.9131
OTC	0.4384	0.0958	0.3662	0.0333	21.9193	8.7732
RMC	0.1609	0.0702	0.1521	0.0086	15.3268	0.8534
RTC	0.3003	0.0515	0.2003	0.0171	23.6818	1.4036
FM(G)	0.4045	0.0891	0.0523	0.0037	0.3081	0.1407
FT(G)	0.4180	0.0870	0.0195	0.0024	0.0936	0.0396
FM(N)	0.3298	0.0687	0.0245	0.0044	0.0022	0.0115
FT(N)	0.3975	0.0786	0.0173	0.0029	0.0001	0.0001

performs considerably below the others, with our two approaches (FM) providing performance fairly close to OMC. This suggests that recommendation quality can be preserved, but leaves open the question of whether we can add fairness.

Hence, we turn to the impact on fairness of the four approaches. Figure 5.5 presents the KS statistic and MAD (recall, lower is better). We can see that all three fairness-aware approaches – RMC, FM(G) and FM(N) – have a strong impact on the KS statistic in comparison with OMC. And for MAD, we see that both FM(G) and FM(N) achieve much better ratings difference in comparison with RMC, indicating that we can induce aggregate statistics that are fair between the two sides of the sensitive attribute (old vs. new).

Last, we exam how well do these approaches perform from the perspective of sensitive information elimination. The left figure in Figure 5.6 shows the SCos statistic, while the right figure shows the SCorr statistic. Both of them demonstrate that the proposed FATR framework can eliminate sensitive information to a great extent, but RMC can only reduce the SCos to around half of that of OMC and SCorr to around one third of that of OMC.

5.2.5.3 RQ2: Compare Matrix vs. Tensor-Based Methods

We next turn to evaluating the expert recommendation task over the real-world Twitter dataset. Here we consider the tensor-based approaches – OTC, RTC, plus FT(G) and FT(N). To further evaluate the impact of moving from a matrix view to a tensor view, we also consider the purely matrix-based approaches, which compute users preferences on experts for each topic independently. We set 20 as the latent dimension for all the methods and fine tune all other parameters; for our proposed methods we set $\lambda = 1$, $\gamma = 0.05$ and learning rate as 0.001 for FM(G), and $\lambda = 0.00001$ and $\gamma = 0.01$ for FM(N). We show the results for all of our metrics in Table 5.2.

First, let's focus on the differences between matrix and tensor approaches. We observe that the tensor-based approaches mostly provide better recommendation quality (Precision@k and Recall@k) in comparison with the matrix-based approaches. Since the expert dataset is naturally multi-aspect, the tensor approaches better model the multi-way relationships among users, experts, and topics. We see that the fairness quality (KS and MAD) of matrix-based methods are better than tensor-based ones for the baselines methods (OMC vs OTC, and RMC vs RTC), but the fairness improves for our proposed methods when we move from matrix to tensor. We see a similar result for the impact on eliminating sensitive information (SCos and SCorr).

Second, let's consider the empirical results across approaches. We see that: (i) the proposed methods are slightly worse than OTC from the perspective of recommendation quality, but keep the difference small, and FM methods also have comparable recommendation performance with OMC; (ii) FT(G) and FT(N) provide the best fairness enhancement results, and FM(G) and FM(N) also alleviate the unfairness a lot compared with other matrix-based methods. RTC and RMC improve the fairness as well, but their effects are not as good as the proposed methods; (iii) the proposed approaches can effectively eliminate the sensitive information; and (iv) comparing the two variations of FATR, FT(G) always provides better recommendation quality but performs worse than FT(N) in terms of fairness enhancement and sensitive information elimination, which may be because Newton's method has stronger effects on optimization leading to more effective minimization of the orthogonal constraint term $\mathfrak{F} = \frac{\lambda}{2}\|\mathbf{A}_n''^{\top}\mathbf{A}_n'\|_{\mathrm{F}}^2$ in Equation (5.4).

In addition, the $\frac{\gamma}{2}\sum_{i=1}^{N}\|\mathbf{A}_i\|_{\mathrm{F}}^2$ term in our proposed objective function (5.4) may influence the recommendation performance, but the baselines do not have it, which may be an unfair comparison. Therefore, we do another experiment using OTC, RTC, OMC, and RMC with the L2-norm term. The recommendation performance results and fairness enhancement results are shown in Figure 5.7. We can conclude similarly that the proposed methods still perform well in terms of

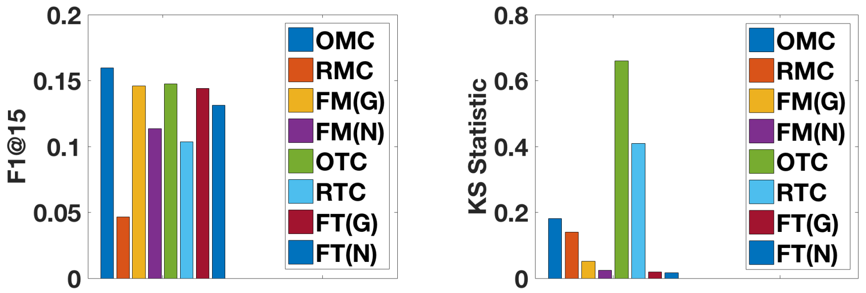
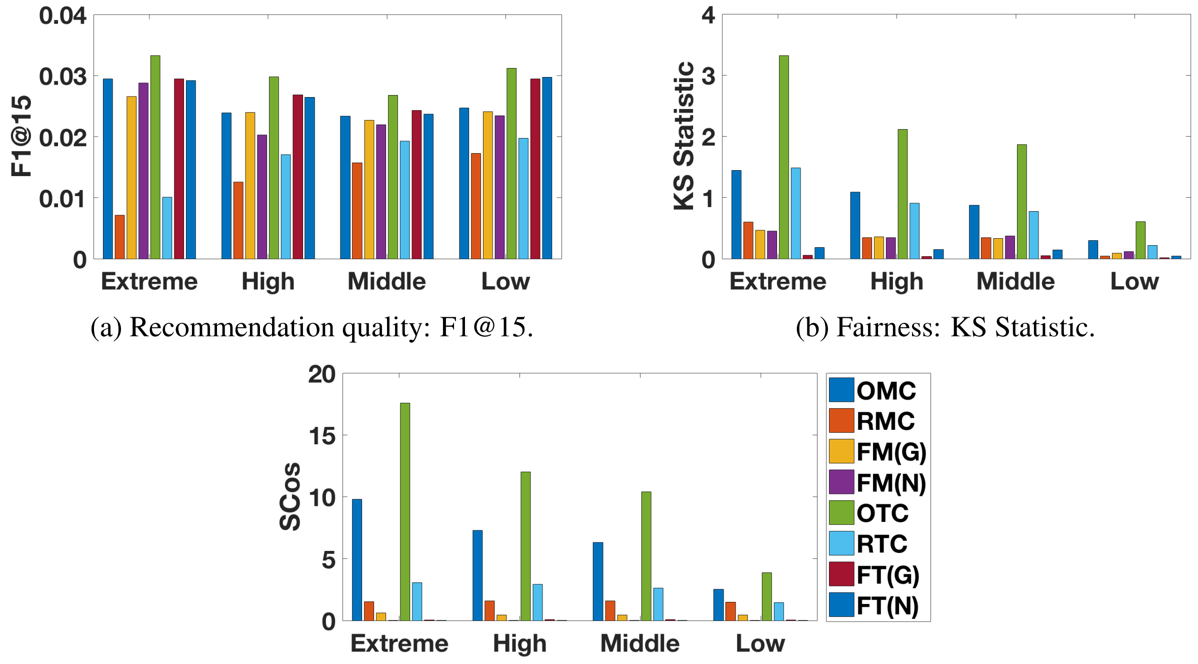


Figure 5.7: F1@15 and KS statistics of the proposed methods and the baselines with L2-norm terms. Reprinted with permission from [10].



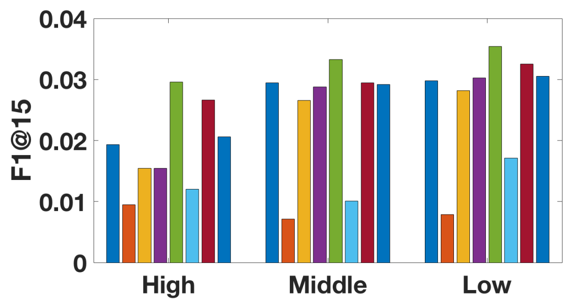
(a) Recommendation quality: F1@15.

(b) Fairness: KS Statistic.

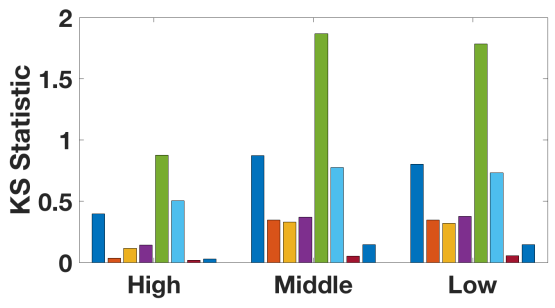
(c) Sensitive information elimination: SCos.

Figure 5.8: Evaluating the impact of bias (Synthetic Experts dataset). Reprinted with permission from [10].

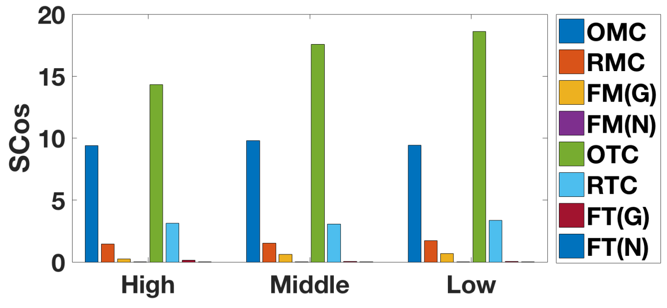
both recommendation quality and fairness enhancement. Besides, we find that compared with the baselines without L2-norm terms, the baselines with L2-norm have better recommendation quality but higher bias.



(a) Recommendation quality: F1@15.



(b) Fairness: KS Statistic.



(c) Sensitive information elimination: SCos.

Figure 5.9: Evaluating the impact of sparsity under extreme bias (Synthetic Experts dataset). Reprinted with permission from [10].

5.2.5.4 RQ3: Performance with Varying Bias and Sparsity

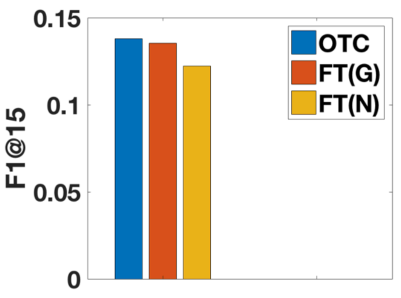
Next, we consider the impact of bias and sparsity through a series of experiments over the synthetic expert datasets. For parameters setting, the latent factor dimension is set as 20, we set $\lambda = 0.25$, $\gamma = 0.05$, learning rate as 0.002 for FT(G), and $\lambda = 0.0001$ and $\gamma = 0.1$ for FT(N). We set the latent dimension smaller than 30 on purpose, which is the number of factors we use when generating the synthetic dataset, because in practice, researchers tend to use low dimensional latent factor to model user-item interactions.

We begin by investigating the impact of bias – do our methods perform well even in cases of extreme bias? Or do they require only moderate amounts? We fix the sparsity level at 0.02 and vary the bias levels from Low, Middle, High, and Extreme. We show in Figure 5.8a the *F1@15* of all eight methods on these four datasets. The results show that OTC always performs best, but FT(G) does not reduce the *F1 score* much compared with other methods. Overall, tensor-based methods outperform matrix-based methods. And within matrix-based methods, FM(G) is just a little worse than OMC, and much better than RMC. Further, we can observe that as the bias level goes down, the recommendation quality is improved for all six fairness-aware methods in comparison with OTC and OMC. For example the F1@15 score difference between OTC and FT(G) are 0.0041, 0.0034, 0.0031, and 0.0015 for the extreme, high, medium, and low bias situations respectively. Figure 5.8b shows that for all the bias levels, the proposed FT(G) and FT(N) can enhance the fairness to a great extent. We can also observe that RTC and RMC can reduce the unfairness compared with conventional completion methods, but their performances are not comparable with the proposed methods. One outlier is the result produced by RMC in the low bias dataset. Although it reduces the KS as low as proposed methods do, its recommendation quality is not ideal. We also study how well do these methods eliminate sensitive information as demonstrated in Figure 5.8c. The figure shows that the proposed methods (both tensor-based and matrix-based) have the lowest SCos values, meaning that our methods can effectively eliminate the sensitive information. From these results, we can conclude that the proposed approaches provide good and consistent performance over all the bias levels.

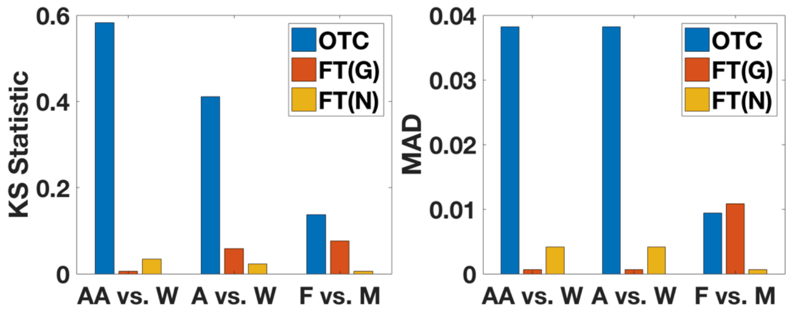
Furthermore, we also analyze the results for datasets with various sparsities with bias level fixed at the extreme level. The results are shown in Figure 5.9. We can draw the similar conclusion from it that the proposed methods reduce the unfairness without much loss of the prediction accuracy for different sparsities. However, in addition to this conclusion, these results also imply that with the dataset being denser, the unfairness is more severe. Combining the observations from Figure 5.8 and Figure 5.9, we can learn that: (i) tensor completion possesses more algorithmic bias than matrix completion does; and (ii) the proposed FATR methods have consistent fairness-enhancement and sensitive information eliminating performance on datasets with various bias levels and sparsities. We also compute MAD and SCorr statistics, showing similar patterns as KS and SCos.

5.2.5.5 RQ4: Multiple Features and Multiple Categories

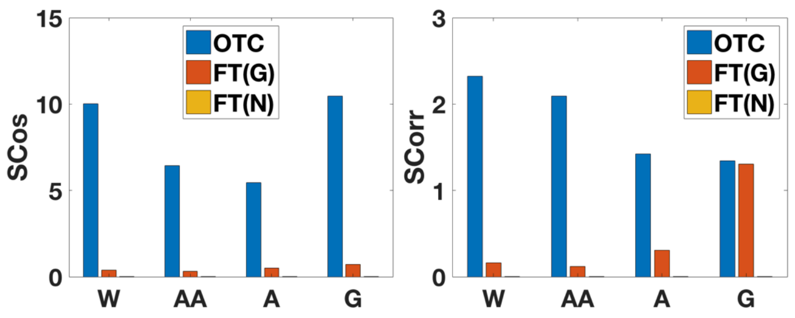
Finally, by the same dataset as used in Section 5.2.5.3, we investigate how the proposed model performs with multiple features and multiple categories (as introduced in Section 5.2.4). We consider both gender and ethnicity as sensitive attributes. For ease of experimentation, we consider



(a) Recommendation quality: F1@15.



(b) Fairness Quality: KS and MAD.



(c) Sensitive information elimination: SCos and SCorr.

Figure 5.10: Evaluating the generalizing ability to multi features and multi categories. Reprinted with permission from [10].

gender (G) as a binary feature (M=Male, F=Female). For ethnicity, we consider three categories: White (W), African-American (AA), and Asian (A). Our dataset contains 126 whites with 11,612 positive feedbacks, 80 Asian people with 2,238 feedbacks, and 46 African-Americans with 3,017 positive feedbacks. The distribution of the gender is: 163 males and 83 females. Males have 10,160 positive ratings and females have 6,707 positive ratings. Other settings of the experiment are the same as single-feature experiment as described in Section 5.2.5.3.

For the parameters settings, we set the latent factor dimension as 20 for OTC, but 25 for FT(G) and FT(N) because there are 5 dimensions occupied by the sensitive dimensions, and we want similar degree of freedom for all the methods. We set $\lambda = 0.05$, $\gamma = 0.05$, and the learning rate 0.002 for FT(G), and $\lambda = \gamma = 1$ for FT(N). Because regularization-based models cannot be easily applied to this scenario, we compare FT(G) and FT(N) with OTC.

Figure 5.10a illustrates that the proposed methods can keep a relatively high recommendation quality compared with the OTC. Figure 5.10b shows that FT(N) model have a good fairness enhancement performance for both attributes. FT(G) works well on the ethnicity feature but a little unsatisfactory for the gender feature. One possible reason is that FT(G) requires more effort for parameter tuning. Moreover, the bias related to the ethnicity feature is more severe than the unfairness related to the gender feature, which makes it harder for the model to decrease the unfairness for the gender feature. Figure 5.10c shows the relationships between the latent factor matrices from the three methods and all the sensitive features. It implies that the FATR models can alleviate the impact of the sensitive information from all the sensitive attributes. Further, we see that FT(N) works well for all attributes including gender (which is challenging for the other approaches).

5.2.6 Summary

In this section, based on well known concepts of statistical parity and equal opportunity, we first propose two fairness metrics designed specifically for personalized ranking recommendation tasks. Then we empirically show that the influential Bayesian Personalized Ranking model is vulnerable to the inherent data imbalance and tends to generate unfair recommendations w.r.t. the proposed fairness metrics. Next we propose a novel fairness-aware personalized ranking model incorporating adversarial learning to augment the proposed fairness metrics. At last, extensive experiments show the effectiveness of the proposed model over other state-of-the-art alternatives.

5.3 Measuring and Enhancing Fairness in Personalized Ranking Recommender Systems

Although the proposed FATR produces the outstanding performance for enhancing recommendation fairness, one big challenge unsolved is that the fairness is measured and enhanced based on predicted scores. However, the predicted score is the intermediate step toward a ranking list of items, which is the final recommendation result we show to users. Fair scores does not necessarily lead to fair ranking result. Therefore, it is required to further study how to measure and enhance fairness on ranking-based recommender systems. In this section, we introduce our work on developing new fairness measurements and new fairness-enhancement algorithms for personalized ranking recommender systems.

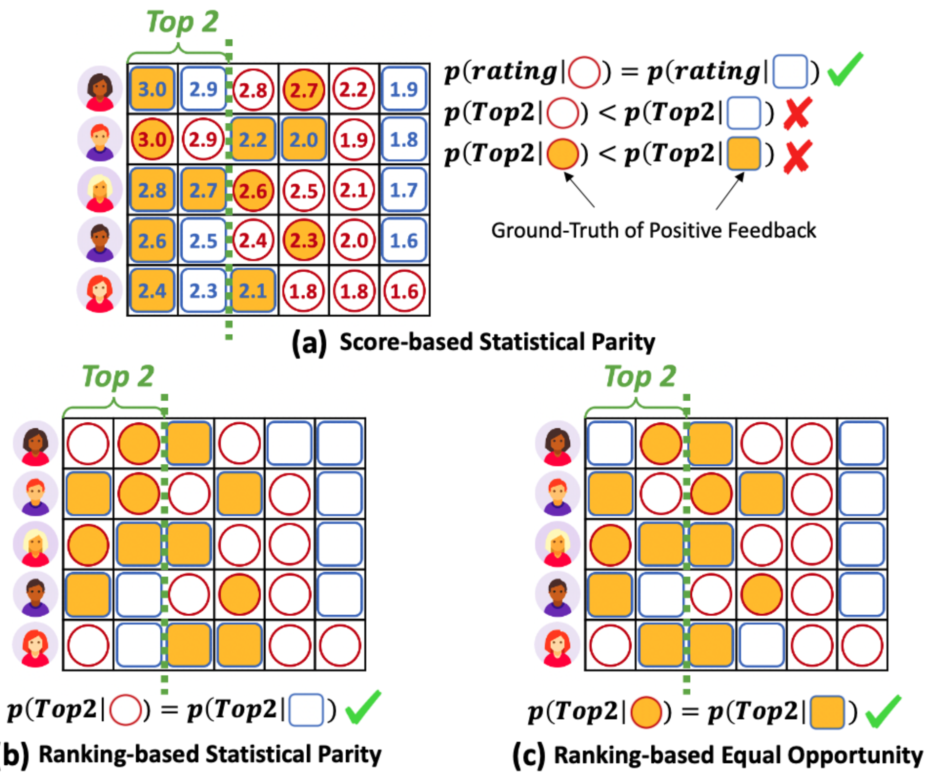


Figure 5.11: (a) is an example following score-based statistical parity from previous works [11, 10]. (b) and (c) are examples of the proposed metrics: (b) is ranking-based statistical parity, and (c) is ranking-based equal opportunity. Reprinted with permission from [12].

5.3.1 Introduction

Previous works on recommendation fairness [102, 10, 120, 119, 11, 121] mainly focus on investigating how to produce similar predicted score distributions for different groups of items (in other words, by removing the influence of group information when predicting preference scores). The main drawback of these works is that they mainly focus on the perspective of *predicted preference scores* [102, 120, 119, 121, 11, 10]. In practice, however, predicted scores are an intermediate step towards a ranked list of items that serves as the final recommendation result, and having similar predicted scores does not necessarily lead to a fair ranking result, suggesting the importance of directly measuring fairness over rankings instead of scores.

Take the most frequently adopted concept – statistical parity (or demographic parity) [102, 10, 11] – as an example. Fairness of statistical parity encourages identical predicted score distributions for different groups. Figure 5.11a shows an example following the statistical parity constraint, where rows represent users, and blue squares and red circles represent two different groups of items. Each group has three items. The numbers in the matrix are the predicted scores from users to items. The items are ranked by the scores in descending order for each user. The top-2 items are recommended to users, and the yellow background in some circle and square items represents positive ground-truth. In this example, the blue square group and red circle group have exactly the same score distribution (a uniform distribution over the range $[1.6, 3.0]$). However, the recommendation result is unfair for the red circle group, which only has two items being recommended. Therefore, we propose two new fairness metrics calculated directly over the ranking results – *ranking-based statistical parity (RSP)* and *ranking-based equal opportunity (REO)*.

Motivating Example: Ranking-Based Statistical Parity

Unlike traditional score-based statistical parity, RSP encourages the probabilities for different groups being recommended (being ranked within top k) to be similar. Figure 5.11b provides an example following RSP, where the probabilities of being ranked within the top-2 for the two groups are both $\frac{1}{3}$. However, while RSP encourages a fair ranking result, it neglects the intrinsic group quality or user preference imbalance of the data, which may exert unfairness to the popular group. For instance, in Figure 5.11a and Figure 5.11b, the yellow background in some entries represents that the marked item has the ground-truth that it is liked by the corresponding user (i.e. it forms a matched user-item pair). Then, we see that the blue square group is more preferred by users than the red circle group since the numbers of positive user-item pairs in the two groups are $8 : 4$. Hence, keeping the same recommendation probability in Figure 5.11b is unfair for the blue square group.

Motivating Example: Ranking-Based Equal Opportunity

Taking into consideration the ground-truth of user-item matching, we propose the metric REO that encourages the probabilities of being correctly recommended to matched users to be the same across item groups. A positive example is given in Figure 5.11c, where the probability of being ranked in top-2 to matched users is $\frac{1}{2}$ for both item groups. Figure 5.11a and Figure 5.11b show the negative examples, where in Figure 5.11a the ranking result shows favor to the blue square group, while Figure 5.11b is biased towards the red circle group.

Due to their distinct characteristics, RSP and REO are applicable for different scenarios. RSP-based fairness is usually important and needs to be enhanced for situations where there are sensitive attributes attached to the subjects being recommended, such as considering fairness across genders or races of job candidates when they are recommended to companies for hiring. Because in this case, social ethics require equal exposure probability for different demographic groups in the system. While REO-based fairness applies to more general scenarios without sensitive attribute, like movies or songs recommendation. Unfairness regarding REO in these systems can potentially exerts damaging influence to both users and item providers. On the one hand, user needs corresponding to minority item groups are not fully acknowledged, leading to lower user satisfaction. On the other hand, item providers of minority groups may not receive enough exposure hurting their economic gains.

Contributions

With these two metrics in mind, we empirically demonstrate that a fundamental recommendation model – Bayesian Personalized Ranking (BPR) [27] – is vulnerable to unfairness, which motivates our efforts to address it. Then, we show how to improve the fairness based on the two introduced metrics through a debiased personalized ranking model (DPR) that has two key features: a multi-layer perceptron adversary that seeks to enhance the score distribution similarity among item groups and a KL Divergence based regularization term that aims to normalize the score distribution for each user. Incorporating these two components together, RSP (or REO depending on how we implement the adversary learning) based fairness can be significantly improved while preserving recommendation quality at the same time. Extensive experiments on three public datasets show the effectiveness of the proposed model over state-of-the-art alternatives. In general, DPR is able to reduce the unfairness corresponding to the two metrics for BPR by 67.3% on average (with an improvement of 48.7% over the best baseline), while only decreasing $F1@15$ versus BPR by 4.1% on average (with an improvement of 16.4% over the best baseline).

5.3.2 Fairness in Personalized Ranking

In this section, we first describe the personalized ranking problem and ground our discussion through a treatment of Bayesian Personalized Ranking (BPR). Next, we introduce two proposed fairness metrics for personalized ranking. Last, we empirically demonstrate that BPR is vulnerable to data bias and tends to produce unfair recommendations.

5.3.2.1 Bayesian Personalized Ranking

Given N users $\mathcal{U} = \{1, 2, \ldots, N\}$ and M items $\mathcal{I} = \{1, 2, \ldots, M\}$, the personalized ranking problem is to recommend a list of k items to each user u based on the user's historical behaviors $\mathcal{I}_u^+ = \{i, j, \ldots\}$, where $i, j, \ldots$ are the items u interacts with before (and so can be regarded as implicit positive feedback). Bayesian Personalized Ranking (BPR) [27] is one of the most influential methods to solve this problem, which is the foundation of many cutting edge personalized ranking algorithms (e.g. [30, 18]). BPR adopts matrix factorization [29] as the base and minimizes a pairwise ranking loss, formalized as:

$$\min_{\mathbf{\Theta}} \mathcal{L}_{BPR} = -\sum_{u \in \mathcal{U}} \sum_{\substack{i \in \mathcal{I}_u^+ \\ j \in \mathcal{I} \setminus \mathcal{I}_u^+}} ln\, \sigma(\widehat{y}_{u,i} - \widehat{y}_{u,j}) + \frac{\lambda_{\mathbf{\Theta}}}{2} \|\mathbf{\Theta}\|_{\mathrm{F}}^2, \tag{5.19}$$

where $\widehat{y}_{u,i}$ and $\widehat{y}_{u,j}$ are the predicted preference scores calculated by the matrix factorization model for user u to positive item i and sampled negative item j; $\sigma(\cdot)$ is the Sigmoid function; $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm; $\mathbf{\Theta}$ represents the model parameters, i.e., $\mathbf{\Theta} = \{\mathbf{P}, \mathbf{Q}\}$, where $\mathbf{P}$ and $\mathbf{Q}$ are the latent factor matrices for users and items; and $\lambda_{\mathbf{\Theta}}$ is the trade-off weight for the l2 regularization.

With the trained BPR, we can predict the preference scores toward all un-interacted items and rank them in descending order for user u. A list of items with the top k largest scores $\{R_{u,1}, R_{u,2}, \ldots, R_{u,k}\}$ will be recommended to user u, where $R_{u,k}$ is the item id at the ranked k position.

5.3.2.2 Fairness Metrics for Personalized Ranking

However, there is no notion of fairness in such a personalized ranking model. Here, we assume a set of A sensitive groups $\mathcal{G} = \{g_1, g_2, \ldots, g_A\}$, and every item in $\mathcal{I}$ belongs to one or more groups. A group here could correspond to gender, ethnicity, or other item attributes. We define a function $G_{g_a}(i)$ to identify whether item i belongs to group g_a, if it does, the function returns 1, otherwise 0. The high-level goal of a fairness-aware recommender is to enhance the fairness among different groups such that no group is under-recommended.

Previous definitions of recommendation fairness [10, 11, 141] have two main drawbacks: (i) the fairness metrics are calculated over the predicted scores, which are not aligned with the ranking results; and (ii) the definitions are mainly based on the concept of statistical parity, which does not take the ground-truth of user-item matching into account. Therefore, we propose two new fairness metrics over the ranking of items from different groups.

Ranking-based Statistical Parity (RSP)

Statistical parity requires the probability distributions of model outputs for different input groups to be the same. In a similar way, for the personalized ranking task, statistical parity can be defined as forcing the ranking probability distributions of different item groups to be the same. Because conventionally only the top-k items will be recommended to users, we focus on the probabilities of being ranked in top-k, which is also aligned with basic recommendation quality evaluation metrics such as *precision@k* and *recall@k*. As a result, we propose the ranking-based statistical parity metric – RSP, which encourages $P(R@k|g = g_1) = P(R@k|g = g_2) = \ldots = P(R@k|g = g_A)$, where $R@k$ represents 'being ranked in top-k', and $P(R@k|g = g_a)$ is the probability of items in group g_a being ranked in top-k. Formally, we calculate the probability as follows:

$$P(R@k|g = g_a) = \frac{\sum_{u=1}^{N} \sum_{i=1}^{k} G_{g_a}(R_{u,i})}{\sum_{u=1}^{N} \sum_{i \in \mathcal{I} \setminus \mathcal{I}_u^+} G_{g_a}(i)}, \tag{5.20}$$

where $\sum_{i=1}^{k} G_{g_a}(R_{u,i})$ calculates how many un-interacted items from group g_a are ranked in top-k for user u, and $\sum_{i \in \mathcal{I} \setminus \mathcal{I}_u^+} G_{g_a}(i)$ calculates how many un-interacted items belong to group g_a for u. Last, we compute the *relative standard deviation* (to keep the same scale for different k) over the probabilities to determine $RSP@k$:

$$RSP@k = \frac{std(P(R@k|g = g_1), \ldots, P(R@k|g = g_A))}{mean(P(R@k|g = g_1), \ldots, P(R@k|g = g_A))}, \tag{5.21}$$

where $std(\cdot)$ calculates the standard deviation, and $mean(\cdot)$ calculates the mean value.

Ranking-based Equal Opportunity (REO)

Our second metric is based on the concept of equal opportunity [93, 94, 95], which encourages the true positive rates (TPR) of different groups to be the same. Take a binary classification task with two groups as an example, equal opportunity requires:

$$P(\widehat{c} = 1|g = 0, c = 1) = P(\widehat{c} = 1|g = 1, c = 1), \tag{5.22}$$

where c is the ground-truth label, $\widehat{c}$ is the predicted label; $P(\widehat{c} = 1|g = 0, c = 1)$ represents the TPR for group 0, $P(\widehat{c} = 1|g = 1, c = 1)$ is the TPR for group 1. Similarly, in the personalized ranking system, equal opportunity demands the ranking based TPR for different groups to be the same. We can define the TPR as the probability of being ranked in top-k given the ground-truth that the user likes the item, noted as $P(R@k|g = g_a, y = 1)$, where $y = 1$ represents items are liked by users. The probability can be calculated by:

$$P(R@k|g = g_a, y = 1) = \frac{\sum_{u=1}^{N} \sum_{i=1}^{k} G_{g_a}(R_{u,i}) Y(u, R_{u,i})}{\sum_{u=1}^{N} \sum_{i \in \mathcal{I} \setminus \mathcal{I}_u^+} G_{g_a}(i) Y(u, i)}, \tag{5.23}$$

where $Y(u, R_{u,i})$ identifies the ground-truth label of a user-item pair $(u, R_{u,i})$, if item $R_{u,i}$ is liked by user u, returns 1, otherwise 0 (in practice, $Y(u, i)$ identifies whether a user-item pair (u, i) is in the test set); $\sum_{i=1}^{k} G_{g_a}(R_{u,i}) Y(u, R_{u,i})$ counts how many items in test set from group g_a are ranked in top-k for user u, and $\sum_{i \in \mathcal{I} \setminus \mathcal{I}_u^+} G_{g_a}(i) Y(u, i)$ counts the total number of items from group

g_a in test set for user u. Similar to RSP, we calculate the relative standard deviation to determine $REO@k$:

$$REO@k = \frac{std(P(R@k|g = g_1, y = 1) \ldots P(R@k|g = g_A, y = 1))}{mean(P(R@k|g = g_1, y = 1) \ldots P(R@k|g = g_A, y = 1))}. \tag{5.24}$$

For classification tasks, TPR is the recall of classification, and for personalized ranking, the probability $P(R@k|g = g_a, y = 1)$ is *recall@k* of group g_a. In other words, mitigating REO-based bias requires $recall@k$ for different groups to be similar.

Note that for both $RSP@k$ and $REO@k$, *lower values indicate the recommendations are fairer*. In practice, RSP is particularly important in scenarios where humans or items with sensitive attributes are recommended (such as political news). Because RSP-based fairness in these scenarios leads to social issues like gender discrimination during recruiting or political ideology unfairness during election campaigns. Conversely, REO-based fairness is supposed to be enhanced in general item recommendation systems so that no user need is ignored, and all items have the chance to be exposed to matched users who like them.

5.3.2.3 Data-driven Study

In this section, we empirically show that BPR is vulnerable to imbalanced data and tends to produce unfair recommendation based on metrics RSP and REO. Since there is no standard public dataset related to recommendation fairness with sensitive attributes, we adopt three public real-world datasets that have been extensively used in previous works [102, 18, 142]. However, conclusions we draw should still hold if we analyze the fairness on datasets with sensitive features because the fundamental problem definition and the mechanism leading to unfairness are exactly the same as the experiments here.

- **MovieLens 1M (ML1M)** [96] is a movie rating dataset, where we treat all ratings as positive feedback indicating users are interested in rated movies. We consider the recommendation bias for movie genres of ‘Sci-Fi’, ‘Adventure’, ‘Crime’, ‘Romance’, ‘Childrens’, and ‘Horror’, and remove other films, resulting in $6,036$ users, $1,481$ items, and $526,490$ interactions.

Table 5.3: Group information in the three datasets. Reprinted with permission from [12].

	Group	#Item	#Feedback	$\frac{\#feedback}{\#item}$
ML1M	Sci-Fi	271	157,290	580.41
	Adventure	276	133,946	485.31
	Crime	193	79,528	412.06
	Romance	447	147,501	329.98
	Children's	248	72,184	291.06
	Horror	330	76,370	231.42
	Relative std	-	-	**0.33**
Yelp	American(New)	1610	91,519	56.84
	Japanese	946	45,508	48.11
	Italian	1055	46,434	44.01
	Chinese	984	36,729	37.33
	Relative std	-	-	**0.17**
Amazon	Grocery	749	49,646	66.28
	Office	892	37,776	42.35
	Pet	518	16,260	31.39
	Tool	606	14,771	24.37
	Relative std	-	-	**0.44**

- **Yelp** (`https://www.yelp.com/dataset/challenge`) is a review dataset for businesses. We regard the reviews as the positive feedback showing user interests and only consider restaurant businesses. We investigate the recommendation bias among food genres of 'American(New)', 'Japanese', 'Italian', and 'Chinese', resulting in $8,263$ users, $4,420$ items, and $211,721$ interactions.

- **Amazon** [98] contains product reviews on the Amazon e-commerce platform. We regard user purchase behaviors as the positive feedback, and consider recommendation bias among product categories of 'Grocery', 'Office', 'Pet', and 'Tool', resulting in $4,011$ users, $2,765$ items, and $118,667$ interactions.

Moreover, Table 5.3 lists the details of each group in the datasets, including the number of items, the number of feedback, and the ratio between them $\frac{\#feedback}{\#item}$. We use this ratio to identify the intrinsic data imbalance. The higher the ratio is, the more this group is favoured by users, and the relative standard deviation of ratios for all groups can indicate overall bias in the dataset. Hence, the Amazon and ML1M datasets contain relatively high bias; and Yelp has lower bias,

Table 5.4: Ranking probability distributions and RSP and REO metrics on three datasets by BPR. Reprinted with permission from [12].

		$P(R@k\|g)$			$P(R@k\|g, y=1)$		
	Genres	@5	@10	@15	@5	@10	@15
ML1M	Sci-Fi	.00654	.01306	.01949	.09497	.16819	.22922
	Adventure	.00516	.01022	.01521	.08884	.15808	.21657
	Crime	.00456	.00888	.01318	.07469	.13017	.17941
	Romance	.00327	.00665	.01002	.06448	.12003	.16366
	Children's	.00251	.00494	.00742	.05852	.10470	.14464
	Horror	.00176	.00354	.00533	.05399	.10132	.13985
	RSP or REO	**.41054**	**.40878**	**.40579**	**.20885**	**.19316**	**.18933**
Yelp	American(New)	.00154	.00302	.00449	.06345	.10904	.14497
	Japanese	.00111	.00219	.00328	.04770	.08207	.11106
	Italian	.00093	.00194	.00297	.03890	.07087	.09658
	Chinese	.00072	.00146	.00222	.03376	.05626	.07961
	RSP or REO	**.28005**	**.26376**	**.25224**	**.24515**	**.24290**	**.22253**
Amazon	Grocery	.00283	.00572	.00869	.03931	.07051	.09297
	Office	.00165	.00336	.00506	.01196	.02039	.03180
	Pet	.00185	.00348	.00501	.04815	.07807	.10215
	Tool	.00082	.00165	.00250	.00552	.01105	.01519
	RSP or REO	**.40008**	**.40672**	**.41549**	**.68285**	**.65756**	**.62175**

but American(New) restaurants still have $\frac{\#feedback}{\#item}$ around 1.5 times higher than that of Chinese restaurants.

We run BPR on these datasets and analyze the ranking probability distributions. The detailed model hyper-parameter settings and data splitting are described in Section 5.3.4.1. Table 5.4 presents $P(R@k|g)$ and $P(R@k|g, y = 1)$ for different groups on three datasets by BPR, where we consider $k = 5, 10$, and 15. We also list the metrics $RSP@k$ and $REO@k$. From the table, we have three major observations:

(i) For all datasets, the ranking probabilities are very different among groups, e.g., in ML1M, $P(R@5|g =$ Sci-Fi$)$ is four times higher than $P(R@5|g =$ Horror$)$, and $P(R@5|g =$ Sci-Fi$, y = 1)$ is two times higher than $P(R@5|g =$ Horror$, y = 1)$. And the high values of $RSP@k$ and $REO@k$ for all k and datasets demonstrate the biased recommendations by BPR.

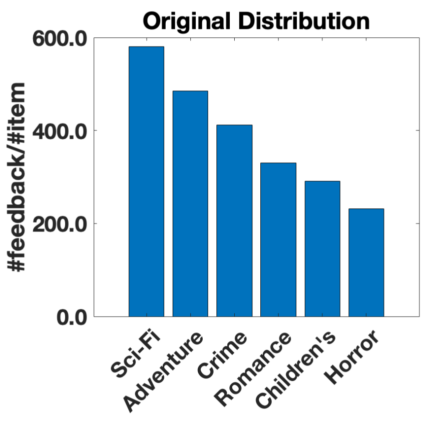
(ii) The distributions of $P(R@k|g)$ and $P(R@k|g, y = 1)$ for all datasets basically follow the distributions of $\frac{\#feedback}{\#item}$ shown in Table 5.3, and sometimes the deviations of the ranking probability distributions are even larger than $\frac{\#feedback}{\#item}$ distributions, for example, the relative standard deviation of $P(R@15|g)$ in ML1M is 0.4058 while that of $\frac{\#feedback}{\#item}$ is 0.3344, which indicates that BPR preserves or even amplifies the inherent data imbalance.

(iii) As k decreases, the values of $RSP@k$ and $REO@k$ increase. In other words, the results are unfairer for items ranked at top positions. This phenomenon is harmful for recommenders since attention received by items increases rapidly with rankings getting higher [143], and top-ranked items get most of attention from users.

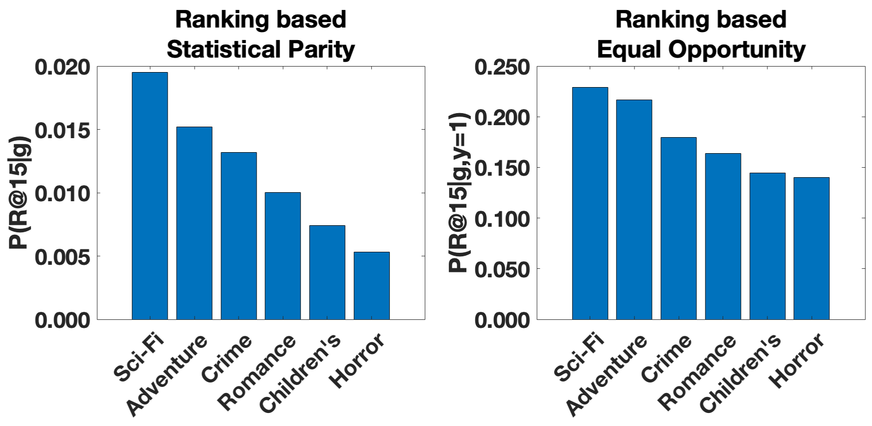
Moreover, we also plot the original $\frac{\#feedback}{\#item}$ distribution of ML1M in Figure 5.12a and the ranking probability distributions by BPR in Figure 5.12b, which visually confirms our conclusion that BPR inherits data bias and produces unfair recommendations. This conclusion motivates the design of a fairness-aware personalized ranking framework as the models we propose here. Figure 5.12c shows the ranking probability distributions generated by the proposed Debiased Personalized Ranking models, illustrating more evenly distributed and fairer recommendations compared to BPR.

5.3.3 Proposed Method

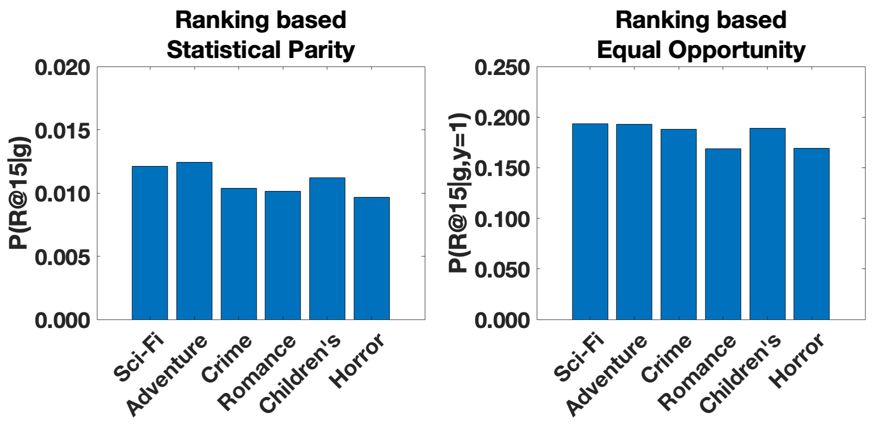
Previous works on fairness-aware recommendation [10, 11, 119] mainly focus on forcing different groups to have similar score distributions, which cannot necessarily give rise to fair personalized rankings, as shown in Figure 5.11a. One key reason is that users have different rating scales, which means the high score from one user to an item does not necessarily result in a high ranking, and a low score does not lead to a low ranking. Conversely, if every user has an identical score distribution, the value ranges of the scores within the top-k for all users will be the same, as demonstrated in Figure 5.13a. Then, the top-k scores in different group score distributions (noted as $p(\widehat{y}|g)$) will also cover the same value range. Last, as illustrated in Figure 5.13b, if we enforce identical score distribution for different groups, the proportions of top-k scores in the whole distribution for different groups will be the same, i.e., we have the same probability $p(R@k|g)$ for different groups (the definition of RSP). Similarly, if the positive user-item pairs in different groups have the same score distribution (noted as $p(\widehat{y}|g, y = 1)$), we will have the same probability



(a) Original.



(b) BPR results.



(c) DPR results.

Figure 5.12: The original distribution of #feedback/#item over different groups of ML1M data, and the ranking top15 probability distributions (both statistical parity and equal opportunity based) produced by BPR and proposed DPR. Reprinted with permission from [12].

$p(R@k|g, y = 1)$ for different groups (the definition of REO), as presented in Figure 5.13c. Based on this intuition, the proposed DPR first enhances the score distribution similarity between different groups by adversarial learning, and then normalizes user score distributions to the standard

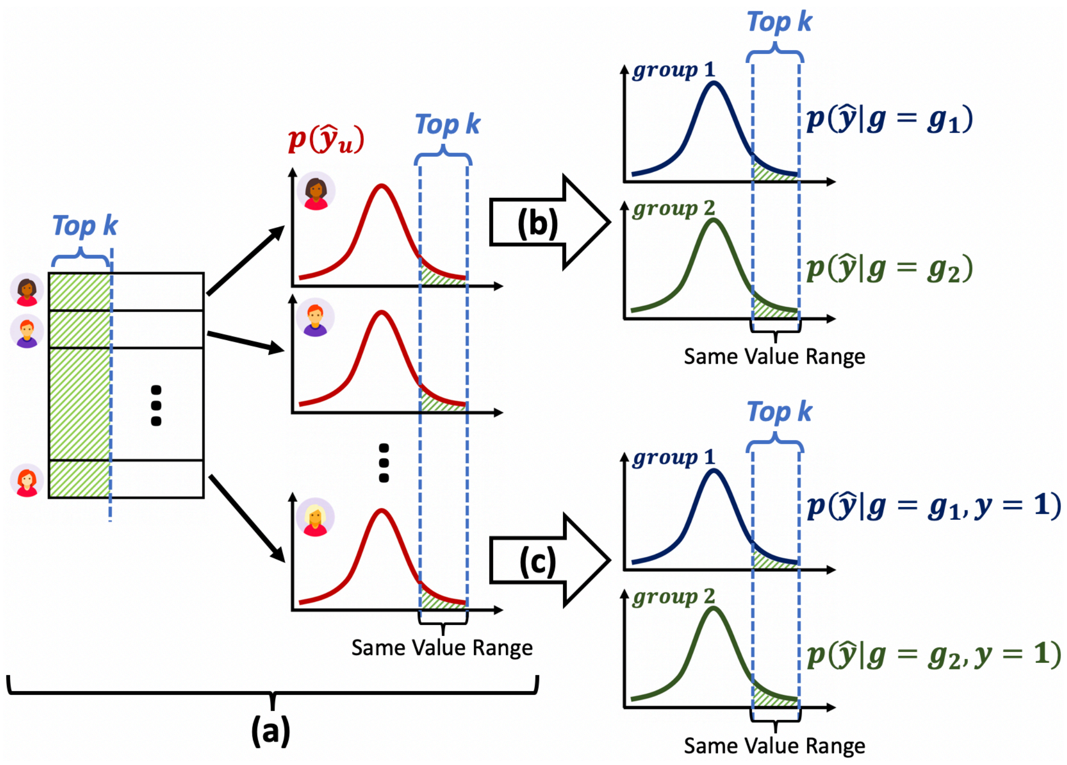


Figure 5.13: Illustration of the intuition of the proposed DPR. Reprinted with permission from [12].

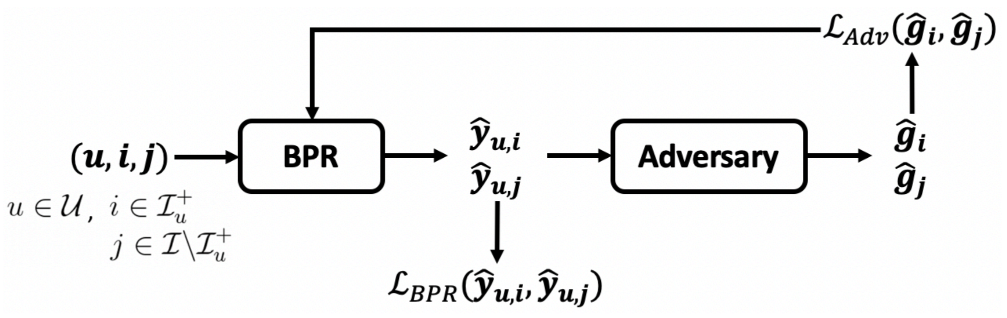
normal distribution by a Kullback-Leibler Divergence (KL) loss. We introduce the two components of DPR and the model training process in the following subsections.

5.3.3.1 Enhancing Score Distribution Similarity

Adversarial learning has been widely applied in supervised learning [95, 144, 93] to enhance model fairness, with theoretical guarantees and state-of-the-art empirical performance. Inspired by these works, we propose to leverage adversarial learning to enhance the score distribution similarity between different groups. We first take the metric RSP as the example to elaborate the proposed method, and then generalize it to REO. Last, we show the advantages of the proposed adversarial learning over previous methods.

Adversary for RSP

The intuition of adversarial learning is to play a minimax game between the BPR model and a discriminator. The discriminator is to classify the groups of the items based on the predicted user-item scores by BPR. And BPR does not only need to minimize the recommendation error, but



$\mathcal{L}_{Adv}(\hat{g}_i, \hat{g}_j)$
(u, i, j)
$u \in \mathcal{U},\ i \in \mathcal{I}_u^+$
$j \in \mathcal{I} \backslash \mathcal{I}_u^+$
BPR
$\hat{y}_{u,i}$
$\hat{y}_{u,j}$
Adversary
$\hat{g}_i$
$\hat{g}_j$
$\mathcal{L}_{BPR}(\hat{y}_{u,i}, \hat{y}_{u,j})$

Figure 5.14: The architecture of the adversarial learning. Reprinted with permission from [12].

also needs to prevent the discriminator from correctly classifying the groups. If the discriminator cannot accurately recognize the groups given the outputs from BPR, then the predicted score distributions will be identical for different groups. Figure 5.14 presents the architecture of the adversarial learning framework, where each training user-item pair (u, i) is first input to a conventional BPR model; then the output of BPR, $\widehat{y}_{u,i}$ is fed into a multi-layer perceptron (MLP) to classify the groups $\widehat{\mathbf{g}}_i$ of the given item i. $\widehat{\mathbf{g}}_i \in [0,1]^A$ is the output of the last layer of MLP activated by the *sigmoid* function, representing the probability of i belonging to each group, e.g., $\widehat{\mathbf{g}}_{i,a}$ means the predicted probability of i belonging to group g_a. The MLP is the adversary, which is trained by maximizing the likelihood $\mathcal{L}_{Adv}(\mathbf{g}_i, \widehat{\mathbf{g}}_i)$, and BPR is trained by minimizing the ranking loss shown in Equation 2.8 as well as minimizing the adversary objective $\mathcal{L}_{Adv}(\mathbf{g}_i, \widehat{\mathbf{g}}_i)$. $\mathbf{g}_i \in \{0,1\}^A$ is the ground-truth groups of item i, if i is in group g_a, $\mathbf{g}_{i,a} = 1$, otherwise 0. We adopt the log-likelihood as the objective function for the adversary:

$$\max_{\mathbf{\Psi}} \mathcal{L}_{Adv}(i) = \sum_{a=1}^{A} (\mathbf{g}_{i,a} log\, \widehat{\mathbf{g}}_{i,a} + (1 - \mathbf{g}_{i,a}) log\,(1 - \widehat{\mathbf{g}}_{i,a})), \tag{5.25}$$

where we denote $\mathcal{L}_{Adv}(\mathbf{g}_i, \widehat{\mathbf{g}}_i)$ as $\mathcal{L}_{Adv}(i)$ for short, and $\mathbf{\Psi}$ is the parameters of the MLP adversary. Combined with the BPR model, the objective function can be formulated as:

$$\begin{gathered}\min_{\mathbf{\Theta}} \max_{\mathbf{\Psi}} \sum_{u \in \mathcal{U}} \sum_{\substack{i \in \mathcal{I}_u^+ \\ j \in \mathcal{I} \backslash \mathcal{I}_u^+}} \mathcal{L}_{BPR}(u, i, j) + \alpha(\mathcal{L}_{Adv}(i) + \mathcal{L}_{Adv}(j)), \\ \text{where } \mathcal{L}_{BPR}(u, i, j) = -ln\, \sigma(\widehat{y}_{u,i} - \widehat{y}_{u,j}) + \frac{\lambda_{\mathbf{\Theta}}}{2} \|\mathbf{\Theta}\|_{\mathrm{F}}^2,\end{gathered} \tag{5.26}$$

and α is the trade-off parameter to control the strength of the adversarial component.

Adversary for REO

As for REO, we demand the score distributions of positive user-item pairs rather than all the user-item pairs to be identical for different groups. Therefore, instead of feeding both scores for positive and sampled negative user-item pairs $\widehat{y}_{u,i}$ and $\widehat{y}_{u,j}$, we only need to feed $\widehat{y}_{u,i}$ into the adversary as:

$$\min_{\mathbf{\Theta}} \max_{\mathbf{\Psi}} \sum_{u \in \mathcal{U}} \sum_{\substack{i \in \mathcal{I}_u^+ \\ j \in \mathcal{I} \setminus \mathcal{I}_u^+}} \mathcal{L}_{BPR}(u,i,j) + \alpha \mathcal{L}_{Adv}(i). \tag{5.27}$$

Advantages of adversarial learning

There are two existing approaches to achieve a similar effect: a regularization-based method [120, 11, 102]; and a latent factor manipulation method [10]. The advantages of the proposed adversarial learning over previous works can be summarized as: (i) it can provide more effective empirical performance than other methods; (ii) it is flexible to swap in different bias metrics (beyond just RSP and REO); (iii) it can handle multi-group circumstances; and (iv) it is not coupled with any specific recommendation models and can be easily adapted to methods other than BPR (such as more advanced neural networks).

5.3.3.2 Individual User Score Normalization

After the enforcement of distribution similarity, the next step towards personalized ranking fairness is to normalize the score distribution for each user. We can assume the score distribution of every user follows the normal distribution because based on the original BPR paper [27], every factor in the user or item latent factor vector follows a normal distribution, then $\mathbf{P}_u^\top \mathbf{Q}_i$ (for a given user u, $\mathbf{P}_u$ is a constant and $\mathbf{Q}_i$ is a vector of normal random variables) follows a normal distribution as well. Thus we can normalize the score distribution of each user to the standard normal distribution by minimizing the KL Divergence between the score distribution of each user

Algorithm 3: Training algorithm for DPR-RSP.

```
   Input: Training data $\mathcal{D}$, adversarial regularizer $\alpha$, KL-loss regularizer $\beta$, $L_2$ regularizer
          $\lambda_{\mathbf{\Theta}}$, and learning rate for BPR $\eta_{BPR}$, learning rate for the adversary $\eta_{Adv}$ ;
   Output: BPR parameters $\mathbf{\Theta}$;
1  Randomly Initialize model parameters $\mathbf{\Theta}$ for BPR, and $\mathbf{\Psi}$ for the adversary;
2  repeat
3      for $(u', i', j')$ in $\mathcal{D}$ do
4          Update $\mathbf{\Psi}$ based on $\boldsymbol{\mathcal{L}}_{Adv}(i')$ by backpropagation;
5          Update $\mathbf{\Psi}$ based on $\boldsymbol{\mathcal{L}}_{Adv}(j')$ by backpropagation;
6      Randomly draw a mini-batch $\mathcal{D}^{mini}$ from $\mathcal{D}$;
7      for $(u, i, j)$ in $\mathcal{D}^{mini}$ do
8          Update $\mathbf{\Theta}$ based on gradient $\frac{\partial(\boldsymbol{\mathcal{L}}_{BPR}(u,i,j)+\alpha(\boldsymbol{\mathcal{L}}_{Adv}(i)+\boldsymbol{\mathcal{L}}_{Adv}(j))+\beta\boldsymbol{\mathcal{L}}_{KL})}{\partial\mathbf{\Theta}}$;
9  until converge;
10 Return $\mathbf{\Theta}$;
```

and a standard normal distribution as the KL-loss:

$$\mathcal{L}_{KL} = \sum_{u \in \mathcal{U}} D_{\text{KL}}(q_{\mathbf{\Theta}}(u)||\mathcal{N}(0,1)), \tag{5.28}$$

where $q_{\mathbf{\Theta}}(u)$ is the empirical distribution of predicted scores for user u, and $D_{\text{KL}}(\cdot||\cdot)$ computes KL Divergence between two distributions.

5.3.3.3 Model Training

Combining the KL-loss with Equation 5.26 leads to the complete DPR models to optimize RSP, noted as DPR-RSP:

$$\min_{\mathbf{\Theta}} \max_{\mathbf{\Psi}} \sum_{u \in \mathcal{U}} \sum_{\substack{i \in \mathcal{I}_u^+ \\ j \in \mathcal{I} \setminus \mathcal{I}_u^+}} (\mathcal{L}_{BPR}(u,i,j) + \alpha(\mathcal{L}_{Adv}(i) + \mathcal{L}_{Adv}(j))) + \beta \mathcal{L}_{KL}, \tag{5.29}$$

where β is the trade-off parameter to control the strength of KL-loss. Similarly, we can optimize REO by combining KL-loss with Equation 5.27 to arrive at a DPR-REO model as well. Note that although the proposed DPR is built with BPR as the model foundation, it is in fact flexible enough to be adapted to other recommendation algorithms, such as more advanced neural networks [17].

Then, we train the model in a mini-batch manner. The model training process for DPR-RSP can be summarized in Algorithm 3. In each epoch, there are two phases: we first update the

Table 5.5: Characteristics of the three 2-group datasets. Reprinted with permission from [12].

	#Users	#Items	#Ratings	Density
ML1M-2	5,562	543	215,549	7.14%
Yelp-2	6,310	2,834	117,978	0.66%
Amazon-2	3,845	2,487	84,656	0.89%

weights in the MLP adversary to maximize the classification objective, then update the BPR to minimize pairwise ranking loss, classification objective and KL-loss all together. Following the adversarial training proposed in [144], in each iteration, we first update the MLP adversary by the whole dataset, then update BPR by a mini-batch, which empirically leads to faster convergence. And in practice, we usually first pre-train the BPR model for several epochs.

Algorithm 3 can be easily extended to the DPR-REO model by two minor modifications: first remove the negative samples update steps for MLP adversary (line 5); then replace the update rule of line 8 with the gradient:

$$\frac{\partial((\boldsymbol{\mathcal{L}}_{BPR}(u,i,j) + \alpha\boldsymbol{\mathcal{L}}_{Adv}(i)) + \beta\boldsymbol{\mathcal{L}}_{KL})}{\partial\Theta}. \tag{5.30}$$

5.3.4 Experiments

In this section, we empirically evaluate the proposed model w.r.t. the two proposed bias metrics as well as the recommendation quality. We aim to answer three key research questions: **RQ1** What are the effects of the proposed KL-loss, adversary, and the complete model DPR on recommendations? **RQ2** How does the proposed DPR perform compared with other state-of-the-art debiased models from the perspectives of improving fairness and recommendation quality preserving? and **RQ3** How do hyper-parameters affect the DPR framework?

5.3.4.1 Experimental Settings

Datasets

The three datasets used in the experiments have been introduced in Section 5.3.2.3. Since the state-of-the-art baselines can only work for binary group cases, to answer **RQ2**, we create subsets keeping the most popular and least popular groups in the original datasets: **ML1M-2** ('Sci-Fi' vs 'Horror'), **Yelp-2** ('American(New)' vs. 'Chinese'), and **Amazon-2** ('Grocery' vs. 'Tool'). The specifics of the 2-group datasets are presented in Table 5.5. All datasets are randomly split into 60%, 20%, 20% for training, validation, and test sets. Note that there is no standard public dataset with sensitive features, thus we use public datasets for general recommendation scenarios to evaluate the performance of enhancing RSP-based fairness. However, conclusions we draw should still hold if we analyze the fairness-enhancement performance on datasets with sensitive features because the fundamental problem definition and the mechanism leading to unfairness are exactly the same as the experiments here.

Metrics

In the experiments, we need to consider both recommendation quality and recommendation fairness. For recommendation fairness, we report $RSP@k$ and $REO@k$ as described in Section 5.3.2.2. As for the recommendation quality we adopt $F1@k$. We report the results with $k = 5, 10$, and 15. Note that we also measure NDCG in the experiments, which shows the same pattern as F1, hence we only report $F1@k$ for conciseness.

Baselines

We compare the proposed DPR with unfair method BPR shown in Section 5.3.2.1 and two state-of-the-art fairness-aware recommendation methods:

FATR [10]. This is a tensor-based method, which enhances the score distribution similarity for different groups by manipulating the latent factor matrices. We adopt the 2D matrix version of this approach. Note that FATR is designed for statistical parity based metric, hence we do not have high expectation for the performance w.r.t. equal opportunity.

Reg [120, 11, 102]. The most commonly used fairness-aware method for two-group scenarios, which penalizes recommendation difference by minimizing a regularization term. Following [11], we adopt the squared difference between the average scores of two groups for all items as the regularization to improve RSP, denoted as **Reg-RSP**. For REO, we adopt the squared difference between the average scores of positive user-item pairs as the regularization, denoted as **Reg-REO** (it is similar to DPR-REO but enhances the distribution similarity by static regularization rather than adversary).

To have a fair comparison, we modify the loss functions of all baselines to the BPR loss in Equation 5.19. Moreover, to align the baselines with the bias metrics for ranking, we further add the proposed KL-loss introduced in Section 5.3.3.2 to both baselines.

Reproducibility

Go to `https://github.com/Zziwei/Item-Underrecommendation-Bias` for all code and data. We implement the proposed model using Tensorflow [59] and adopt Adam [60] optimization algorithm. We tune the hyper-parameters of the models involved by the validation set, the basic rules are: (i) we search the hidden dimension over $\{10, 20, 30, 40, 50, 60, 70, 80\}$; (ii) search the L_2 regularizer $\lambda_{\mathbf{\Theta}}$ over $\{0.01, 0.05, 0.1, 0.5, 1.0\}$; (iii) search the adversary regularizer α over range $[500, 10000]$ with step 500; (iv) search the KL-loss regularizer β over range $[10, 70]$ with step 10; and (v) search the model specific weight in FATR over $\{0.01, 0.05, 0.1, 0.5, 1.0\}$, and model specific weight for Reg-RSP and Reg-REO over the range $[1000, 10000]$ with step 2000. Note that selections of α and β should consider the balance between recommendation quality and recommendation fairness.

There are two sets of experiments: experiments over multi-group datasets (ML1M, Yelp, and Amazon) to answer **RQ1** and **RQ3**; and experiments over binary-group datasets (ML1M-2, Yelp-2, and Amazon-2) to answer **RQ2**.

In the first set of experiments, for all three datasets: we set 20 as the hidden dimensions for BPR, DPR-RSP, and DPR-REO; we set the learning rate 0.01 for BPR, and η_{BPR} 0.01 as well for DPR-RSP and DPR-REO. For all methods, we set $\lambda_{\mathbf{\Theta}} = 0.1$ for ML1M and Amazon; set $\lambda_{\mathbf{\Theta}} = 0.05$ for Yelp. As for adversary learning rate η_{Adv}, we set 0.005 for ML1M and Yelp, 0.001 for Amazon. For all three datasets, we set $\alpha = 5000$ for DPR-RSP. As for DPR-REO, we set $\alpha = 1000$ for ML1M, 5000 for Yelp, and 10000 for Amazon.

In the second set of experiments, we set different hidden dimensions for different datasets, but

Table 5.6: Comparison between BPR w/o KL-loss for JS Divergences among user score distributions over three datasets. Reprinted with permission from [12].

	ML1M	Yelp	Amazon
BPR	0.1540	0.0808	0.0836
BPR w/ KL-loss	0.0571	0.0254	0.0313
Δ	**-62.92%**	**-68.56%**	**-62.56%**

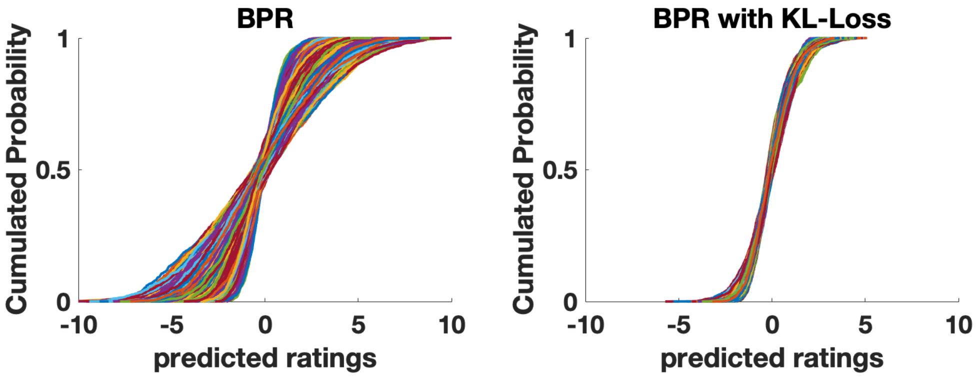


Figure 5.15: CDFs of user score distributions predicted by BPR and BPR with KL-loss over ML1M dataset. Reprinted with permission from [12].

for the same dataset all methods have the same dimension: we set 10 for ML1M-2, 40 for Yelp-2, and 60 for Amazon-2. We set the learning rate 0.01 for baselines, and 0.01 as η_{BPR} for DPR-RSP and DPR-REO. As for adversary learning rate η_{Adv}, we set 0.005 for all three datasets.

For all methods in all experiments, we have negative sampling rate 5 and mini-batch size 1024. For all fairness-aware methods, we set $\beta = 30$. And we adopt a 4-layer MLP with 50 neurons with ReLU activation function in each layer as the adversary for DPR.

5.3.4.2 *RQ1: Effects of Model Components*

In this subsection, we aim to answer three questions: whether the KL-loss can effectively normalize user score distribution? whether the adversary can effectively enhance score distribution similarity among groups? and whether DPR-RSP and DPR-REO can effectively improve the fairness metrics RSP and REO?

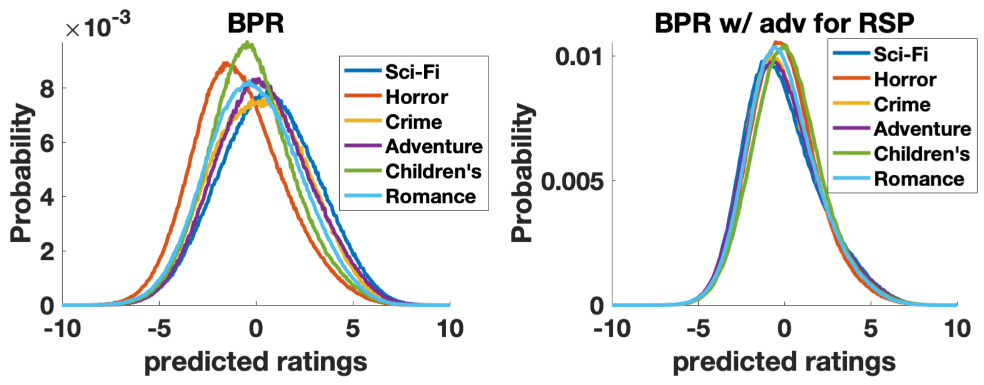


Figure 5.16: PDFs of $p(\widehat{y}|g)$ for different groups by BPR and BPR w/ adv for RSP over ML1M. Reprinted with permission from [12].

Effects of KL-loss

The KL-loss is to normalize the user score distribution. Hence, we adopt the Jensen-Shannon Divergence (JS Divergence) to measure the deviation between user score distributions, where lower JS Divergence indicates that the user score distributions are normalized better. We compare BPR and BPR with KL-loss over all three datasets, the results are shown in Table 5.6, and the improvement rates (noted as Δ) are also calculated. We can observe that with the KL-loss, the divergence among user score distributions is largely reduced, demonstrating the effectiveness of KL-loss. To better show the effects of KL-loss, we visualize the score distribution for every user produced by BPR with and without KL-loss for ML1M in Figure 5.15, where each curve represents the Cumulative Distribution Function (CDF) of a single user's scores. The closely centralized CDFs in the right figure verify the effectiveness of the proposed KL-loss.

Effects of Adversary

The adversary in DPR is to enhance the score distribution similarity among different groups. To evaluate the effectiveness of the adversarial learning, we compare the performances of BPR and BPR with adversary for both metrics (noted as *BPR w/ adv for RSP* and *BPR w/ adv for REO*). More specifically, we compare BPR with BPR w/ adv for RSP w.r.t. JS Divergence among $p(\widehat{y}|g)$ for different groups, and compare BPR with BPR w/ adv for REO w.r.t. JS Divergence among $p(\widehat{y}|g, y = 1)$ for different groups. Results are shown in Table 5.7, where the top three

Table 5.7: Comparison between BPR and BPR w/ adv for JS Divergences of score distribution among different groups. Reprinted with permission from [12].

		ML1M	Yelp	Amazon
RSP setting	BPR	0.0222	0.0011	0.0215
	BPR w/ adv	0.0090	0.0004	0.0046
	Δ	**-59.46%**	**-63.64%**	**-78.60%**
REO setting	BPR	0.0128	0.0045	0.0378
	BPR w/ adv	0.0047	0.0041	0.0087
	Δ	**-63.28%**	**-8.89%**	**-76.98%**

Table 5.8: Comparison between BPR and DPR-RSP w.r.t. $F1@15$ and $RSP@15$. Reprinted with permission from [12].

		ML1M	Yelp	Amazon
F1@15	BPR	0.1520	0.0371	0.0230
	DRP-RSP	0.1439	0.0354	0.0221
	Δ	**-5.31%**	**-4.32%**	**-3.90%**
RSP@15	BPR	0.4058	0.2522	0.4155
	DPR-RSP	0.0936	0.0856	0.0607
	Δ	**-76.92%**	**-66.07%**	**-85.40%**

rows are calculated on all user-item pairs not in the training set (fit the RSP setting), the bottom three rows are calculated on user-item pairs only in the test set (fit the REO setting). The table demonstrates the extraordinary effectiveness of the proposed adversarial learning for enhancing distribution similarity under both settings. To further validate this conclusion, we visualize the distributions of $p(\widehat{y}|g)$ for different groups from ML1M in Figure 5.16 (distributions of $p(\widehat{y}|g, y = 1$ have the same pattern), where the Probability Distribution Function (PDF) of every group's score distribution is plot as a single curve. We can find that PDFs by BPR w/ adv are close to each other, while PDFs by the ordinary BPR differ considerably.

Effects of DPR

The effects of the complete DPR should be evaluated from the perspectives of both recommendation quality and recommendation fairness. We first investigate the performance of DPR-RSP.

Table 5.9: Comparison between BPR and DPR-REO w.r.t. $F1@15$ and $REO@15$. Reprinted with permission from [12].

		ML1M	Yelp	Amazon
F1@15	BPR	0.1520	0.0371	0.0230
	DRP-REO	0.1527	0.0363	0.0208
	Δ	**+0.49%**	**-1.94%**	**-9.81%**
REO@15	BPR	0.1893	0.2225	0.6217
	DPR-REO	0.0523	0.0874	0.3577
	Δ	**-72.38%**	**-60.73%**	**-42.47%**

$F1@15$ and $RSP@15$ results of both BPR and DPR-RSP over three datasets are listed in Table 5.8, where the change rates for them are calculated. From the table we have three observations: (i) DPR-RSP improves the fairness over BPR greatly (decreases $RSP@15$ by 76% on average); (ii) DPR-RSP effectively preserves the recommendation quality (only drops $F1@15$ by 4% on average); and (iii) for different datasets with different degrees of data imbalance, DPR-RSP can reduce the unfairness to a similar level ($RSP@15$ for three datasets by DPR-RSP are all smaller than 0.1).

Similar conclusions can be drawn for DPR-REO based on Table 5.9, where comparison between BPR and DPR-REO w.r.t. $F1@15$ and $REO@15$ are listed. We can observe that DPR-REO is able to decrease metric $REO@15$ to a great extent while preserving high $F1@15$ as well. Generally speaking, DPR-REO demands less recommendation quality sacrifice because the definition of REO is less stringent and enhancing fairness is easier to achieve than RSP. However, there is one exception that in Amazon dataset, DPR-REO drops $F1@15$ by 9.8%. It may be because for the Amazon dataset, every item group has its own collection of users, and there are few users giving feedback to more than one group, which exerts difficulty for DPR-REO training.

5.3.4.3 RQ2: Comparison with Baselines

We next compare the proposed DPR with state-of-the-art alternatives to answer two questions: (i) how does the proposed adversarial learning perform in comparison with baselines for predicted score distribution similarity enhancement? and (ii) how does the proposed DPR perform for both fairness metrics compared with baselines? Because baselines Reg-RSP and Reg-REO can only

Table 5.10: Comparison between DPR and baselines for JS Divergences of score distribution among groups. Reprinted with permission from [12].

		ML1M-2	Yelp-2	Amazon-2
RSP setting	BPR	0.0564	0.0034	0.0514
	FATR	**0.0218**	0.0027	**0.0332**
	Reg-RSP	0.0276	**0.0026**	0.0378
	DPR-RSP	**0.0155**	**0.0020**	**0.0079**
	Δ	-28.90%	-23.08%	-76.20%
REO setting	BPR	0.0422	0.0216	0.1531
	FATR	**0.0044**	0.0078	0.1844
	Reg-REO	0.0179	**0.0062**	**0.0219**
	DPR-REO	**0.0011**	**0.0018**	**0.0038**
	Δ	-75.00%	-70.97%	-82.65%

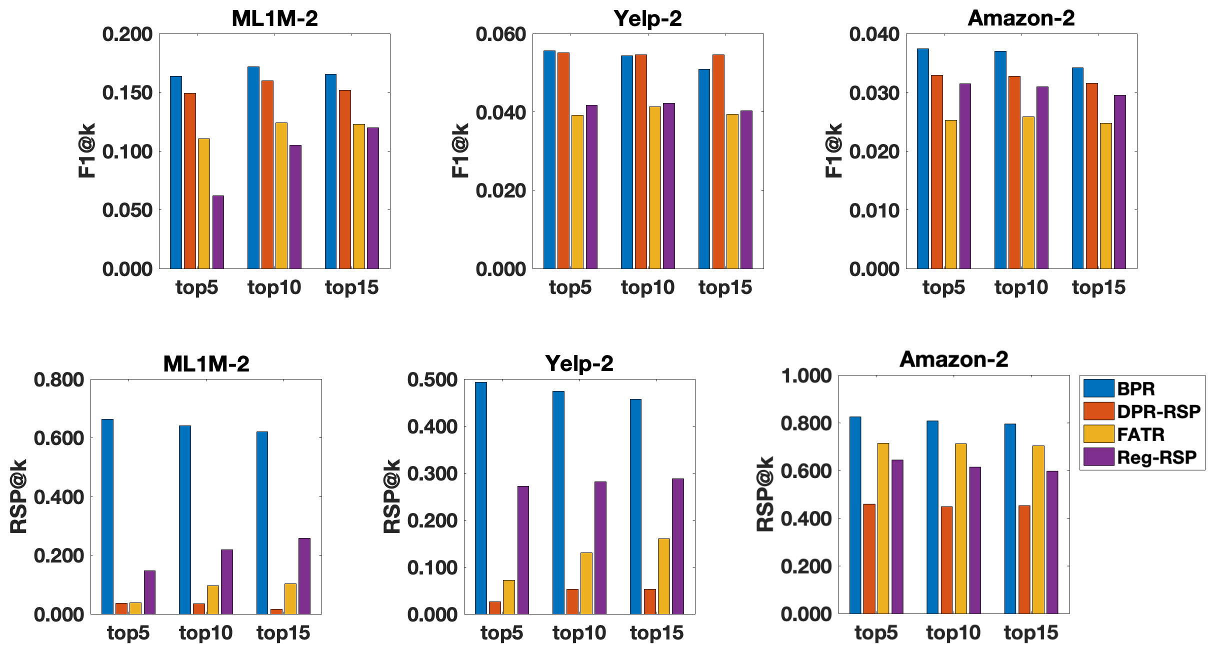


Figure 5.17: F1@k and RSP@k of four different models over three datasets. Reprinted with permission from [12].

work for binary-group cases, we conduct the experiment over ML1M-2, Yelp-2, and Amazon-2 datasets in this subsection.

To answer the first question, we report JS Divergences of score distributions for different groups in Table 5.10, where the top five rows are calculated on all user-item pairs not in the training set

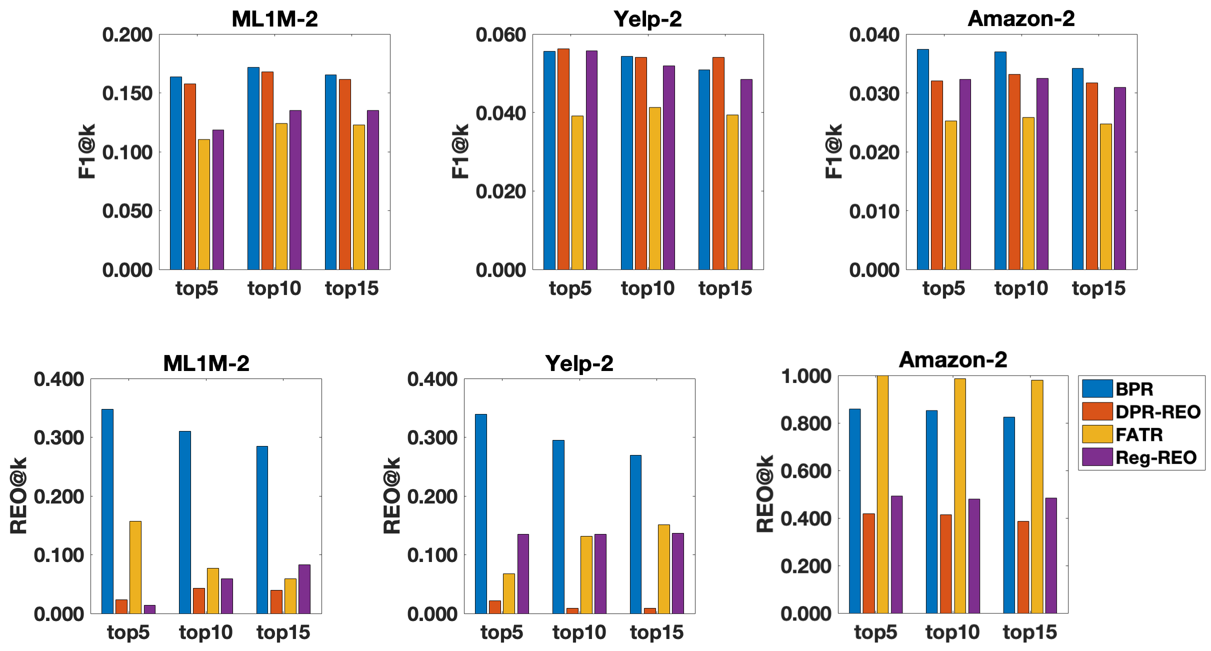


Figure 5.18: F1@k and REO@k of four different models over three datasets. Reprinted with permission from [12].

(fitting the RSP setting), and the bottom five rows are calculated on user-item pairs only in the test set (fitting the REO setting). The improvement rates of DPR over the best baselines also are calculated. From the table we can conclude that the proposed adversarial learning can more effectively enhance score distribution similarity than baselines. Although less competitive, both FATR and Reg models can improve the distribution similarity to some degree compared with BPR.

As for the second question, we show $F1@k$, $RSP@k$, and $REO@k$ comparison between all methods over all datasets in Figure 5.17 and Figure 5.18. On the one hand, from the leftmost three figures in both Figure 5.17 and Figure 5.18, we can observe that DPR-RSP and DPR-REO preserve relatively high $F1@k$ from BPR and outperform other baselines significantly. On the other hand, from the rightmost three figures, we are able to see that DPR-RSP and DPR-REO enhance RSP and REO to a great extent respectively, which also outperform other fairness-aware methods considerably. Besides, one potential reason for better recommendation quality for DPR on Yelp is that the intrinsic bias in Yelp is small, thus DPR can promote unpopular groups and keep the original high rankings for popular groups simultaneously, leading to better performance.

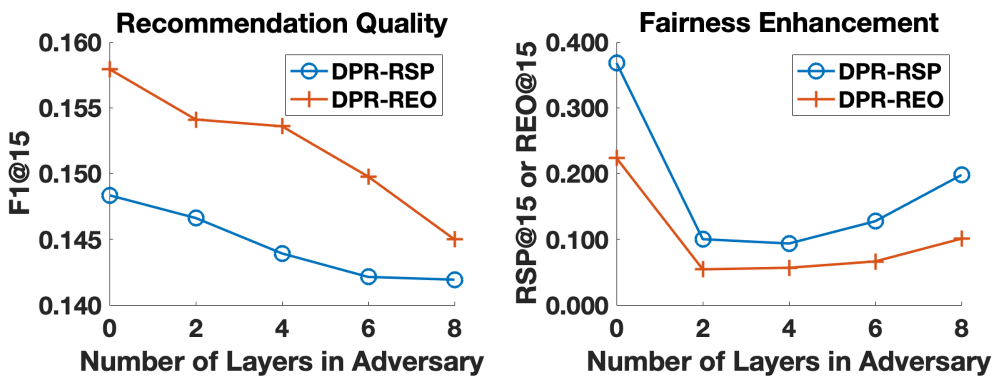


Figure 5.19: $F1@15$, $RSP@15$, and $REO@15$ of DPR-RSP and DPR-REO w.r.t. different numbers of layers over ML1M. Reprinted with permission from [12].

5.3.4.4 RQ3: Impact of Hyper-Parameters

Finally, we investigate the impact of three hyper-parameters: (i) the number of layers in the MLP adversary; (ii) the adversary trade-off regularizer α; and (iii) the KL-loss trade-off regularizer β. For conciseness, we only report experimental results on ML1M dataset, but note that the results on other datasets show similar patterns.

Impact of Layers in Adversary

First, we experiment with the number of layers in MLP adversary varying in $\{0, 2, 4, 6, 8\}$, and the other parameters are the same as introduced in Section 5.3.4.1 including that the number of neurons in each MLP layer is still 50. Generally speaking, with more layers, the adversary is more complex and expressive, which intuitively results in better fairness performance. The $F1@15$ results of DPR-RSP and DPR-REO w.r.t. different numbers of layers are shown at the left in Figure 5.19, and $RSP@15$ and $REO@15$ results are presented at the right in Figure 5.19. From these figures, we can infer that with a more powerful adversary, the recommendation quality drops more; however, the fairness improvement effect first gets promoted but then weakened due to difficulty of model training. The best value is around 2 to 4. Besides, we can also find that it is easier to augment the metric REO than RSP with less recommendation quality sacrificed, which is consistent with the observation in Section 5.3.4.2.

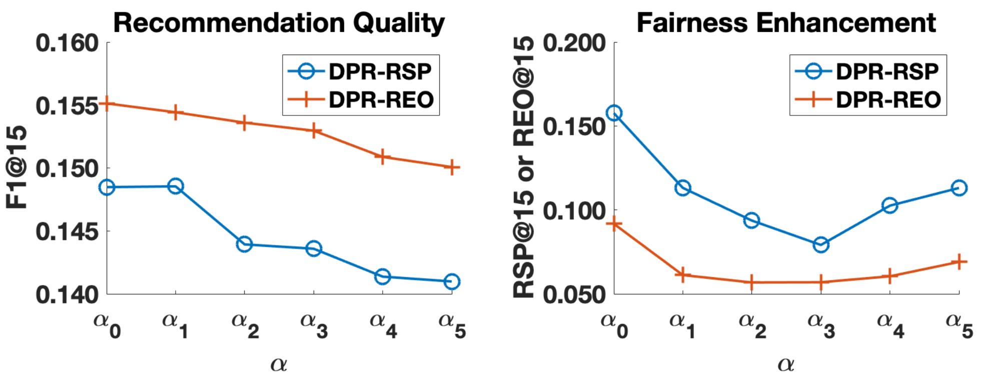


Figure 5.20: $F1@15$, $RSP@15$ and $REO@15$ of DPR-RSP and DPR-REO w.r.t. different α over ML1M. Reprinted with permission from [12].

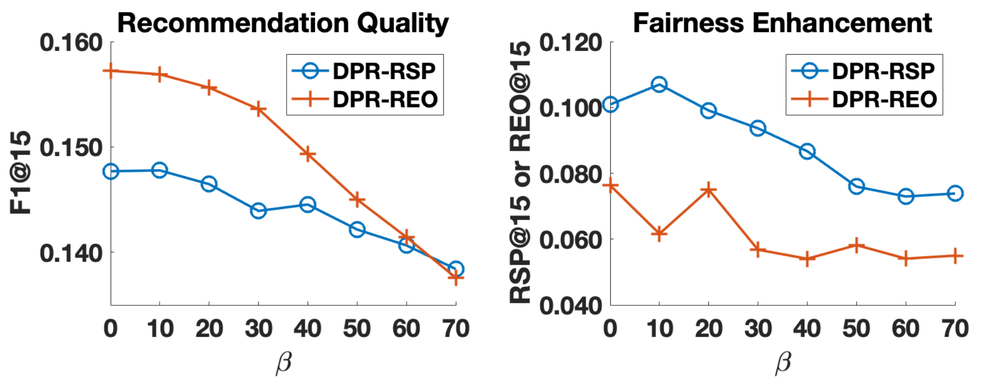


Figure 5.21: $F1@15$, $RSP@15$ and $REO@15$ of DPR-RSP and DPR-REO w.r.t. different β over ML1M. Reprinted with permission from [12].

Impact of α

Then, we vary the adversary trade-off regularizer α and plot the results in Figure 5.20, where the x-axis coordinates $\{\alpha_0, \alpha_1, \ldots, \alpha_5\}$ are $\{1000, 3000, 5000, 7000, 9000, 11000\}$ for DPR-RSP and $\{200, 600, 1000, 1400, 1800, 2200\}$ for DPR-REO. The left figure demonstrates the $F1@15$ results with different α, which shows that with larger weight for the adversary, the recommendation quality decreases more. For the fairness improving performance, as presented at the right in Figure 5.20, with larger α, both DPR-RSP and DPR-REO first improve the fairness, but then increase it again, which is most likely due to the dominating of adversary over KL-loss in the objective function. To balance the recommendation quality and recommendation fairness, setting

$\alpha = 5000$ for DPR-RSP and $\alpha = 1000$ for DPR-REO are reasonable choices.

Impact of β

Last, we study the impact of the KL-loss trade-off regularizer β and vary the value in the set $\{0, 10, 20, 30, 40, 50, 60, 70\}$. The left figure in Figure 5.21 shows the change tendency of $F1@15$, which implies that larger β leads to lower recommendation quality. The fairness improving performance of DPR-RSP and DPR-REO with different β are shown at the right in Figure 5.21, from which we can observe that with higher β, the fairness is enhanced better, and converges to a certain degree. However, the impact of β is not as strong as that of α (the value changes of $RSP@15$, and $REO@15$ in Figure 5.21 are smaller than those in Figure 5.20).

5.3.5 Summary

In this section, we propose a novel framework – FATR – to enhance the fairness for implicit recommender systems while maintaining recommendation quality. FATR effectively eliminates sensitive information and provides fair recommendation with respect to the sensitive attribute. Further, unlike previous efforts, the proposed model can also handle multi-feature and multi-category cases. Extensive experiments show the effectiveness of FATR compared with state-of-the-art alternatives.

5.4 Enhancing Fairness in Cold-start Recommender Systems

Last, we turn our focus to cold-start recommender systems and investigate whether new items with no historical feedback can be fairly treated and how to enhance such recommendation fairness for cold-start items.

5.4.1 Introduction

Previous works have revealed that widely used recommendation algorithms can indeed produce unfair recommendations toward different items (like job candidates of different genders), e.g., [12, 24, 101, 26, 11]. However, these existing works consider fairness only during the middle of the life cycle of an item, that is in the *warm start* recommender scenario. In this scenario, prior

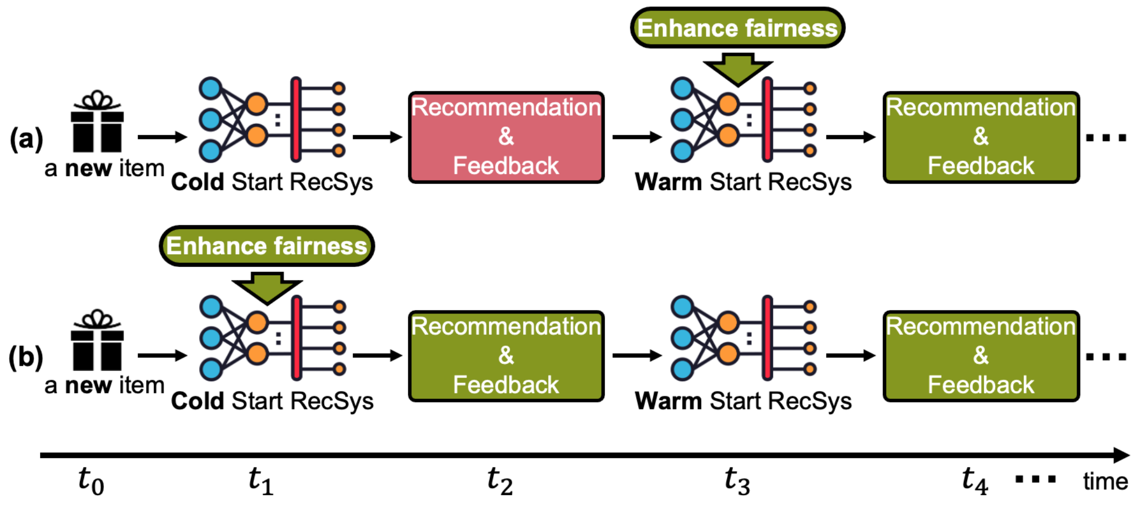


Figure 5.22: (a) Existing works consider fairness during warm start recommendation period; (b) we study fairness among new items, i.e., the cold-start recommendation period. Reprinted with permission from [13].

works show that the main driver of unfairness is the data bias in historical feedback (like clicks or views), and recommendation algorithms unaware of this bias can inherit and amplify this bias to produce unfair recommendations. But what if there is no historical feedback? Can we fairly recommend new items (we use ‘new items’ and ‘cold-start items’ interchangeably) in such a *cold-start* scenario?

To illustrate, Figure 5.22a shows the life cycle of an item: the item first appears in the system at t_0; in the absence of historical feedback, a cold-start recommendation algorithm recommends this item to users and receives the first collection of feedback at t_2; then, a warm start recommender can be trained at t_3 by this first collection of feedback, further recommending the item to users and collecting new feedback at t_4; the system continues this loop of collecting new feedback and training a new warm start model until the item leaves the system. While methods in existing works can introduce fairness at time t_3 and later, is the item treated fairly before then? Here, we show that the data bias can be transferred from warm start items to new items through item content features by machine learning based cold-start recommendation algorithms, inducing unfair recommendations among these new items.

This *fairness gap* can be especially problematic since unfairness introduced by cold-start recommenders will be perpetuated and accumulated through the entire life cycle of an item, resulting in growing difficulty for mitigating unfairness as the life cycle goes on. Instead, providing fair recommendations among new items could give rise to a virtuous circle of collecting (relatively) unbiased feedback and training fairer models later in the life cycle. Hence, as shown in Figure 5.22b, we propose to investigate fairness in cold-start recommender systems and aim to enhance fairness among new items at the beginning of their life cycles.

Fairness goal

Here, we follow two well-known concepts – equal opportunity [94] and Rawlsian Max-Min fairness principle of distributive justice [145] – to introduce the *Max-Min Opportunity Fairness* in the context of cold-start scenarios. The fairness goal is to provide recommendations that maximize the true positive rate (that is, the probability of being accurately recommended to matched users who will like the item during testing) of the worst-off items so that no item is under-served by the recommendation model. The advantages of this fairness goal are twofold: (i) by following equal opportunity to measure fairness by the true positive rate, the fairness is directly aligned with the feedback or economic gain items receive as well as user satisfaction; and (ii) by following Rawlsian Max-Min fairness to accept inequalities, the fairness does not require decreasing utility for the better-served items and thus can better preserve the overall utility. Thus, by improving the Max-Min Opportunity Fairness, we aim to improve item fairness without decreasing overall satisfaction.

Contributions

To the best of our knowledge, this is the first work to study recommendation fairness among new items in cold-start recommenders. In sum, we make the following contributions:

- We introduce the problem of fairness among new items in cold-start scenarios, and we conduct a comprehensive data-driven study to demonstrate the prevalence of unfairness among new items in cold-start recommender systems.
- To mitigate unfairness among new items, we propose a novel learnable post-processing framework as a solution blueprint, which promises better practical feasibility than existing

strategies. Based on this blueprint, we demonstrate two concrete approaches for enhancing item fairness: a score scaling model based on existing work for addressing popularity bias [80] and a novel joint-learning generative model that can effectively improve fairness.

- Extensive experiments over four datasets show that both proposed methods can simultaneously enhance item fairness while preserving recommendation utility, demonstrating the viability of filling the fairness gap. Furthermore, we also demonstrate the capability of the proposed methods to enhance group-level fairness in addition to individual-level fairness.

5.4.2 Fairness Among New Items

In this section, we first introduce the cold-start recommendation problem. Second, we formalize fairness among new items. Then, we introduce how to measure recommendation utility from the view of items. Last, we conduct a data-driven study over four public datasets and four cold-start recommendation algorithms to empirically demonstrate the prevalence of unfairness among new items.

5.4.2.1 Cold-start Recommendation

We focus on the cold-start item recommendation task [5], where all users are warm start during training and testing, but new items never seen during training are to be recommended during testing. Assume we have N users $\mathcal{U} = \{1, 2, \ldots, N\}$ and M_w warm start items $\mathcal{I}_w = \{1, 2, \ldots, M_w\}$, where each item has at least one historical interaction record in the training data. We denote $\mathcal{O}_{tr} = \{(u, i)\}$ as the training set, where $u \in \mathcal{U}$ indexes one user, and $i \in \mathcal{I}_w$ indexes one warm start item. We also have M_c cold-start (new) items $\mathcal{I}_c = \{1, 2, \ldots, M_c\}$, none of which are included in the training set $\mathcal{O}_{tr}$. We denote $\mathcal{O}_{te} = \{(u, i)\}$ as the test set, where $u \in \mathcal{U}$ and $i \in \mathcal{I}_c$. For each item i, we have a subset of users $\mathcal{U}_i^+$ to indicate the matched users (users who have already interacted the item during training or will interact with the item during testing) for this item: for a warm start item $i \in \mathcal{I}_w$, $\mathcal{U}_i^+$ are matched users in the training set $\mathcal{O}_{tr}$; for a new item $i \in \mathcal{I}_c$, $\mathcal{U}_i^+$ are matched users in the test set $\mathcal{O}_{te}$. *The goal of a cold-start recommender [7, 5, 49, 64] is to provide a ranked list of cold-start items to each user as recommendations.*

To provide cold-start recommendations, typical machine learning based methods [6, 52, 49, 50, 51, 5, 3, 4, 7, 64] need to utilize user-item interactions $\mathcal{O}_{tr}$ of existing warm start users and items, and content features of both warm and cold-start items. These content features – such as item descriptions, reviews, or from other sources – are often readily available even for new items. The main idea of these cold-start recommendation algorithms is to learn a transformation from item content features of warm start items to user-item interactions between warm start users and items during training, and then apply this learned transformation process to the content features of new items to predict possible interactions between users and new items as recommendations during testing. Note that item content features of warm start items and new items share the same feature space. As a result, bias inherent in training data of warm users and items collected from an existing fairness-unaware recommender system will be transferred through the item content features to recommendations for new items, generating unfair recommendations among new items.

5.4.3 Formalizing Fairness

A natural following question is how to determine if recommendations are fair or not among new items? Here, we follow two well-known concepts to formalize fairness: equal opportunity [94] and Rawlsian Max-Min fairness principle of distributive justice [145].

In a classification task, **equal opportunity** requires a model to produce the same true positive rate (TPR) for all individuals or groups. Equal opportunity fairness has already been recognized as unquestionably important in recommender systems by previous works [12, 24, 122, 26]. The goal is to ensure that items from different groups can be equally recommended to matched users during testing (the same true positive rate): for example, candidates of different genders are equally recommended to job openings that they are qualified for. In contrast, demographic parity fairness [12, 11] only focuses on the difference in the amount of exposure to users without considering the ground-truth of user-item matching. However, because only the exposure to matched users (as considered by equal opportunity fairness) can influence the feedback or economic gain of items, in recommendation tasks, equal opportunity is better aligned than demographic parity fairness.

Rawlsian Max-Min fairness requires a model to maximize the minimum utility of individuals or groups so that no subject is under-served by the model. Unlike equality (or parity) based notions of fairness [24, 122, 26, 12, 119, 120, 121, 11] aiming to eliminate difference among individuals or groups but neglecting a decrease of utility for better-served subjects, Rawlsian Max-Min fairness accepts inequalities and thus does not requires decreasing utility of better-served subjects. So, Rawlsian Max-Min fairness is preferred in applications where perfect equality is not necessary, such as recommendation tasks, and it can also better preserve the overall model utility.

Hence, following these two concepts, for the cold-start recommendation task, we have the fairness definition:

Definition 1 (Max-Min Opportunity Fairness)**.** *Suppose $\mathcal{H}$ is a set of models, $TPR(i)$ is the expected true positive rate a new item i gets from a model h, then the model h^* is said to satisfy Max-Min Opportunity Fairness if it maximizes the true positive rate of the worst-off item:*

$$h^* = \arg\max_{h \in \mathcal{H}} \min_{i \in \mathcal{I}_c} TPR(i) \tag{5.31}$$

Hence, the goal of enhancing such a Max-Min Opportunity Fairness is to improve the true positive rate of the worst-off item in recommendations. And we measure this Max-Min Opportunity Fairness for a cold-start recommender by calculating the average true positive rate of the $t\%$ worst-off items, which are the $t\%$ items with the lowest true positive rates among all cold-start items during testing. We measure the fairness over $t\%$ items instead of just the worst item to make the metric more flexible and robust to noise.

Then, the next question is how to calculate the true positive rate of an item? We calculate the true positive rate for a new item by averaging a scoring function of ranking positions[2] across all matched users in the test set. Concretely, we propose the true positive rate metric Mean Discounted

[2]We use a reciprocal log function as the scoring function to calculate the true positive rate resulting in a metric aligned with NDCG. Other choices lead to true positive rate calculations aligned with other existing utility metrics: a step function is aligned with Recall@k; a reciprocal function is aligned with mean reciprocal rank.

Table 5.11: Characteristics of the four public datasets. Reprinted with permission from [13].

	#user	Train		Validation		Test	
		#item	#record	#item	#record	#item	#record
ML1M	6,018	1,811	529,952	302	106,695	905	296,870
ML20M	112,292	6,083	10,697,409	1,014	1,797,626	3,041	5,490,603
CiteULike	5,551	13,584	164,210	1,018	13,037	2,378	27,739
XING	89,867	10031	1,893,135	1,671	376,994	5,016	1,131,487

Gain (MDG) for an item i:

$$MDG_i = \frac{1}{|\mathcal{U}_i^+|} \sum_{u \in \mathcal{U}_i^+} \frac{\delta(\widehat{z}_{u,i} <= k)}{log(1 + \widehat{z}_{u,i})}, \tag{5.32}$$

where $\mathcal{U}_i^+$ is the set of matched users for the new item i in the test set; $\widehat{z}_{u,i}$ is the ranking position of i (among all new items $\mathcal{I}_c$, ranging from 1 to M_c) for user u by a cold-start recommendation model; $\delta(x)$ returns 1 if x is true, otherwise 0. That is to say, we only consider the discounted gain $1/log(1 + \widehat{z}_{u,i})$ for ranking positions within the top-k and assign 0 for positions after k (we fix $k = 100$) so that MDG is aligned with the well-known metric NDCG@k [146]. $MDG_i = 0$ means that item i is never recommended to matched users who like it during testing; $MDG_i = 1$ means that i is ranked at the top position to all matched users during testing.

With the introduced metric MDG, we measure the Max-Min Opportunity Fairness in Definition 1 for a cold-start recommender by calculating the average MDG of the $t\%$ worst-off items, which are the $t\%$ items with the lowest MDG among all cold-start items during testing. In the empirical study, we report results with $t\% = 10\%$ and 20%, denoted as **MDG-min10%** and **MDG-min20%**. Higher values indicate the evaluated system is fairer. We also report the average MDG for the 10% best-served items for comparison, which are the 10% items with the highest MDG, denoted as **MDG-max10%**.

5.4.3.1 Measuring Utility for Items

The typical way to evaluate the quality of a recommender system, such as with NDCG@k, is to first calculate the recommendation utility for each user and then average across users as the measured utility for a recommendation algorithm. Thus, this metric represents the expectation of recommendation utility a random user can receive from an algorithm, which is essentially from the view of users. We call these conventional metrics like NDCG@k as **user-view utility**. In contrast, we can also evaluate the recommendation utility from the view of items to show how well items are generally served by a recommendation algorithm. Because we study item fairness here, it is natural to consider this **item-view utility** as one evaluation aspect in empirical studies. Here, given the true positive rate metric MDG introduced in Section 5.4.3, we report **MDG-all** $= \sum_{i \in \mathcal{I}_c} MDG_i / |\mathcal{I}_c|$ as the item-view utility metric, which shows the expectation of recommendation utility a random item can get from an algorithm.

5.4.3.2 Data-Driven Study

Given the formalization of fairness in cold-start recommendation, how much unfairness can be generated by different cold-start recommendation algorithms? Here, we conduct a data-driven study on four public datasets and four different cold-start models to show the prevalence of unfairness in cold-start recommendation.

Datasets

We adopt four widely used datasets for cold-start recommendation: **ML1M** [96], **ML20M** [96], **CiteULike** [66], and **XING** [68]. ML1M and ML20M are movie rating datasets, where we consider all ratings as positive signals. Both datasets contain tag genome scores [147] (showing relevance of an item to a fixed set of tags) as the item content features. CiteULike is a dataset recording user preferences toward scientific articles, and following [5], we use the abstracts of these articles as item content features. XING is a user-view-job dataset, and it includes career level, tags, and other related information as the item content features. For ML1M, ML20M, and XING, we randomly select 10% items and 30% items as the cold-start (new) items, with all the user-item interactions of these items, to be the validation set and test set respectively. For CiteULike, we directly adopt the dataset splitting from [5]. The detailed statistics of the four datasets are shown in Table 5.11.

Table 5.12: Empirical results of four algorithms on ML1M (DN stands for DropoutNet, DM stands for DeepMusic). Reprinted with permission from [13].

		Heater	DN	DM	KNN	Optimal	Random
User utility	NDCG@15	0.5516	0.5488	0.5312	0.4402	1.000	0.0550
	NDCG@30	0.5332	0.5316	0.5167	0.4226	1.000	0.0586
Item utility	MDG-all	0.0525	0.0552	0.0572	0.0646	0.1932	0.0236
Fairness	MDG-min10%	0.0000	0.0000	0.0000	0.0001	0.1388	0.0118
	MDG-min20%	0.0000	0.0000	0.0001	0.0020	0.1498	0.0145
	MDG-max10%	0.2272	0.2294	0.2323	0.2091	0.2471	0.0386

Cold-start recommendation models

There are many different cold-start recommendation models in the literature, and it is impossible to test all of them. Generally, existing algorithms can be categorized into joint-training, separate-training, combined, and heuristic non-parametric methods as introduced in Section 5.4.2.1. Thus, here, we pick four representative algorithms from each category: a typical joint-training method **DropoutNet** [5]; a typical separate-training method **DeepMusic** [49]; a combination of joint-training and separate-training method **Heater** [7]; and a heuristic non-parametric method **KNN** [64].

Experiment protocol

Following these cold-start recommendation works [7, 5, 49, 64], during testing, we evaluate recommendations for only new items without mixing warm start items in, which allows us to focus on the behavior of cold-start recommenders and deepen our understandings of fairness among new items.

Empirical results

First, we report the results of four cold-start recommendation algorithms on the ML1M dataset in Table 5.12. Besides, we also show two special cases: (i) the **Optimal** case we can achieve, for which we directly get access to the positive user-item pairs in the test set, and rank the matched items to the most top positions with a random order for each user; and (ii) the **Random** case, for which we randomly rank the items for each user. The first three rows in Table 5.12 show the results of user-view utility (NDCG@15 and NDCG@30) and item-view utility (MDG-all). *User-*

view utility and item-view utility reveal different aspects of a recommender system, which usually show opposite patterns.[3] For example, in Table 5.12, Heater achieves the best user-view utility, however, it has the lowest item-view utility (while KNN is the opposite). Since items are treated unfairly by Heater, a small subset of items (potential popular items) with large numbers of test samples receive very high MDG and the majority of items get very low MDG, leading to high user-view utility but low item-view utility. In other words, the different patterns of user-view and item-view utility is due to the unfairness in recommendations.

Hence, we next analyze the fairness as shown in the last three rows in Table 5.12, where we present MDG-min10%, MDG-min20% to show the average MDG of the worst-off items; and we also present MDG-max10% to show the average MDG of the best-served items for comparison. We can observe that Heater and DropoutNet produce zero MDG for the 10% and 20% worst-off items, which means these items are never exposed to matched users who will like them during testing. DeepMusic and KNN perform slightly better, but the MDG-min10% and MDG-min20% are still very low. However, MDG-max10% values are very high for all four algorithms. *The large difference between MDG-min and MDG-max illustrates the unfairness among new items in these four cold-start recommendation models.* For the Optimal case, because the recommendation is optimal for every user and item, the fairness is also guaranteed. Moreover, we can also observe that MDG-min10% and MDG-min20% in the Random case are higher than the personalized cold-start recommendation models, which on one hand shows the unfairness in these cold-start models, and on the other hand, demonstrates one possible way to enhance the fairness by introducing randomness to recommendations.

To further understand the unfairness issue in these algorithms, we plot the MDG of each item by different algorithms in Figure 5.23, where for each algorithm, items are sorted based on their MDG in ascending order. Each dot in Figure 5.23 represents one item for one algorithm; the y-axis shows the MDG an item receives from one of the six algorithms; and the x-axis shows the position

[3] Although in practice, user-view and item-view utility often show opposite patterns, indeed, they are not opposite to each other. Theoretically, an optimal model can achieve the best user-view and item-view utility at the same time, and achieve fairness among new items as well. One example of such an optimal model is the ‘Optimal’ in Table 5.12.

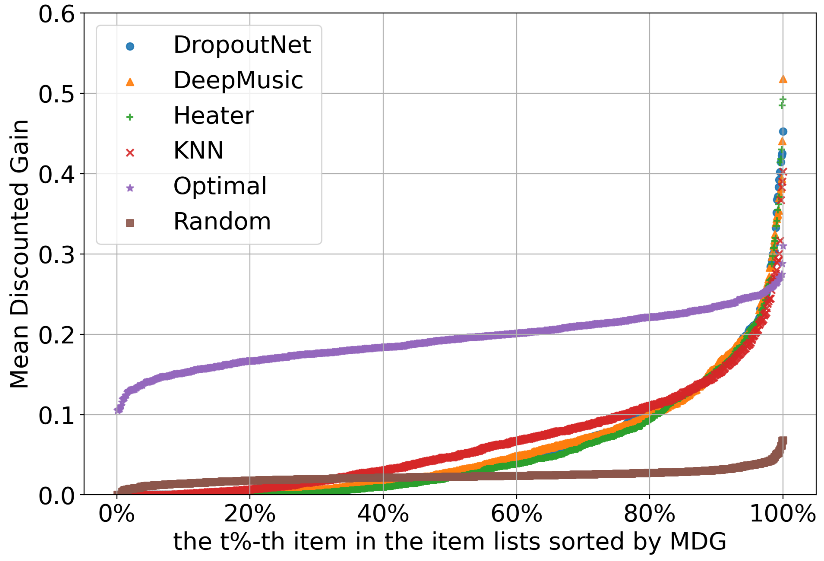


Figure 5.23: For ML1M and each model, sort items by MDG in ascending order and plot their corresponding MDG. Reprinted with permission from [13].

Table 5.13: Empirical results of Heater on four datasets. Reprinted with permission from [13].

		ML1M	ML20M	CiteULike	XING
User utility	NDCG@15	0.5516	0.4408	0.2268	0.2251
	NDCG@30	0.5332	0.4308	0.2670	0.2762
Item utility	MDG-all	0.0525	0.0187	0.1833	0.1333
Fairness	MDG-min10%	0.0000	0.0000	0.0046	0.0028
	MDG-min20%	0.0000	0.0000	0.0251	0.0129
	MDG-max10%	0.2272	0.1455	0.5106	0.3821

of an item in a sorted item list of an algorithm (e.g., the dot corresponding to 100% represents the item with the largest MDG for one algorithm, 0% represents the item with the lowest MDG). From the figure, we can see that all four cold-start recommendation algorithms produce skewed distributions of MDG across recommended new items: most items receive low or even zero MDG, and only a few items receive extremely high MDG, confirming the existence of unfairness. The Optimal shows the best result we can achieve on the given test set, where we can observe that the overall distribution is much more flat, with higher MDG for worst-off items but lower MDG for the best-served items compared with the other four algorithms. *The goal of fairness enhancement is to generate a distribution as close as possible to the optimal case.*

Last, we report results for the best-performing model Heater on all four datasets in Table 5.13.

From the table, we can see that for all four datasets, MDG-min10% and MDG-min20% are very small (or even zero) compared with MDG-max10%, demonstrating that *the unfairness among new items is prevalent across datasets from different domains and with different characteristics*. Besides, comparing results of different datasets, we find that *fairness is highly related to the density of training data and quality of item content features*: training data with high density or with high-quality item content features can lead to more unfair recommendations, such as ML1M and ML20M which are very dense and have high-quality item content features (informative tag genome scores [147]). When the training data is dense or item content features are informative, a cold-start recommendation model can more effectively learn information including data bias in training data and hence deliver unfairer recommendations.

5.4.4 Fairness Enhancement Approaches

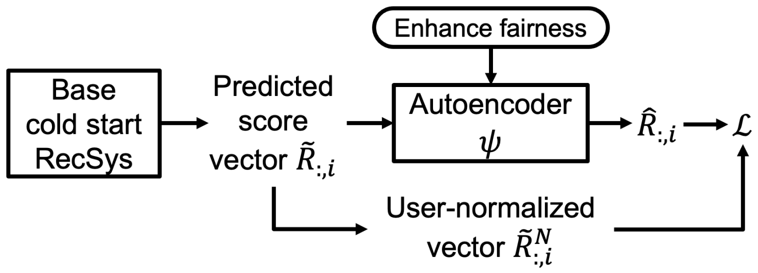
Given the observation of unfairness, we study in this section how to enhance the fairness for new items. We first propose a novel learnable post-processing framework to enhance the fairness for new items, which is not a concrete model but a high-level solution blueprint. Then, based on this blueprint, we propose two concrete models: a novel joint-learning generative method; and a score scaling model, which adapts a previous work for popularity bias [80] into the proposed framework, as a baseline for comparison.

5.4.4.1 Learnable Post-processing Framework

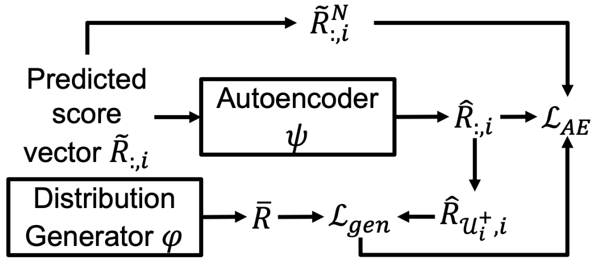
We first elaborate the overall structure of the proposed framework, and then explain how to enhance fairness in this framework.

Main structure of the framework

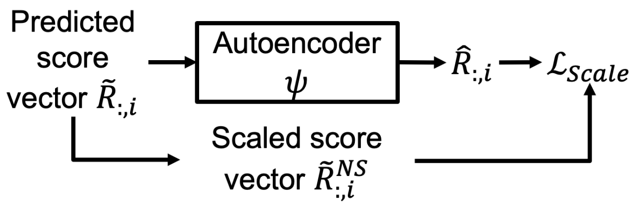
Most existing works improve item fairness [24, 12, 26, 102], popularity bias [148, 71, 80], or diversity [149, 150] for warm start recommender systems by either in-processing methods or non-parametric post-processing methods. In-processing methods [102, 24, 71] modify the original recommendation models to achieve fairness or other goals. The major drawback of this type of method is that it requires re-training the whole recommender system with all training data, which



(a) The learnable post-processing framework.



(b) The Gen method.


Predicted score vector $\tilde{R}_{:,i}$
Autoencoder ψ
$\hat{R}_{:,i}$
$\mathcal{L}_{Scale}$
Scaled score vector $\tilde{R}^{NS}_{:,i}$

(c) The Scale method.

Figure 5.24: The structures of the proposed learnable post-processing framework and two concrete models. Reprinted with permission from [13].

is highly resource intensive and not practically feasible in real-world systems. Another widely investigated class is the non-parametric post-processing method [129, 26, 101], which keeps the original recommender systems unchanged, but conducts a heuristic re-ranking to the output of the original model to achieve fairness or other goals. This type of approach is more practically feasible because it does not require re-training the base model. But limitations are that it usually produces inferior performance than learning based in-processing methods and can be difficult to adapt to equal opportunity based fairness. To overcome the disadvantages of these two types of approaches, we propose a learnable post-processing framework to enhance fairness.

The proposed learnable post-processing framework is shown in Figure 5.24a, which has three main components: (i) Similar to the non-parametric post-processing method, we keep the original base model unchanged, which can be any existing cold-start recommendation model, such as Heater, DropoutNet, DeepMusic, or KNN used in Section 5.4.3.2. (ii) Instead of the heuristic re-ranking in the non-parametric post-processing method, we learn an autoencoder (denoted as ψ) to conduct the 're-ranking'. We input the predicted score vector of an item i from the base model to the autoencoder ψ, where the score vector is denoted as $\widetilde{R}_{:,i} \in \mathbb{R}^N$ and the size of the vector is the total number of users N. Autoencoder ψ outputs the reconstructed score vector $\widehat{R}_{:,i} \in \mathbb{R}^N$ for final rankings. (iii) By introducing fairness to the training process of the autoencoder ψ, we enable ψ to produce fairer results for new items.

During training, we can only get access to the warm start items, so we use $\widetilde{R}_{:,i}$ for $i \in \mathcal{I}_w$ to train the autoencoder ψ. Then, during testing, we receive predicted scores from the base model for new items $\widetilde{R}_{:,i}$ with $i \in \mathcal{I}_c$, and feed them to the trained ψ to have fairness-enhanced scores for recommending these new items. Note that if we do not introduce fairness to the learning process of ψ and just make the output of the autoencoder $\widehat{R}_{:,i}$ as close as possible to the input $\widetilde{R}_{:,i}$, then the learnable post-processing framework will reproduce the same recommendations as the base model.

Enhancing fairness in this framework

The next question is how can we enhance the fairness among new items in such a framework? The ultimate goal of fairness is to promote the true positive rate for worst-off items so that they can receive similar true positive rate as best-served items. From the formulation of true positive rate (such as Equation 5.32), we know that if we can improve the expectation of ranking position to matched users for under-served items during testing, then fairness can be enhanced. However, ranking position is hard to control in a recommendation model. So we need to further align the ranking positions with scores predicted by models, then ensure the fairness by increasing the expectation of predicted scores to matched users for under-served items. Therefore, to achieve fairness, we need to accomplish two requirements:

Requirement-1: promote under-served items so that their distributions of matched-user predicted scores, denoted as $P(\widehat{R}_{\mathcal{U}_i^+,i})$, are as close as possible to the distributions of best-served items.

Requirement-2: for every user, the predicted scores, denoted as $\widehat{R}_{u,:}$, follow the same distribution.

By achieving Requirement-1, we try to maximize the expectation of predicted scores to matched users for under-served items. By achieving Requirement-2, we ensure that a specific predicted score value corresponds to a specific ranking position for all users. Hence, with both requirements fulfilled, under-served items are promoted to have higher expectation of ranking position to

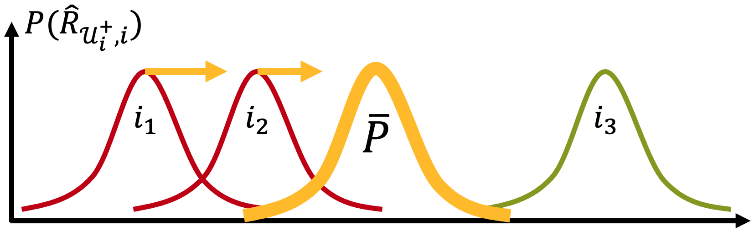


Figure 5.25: In each training epoch, update ψ to push $P(\widehat{R}_{\mathcal{U}_i^+,i})$ of under-estimated items (i_1 and i_2) as close as possible to the target $\overline{P}$ generated by φ. Reprinted with permission from [13].

matched users during testing, i.e., fair recommendations are provided.

To achieve these two requirements for new items during testing, we need to train the autoencoder ψ so that ψ achieves these requirements for warm start items during training. Requirement-2 is easy to accomplish. First, before training ψ, we normalize the score distributions of users in the outputs of the base model to a standard normal distribution (score distributions for users are usually considered as normal distributions [58, 12]). And we use the normalized scores as the ground truth to train ψ. By this, ψ will learn to output scores normalized for users. Thus, during training, we have the predicted score matrix for warm start items $\widetilde{R} \in \mathbb{R}^{N\times M_w}$ from the base model, and we normalize it for each user:

$$\widetilde{R}^N_{u,:} = (\widetilde{R}_{u,:} - Mean(\widetilde{R}_{u,:}))/Std(\widetilde{R}_{u,:}), \tag{5.33}$$

where $Mean(\cdot)$ calculates the mean value of a sequence; $Std(\cdot)$ calculates the standard deviation of a sequence; and $\widetilde{R}^N_{u,:}$ with all $u \in \mathcal{U}$ form the normalized score matrix $\widetilde{R}^N$ to train the autoencoder ψ. A good property of this method is that because the recommendation is based on a ranked list of items for each user, the normalization of scores for each user will not influence the ranking order.

Now, the remaining challenge is how to accomplish Requirement-1. To tackle this, following the learnable post-processing framework, we propose two concrete models: a novel joint-learning generative method (**Gen**); and a score scaling method (**Scale**).

5.4.4.2 The Joint-learning Generative Method

The overall framework of the joint-learning generative method is shown in Figure 5.24b where there are two main components: an autoencoder ψ and a distribution generator φ. The intuition of Gen is: in each training epoch, the distribution generator φ first generates a target distribution $\overline{P}$; then we update the autoencoder ψ so that it promotes items that are under-estimated in prior epochs by pushing their matched-user score distributions $P(\widehat{R}_{\mathcal{U}_i^+,i})$ as close as possible to the target distribution $\overline{P}$, and at the same time, have ψ preserve the recommendation utility as much as possible; last we update the distribution generator φ to generate a new target distribution $\overline{P}$ as the average of all items' matched-user score distributions $P(\widehat{R}_{\mathcal{U}_i^+,i})$. We show an example in Figure 5.25, where there are 3 items i_1, i_2 and i_3 and we show their matched-user score distributions at current training epoch. i_1 and i_2 have worse distributions (lower expectations of matched-user scores) than i_3, and we have the target distribution $\overline{P}$ which is the average of all three items. So, in this epoch, we need to update ψ so that $P(\widehat{R}_{\mathcal{U}_i^+,i})$ of i_1 and i_2 are promoted to be close to $\overline{P}$. After this, we also need to update the $\overline{P}$ to be the average of the new $P(\widehat{R}_{\mathcal{U}_i^+,i})$ of these three items. By jointly learning these two components, we can push the matched-user score distribution $P(\widehat{R}_{\mathcal{U}_i^+,i})$ of under-estimated items (like i_1 and i_2) to be closer and closer to best-served items (like i_3) epoch by epoch, and eventually achieve a fair status.

Concretely, the autoencoder ψ is the same as the one in the framework in Section 5.4.4.1, which takes predicted score vectors $\widetilde{R}_{:,i}$ from a base model as input and outputs $\widehat{R}_{:,i}$, with user-normalized score vectors $\widetilde{R}^N_{:,i}$ as training ground-truth.

The distribution generator φ is to generate a target distribution $\overline{P}$ (the average distribution of $P(\widehat{R}_{\mathcal{U}_i^+,i})$ over all items from last epoch) for under-estimated items to be promoted to. φ is a multi-layer perceptron (MLP) with 1-dimension input and output layers, which takes S random seeds from a standard normal distribution as inputs and outputs S samples $\overline{R} \in \mathbb{R}^S$ to represent the target distribution $\overline{P}$. To generate such a $\overline{R}$, at the end of each training epoch, we first retrieve the matched-user entries $\widehat{R}_{\mathcal{U}_i^+,i}$ in the output vectors $\widetilde{R}^N_{:,i}$ from the autoencoder ψ for all warm items $\mathcal{I}_w$. Then, for each item i, we update φ to minimize the sum of distribution distances between generated samples $\overline{R}$ from φ and matched-user scores $\widehat{R}_{\mathcal{U}_i^+,i}$ so that the underlying distribution of $\overline{R}$ has the minimum sum of distances to all items. For example, in Figure 5.25, the generated $\overline{P}$ has the minimum sum of distances to these three items, that is, $\overline{P}$ is the average distribution of these items. Here, we adopt the Maximum Mean Discrepancy (MMD) [151], an effective kernel based statistic test method, to calculate the distribution distance by samples from two distributions as the loss for the distribution generator:

$$\begin{aligned}\min_{\varphi} \mathcal{L}_{gen} &= \sum_{i \in \mathcal{I}_w} MMD(\overline{R}, \widehat{R}_{\mathcal{U}_i^+,i}) \\ &= \sum_{i \in \mathcal{I}_w} (\frac{1}{S^2} \sum_{x,y=1}^{S} f(\overline{R}[x], \overline{R}[y]) - \frac{2}{S^2} \sum_{x=1}^{S} \sum_{y=1}^{S} f(\overline{R}[x], \widehat{R}_{\mathcal{U}_i^+,i}[y]) \\ &+ \frac{1}{S^2} \sum_{x,y=1}^{S} f(\widehat{R}_{\mathcal{U}_i^+,i}[x], \widehat{R}_{\mathcal{U}_i^+,i}[y])),\end{aligned} \tag{5.34}$$

where $f(a,b) = exp(-(a-b)^2/l^2)$ is a Gaussian kernel with $l = 1$. Note that we do not need a regularization for φ because there is no overfitting problem for it.

After generating samples $\overline{R}$ following the target distribution $\overline{P}$, we then update the autoencoder ψ by:

$$\min_{\psi} \mathcal{L}_{AE} = \sum_{i \in \mathcal{I}_w} (\|\widetilde{R}^N_{:,i} - \widehat{R}_{:,i}\|_{\text{F}} + \alpha(MMD(\overline{R}, \widehat{R}_{\mathcal{U}_i^+,i}) \cdot \delta(i \in \mathcal{I}_{UE}))) + \lambda \|\psi\|_{\text{F}}, \tag{5.35}$$

where $\|\psi\|_{\text{F}}$ is the L2 regularization and λ is the regularization weight; the RMSE part $\|\widetilde{R}^N_{:,i} - \widehat{R}_{:,i}\|_{\text{F}}$ is to preserve the recommendation utility from the base model; α is the fairness-strength weight to control the fairness enhancement strength: the larger the more strength for improving fairness; $\mathcal{I}_{UE}$ is a subset of items that are under-estimated by the autoencoder ψ from last epoch: we first calculate the mean value of the matched-user scores for each item i as $m_i = Mean(\widehat{R}_{\mathcal{U}_i^+,i})$, then determine the under-estimated item set $\mathcal{I}_{UE}$ with items whose mean values are lower than the average of all items, i.e., $i \in \mathcal{I}_{UE}$ if $m_i < Mean(m_{\mathcal{I}_w})$. As a result, by this loss, we push $P(\widehat{R}_{\mathcal{U}_i^+,i})$ for $i \in \mathcal{I}_{UE}$ to be close to $\overline{P}$.

5.4.4.3 *The Score Scaling Method*

In addition to Gen, we adapt an existing work for addressing popularity bias [80] into the proposed learnable post-processing framework to enhance fairness among new items. This score scaling method can also serve as a baseline method.

To counteract popularity bias, the work [80] re-scales the training data based on item popularity: it up-scales ratings for unpopular items and down-scales ratings for popular items of high popularity, and then it trains a recommendation model on the scaled data to deliver debiased recommendations. Similarly, we can scale the user-normalized predicted score vectors $\widetilde{R}^N_{:,i}$ to be fairer ground-truth to train a fair autoencoder ψ in the proposed post-processing framework. In detail, for the user-normalized score vector $\widetilde{R}^N_{:,i}$ of each warm start item $i \in \mathcal{I}_w$, we scale the matched-user entries:

$$\widetilde{R}^{NS}_{\mathcal{U}^+_i,i} = \widetilde{R}^N_{\mathcal{U}^+_i,i} \times \frac{Max(\{Mean(\widetilde{R}^N_{\mathcal{U}^+_j,j})^\beta | j \in \mathcal{I}_w\})}{Mean(\widetilde{R}^N_{\mathcal{U}^+_i,i})^\beta}, \tag{5.36}$$

where $\widetilde{R}^N_{\mathcal{U}^+_i,i}$ are the matched-user entries in $\widetilde{R}^N_{:,i}$; $Mean(\widetilde{R}^N_{\mathcal{U}^+_i,i})$ calculates the mean value of matched-user entries; $Max(\cdot)$ returns the maximum value in a sequence, and for the numerator, we use $Max(\{Mean(\widetilde{R}^N_{\mathcal{U}^+_j,j})^\beta | j \in \mathcal{I}_w\})$ so that no item is down-scaled but only under-estimated items are up-scaled during the scaling; β is the fairness-strength weight – the larger the more strength to enhance fairness; $\widetilde{R}^{NS}_{\mathcal{U}^+_i,i}$ denotes the scaled entries for matched users, and we write them back to $\widetilde{R}^N_{:,i}$ to have the final score vector $\widetilde{R}^{NS}_{:,i}$ for i as the ground-truth for training the autoencoder ψ. The overall framework of the proposed Scale is shown in Figure 5.24c.

By this, we have fairness-enhanced scores for warm items. The autoencoder learned with these scaled scores as ground truth will bring fairer recommendations for new items during testing. We adopt the RMSE loss for learning the proposed Scale model:

$$\min_{\psi} \mathcal{L}_{Scale} = \sum_{i \in \mathcal{I}_w} \|\widetilde{R}^{NS}_{:,i} - \widehat{R}_{:,i}\|_{\text{F}} + \lambda \|\psi\|_{\text{F}}. \tag{5.37}$$

5.4.5 Experiments

In this section, we conduct extensive experiments to answer three key research questions: **RQ1** how does the Gen model enhance fairness for new items and preserve utility, compared with Scale and other baselines? **RQ2** what is the impact of the hyper-parameters in the two proposed methods? and **RQ3** what is the impact of the proposed fairness-enhancement methods on group-level fairness?

5.4.5.1 Experimental Settings

Data and Metrics

Similar to the data-driven study in Section 5.4.3.2, we use the same four datasets (ML1M, ML20M, CiteULike, and XING) and same metrics: we report NDCG@k with k=15 and 30 for evaluating user-view utility; MDG-all for item-view utility; MDG-min10% and MDG-min20% (the larger the fairer a model is) for fairness; and we also report MDG-max10% for comparison.

Baselines

We use the same four cold-start recommendation base models (Heater, DropoutNet, DeepMusic, and KNN) as introduced in Section 5.4.3.2. And we investigate the fairness-enhancement performance of the proposed Gen and Scale. We consider the Scale method, which adapts a previous work for addressing popularity bias into the proposed learnable post-processing framework, as one baseline to compare with Gen. Besides, we saw in Section 5.4.3.2 that random rankings can also improve the true positive rates of worst-off items compared with personalized cold-start recommendation algorithms. Hence, we also consider a **Noise** method which adds random noise to the output of base cold-start recommendation models as another baseline, in which there is a fairness-strength weight γ to control the amount of noise added.

Reproducibility

All models are implemented by Tensorflow [59] and optimized by Adam algorithm [60]. For the four cold-start recommendation base models, we follow the hyper-parameter tuning strategies in [7]. For the two proposed fairness-enhancement models, we assign the autoencoder ψ a single hidden layer of dimension 100 with a linear activation function. For Gen, we assign the distribution generator φ two hidden layers of dimension 50 with a *tanh* activation function. We tune the fairness-strength weight α, β, γ in Gen, Scale, and Noise models by grid search on validation sets. Because there is a trade-off between the user-view utility and fairness, when tuning the

Table 5.14: Empirical results on ML1M dataset for all models. Reprinted with permission from [13].

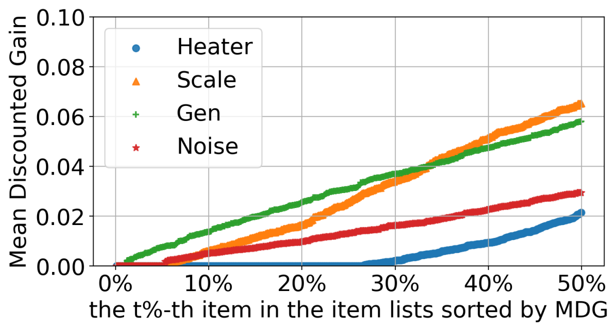
	NDCG		MDG-all	Fairness: MDG		
	@15	@30		min10%	min20%	max10%
Heater	0.5516	0.5332	0.0525	0.0000	0.0000	0.2272
Noise	0.4240	0.4084	0.0482	0.0017	0.0046	0.1730
Scale	0.5282	0.5135	0.0755	0.0015	0.0066	0.2025
Gen	0.5379	0.5206	0.0719	0.0073	0.0136	0.2036
DN	0.5488	0.5316	0.0552	0.0000	0.0000	0.2294
Noise	0.4586	0.4420	0.0513	0.0010	0.0037	0.1876
Scale	0.5315	0.5150	0.0766	0.0015	0.0069	0.2057
Gen	0.5345	0.5175	0.0745	0.0075	0.0138	0.2055
DM	0.5312	0.5167	0.0572	0.0000	0.0001	0.2323
Noise	0.4406	0.4304	0.0543	0.0007	0.0032	0.1937
Scale	0.5058	0.4946	0.0726	0.0010	0.0047	0.2140
Gen	0.5144	0.5024	0.0730	0.0027	0.0071	0.2136
KNN	0.4402	0.4226	0.0646	0.0001	0.0020	0.2091
Noise	0.3450	0.3378	0.0591	0.0016	0.0053	0.1643
Scale	0.4181	0.4027	0.0712	0.0023	0.0084	0.1791
Gen	0.4158	0.4002	0.0724	0.0075	0.0140	0.1831

fairness-strength weights α and β, we try to preserve a relatively high NDCG@k for both methods and analyze the fairness performance of them. *All code, data, and settings will be available at* `https://github.com/Zziwei/Fairness-in-Cold-Start-Recommendation`.

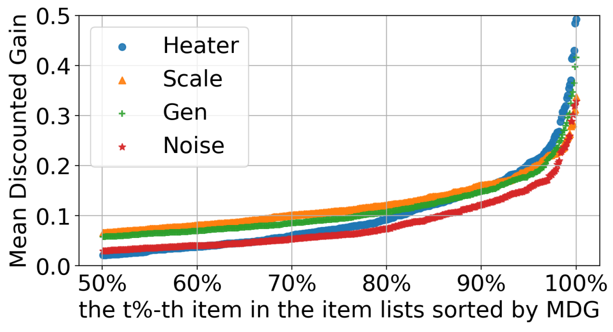
5.4.5.2 *RQ1: Fairness-Enhancement Performance*

First, we investigate how do the two proposed methods perform in terms of improving fairness and preserving recommendation utility. We first apply the two proposed methods and baseline Noise to each of the four cold-start recommendation models. Results on the Ml1M dataset are reported in Table 5.14, where we show results of the four cold-start recommendation base models. The three rows following each of the cold-start recommendation base models are results of Noise, Scale, and Gen with the given cold-start recommendation model as the base model (forming a four-row group).

From Table 5.14, comparing the results before and after applying the two proposed methods Scale and Gen, we have four major observations: (i) for all cold-start recommendation methods, after applying the proposed models, the user-view utility decreases but with small percentages;



(a) 0∼50%-th items.



(b) 50∼100%-th items.

Figure 5.26: Heater as base, MDG of items by different models. Reprinted with permission from [13].

(ii) after applying Scale and Gen, the fairness among new items (evaluated by MDG-min10% and MDG-min20%) is significantly improved; (iii) after applying proposed methods, the utility for the best-served items (measured by MDG-max10%) decreases, but with a limited percentage as well; and (iv) the item-view utility (evaluated by MDG-all) significantly increases after applying Scale and Gen. This item-view utility improvement is due to that items originally under-served by base models receive more utility from the two proposed models, leading to the improvement of overall item-view utility even though the best-served items receive lower utility. Based on these observations, we can conclude that Scale and Gen can significantly enhance the fairness among new items; and they can also effectively preserve the user-view utility and improve the item-view utility.

Next, we compare the performance between Noise, Scale, and Gen in Table 5.14. We can find that Scale and Gen promote fairness to a greater extent with higher utility (both user-view and item view) preserved than Noise, showing the effectiveness of the proposed framework and methods. Then, comparing Scale and Gen, under the circumstance that they produce similar user-view utility: (i) Scale provides slightly higher item-view utility than Gen; but (ii) Gen delivers better fairness-enhancement performance than Scale: Gen outperforms Scale for 295.68% for MDG-min10% and 80.95% for MDG-min20%. As a result, we conclude that Gen can enhance fairness for new items more effectively than Scale.

To better understand the effects of the proposed Scale and Gen, on the ML1M dataset, we sort the recommended new items by MDG in ascending order and plot them in Figure 5.26 for Heater

Table 5.15: Results on 4 datasets for Heater as base model. Reprinted with permission from [13].

		NDCG		MDG-all	Fairness: MDG		
		@15	@30		min10%	min20%	max10%
ML1M	Heater	0.5516	0.5332	0.0525	0.0000	0.0000	0.2272
	Noise	0.4240	0.4084	0.0482	0.0017	0.0046	0.1730
	Scale	0.5282	0.5135	0.0755	0.0015	0.0066	0.2025
	Gen	0.5379	0.5206	0.0719	0.0073	0.0136	0.2036
ML20M	Heater	0.4408	0.4308	0.0187	0.0000	0.0000	0.1455
	Noise	0.3600	0.3509	0.0184	0.0000	$6e^{-6}$	0.1144
	Scale	0.4166	0.4104	0.0302	0.0000	$2.5e^{-5}$	0.1443
	Gen	0.4265	0.4165	0.0296	0.0003	0.0014	0.1430
CiteULike	Heater	0.2268	0.2670	0.1833	0.0046	0.0251	0.5106
	Noise	0.2095	0.2481	0.1779	0.0051	0.0259	0.4958
	Scale	0.2202	0.2610	0.1867	0.0055	0.0270	0.5099
	Gen	0.2187	0.2599	0.1869	0.0111	0.0332	0.5034
XING	Heater	0.2251	0.2762	0.1333	0.0028	0.0129	0.3821
	Noise	0.2051	0.2553	0.1301	0.0038	0.0142	0.3561
	Scale	0.2183	0.2701	0.1414	0.0038	0.0162	0.3801
	Gen	0.2185	0.2712	0.1487	0.0093	0.0256	0.3755

and three fairness-enhancement methods with Heater as the base model. Figure 5.26a shows the MDG for items belonging to the first half of the sorted list (from 0%-th to 50%-th items), and Figure 5.26b shows the MDG for items belonging to the left half of the sorted list (from 50%-th to 100%-th items). From these two figures, we see that compared to the base model Heater: both Scale and Gen significantly increase MDG (better than Noise) for most items (around from 0%-th to 90%-th in the sorted item list), which are originally under-served by Heater; and MDG for items that are best-served by Heater (around 90%-th to 100%-th) are only slightly decreased by Scale and Gen. Moreover, another interesting observation is that Gen improves the MDG of the worst-served items better than Scale as shown in Figure 5.26a, while Scale promotes the MDG for items between the worst-served and best-served (around 40%-th to 90%-th) more than Gen. This is the reason why Gen outperforms Scale for fairness but Scale performs better for item-view utility as in Table 5.14.

Last, in Table 5.15, we report the results of Heater and the three fairness-enhancement methods with Heater as base model on all datasets. Similar conclusions can be drawn from these results: for all different datasets, the proposed Scale and Gen can more effectively enhance the fairness for

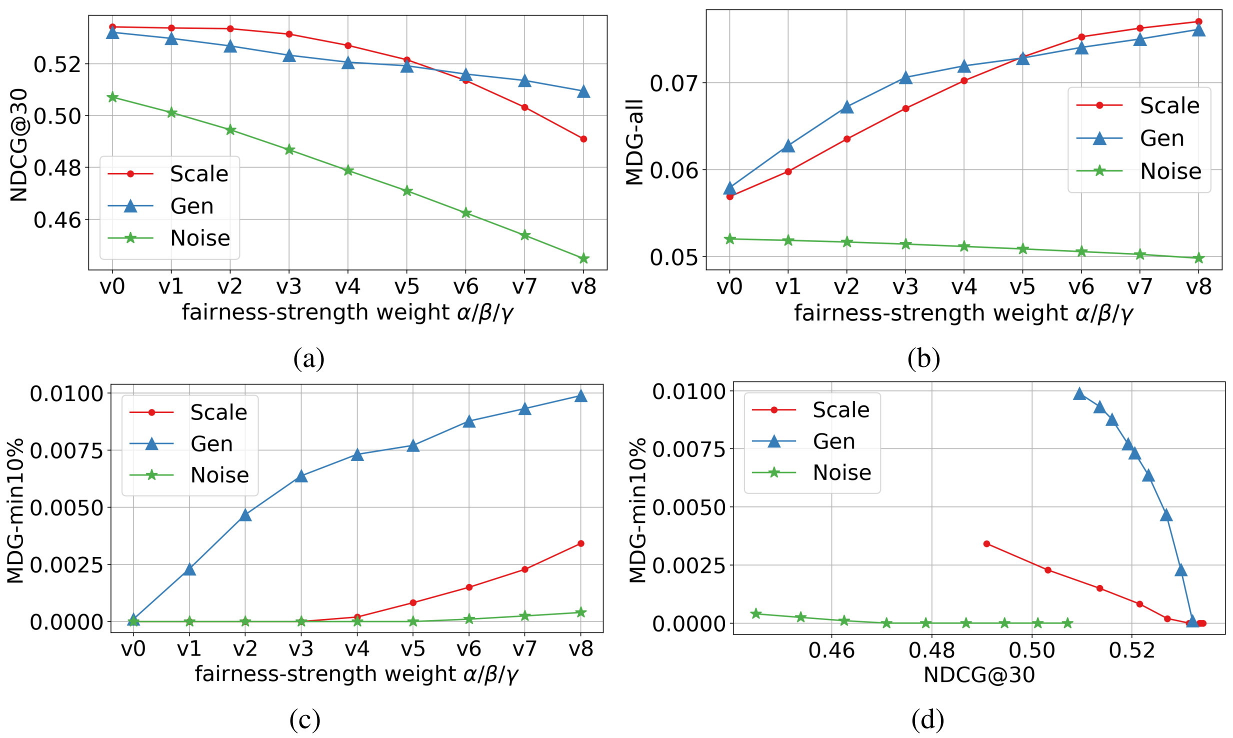


Figure 5.27: Investigate impact of α, β, and γ on ML1M dataset: (a) shows the impact on NDCG@30; (b) shows the impact on MDG-all; and (c) shows the impact on NDCG@30 and MDG-max10% together. Reprinted with permission from [13].

new items and preserve the utility.

5.4.5.3 RQ2: Impact of Hyper-parameters

We next turn to study the impact of the fairness-strength weights α in Gen, β in Scale, and γ in Noise. Recall that larger α, β, and γ lead to more strength for enhancing fairness. We run experiments for Gen with α varying from 40 to 360 with step 40, Scale with β varying from 1 to 5 with step 0.5, and Noise with γ varying from 0.4 to 0.8 with step 0.5. In Figure 5.27a, we show the results of NDCG@30 on the ML1M dataset, where v_0 to v_8 correspond to different values of α, β, and γ in ascending order. The figure shows that with larger fairness-strength weights, the user-view utility gets smaller for all methods. Then, we present how the item-view utility (measured by MDG-all) changes when we increase the fairness-strength weights in Figure 5.27b. We can observe that with weights increasing, MDG-all keeps increasing for both Scale and Gen, but slowly decreases for Noise. Next, we show how fairness (MDG-min10%) changes with weights increasing in Figure 5.27c, which demonstrates that all models improve fairness when the weights

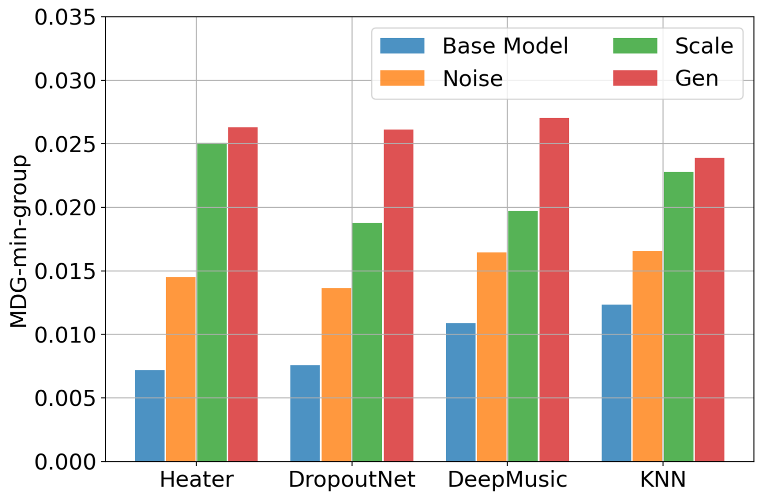


Figure 5.28: Investigate group-level fairness on ML1M. Reprinted with permission from [13].

get larger.

For Figure 5.27a, Figure 5.27b, and Figure 5.27c, note that we cannot directly compare the NDCG@30, MDG-all, and MDG-min10% between Scale, Gen, and Noise because the x-axis (values of α, β, and γ) is not the same for these two methods. Therefore, to further compare these three methods, we plot Figure 5.27d: the y-axis is MDG-min10%; the x-axis is NDCG@30; each dot represents an experiment result of a model with a specific fairness-strength weight; and weights are in decreasing order from left to right (for example, the leftmost dot for Gen corresponds to the experiment with $\alpha = 360$). Now, we can conclude from this figure that with the same user-view utility preserved, Gen enhances fairness more effectively than Scale and Noise for different fairness-strength weights.

5.4.5.4 RQ3: Impact on Group-level Fairness

The fairness we discussed so far is for individual items: we consider the difference among individual items as unfairness. Another widely investigated concept is group-level fairness: consider the difference among different items groups determined by item attributes as unfairness [26, 24, 12]. Here, we want to study how do our proposed methods impact group-level fairness in cold-start recommendation.

To be consistent to the measurement of fairness we study here, we evaluate the group-level fairness by calculating the average MDG for the worst-off item group (the item group with the lowest average MDG), which is also similar to the measurement of group-level fairness in a classification task [152]. We denote the group-level fairness metric as **MDG-min-group**. For the ML1M dataset, we show the group-level fairness results of all four cold-start recommendation base models and their corresponding fairness-enhanced results by the three methods in Figure 5.28, where we group items by the movie genres provided by ML1M dataset [96], and the worst-off movie genre for all methods is 'Documentary'. From Figure 5.28, we see that after applying the two proposed methods and baseline Noise, the group-level fairness is significantly improved, and Gen performs better than Scale and Noise. This result is reasonable and expected, because these methods improve MDG for all under-served items, resulting in under-served groups consisting of under-served items being promoted in general. This property is very helpful when group-level fairness is required but no group attribute is accessible.

5.4.6 Summary

In this section, we investigate the fairness among new items in cold start recommenders. We first empirically show the prevalence of unfairness in cold start recommenders. Then, to enhance the fairness among new items, we propose a novel learnable post-processing framework as a solution blueprint and propose two concrete models – Scale and Gen – following this blueprint. Last, extensive experiments show the effectiveness of the two proposed models for enhancing fairness and preserving recommendation utility.

5.5 Conclusions

In this chapter, we study how to measure and enhance recommendation fairness in different recommendation scenarios. First, we focus on how to enhance fairness for a multi-dimension recommendation task. We propose a fairness-aware tensor-based recommendation algorithm, which is able to isolate and extract sensitive information in the latent factors and delivers fair recommendation result by the non-sensitive latent factors. Then, we switch our attention to personalized ranking recommender systems to directly measure and enhance fairness on ranking result instead of the the predicted score, which is intermediate result in a real-world system. Specifically, we propose two new fairness measurements based on well-known concepts of statistical parity and equal opportunity. Correspondingly, we also design a novel algorithm adopting the adversarial learning technique, which is able to produce outstanding performance and is flexible to be applied to any exisitng recommendation algorithms. At last, we target recommendation fairness in a cold-start recommender system and aim to improve the fairness among new items with no historical feedback data. To tackle this problem, we propose a novel learnable post-processing framework as a solution blueprint and propose two concrete models – Scale and Gen – following this blueprint. Extensive experiments using real-world data have been conducted to show the state-of-the-art performance of our developed algorithms.

6. CONCLUSION AND FUTURE RESEARCH OPPORTUNITIES

Recommender systems are essential conduits: they can shape the media we consume, the jobs we seek, and even the friendships and professional contacts that form our social circles. With such a wide impact, recommender systems can exert strong, but often unforeseen, and sometimes even detrimental influence on the society in terms of culture, lifestyles, politics, education, ethics, economic well-being, and even social justice. Hence, in this dissertation research, we aim to lay the foundation for new responsible recommender systems by identifying, analyzing, and alleviating potential risks and harms on users, item providers, the platforms, and ultimately the society.

Specifically, we make three unique contributions toward responsible recommender systems. First, we study how to counteract the exposure bias in user-item interaction data. We first develop a novel and effective combinational joint learning framework to deliver unbiased recommendation with biased training data. And then, we further explore how to provide high-accuracy recommendations in the scenario with extreme exposure bias. For this, we propose a new cold-start recommendation algorithm – Heater – utilizing a randomized training mechanism and a Mixture-of-Experts Transformation structure. By extensive experiments, we show how our proposed methods effectively counteract the exposure bias and outperform state-of-the-art baselines.

For the second contribution of this dissertation, we move our focus from the bias in data to the bias in machine learning algorithms. We investigate how machine learning based recommendation algorithms introduce bias on items and users in the system. We first introduce the popularity-opportunity bias on items with empirical and theoretical study showing the existence of this bias. Then, we develop a simple but powerful post-processing method for debiasing. Besides, we also introduce the mainstream bias on users produced by recommendation algorithms, for which we explore both global and local methods to address this problem. Extensive experiments demonstrate that our proposed algorithms can effectively relieve the two different types of algorithmic bias.

At the end, our third contribution is to study how to measure and enhance fairness in recommender systems in different scenarios. First, we study how to enhance fairness in a multi-dimension recommender system and develop a fairness-aware tensor-based model to achieve the goal. Then, we turn to the personalized ranking recommender system. We introduce two new fairness measurements directly based on ranking results and propose an adversarial learning based algorithm for enhancing fairness in the personalized ranking system. At last, we investigate fairness in the cold-start recommender system and develop a novel learnable post-processing framework and a joint-learning generative model to improve fairness among cold-start items without any historical interaction data. We conduct extensive experiments and show the outstanding performance of the proposed models for enhancing fairness in different scenarios.

While in this dissertation, we make a substantial contribution toward identifying, analyzing, and mitigating data bias, algorithmic bias, and fairness in recommendation, there are many remaining challenges and open questions waiting to be studied. For the future research directions, we are specifically interested in three topics:

- **Exploring the inherent relationship between different types of algorithmic bias.** Although many different types of bias produced by the recommendation algorithms are identified, and special characteristics corresponding to these various types of bias are revealed, we hypothesize that there is an inherent close relationship between them. For example, in Chapter 4, we show two very different types of bias – the popularity-opportunity bias on items and the mainstream bias on users. Although they are defined in terms of different stakeholders, it is not difficult to notice that the items a mainstream user likes are typically popular items, and a popular item is usually liked by mainstream users. Hence, these two types of bias are not independent. This induces us to plan to theoretically and empirically analyze this relationship and develop a uniform solution to address the two types of bias simultaneously by utilizing the uncovered relationship.

- **Analyzing and Alleviating Long-term and Dynamic Impacts.** Most of existing work in the community (including works in this dissertation) focus on studying the unfairness/bias in an offline and static manner. That is, the potential vicious impacts of algorithms are investigated by conducting a one-round experiment with offline data. However, these systems influence people over the long-term and dynamically adapt: users can be gradually influenced by these systems, with this influence changing over time. This influence can not be identified nor thoroughly analyzed by an offline, static experiment. In the future, we plan to further contribute to this under-explored area by developing new tools to analyze the long-term and dynamic impacts of recommender systems and new algorithms to address potential adverse impacts in a long-term and dynamic way. Specifically, we are interested in answering the following research questions: How to design simulation experiments to analyze long-term and dynamic impacts without access to real-world systems? How to audit real-world platforms for their potential risks without access to internal systems? And how the unfairness in recommender systems incurs polarization globally and homogenization locally among people over time? And how to alleviate these issues in a long-term and dynamic way?

- **Improving security and privacy in recommender systems.** Besides developing responsibility by enhancing fairness and alleviating bias, another important topic toward responsibility is to build trust between users and recommender systems. First, to be trustworthy, users need to know that the systems are robust to data noise from users or item providers, such as intentional and unintentional false behaviors from users and false information about items from providers; and the systems should not be susceptible to attacks, such as fake reviews to promote or demote target items. Second, to gain trust, the systems should provide privacy protection for users. To achieve this, recommender systems need to enable users to decide and control what data about or generated by them can be used by the machine learning models. In the future, we plan to develop new algorithms to augment the robustness and reliability, and also to endow the algorithms the ability to dynamically learn new knowledge and forget learned knowledge based on users' changing permission of access to data. Specifically, we are interested in tackling the following research questions: How to improve robustness of recommendation algorithms to data noise, including intentional and unintentional false behaviors from users and false information about items from providers? How to upgrade existing algorithms to defend against various attacks and threats? And how to enable recommendation algorithms to protect user privacy by equipping them flexibility to dynamically learn new knowledge and forget learned knowledge based on users' changing permission of access to data?

REFERENCES

[1] Z. Zhu, Y. He, Y. Zhang, and J. Caverlee, "Unbiased implicit recommendation and propensity estimation via combinational joint learning," p. 551–556, Fourteenth ACM Conference on Recommender Systems, 2020.

[2] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *Advances in neural information processing systems*, pp. 6904–6914, 2017.

[3] S. Sedhain, A. K. Menon, S. Sanner, L. Xie, and D. Braziunas, "Low-rank linear cold-start recommendation from social data," Proceedings of the AAAI Conference on Artificial Intelligence, 2017.

[4] J. Li, M. Jing, K. Lu, L. Zhu, Y. Yang, and Z. Huang, "From zero-shot learning to cold-start recommendation," Proceedings of the AAAI conference on artificial intelligence, 2019.

[5] M. Volkovs, G. Yu, and T. Poutanen, "Dropoutnet: Addressing cold start in recommender systems," p. 4957–4966, Advances in neural information processing systems, 2017.

[6] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, "Learning attribute-to-feature mappings for cold-start recommendations.," vol. 10, p. 176–185, 2010 IEEE International Conference on Data Mining, 2010.

[7] Z. Zhu, S. Sefati, P. Saadatpanah, and J. Caverlee, "Recommendation for new users and new items via randomized training and mixture-of-experts transformation," p. 1121–1130, Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020.

[8] Z. Zhu, Y. He, X. Zhao, Y. Zhang, J. Wang, and J. Caverlee, "Popularity-opportunity bias in collaborative filtering," p. 85–93, Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021.

[9] Z. Zhu and J. Caverlee, "Fighting mainstream bias in recommender systems via local fine tuning," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, p. 1497–1506, 2022.

[10] Z. Zhu, X. Hu, and J. Caverlee, "Fairness-aware tensor-based recommendation," p. 1153–1162, Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018.

[11] T. Kamishima and S. Akaho, "Considerations on recommendation independence for a find-good-items task," FATREC Workshop on Responsible Recommendation Proceedings, 2017.

[12] Z. Zhu, J. Wang, and J. Caverlee, "Measuring and mitigating item under-recommendation bias in personalized ranking systems," p. 449–458, Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, 2020.

[13] Z. Zhu, J. Kim, T. Nguyen, A. Fenton, and J. Caverlee, "Fairness among new items in cold start recommender systems," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 767–776, 2021.

[14] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, and M. Ispir, "Wide deep learning for recommender systems," p. 7–10, Proceedings of the 1st workshop on deep learning for recommender systems, 2016.

[15] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[16] W. Niu, J. Caverlee, and H. Lu, "Neural personalized ranking for image recommendation," Proceedings of the eleventh ACM international conference on web search and data mining, 2018.

[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," p. 173–182, Proceedings of the 26th international conference on world wide web, 2017.

[18] X. He, Z. He, X. Du, and T.-S. Chua, “Adversarial personalized ranking for recommendation,” The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018.

[19] “Google’s algorithms discriminate against women and people of colour.”

[20] “Amazon scraps secret ai recruiting tool that showed bias against women.”

[21] M. D. Ekstrand and D. Kluver, “Exploring author gender in book rating and recommendation,” *User Modeling and User-Adapted Interaction*, pp. 1–44, 2021.

[22] “Youtube under fire for recommending videos of kids with inappropriate comments.”

[23] “The ‘filter bubble’ explains why trump won and you didn’t see it coming.”

[24] A. Beutel, J. Chen, T. Doshi, H. Qian, L. Wei, Y. Wu, L. Heldt, Z. Zhao, L. Hong, and E. H. Chi, “Fairness in recommendation ranking through pairwise comparisons,” p. 2212–2220, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.

[25] R. Burke, “Multisided fairness for recommendation,” 2017.

[26] S. C. Geyik, S. Ambler, and K. Kenthapadi, “Fairness-aware ranking in search recommendation systems with application to linkedin talent search,” Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining, 2019.

[27] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, 2009.

[28] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” *Advances in artificial intelligence*, vol. 2009, 2009.

[29] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” vol. 42, Computer, IEEE Computer Society Press Los Alamitos, CA, USA, 2009.

[30] R. He and J. McAuley, “Vbpr: Visual bayesian personalized ranking from implicit feedback,” Proceedings of the AAAI Conference on Artificial Intelligence, 2016.

[31] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei, “Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence,” Proceedings of the 10th ACM conference on recommender systems, 2016.

[32] J. Wang and J. Caverlee, “Recommending music curators: A neural style-aware approach,” European Conference on Information Retrieval, 2020.

[33] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, “Next-item recommendation with sequential hypergraphs,” Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, 2020.

[34] F. Zhao and Y. Guo, “Improving top-n recommendation with heterogeneous loss,” Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 2016.

[35] T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, and T. Joachims, “Recommendations as treatments: Debiasing learning and evaluation,” 2016.

[36] H. Steck, “Training and testing of recommender systems on data missing not at random,” p. 713–722, Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010.

[37] H. Steck, “Evaluation of recommendations: rating-prediction and ranking,” p. 213–220, Proceedings of the 7th ACM conference on Recommender systems, 2013.

[38] X. Wang, R. Zhang, Y. Sun, and J. Qi, “Doubly robust joint learning for recommendation on data missing not at random,” p. 6638–6647, International Conference on Machine Learning, PMLR, 2019.

[39] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani, “Probabilistic matrix factorization with non-random missing data,” p. 1512–1520, International Conference on Machine Learning, 2014.

[40] Y. Saito, “Asymmetric tri-training for debiasing missing-not-at-random explicit feedback,” Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020.

[41] Y. Wang, D. Liang, L. Charlin, and D. M. Blei, “The deconfounded recommender: A causal inference approach to recommendation,” arXiv preprint arXiv:1808.06581, 2018.

[42] D. Liang, L. Charlin, and D. M. Blei, “Causal inference for recommendation,” Causation: Foundation to Application, Workshop at UAI. AUAI, 2016.

[43] B. M. Marlin and R. S. Zemel, “Collaborative prediction and ranking with non-random missing data,” Proceedings of the third ACM conference on Recommender systems, 2009.

[44] H. Steck, “Item popularity and recommendation accuracy,” Proceedings of the fifth ACM conference on Recommender systems, 2011.

[45] X. Zhao, Z. Zhu, Y. Zhang, and J. Caverlee, “Improving the estimation of tail ratings in recommender system with multi-latent representations,” p. 762–770, Proceedings of the 13th International Conference on Web Search and Data Mining, 2020.

[46] L. Yang, Y. Cui, Y. Xuan, C. Wang, S. Belongie, and D. Estrin, “Unbiased offline recommender evaluation for missing-not-at-random implicit feedback,” Proceedings of the 12th ACM conference on recommender systems, 2018.

[47] Y. Saito, “Unbiased pairwise learning from biased implicit feedback,” p. 5–12, Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval, 2020.

[48] Y. Saito, “Unbiased pairwise learning from implicit feedback,” in *NeurIPS 2019 Workshop on Causal Machine Learning*, 2019.

[49] V. , S. Dieleman, and B. Schrauwen, “Deep content-based music recommendation,” p. 2643–2651, Advances in neural information processing systems, 2013.

[50] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," p. 650–658, Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008.

[51] M. Saveski and A. Mantrach, "Item cold-start recommendations: learning local collective embeddings," p. 89–96, Proceedings of the 8th ACM Conference on Recommender systems, 2014.

[52] I. Barjasteh, R. Forsati, F. Masrour, A.-H. Esfahanian, and H. Radha, "Cold-start item and user recommendation with decoupled completion and transduction," p. 91–98, Proceedings of the 9th ACM Conference on Recommender Systems, 2015.

[53] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," p. 111–112, ACM, 2015.

[54] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "Melu: Meta-learned user preference estimator for cold-start recommendation," p. 1073–1082, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.

[55] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," p. 263–272, 2008 Eighth IEEE international conference on data mining, 2008.

[56] S. Kabbur, X. Ning, and G. Karypis, "Fism: factored item similarity models for top-n recommender systems," p. 659–667, Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013.

[57] S. Bonner and F. Vasile, "Causal embeddings for recommendation," p. 104–112, Proceedings of the 12th ACM conference on recommender systems, 2018.

[58] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," p. 1257–1264, Advances in neural information processing systems, 2007.

[59] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, "Tensorflow: a system for large-scale machine learning.," vol. 16,

p. 265–283, 12th USENIX symposium on operating systems design and implementation (OSDI 16), 2016.

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[61] M. Kompan and M. Bieliková, "Content-based news recommendation," International conference on electronic commerce and web technologies, 2010.

[62] M. J. Pazzani and D. Billsus, *Content-based recommendation systems*, p. 325–341. The adaptive web, Springer, 2007.

[63] R. Van Meteren and M. Van Someren, "Using content-based filtering for recommendation," in *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*, pp. 47–56, 2000.

[64] S. Sedhain, S. Sanner, D. Braziunas, L. Xie, and J. Christensen, "Social collaborative filtering for cold-start recommendations," p. 345–348, Proceedings of the 8th ACM Conference on Recommender systems, 2014.

[65] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," 2017.

[66] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," p. 448–456, Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 2011.

[67] I. Cantador, P. L. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems," 2011.

[68] F. Abel, Y. Deldjoo, M. Elahi, and D. Kohlsdorf, "Recsys challenge 2017: Offline and online evaluation," p. 372–373, Proceedings of the eleventh acm conference on recommender systems, 2017.

[69] Celma and P. Cano, “From hits to niches?: or how popular artists can bias music recommendation and discovery,” Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition, 2008.

[70] H. Abdollahpouri, R. Burke, and B. Mobasher, “Managing popularity bias in recommender systems with personalized re-ranking,” The thirty-second international flairs conference, 2019.

[71] H. Abdollahpouri, R. Burke, and B. Mobasher, “Controlling popularity bias in learning-to-rank recommendation,” Proceedings of the eleventh ACM conference on recommender systems, 2017.

[72] C. Anderson, *The long tail: Why the future of business is selling less of more*. Hachette Books, 2006.

[73] E. Brynjolfsson, Y. J. Hu, and M. D. Smith, “From niches to riches: Anatomy of the long tail,” 2006.

[74] Y.-J. Park and A. Tuzhilin, “The long tail of recommender systems and how to leverage it,” Proceedings of the 2008 ACM conference on Recommender systems, 2008.

[75] D. Jannach, L. Lerche, I. Kamehkhosh, and M. Jugovac, “What recommenders recommend: an analysis of recommendation biases and possible countermeasures,” 2015.

[76] G. Shani and A. Gunawardana, *Evaluating recommendation systems*. Recommender systems handbook, Springer, 2011.

[77] L. Chen, Y. Yang, N. Wang, K. Yang, and Q. Yuan, “How serendipity improves user satisfaction with recommendations? a large-scale user evaluation,” The world wide web conference, 2019.

[78] H. Abdollahpouri and R. Burke, “Reducing popularity bias in recommendation over time,” arXiv preprint arXiv:1906.11711, 2019.

[79] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, “Correcting popularity bias by enhancing recommendation neutrality,” RecSys Posters, 2014.

[80] H. Steck, “Collaborative filtering via high-dimensional regression,” 2019.

[81] H. Abdollahpouri, M. Mansoury, R. Burke, and B. Mobasher, “The unfairness of popularity bias in recommendation,” 2019.

[82] M. Schedl and C. Bauer, “Online music listening culture of kids and adolescents: Listening analysis and music recommendation tailored to the young,” 1st International Workshop on Children and Recommender Systems (KidRec 2017), co-located with 11th ACM Conference on Recommender Systems (RecSys 2017), 2017.

[83] M. D. Ekstrand, M. Tian, I. M. Azpiazu, J. D. Ekstrand, O. Anuyah, D. McNeill, and M. S. Pera, “All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and effectiveness,” p. 172–186, PMLR, 2018.

[84] Y. Li, H. Chen, Z. Fu, Y. Ge, and Y. Zhang, “User-oriented fairness in recommendation,” p. 624–632, Proceedings of the Web Conference 2021, 2021.

[85] Z. Fu, Y. Xian, R. Gao, J. Zhao, Q. Huang, Y. Ge, S. Xu, S. Geng, C. Shah, and Y. Zhang, “Fairness-aware explainable recommendation over knowledge graphs,” p. 69–78, Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020.

[86] X. Zhao, Z. Zhu, M. Alfifi, and J. Caverlee, “Addressing the target customer distortion problem in recommender systems,” p. 2969–2975, Proceedings of The Web Conference 2020, 2020.

[87] R. Z. Li, J. Urbano, and A. Hanjalic, “Leave no user behind: Towards improving the utility of recommender systems for non-mainstream users,” p. 103–111, Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021.

[88] M. Choi, Y. Jeong, J. Lee, and J. Lee, "Local collaborative autoencoders," p. 734–742, Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021.

[89] E. Christakopoulou and G. Karypis, "Local latent space models for top-n recommendation," p. 1235–1243, Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018.

[90] J. Lee, S. Kim, G. Lebanon, and Y. Singer, "Local low-rank matrix approximation," p. 82–90, International conference on machine learning, 2013.

[91] J. Lee, S. Kim, G. Lebanon, Y. Singer, and S. Bengio, "Llorma: Local low-rank matrix approximation," Journal of Machine Learning Research, 2016.

[92] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," p. 689–698, Proceedings of the 2018 world wide web conference, 2018.

[93] A. Beutel, J. Chen, Z. Zhao, and E. H. Chi, "Data decisions and theoretical implications when adversarially learning fair representations," 2017.

[94] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," p. 3315–3323, Advances in neural information processing systems, 2016.

[95] B. H. Zhang, B. Lemoine, and M. Mitchell, "Mitigating unwanted biases with adversarial learning," Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, 2018.

[96] H. F. Maxwell and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, p. 1–19, 2015.

[97] J. Tang, H. Gao, and H. Liu, "mtrust: discerning multi-faceted trust in a connected world," Proceedings of the fifth ACM international conference on Web search and data mining, 2012.

[98] J. McAuley, C. Targett, Q. Shi, and V. Den, “Image-based recommendations on styles and substitutes,” Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, 2015.

[99] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” 1987.

[100] B. Rastegarpanah, K. P. Gummadi, and M. Crovella, “Fighting fire with fire: Using antidote data to improve polarization and fairness of recommender systems,” Proceedings of the twelfth ACM international conference on web search and data mining, 2019.

[101] W. Liu and R. Burke, “Personalizing fairness-aware re-ranking,” 2018.

[102] S. Yao and B. Huang, “Beyond parity: Fairness objectives for collaborative filtering,” p. 2921–2930, Advances in neural information processing systems, 2017.

[103] P. Cremonesi, Y. Koren, and R. Turrin, “Performance of recommender algorithms on top-n recommendation tasks,” p. 39–46, Proceedings of the fourth ACM conference on Recommender systems, 2010.

[104] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, “Neural graph collaborative filtering,” p. 165–174, Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, 2019.

[105] Y. D. Challenge, “Yelp dataset challenge,” 2013.

[106] I. Ben-Gal, *Outlier detection*, p. 131–146. Springer, 2005.

[107] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “Lof: identifying density-based local outliers,” p. 93–104, Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000.

[108] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, “Deep one-class classification,” p. 4393–4402, International conference on machine learning, PMLR, 2018.

[109] S. Yang, L. Liu, and M. Xu, "Free lunch for few-shot learning: Distribution calibration," International Conference on Learning Representations, 2020.

[110] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," p. 1126–1135, International conference on machine learning, 2017.

[111] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018.

[112] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and L. Antiga, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, p. 8026–8037, 2019.

[113] B. Friedman and H. Nissenbaum, "Bias in computer systems," *ACM Transactions on Information Systems (TOIS)*, vol. 14, no. 3, p. 330–347, 1996.

[114] S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, and A. Huq, "Algorithmic decision making and the cost of fairness," 2017.

[115] D. Pedreshi, S. Ruggieri, and F. Turini, "Discrimination-aware data mining," p. 560–568, Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008.

[116] C. Russell, M. J. Kusner, J. Loftus, and R. Silva, "When worlds collide: integrating different counterfactual assumptions in fairness," p. 6417–6426, Advances in neural information processing systems, 2017.

[117] M. B. Zafar, I. Valera, M. Rodriguez, K. Gummadi, and A. Weller, "From parity to preference-based notions of fairness in classification," p. 228–238, Advances in Neural Information Processing Systems, 2017.

[118] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning fair representations," p. 325–333, International conference on machine learning, 2013.

[119] T. Kamishima, S. Akaho, H. Asoh, and I. Sato, "Model-based approaches for independence-enhanced recommendation," p. 860–867, 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), 2016.

[120] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, "Efficiency improvement of neutrality-enhanced recommendation," Human Decision Making in Recommender Systems (Decisions@ RecSys' 13), 2013.

[121] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, "Recommendation independence," Conference on fairness, accountability and transparency, 2018.

[122] F. Prost, H. Qian, Q. Chen, E. H. Chi, J. Chen, and A. Beutel, "Toward a better trade-off between performance and fairness with kernel-based distribution matching," "ML with Guarantees" workshop at 33rd Conference on Neural Information Processing Systems, 2019.

[123] A. J. Biega, K. P. Gummadi, and G. Weikum, "Equity of attention: Amortizing individual fairness in rankings," p. 405–414, The 41st international acm sigir conference on research & development in information retrieval, 2018.

[124] A. Singh and T. Joachims, "Fairness of exposure in rankings," p. 2219–2228, Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018.

[125] A. Singh and T. Joachims, "Policy learning for fairness in ranking," Advances in Neural Information Processing Systems, 2019.

[126] A. Beutel, E. H. Chi, Z. Cheng, H. Pham, and J. Anderson, "Beyond globally optimal: Focused learning for improved recommendations," Proceedings of the 26th International Conference on World Wide Web, 2017.

[127] A. Krishnan, A. Sharma, A. Sankar, and H. Sundaram, "An adversarial approach to improve long-tail performance in neural collaborative filtering," Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018.

[128] N. Hurley and M. Zhang, “Novelty and diversity in top-n recommendation–analysis and evaluation,” 2011.

[129] H. Steck, “Calibrated recommendations,” p. 154–162, Proceedings of the 12th ACM conference on recommender systems, 2018.

[130] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, “Enhancement of the neutrality in recommendation.,” in *Decisions@ RecSys*, pp. 8–14, 2012.

[131] S. Yao and B. Huang, “New fairness metrics for recommendation that embrace differences,” arXiv preprint arXiv:1706.09838, 2017.

[132] Y. Koren, “Collaborative filtering with temporal dynamics,” *Communications of the ACM*, vol. 53, no. 4, p. 89–97, 2010.

[133] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui, “Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation,” p. 831–840, Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014.

[134] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, “Fast matrix factorization for online recommendation with implicit feedback,” p. 549–558, Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016.

[135] H. Ge, J. Caverlee, and H. Lu, “Taper: A contextual tensor-based approach for personalized expert recommendation.,” p. 261–268, Proceedings of the 10th ACM Conference on Recommender Systems, 2016.

[136] S. Rendle, B. Marinho, A. Nanopoulos, and L. Schmidt-Thieme, “Learning optimal ranking with tensor factorization for tag recommendation,” p. 727–736, Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009.

[137] S. Rendle and L. Schmidt-Thieme, “Pairwise interaction tensor factorization for personalized tag recommendation,” p. 81–90, Proceedings of the third ACM international conference on Web search and data mining, 2010.

[138] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," p. 79–86, Proceedings of the fourth ACM conference on Recommender systems, 2010.

[139] T. Kamishima, S. Akaho, and J. Sakuma, "Fairness-aware learning through regularization approach," p. 643–650, 2011 IEEE 11th International Conference on Data Mining Workshops, 2011.

[140] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," vol. 51, p. 455–500, SIAM review, SIAM, 2009.

[141] R. Burke, N. Sonboli, and A. Ordonez-Gauger, "Balanced neighborhoods for multi-sided fairness in recommendation," p. 202–214, Conference on fairness, accountability and transparency, PMLR, 2018.

[142] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," p. 161–169, Proceedings of the eleventh ACM conference on recommender systems, 2017.

[143] L. Lorigo, M. Haridasan, H. Brynjarsdóttir, L. Xia, T. Joachims, G. Gay, L. Granka, F. Pellacini, and B. Pan, "Eye tracking and online search: Lessons learned and challenges ahead," *Journal of the American Society for Information Science and Technology*, vol. 59, no. 7, p. 1041–1052, 2008.

[144] G. Louppe, M. Kagan, and K. Cranmer, "Learning to pivot with adversarial networks," Advances in neural information processing systems, 2017.

[145] J. Rawls, *Justice as fairness: A restatement.* Harvard University Press, 2001.

[146] C. D. Manning, H. Schütze, and P. Raghavan, *Introduction to information retrieval.* Cambridge university press, 2008.

[147] J. Vig, S. Sen, and J. Riedl, "The tag genome: Encoding community knowledge to support novel interaction," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 2, no. 3, p. 1–44, 2012.

[148] R. Cañamares and P. Castells, “Should i follow the crowd?: A probabilistic analysis of the effectiveness of popularity in recommender systems,” The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018.

[149] T. T. Nguyen, P.-M. Hui, H. F. Maxwell, L. Terveen, and J. A. Konstan, “Exploring the filter bubble: the effect of using recommender systems on content diversity,” p. 677–686, Proceedings of the 23rd international conference on World wide web, 2014.

[150] M. Zhang and N. Hurley, “Avoiding monotony: improving the diversity of recommendation lists,” Proceedings of the 2008 ACM conference on Recommender systems, 2008.

[151] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, “A kernel method for the two-sample-problem,” p. 513–520, Advances in neural information processing systems, 2007.

[152] P. Lahoti, A. Beutel, J. Chen, K. Lee, F. Prost, N. Thain, X. Wang, and E. H. Chi, “Fairness without demographics through adversarially reweighted learning,” Advances in neural information processing systems, 2020.