The Unfairness of Multifactorial Bias in Recommendation

MASOUD MANSOURY, Delft University of Technology, Delft, Netherlands
JIN HUANG, University of Amsterdam, Amsterdam, The Netherlands and University of Cambridge, Cambridge, United Kingdom
MYKOLA PECHENIZKIY, Eindhoven University of Technology, Eindhoven, Netherlands
HERKE VAN HOOF and MAARTEN DE RIJKE, University of Amsterdam, Amsterdam, Netherlands

Popularity bias and positivity bias are two prominent sources of bias in recommender systems. Both arise from input data, propagate through recommendation models, and lead to unfair or suboptimal outcomes. Popularity bias occurs when a small subset of items receives most interactions, while positivity bias stems from the over-representation of high rating values. Although each bias has been studied independently, their *combined effect*, to which we refer to as *multifactorial bias*, remains underexplored. In this work, we examine how multifactorial bias influences item-side fairness, focusing on exposure bias, which reflects the unequal visibility of items in recommendation outputs. Through simulation studies, we find that positivity bias is disproportionately concentrated on popular items, further amplifying their over-exposure. Motivated by this insight, we adapt a percentile-based rating transformation as a pre-processing strategy to mitigate multifactorial bias. Experiments using six recommendation algorithms across four public datasets show that this approach improves exposure fairness with negligible accuracy loss. We also demonstrate that integrating this pre-processing step into post-processing fairness pipelines enhances their effectiveness and efficiency, enabling comparable or better fairness with reduced computational cost. These findings highlight the importance of addressing multifactorial bias and demonstrate the practical value of simple, data-driven pre-processing methods for improving fairness in recommender systems.

CCS Concepts: • **Information systems** → **Recommender systems**; **Evaluation of retrieval results**;

Additional Key Words and Phrases: recommender systems, fairness, multifactorial bias, percentile-based rating transformation

ACM Reference format:
Masoud Mansoury, Jin Huang, Mykola Pechenizkiy, Herke van Hoof, and Maarten de Rijke. 2026. The Unfairness of Multifactorial Bias in Recommendation. *ACM Trans. Inf. Syst.* 44, 8, Article 172 (November 2026), 32 pages. https://doi.org/10.1145/3841477

This research was (partially) supported by the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.1389.20.183, and KICH3.LTP.20.006 and the European Union under grant agreements no. 101070212 (FINDHR) and no. 1012 01510 (UNITE).

Authors' Contact Information: Masoud Mansoury (corresponding author), Delft University of Technology, Delft, Netherlands; e-mail: m.mansoury@tudelft.nl; Jin Huang, University of Cambridge, Cambridge, United Kingdom; e-mail: jh2642@cam.ac.uk; Mykola Pechenizkiy, Eindhoven University of Technology, Eindhoven, Netherlands; e-mail: m.pechenizkiy@tue.nl; Herke van Hoof, University of Amsterdam, Amsterdam, Netherlands; e-mail: h.c.vanhoof@uva.nl; Maarten de Rijke, University of Amsterdam, Amsterdam, Netherlands; e-mail: m.derijke@uva.nl.



This work is licensed under Creative Commons Attribution International 4.0.

© 2026 Copyright held by the owner/author(s).
ACM 1558-2868/2026/11-ART172
https://doi.org/10.1145/3841477

1 Introduction

Recommender systems are known to suffer from different types of bias against users and items [16, 52, 57, 59, 68, 75]. Two prominent forms of such bias are *popularity bias* [14, 16, 30, 59, 75] and *positivity bias* [16, 59]. Popularity bias refers to the situation that the majority of rating data is concentrated on a small subset of highly popular items, leading to their over-representation while less popular items remain under-represented in the input data. Figure 1 shows this phenomenon across four datasets. As shown, most ratings are concentrated on a small fraction of items, with MovieLens exhibiting the strongest popularity bias. For instance, in the MovieLens dataset, approximately 20% of the items make up around 65% of the total ratings. The abundance of rating data on popular items skews the learning process of the recommender models, causing them to primarily learn the characteristics of these popular items [28, 68]. Consequently, these items are disproportionately recommended, reinforcing their dominance in recommendation results.

On the other hand, positivity bias refers to the situation that positive feedback (high rating values) is over-represented in the input data. This occurs because users are more likely to rate items they enjoy while ignoring those they dislike. Figure 2 shows the distribution of ratings across four datasets, highlighting that higher ratings are significantly more prevalent. This pattern is even stronger in the Google Local Data.

These types of bias originate from the input data, are exacerbated or transferred by recommender algorithms, and lead to *unfairness* in recommendation results [29, 30, 79]. Unfair recommendations not only negatively affect the recommendation performance [3, 45] but also lead to or perpetuate undesirable social dynamics [38, 70, 86]. One specific form of this type of unfairness in recommendation output is *exposure bias* [43, 70, 88], which refers to the situation that certain items frequently appear in recommendation results while certain other items are never or rarely recommended to users.

Prior research has discovered various consequences of this disparate representation of items in recommendation results: (i) it negatively affects user experience by limiting exposure to a small subset of items [45, 51, 72, 84] or failing to align with users' preferences [27, 28, 68], (ii) it leads to supplier unfairness, as items from certain suppliers would not be sufficiently (or equally) visible in the recommendation results [43, 54], and (iii) it raises ethical concerns in sensitive domains such as job recommendation or online dating, where the recommended items represent individuals belonging to different sensitive attribute classes, increasing the risk of discrimination [19, 47, 70]. In this article, we explore two key types of bias in input data, popularity bias and positivity bias, their connection to exposure bias in the recommendation results, and strategies to mitigate these types of bias to improve exposure fairness.

Although the unfairness of popularity bias and positivity bias has been *separately* studied in the literature [3, 4, 7, 34, 55, 60], their combined effect on recommendation fairness remains an open research question. This combined effect is referred to as *multifactorial bias*, which encapsulates the essential properties of both types of bias. Huang et al. [28] study the effect of this bias on the performance and accuracy of recommendation models, but its impact on the fairness of recommendation models has yet to be explored. To fill this gap, in this article, we conduct a comprehensive study on the unfairness caused by multifactorial bias in recommendation results and propose a solution to mitigate it.

First, we investigate multifactorial bias in several publicly available datasets. Our analysis reveals that positivity bias is stronger among popular items, signifying that highly rated feedback is disproportionately concentrated on popular items, further reinforcing their over-representation in recommendations. We then conduct a simulation study to examine how this type of bias affects recommendation performance and fairness. Our results show that mitigating positivity bias,

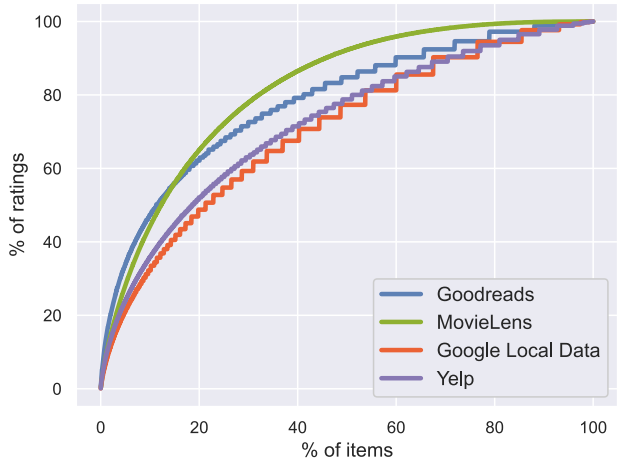


Fig. 1. Popularity distribution of the rating data in multiple datasets used in this article. The horizontal axis shows the percentage of the items, and the vertical axis shows the percentage of the accumulated ratings.



Fig. 2. Distribution of ratings of items in the MovieLens dataset. The horizontal axis shows the rating values in the dataset, and the vertical axis shows the percentage of times each rating value is seen in the dataset.

particularly among popular items (which remain popular in terms of number of interactions/ratings), can improve fairness of exposure for items in recommendation results.

Next, we adapt an existing rating transformation method [49] to address multifactorial bias in rating data, specifically by diminishing the impact of positivity bias on popular items. This method transforms the raw rating data into percentile values in item profiles, leading to less skew in percentile distribution in the input data. Through extensive experiments using six recommendation algorithms across four datasets, we validate the effectiveness of this proposed method in enhancing exposure fairness of recommendation results.

It is important to note that the primary objective of this work is not to introduce a new bias mitigation algorithm. Rather, our aim is to demonstrate how an existing method that balances rating distributions can be effectively adapted to address multifactorial bias, particularly the concentration of positivity bias among popular items, which is a property shared by both the proposed method and the bias itself. In doing so, we focus on an underexplored yet practically important challenge in recommender systems: the interaction between multiple sources of bias, specifically *popularity bias*

and *positivity bias*. While these biases have often been studied independently, their interplay can give rise to distinct fairness issues. Our approach is therefore motivated by the observation that positivity bias may disproportionately affect popular items and that addressing this interaction does not necessarily require a new algorithm, but rather a careful adaptation of existing techniques that are well aligned with the structure of the problem.

Finally, beyond improving fairness, we show that our proposed method enhances the efficiency of existing post-processing bias mitigation approaches. The effectiveness of these approaches depends heavily on the size of the initial recommendation lists generated by the recommendation model. Longer lists yield fairer recommendations, but at the expense of higher computational costs. We show that our pre-processing approach enables achieving fairer recommendation results with much smaller initial recommendation lists, resulting in a more efficient fair recommendation pipeline.

Overall, we make the following contributions:

- —We investigate multifactorial bias in publicly available recommendation datasets, revealing a strong correlation between positivity bias and popularity bias: positivity bias is more prevalent on popular items.
- —Through a simulation study, we demonstrate the relationship between multifactorial bias and exposure fairness in recommendation results, showing that this type of bias negatively affects item exposure fairness.
- —We adapt an existing rating transformation method to mitigate the effect of multifactorial bias in recommendation datasets. Our analysis shows that this proposed method reduces the effect of positivity bias, particularly on popular items, resulting in reduced multifactorial bias in input datasets and fairer item exposure in recommendation results.
- —Lastly, through additional sets of experiments, we show that the proposed pre-processing method improves the efficiency of existing post-processing bias mitigation methods by achieving comparable performance with a substantially shorter initial recommendation list as input for those post-processing bias mitigation methods.

2 Related Work

2.1 Bias and Unfairness

2.1.1 Types of Bias. Selection bias, which refers to bias present in the data associated with recommender systems, is pervasive in recommendation applications and manifests in various forms [16]. It can arise due to self-selection bias, with users choosing to interact with certain items more often [59, 75] or it may be due to algorithmic bias: the recommender system used for logging chooses to show certain items to users more often [9, 68]. Among different types of bias, popularity bias and positivity bias are the most widely recognized and studied, manifesting as the over-representation of interactions with popular items [14, 59, 75] or items that users prefer [59], respectively. Additionally, two forms of self-selection bias are frequently observed: incentive bias, where users are incentivized to provide ratings for benefits and rewards [58], and conformity bias, where users tend to rate items similarly to others in a group [31, 35]. Notably, Zhao et al. [91] show that conformity is also a key driver of popularity bias, as users tend to interact with already popular items regardless of their actual preferences, further inflating the popularity of these items. These types of bias are often characterized by a single underlying factor.

In practice, selection bias is typically more complex, often involving a combination of multiple biases or reflecting a multifactorial structure determined by more than one factor [82, 92]. Selection bias can also be influenced by the additional factor of time, resulting in dynamic popularity bias [26]. Such dynamic effects may also arise because popularity bias is progressively amplified through the model learning process. For instance, Lin et al. [40] provide a spectral analysis revealing that

popular items disproportionately dominate the top singular vectors of the rating matrix. Chen et al. [17] further show that graph convolution disproportionately amplifies the influence of high-degree items in the learning process. Moreover, contextual factors such as position, modality, or surrounding items in a recommendation list can simultaneously affect user interactions, referred to as surrounding item bias or contextual bias [66, 82, 94]. Additionally, selection correlated with both item and rating value factors has been observed across multiple real-world datasets [27, 28, 59]. Specifically, Huang et al. [28] formally define a multifactorial bias that is determined by two factors, viz. item and rating value, which can be seen as a generalization of popularity and positivity bias. In this article, we build on the findings of Huang et al. [28] and focus on multifactorial bias and its resulting unfairness problem.

2.1.2 Unfairness. Despite being two distinct concepts, bias and unfairness are closely interrelated. A widely accepted view is that various forms of bias and the learning process are the key reasons for unfairness issues in machine learning [3, 39, 79]. For instance, prior studies have shown that popularity bias can lead to unfair treatment of both long-tail items and users with little interest in popular items [3, 83]. Recently, Dai et al. [18] offer a unified perspective by framing both bias and unfairness as instances of distribution mismatch: bias arises from a mismatch with the objective target distribution in real scenarios, while unfairness reflects a mismatch with the subjective distribution aligned with human values.

The definition of fairness in recommender systems can vary depending on the multi-stakeholder settings, commonly encompassing user-oriented [3, 38, 39] or item-oriented fairness perspectives [1, 39, 71]. User fairness typically focuses on avoiding algorithmic discrimination against certain users that are defined by demographic attributes (e.g., gender and age) [76, 93]. In contrast, item fairness aims to ensure that items from different categories (e.g., long-tail items) receive equitable exposure in the recommendation results [44]. Each of these fairness perspectives can be considered at both the individual [76, 85] and group levels [70, 87, 88]. At the individual level, fairness is often defined as treating similar individuals similarly, whereas at the group level, fairness refers to ensuring that different user or item groups receive equitable treatment or outcomes, such as equal quality of recommendations or balanced exposure [15, 53]. For example, Steck [76] introduces calibration fairness, an individual-level user fairness notion, which requires that users receive recommendations that are close to what they are expecting to get based on the type of items they have rated. Rather than focusing on one specific fairness consideration, the concept of hybrid fairness acknowledges the diverse fairness demands that arise in the real world and aims to achieve more than one fairness requirement simultaneously [1, 43, 54]. Broadly, Zhao et al. [90] provide a comprehensive overview of the interplay between fairness and diversity in recommender systems, highlighting their connections at both user and item levels.

2.2 Fair Recommendation Methods

Existing methods for improving fairness in recommender systems can be broadly categorized based on their position in the recommendation pipeline: data-oriented, ranking, and reranking methods, which are also referred to as pre-processing, in-processing, and post-processing methods, respectively.

Data-oriented methods mitigate unfairness by modifying or augmenting the training data used to train the recommendation model. For example, Ekstrand et al. [20] apply a re-sampling strategy to adjust the proportion of different user groups in the training data, thereby addressing user fairness issues arising from imbalanced user group distributions. Rastegarpanah et al. [61] propose the use of antidote data (e.g., fake user data) added to the training data to counteract unfairness.

Ranking methods incorporate fairness considerations directly into the learning process of recommendation methods. These methods are typically categorized into regularization-based methods [10, 12, 40, 85, 88], adversarial learning-based methods [11, 81], reinforcement learning-based methods [22], representation learning-based methods [13], causal inference-based methods [91], and more recently, **Large Language Model (LLM)**-based methods. For example, Lu et al. [42] propose a dual debiasing framework that jointly mitigates multiple sources of bias in LLM-based recommendation, while Wang et al. [78] leverage LLMs to denoise sequential recommendation data, improving the robustness of learned user representations.

In contrast, reranking methods adjust the outputs of recommender systems to promote fairness without modifying or retraining the underlying recommendation methods. Zehlike et al. [87] propose FA*IR, a top-k ranking method that maximizes utility while satisfying ranked group fairness. Sonboli et al. [73] define a personalized fairness score based on multiple item attributes and achieve a tradeoff between fairness and utility. Several other works propose diversification reranking approaches to increase the representation of less popular items while maintaining acceptable recommendation accuracy [2, 8, 44]. Additionally, Chen et al. [17] adjust prediction scores at inference time to counteract the amplified influence of popular items introduced by graph convolution. Liu et al. [41] examine model-generated explanations using a causal graph to identify and mitigate sources of bias in the recommendation process. They simulate user feedback on these explanations to better understand and correct biased selection mechanisms. Similarly, inspired by causal inference principles, Zhang et al. [89] analyze both item- and feature-level popularity bias in interaction data and propose separate inverse propensity score-based methods to mitigate their effects. Gao et al. [21] provide a broad discussion of causal inference techniques applied to debiasing in recommender systems, covering propensity-based, intervention-based, and counterfactual methods.

3 Notation and Preliminaries

We consider a typical recommendation scenario, such as movie, music, and product recommendations, where users are allowed to express their opinions on various items through explicit ratings. These ratings are then used to train a recommendation model that generates a recommendation list for each user.

Formally, we denote $\mathcal{U} = \{u_1, \ldots, u_n\}$ as the set of n users, $\mathcal{I} = \{i_1, \ldots, i_m\}$ as the set of m items, and $R \in \mathbb{R}^{n \times m}$ as the user–item rating matrix where entry R_{ui} represents the rating provided by user $u \in \mathcal{U}$ on item $i \in \mathcal{I}$. The rating values are chosen from a predefined set of numerical values within the range $[a, b]$, such that $R_{ui} \in \{r_1, r_2, \ldots, r_O\}$, where O is the number of distinct rating levels. For example, in the MovieLens dataset, a 5-star rating system is used with $a = 1$, $b = 5$, and $R_{ui} \in \{1, 2, 3, 4, 5\}$.

We divide items into two groups based on their popularity: head and tail. Head items, denoted by $\mathcal{I}^H$, consist of popular items that make up roughly 20% of the ratings according to the Pareto Principle [65]. Tail items denoted by $\mathcal{I}^T$ consist of the rest of the less popular items (the ones not in $\mathcal{I}^H$).

Given the rating matrix R as input, the recommendation problem we consider is defined as building a recommendation model that effectively identifies K most relevant items for a target user. The recommendation list for a user u is denoted as L_u, which is a ranked list of K items sorted in descending order based on their predicted relevance score. The set of recommendation lists for all users is denoted as L.

3.1 Multifactorial Bias

According to [16, 28, 75], a typical recommender system with explicit user ratings often exhibits three key characteristics:

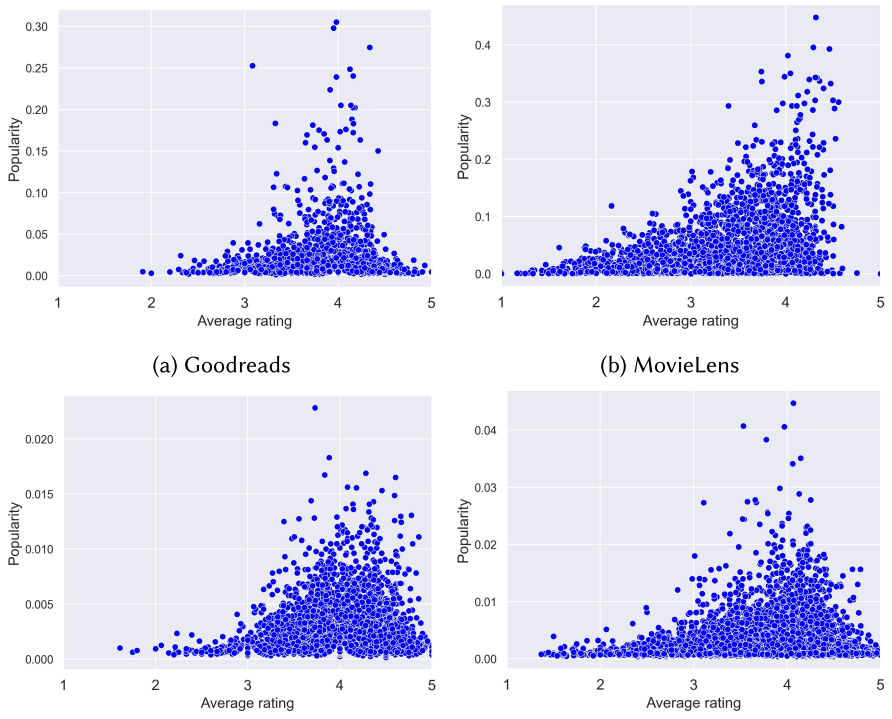


Fig. 3. The relationship between average ratings and popularity of items (fraction of users who interacted with the item) in four public datasets.

(**Characteristic 1**) Few popular items are frequently interacted with, while the majority of unpopular items are rarely interacted with.
(**Characteristic 2**) Users tend to rate items they like, resulting in a skewed distribution of observed ratings toward higher values.
(**Characteristic 3**) The majority of high rating values are assigned to the popular items.

Figure 3 illustrates the relationship between item popularity and average rating values across four public datasets, confirming the presence of Characteristic 3. The popularity of an item i is computed as the fraction of users who interacted with i, i.e., number of users who interacted with i divided by total number of users (n).

These characteristics are commonly recognized as forms of selection bias in recommender systems. Characteristic 1 and Characteristic 2 are often referred to as *popularity bias* and *positivity bias*, respectively, which have been studied extensively in the literature, with various methods proposed to mitigate them [14, 16, 59, 75]. Characteristic 3 has also been observed by Huang et al. [28]. Accordingly, Huang et al. [28] define a notion of *multifactorial bias* that occurs if the process that decides whether a user provides a rating is not a random selection and is determined by the item and rating value factors. This definition provides a general conceptual framework, whereas Characteristic 3 reflects a specific real-world phenomenon commonly observed in practice.

While Characteristic 3 more accurately reflects real-world user behavior, it remains underexplored in the literature. Huang et al. [28] are the first to examine the effect of multifactorial bias on the

Table 1. Characteristics of the Datasets after Pre-Processing

Dataset	#Users	#Items	#Ratings	Density	Ratings
Goodreads	2,225	3,423	137,045	1.8%	1–5
MovieLens	6,040	3,706	1,000,209	4.47%	1–5
Google Local Data	12,791	16,101	391,715	0.19%	1–5
Yelp	24,372	18,569	1,160,487	0.26%	1–5

performance and accuracy of recommendation models. In this article, we further investigate its impact on the fairness of recommendation models.

4 Experiments

In this section, we describe the datasets, recommendation algorithms, evaluation metrics, and the experimental setup that we use in the rest of the paper to perform our experiments.

4.1 Datasets

We use the following datasets for our experiments in this article.

— *Goodreads*:[1] A book recommendation dataset that contains user ratings of books.
— *MovieLens* [24]:[2] A movie recommendation dataset that contains user ratings of movies.
— *Google Local Data* [37]:[3] A subset of the Google Local Data, which consists of business reviews collected from Google Maps that contain user ratings of businesses.
— *Yelp*:[4] A restaurant recommendation dataset that contains user ratings of restaurants.

To alleviate the sparsity issue in the datasets and for more robust evaluation, we perform the following pre-processing on each dataset. From the Goodreads and Google Local Data datasets, we extract users who provided at least 10 ratings and items that received at least 10 ratings. From the Yelp dataset, we create a sample of the dataset in which each user has rated at least 20 items and each item has received at least 20 ratings. Table 1 summarizes the characteristics of the pre-processed datasets that we use in the article.

4.2 Recommendation Algorithms

Since we are studying the impact of positivity bias in recommender systems in this article, to be able to properly analyze the impact of our proposed approaches, we only focus on recommendation algorithms that use rating data for their operation. Algorithms operating on implicit or binary ratings data (e.g., Bayesian Personalized Ranking [62]) would not be affected by positivity bias as they do not use rating data for their internal processing. Therefore, we perform our experiments using the following recommendation algorithms.

— **Biased Matrix Factorization (`BiasedMF`)** [33]: This algorithm extends matrix factorization by incorporating the user bias and the item bias terms into the objective function to better capture the differences in users' behavior and items' characteristics.
— *Integrated Latent Factor and Neighborhood Models* (SVD++) [32]: This algorithm extends matrix factorization by incorporating neighborhood influence into the objective function.

[1] https://github.com/BahramJannesar/GoodreadsBookDataset.
[2] https://grouplens.org/datasets/movielens/1m/.
[3] https://jiachengli1995.github.io/google/index.html.
[4] https://www.yelp.com/dataset.

— **Weighted Matrix Factorization (WRMF)** [25]: A WRMF algorithm that incorporates a weight into the objective function as the confidence in users' interest toward items.
— *List-Wise Matrix Factorization* (ListRank) [69]: A specific form of matrix factorization with list-wise optimization process.
— *User-Based Neighborhood Model* (UserKNN) [63]: A nearest neighbor-based model that uses the user–user similarity matrix to recommend items based on the opinions of similar users.
— *Item-Based Neighborhood Model* (ItemKNN) [67]: A nearest neighbor-based model that uses the item–item similarity matrix to recommend items similar to the ones in a target user's profile.

4.3 Metrics

We evaluate the performance of the recommendation models on accuracy and fairness metrics. We use the following metrics to evaluate the accuracy of the recommendation models.

— *Precision*: The fraction of correctly recommended items (the items in the recommendation lists of users that also exist in the test data).
— **Normalized Discounted Cumulative Gain (nDCG):** A measure of ranking quality of the recommendations. nDCG weights the correctly recommended items based on their position in the recommendation list, recommending these items in higher position would lead to higher weight (higher nDCG).

Moreover, we use the following metrics to evaluate the fairness of the recommendation results.

— **Item Aggregate (IA)** *Diversity* [46]: The fraction of items that appear at least α times in the recommendation lists and can be calculated as:

$$IA(L) = \frac{\sum_{i \in \mathcal{I}} \mathbb{1}\left(\left[\sum_{u \in \mathcal{U}} \sum_{j \in L_u} \mathbb{1}(i = j)\right] \geq \alpha\right)}{m}. \tag{1}$$

This metric is a generalization of standard aggregate diversity as it is used in other work [6, 77] where $\alpha = 1$.
— **Long-Tail Item Aggregate (LIA)** *Diversity* [46]: The fraction of long-tail items that appear at least α times in the recommendation lists and can be calculated as:

$$LIA(L) = \frac{\sum_{i \in \mathcal{I}^T} \mathbb{1}\left(\left[\sum_{u \in \mathcal{U}} \sum_{j \in L_u} \mathbb{1}(i = j)\right] \geq \alpha\right)}{m^T}, \tag{2}$$

where m^T indicates the size of long-tail items in $\mathcal{I}^T$.
— **Equality of Exposure (EE)** [8]: A measure of fair distribution of recommended items. It takes into account how uniformly items appear in recommendation lists. Given $E(i|L)$ as the exposure of item i in the recommendation lists L calculated as:

$$E(i|L) = \frac{\sum_{u \in \mathcal{U}} \sum_{j \in L_u} \mathbb{1}(i = j)}{n \times K}. \tag{3}$$

We first form the distribution of items exposure as $\{E(i|L) \mid \forall i \in \mathcal{I}\}$. Then, we calculate the Gini Index over this distribution to measure its uniformity as follows:

$$Gini(L) = \frac{1}{m-1} \sum_{k=1}^{m} (2k - m - 1) E(i_k|L), \tag{4}$$

where $E(i_k|L)$ is the exposure of the kth least recommended item being drawn from L. A uniform distribution will have a Gini Index equal to 0, which is the ideal case (a lower Gini index is better). Therefore, to be consistent with the metrics definition, we define EE as

$EE = 1 - Gini(L)$. This way, an EE value closer to 1 signifies fairer recommendation results and is more desired (and EE closer to 0 is the least fair recommendation results).

— **User Popularity Lift (UPL)** [3, 5]: A measure of popularity bias amplification in recommendation lists relative to users' historical interactions. It quantifies the percentage change between the average popularity of items in a user's recommendation list and the average popularity of items in their interaction profile. Let $\mathcal{P}_u^R$ and $\mathcal{P}_u^L$ denote the mean popularity of items in the profile and recommendation list of user u, respectively. UPL is computed as follows:

$$UPL = \frac{\sum_{u \in \mathcal{U}} \frac{\mathcal{P}_u^L - \mathcal{P}_u^R}{\mathcal{P}_u^R}}{n}, \tag{5}$$

where $UPL > 0$ indicates that popular items are over-recommended, while $UPL < 0$ indicates that popular items are under-recommended. A desirable value is $UPL = 0$, indicating that the popularity of recommended items aligns with users' historical interactions. To maintain consistency with other metrics, we define **User Popularity Alignment (UPA)** as a normalized variant of UPL, computed as $UPA = \frac{\sum_{u \in \mathcal{U}} UPA_u}{n}$, where UPA_u is defined as:

$$UPA_u = \lceil \rho \rceil - |UPL_u|, \tag{6}$$

where ρ is the maximum absolute value of UPL across all users, i.e., $\rho = \max(|UPL_u| \mid u \in \mathcal{U})$, and $|x|$ denotes the absolute value of x. Higher UPA values indicate stronger alignment between the popularity of users' historical interactions and their recommendation lists, corresponding to UPL values closer to 0.

4.4 Experimental Setup

For our experiments, we split the input data into 80% as a training set and 20% as a test set. The split is performed on users' individual profiles: for each user, we randomly retrieve and use 80% of the ratings in her profile to form the training set, and we use the rest of the ratings to form the test set. The training set is used for building the recommendation model and generating the recommendation list for each user. The test set is used to evaluate the generated recommendations.

In the training phase, we perform extensive experiments with different hyperparameter values for each algorithm and dataset combinations to find the best-performing results. Hence, we perform a grid search over the hyperparameter space for each algorithm. For the neighborhood models (`UserKNN` and `ItemKNN`), we set the *shrinkage rate* as {50, 100, 200} and the number of neighbors as {10, 20, 30, 50, 70, 100, 200}. Also, we use *Pearson Correlation* as the similarity metric for computing the similarity values between users in `UserKNN` and *Cosine similarity* for computing the similarity values between items in `ItemKNN`. For model-based recommendation algorithms (i.e., `BiasedMF`, `SVD++`, `ListRankMF`, and `WRMF`), we set the *number of factors* to {30, 50, 100, 200}, the *number of iterations* to {30, 50, 100, 150, 200}, and the *learning rate* to {0.0001, 0.001, 0.01}. We report the best-performing hyperparameter values for all recommendation models across all datasets in Appendix A.

We use the best-performing recommendation model, to generate the recommendation list of size 10 for each user. Then, we compare the generated recommendation lists with the test data to measure the effectiveness of the recommendation model. We use *Librec-Auto* [48, 50] for performing our experiments and the resources are available at https://github.com/masoudmansoury/Unfairness_MultifactorialBias.

5 The Impact of Multifactorial Bias in Recommendation: A Simulation Study

In this section, we conduct a simulation study on the impact of multifactorial bias on recommendations. In this simulation, we artificially change the high rating values assigned to most popular items to low rating values in the input rating data (i.e., training data) as a pre-processing step and then compare the performance of recommendation models on the original and modified data. For example, on the MovieLens dataset, we modify the rating values in training data by changing rating 5 (i.e., the highest rating) given to popular items to 1 (i.e., the lowest rating) before feeding it into the recommendation model.

What makes this simulation interesting is that the modified data has the same characteristics as the original data (e.g., popularity bias) and only differs in ratings given to the popular items. This signifies that for the modified data, we synthetically mitigated the multifactorial bias. Therefore, the degree of multifactorial bias is the major difference between the modified data and the original data, which we use to show the impact of multifactorial bias on recommendations.

To reliably observe the impact of multifactorial bias, we repeat the simulation by modifying the ratings given to $\beta\%$ of most popular items with varying values of β, i.e., $\beta \in \{1\%, 3\%, 5\%, 7\%, 10\%, 15\%, 20\%, 25\%, 30\%, 35\%, 40\%, 45\%, 50\%\}$. Figure 4 shows the relationship between the average ratings and popularity of each item on four datasets after the aforementioned pre-processing for different β values (i.e., $\beta \in \{1\%, 3\%, 5\%, 7\%, 10\%\}$). As expected, with increasing β value, more high ratings given to the popular items are converted to low rating values. Note that this conversion does not change the shape of the popularity distribution in each of these datasets, and the only difference is in the positivity bias on popular items (i.e., multifactorial bias).

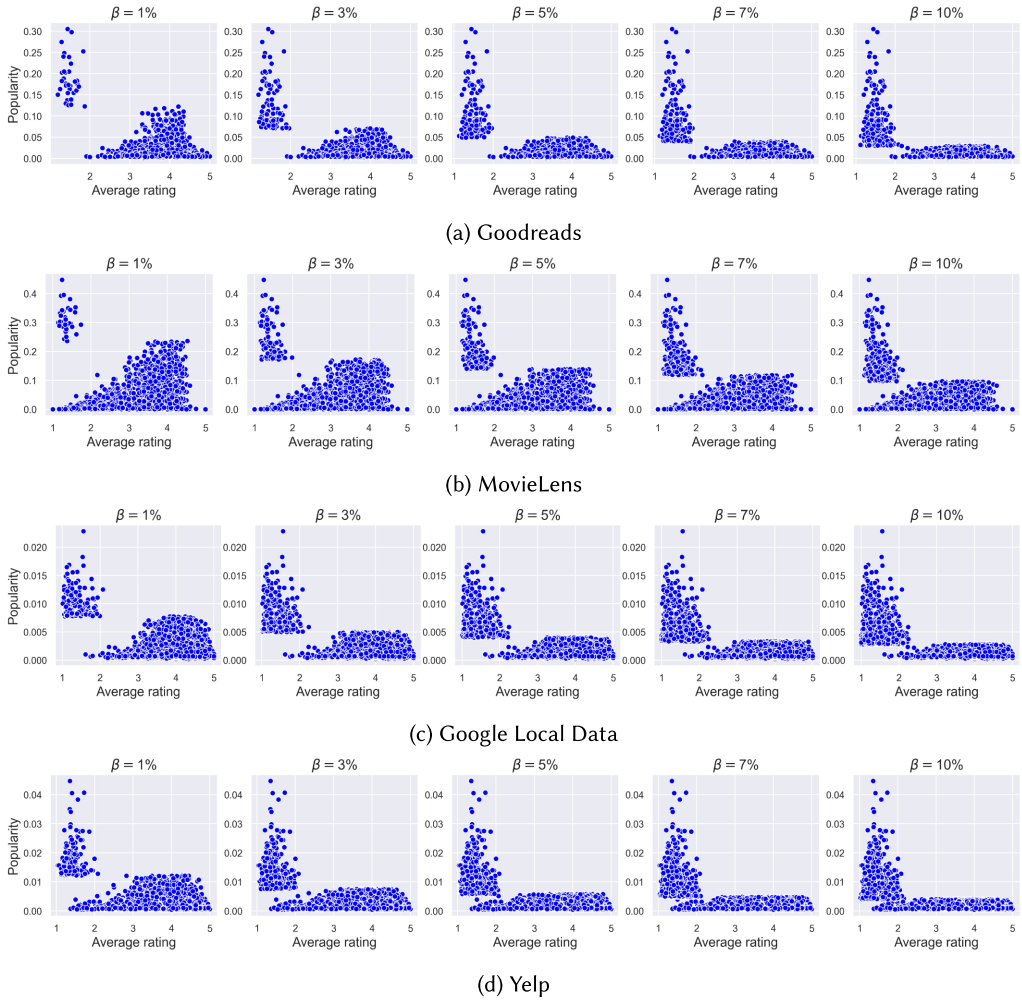
Figure 5 shows the results of our simulation in terms of both accuracy and bias metrics. In all plots, the x-axis indicates the β value, the percentage of the most popular items whose high ratings are changed to low ratings, and the y-axis shows the metric value, either accuracy (e.g., precision and nDCG) or fairness metrics (e.g., IA, LIA, and EE).

According to the plots in Figure 5, as β increases, both precision and nDCG decrease, possibly because less popular items are getting recommended by this change in the ratings of popular items. This loss in accuracy of the recommender system may not necessarily imply a negative consequence as the test set also suffers from popularity bias. On the other hand, from the plots showing IA diversity, LIA, and EE, it is evident that the fairness of recommendations is improved. These results suggest the effect of positivity bias on popular items, reducing the accuracy of the recommendations (due to evaluation with a biased test set) while improving the fairness of exposure for items in recommendation results.

6 Mitigating Unfairness of Multifactorial Bias in Recommendation

As shown in Section 5, while mitigating multifactorial bias helps to improve the exposure fairness of the items in recommendation results, it negatively affects the recommendation accuracy. This indicates that blindly mitigating multifactorial bias may not properly address the problem in real-world scenarios and that a principled solution is needed to address the tradeoff between exposure fairness and accuracy of recommendation results.

We base our bias mitigation approach on the rating transformation method proposed by Mansoury et al. [49]. The authors introduce a rating transformation method that converts the ratings in a user's profile into percentile values to reduce the positivity bias in users' rating behavior. They show that the converted rating data result in a more uniform distribution for ratings in users' profiles and consequently lead to improved ranking quality for the recommendation results. In this article, we also build on the same transformation method to reduce multifactorial bias in rating data. Unlike Mansoury et al. [49], who perform a percentile transformation on user profiles, we



(a) Goodreads

(b) MovieLens

(c) Google Local Data

(d) Yelp

Fig. 4. The relationship between the average ratings and popularity of items (fraction of users who interacted with the item) in four datasets after artificially mitigating for multifactorial bias.

perform a percentile transformation on the profiles of items. The motivation for this idea is the property of this percentile transformation method that produces a more uniform distribution for the transformed ratings (or percentile values) given to the items. In the following, we demonstrate how this property helps mitigating the multifactorial bias in recommender systems.

Given $R_{.,i}$ as the ratings given by different users on item i, the percentile value corresponding to each rating in $R_{.,i}$ is calculated as:

$$Per(r, R_{.,i}) = \frac{100 \times position(r, o(R_{.,i}))}{|R_{.,i}| + 1}, \quad \forall r \in R_{.,i}, \tag{7}$$

where $Per(r, R_{.,i})$ is the percentile value corresponding to rating r in $R_{.,i}$, $o(R_{.,i})$ is sorted $R_{.,i}$ in the ascending order, and $position(r, o(R_{.,i}))$ returns the index of r in $o(R_{.,i})$. One challenge in this transformation is that $R_{.,i}$ often contains repeated ratings, which makes specifying the position of a specific rating unclear. For example, in $R_{.,i} = \langle 1, 2, 2, 2, 2, 3, 3, 4, 5, 5 \rangle$, it is not clear what the

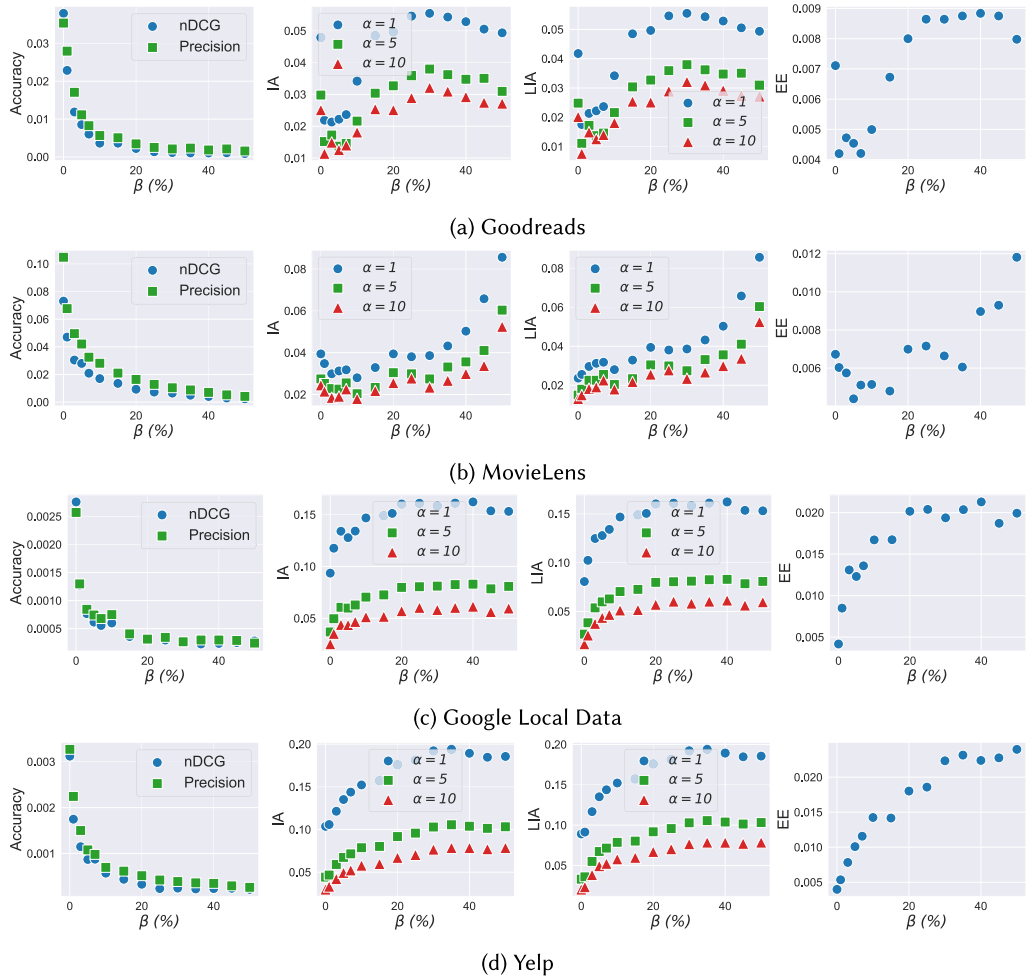
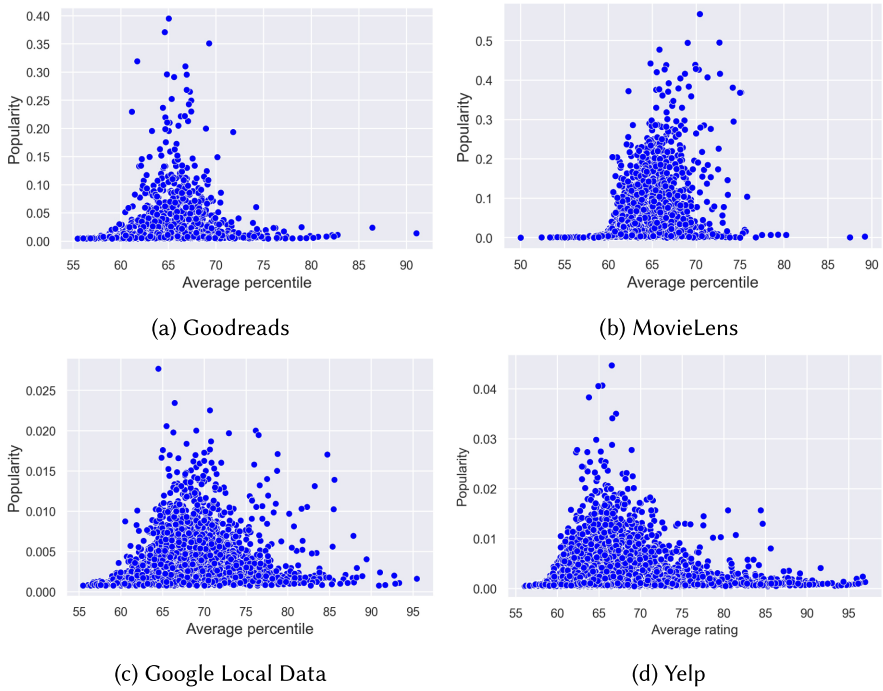


Fig. 5. The performance of the recommendation model on modified rating data in our simulation study on three datasets. Experiments are performed using the `BiasedMF` algorithm.

position of rating 2 should be. Possible solutions for this issue can be returning the first occurrence (e.g., position 2), last occurrence (e.g., position 5), or something in between. Consistent with the findings in [49], our experiments show that considering the last occurrence as the position of the repeated rating yields the best result. Therefore, for the remainder of the article, our percentile transformation is based on the last occurrence of the repeated ratings.

On each dataset used in this article (described in Section 4.1), we transform the rating data into percentile data using the proposed percentile transformation method. In other words, for each user u and item i in rating matrix R, we transform the rating R_{ui} into the percentile value using Equation (7). This will create a user–item percentile matrix. We denote this matrix as $P \in \mathbb{R}^{n \times m}$ with P_{ui} being the percentile value corresponding to R_{ui}.

To see to what degree the transformed percentile data suffer from multifactorial bias, we reproduce the plots in Figure 3 using the percentile matrix. Figure 6 illustrates the relationship between item popularity and average percentile values across four datasets. Unlike the rating data in Figure 3,



(a) Goodreads (b) MovieLens

(c) Google Local Data (d) Yelp

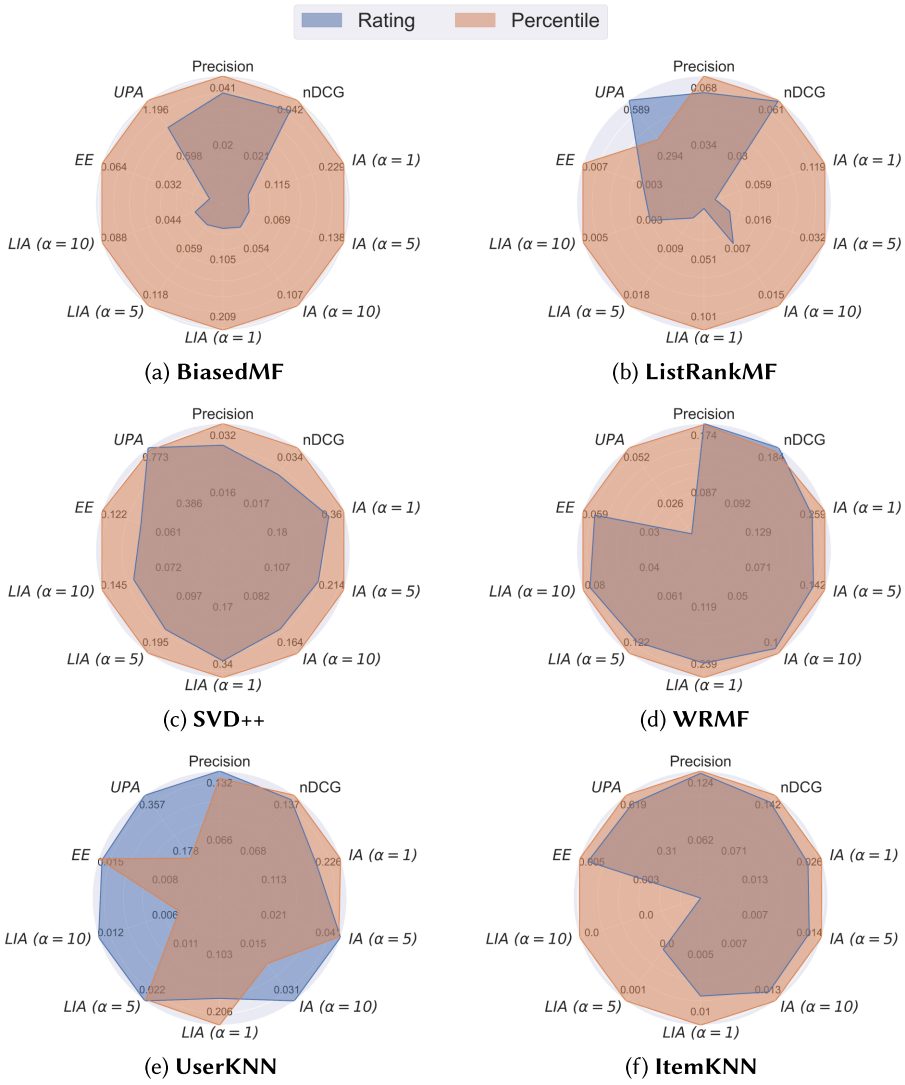
Fig. 6. The relationship between the average percentile values and popularity of each item (fraction of users who interacted with the item) in four public datasets.

the percentile representation does not exhibit multifactorial bias: both popular and unpopular items have percentile values concentrated within a similar range, without disproportionately high values assigned to popular items. This is because item-level percentile transformation decouples rating magnitude from item popularity by normalizing ratings within each item. As a result, the long-tailed skewness observed in raw ratings, where popular items accumulate inflated values due to the interaction between popularity and positivity bias, is reduced, leading to a more uniform distribution. Consequently, the influence of this interaction is weakened, and rating values become more evenly represented across items.

Our hypothesis is that using this percentile matrix as input for recommendation algorithms results in a less biased model that may consequently improve the fairness of recommendation results. To confirm this hypothesis, we separately perform experiments with raw rating data and transformed percentile data as input for the recommendation algorithms and evaluate their fairness. In the following section, we report our experimental results.

6.1 Experimental Results

To show the effectiveness of the percentile transformation in improving fairness of exposure for items and popularity alignment for users, we perform experiments using six recommendation models (described in Section 4.2) on four datasets (Goodreads, MovieLens, Google Local Data, and Yelp). The recommendation models are separately trained with two types of input data: (i) original rating data and (ii) percentile values. The results are evaluated in terms of both accuracy and fairness metrics (described in Section 4.3). Figures 7–10 present our experimental results.



(a) **BiasedMF** (b) **ListRankMF**

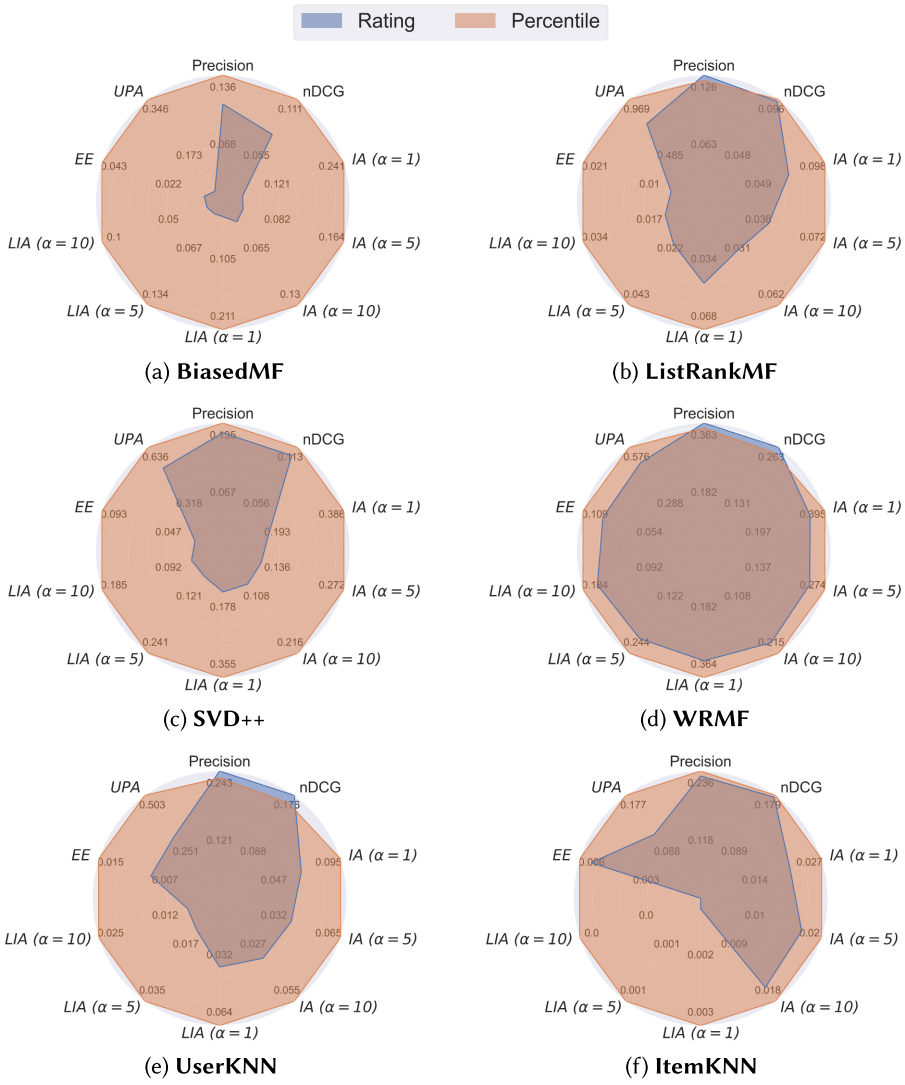
(c) **SVD++** (d) **WRMF**

(e) **UserKNN** (f) **ItemKNN**

Fig. 7. Performance of six recommender algorithms when raw rating data and percentile data are separately used as input on the *Goodreads* dataset. For all metrics, higher values signify a better performance.

First, from the plots in Figures 7–10, it is evident that our proposed percentile transformation significantly improves the fairness of exposure for items across all datasets, recommendation models, and evaluation metrics (except for `ListRankMF` on the Yelp dataset). In particular, in most cases, not only fairness is significantly improved but also accuracy metrics are boosted. For instance, on the Goodreads dataset, both accuracy and fairness are improved when percentile values are used as input for `BiasedMF`, `ListRankMF`, `SVD++`, and `ItemKNN`, rather than original rating values.

This fairness–accuracy tradeoff can also be observed in our results, such as for `WRMF` on the MovieLens dataset. However, the drop in accuracy is negligible in comparison to the gain obtained from the improvements in fairness. For example, for `WRMF` on the Google Local Data, with a 1.22% drop in nDCG, fairness is improved by 16.81%, 19%, and 18.16% for $IA(\alpha = 1)$, $LIA(\alpha = 1)$, and



(a) **BiasedMF** (b) **ListRankMF**

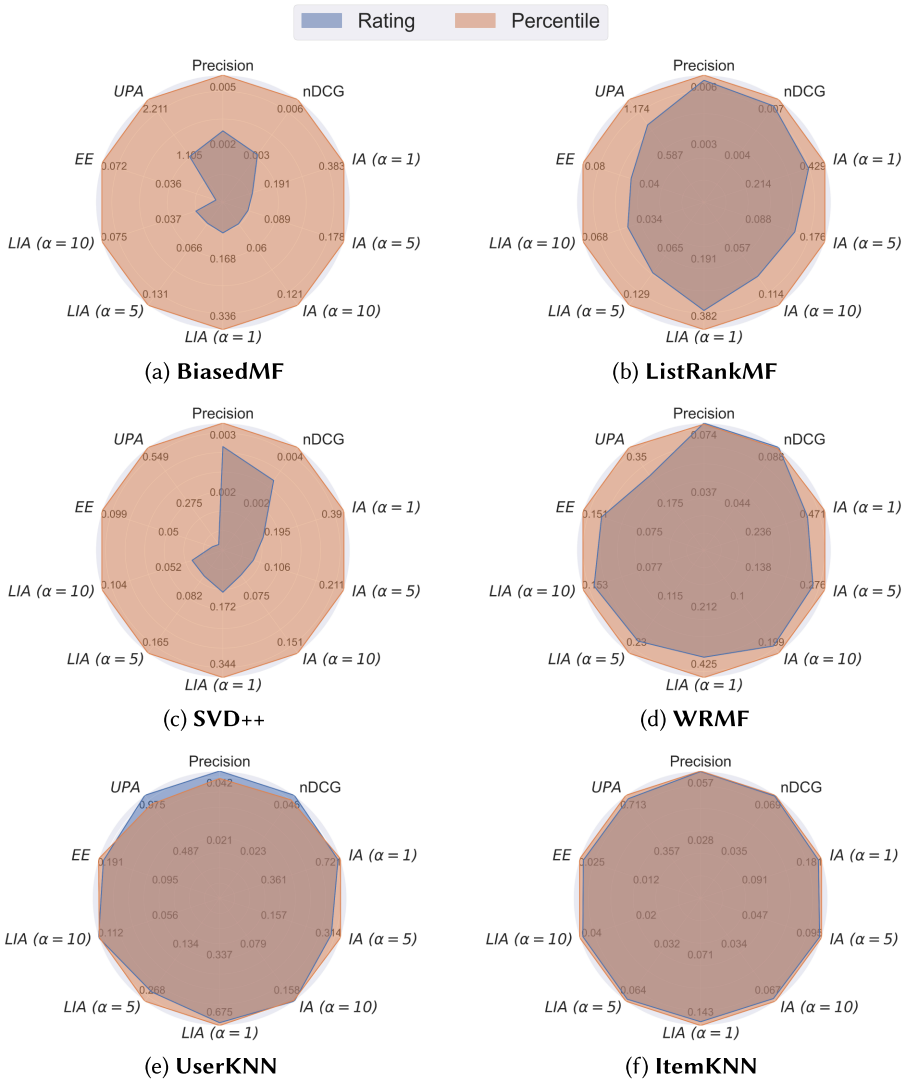
(c) **SVD++** (d) **WRMF**

(e) **UserKNN** (f) **ItemKNN**

Fig. 8. Performance of six recommendation algorithms when raw rating data and percentile data are separately used as input on the *MovieLens* dataset. For all metrics, higher values signify a better performance.

EE, respectively. Or for UserKNN on the MovieLens dataset, with a 7.61% drop in nDCG, fairness is improved by 48.09%, 85.04%, and 76.18% for $IA(\alpha = 1)$, $LIA(\alpha = 1)$, and *EE*, respectively. This confirms the effectiveness of the proposed solution in achieving exposure fairness for items.

For UserKNN on the Goodreads dataset, the results show that our percentile method is outperformed by the original model in terms of $IA(\alpha = 10)$ and $LIA(\alpha = 10)$. This is due to the fact that our percentile method recommends more unique items in recommendation lists (higher $IA(\alpha = 1)$ and $LIA(\alpha = 1)$), but there are not sufficiently many recommendation slots to show each at least 10 times (only 10 slots per user are available). This results in lower $IA(\alpha = 10)$ and $LIA(\alpha = 10)$ values, while higher $IA(\alpha = 1)$, $LIA(\alpha = 1)$, and *EE* values signify that more unique items are equally represented in the recommendation results, which follows the fairness criteria.



(a) **BiasedMF** (b) **ListRankMF**

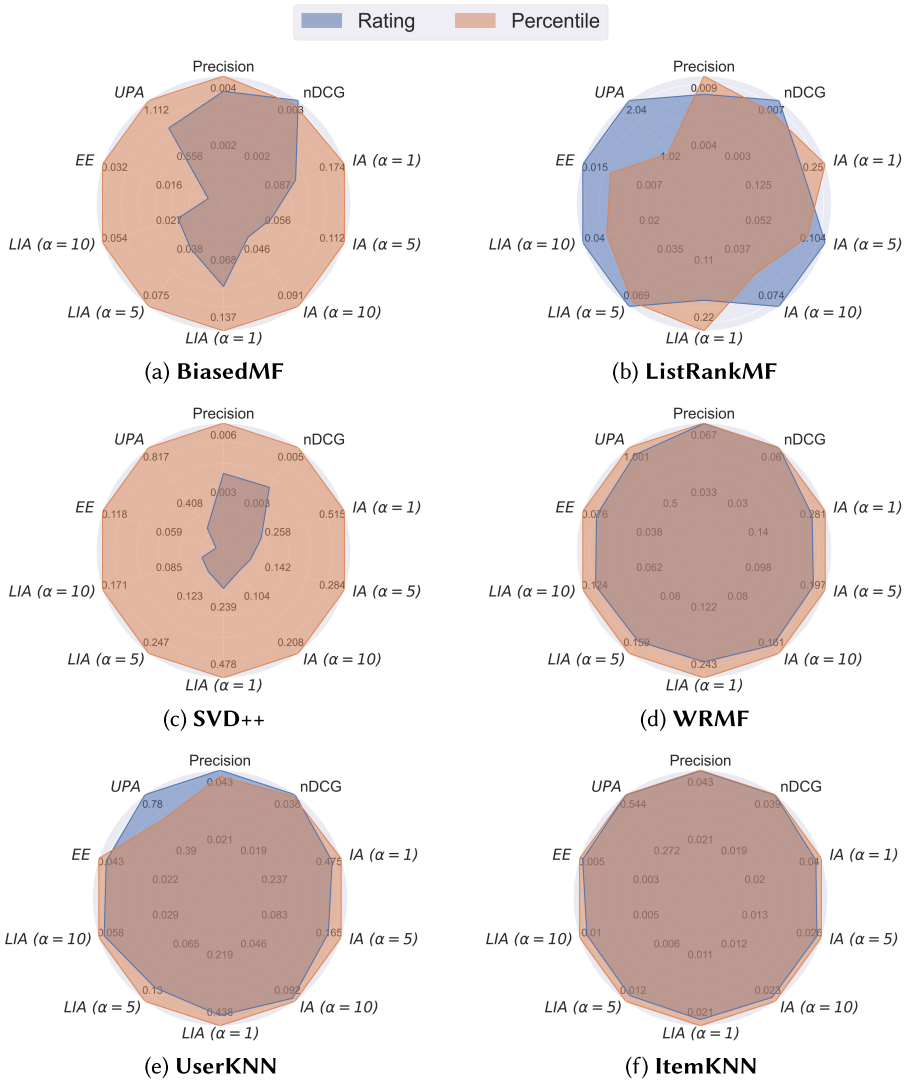
(c) **SVD++** (d) **WRMF**

(e) **UserKNN** (f) **ItemKNN**

Fig. 9. Performance of six recommendation algorithms when raw rating data and percentile data are separately used as input on the *Google Local Data* dataset. For all metrics, higher values signify a better performance.

Across all datasets, the percentile transformation exhibits a relatively smaller impact on neighborhood-based models (`UserKNN` and `ItemKNN`) as well as `WRMF`. For neighborhood models, the use of mean-centering and normalization steps partially compensates for biases in rating distributions, thereby reducing their sensitivity to distortions in the rating scale. As a result, the additional effect of percentile transformation is less pronounced, although it still yields incremental improvements.

For `WRMF`, the effect is even more limited, as the model is inherently less sensitive to absolute rating magnitudes. Instead, it relies on interaction confidence and latent structure, making it more robust to skewness in rating values. Consequently, modifying the rating scale through percentile transformation has a relatively minor impact on the learned representations.



(a) **BiasedMF** (b) **ListRankMF**

(c) **SVD++** (d) **WRMF**

(e) **UserKNN** (f) **ItemKNN**

Fig. 10. Performance of six recommendation algorithms when raw rating data and percentile data are separately used as input on the *Yelp* dataset. For all metrics, higher values signify a better performance.

These observations suggest that the effectiveness of percentile transformation depends on the extent to which a model relies on raw rating values. For models that already incorporate normalization mechanisms or are less dependent on rating magnitude, the benefits tend to be smaller. Nevertheless, the transformation does not degrade performance and can still provide consistent, albeit modest, improvements in fairness.

Additionally, on the user side, UPA measures the popularity alignment between users' historical interactions and their recommendation lists. A higher UPA indicates stronger alignment, meaning that the mean popularity of items in historical interactions is closer to that of items in recommendation lists across all users. The results in Figures 7–10 show that UPA improves in most cases when percentile values are used as input to recommendation models. For instance,

on MovieLens and Google Local Data, this pattern is particularly pronounced, as improvements are consistently observed across all recommendation models. It is worth noting that, unlike the mitigation method in [5], which directly targets user-side popularity alignment, the proposed percentile transformation does not explicitly aim to mitigate user-side bias. Instead, it operates on the item side. Therefore, the observed improvements in user-side fairness can be interpreted as an indirect effect of mitigating bias through the percentile transformation. This improvement in UPA is also evident in the Goodreads and Yelp datasets across all recommendation models, except for `ListRankMF` and `UserKNN`.

To further understand the effect of the percentile transformation method on user-side fairness, we examine the popularity lift across different user groups for recommendation models trained on rating values and percentile values, as shown in Figure 11. Users are grouped following the strategy introduced in [3, 5]: *Blockbuster* users are primarily interested in popular items, *Niche* users prefer less popular items, and *Diverse* users exhibit a balanced interest in both popular and less popular items. To quantify the difference in UPL between models trained on rating values and percentile values, we compute the percentage change as:

$$\delta^{UPL} = \frac{UPL_{Percentile} - UPL_{Rating}}{UPL_{Rating}}, \tag{8}$$

where a lower δ^{UPL} indicates a smaller popularity lift when using percentile values compared to rating values. This suggests that percentile values better align the popularity of recommended items with users' historical interactions. As shown in Figure 11, in most cases, recommendation models trained on percentile values yield lower UPL than those trained on rating values. In particular, niche users, who represent the protected group in this context, are consistently better served by percentile values in terms of aligning their interests with the popularity spectrum. This trend is observed across most datasets and recommendation models, with exceptions including `ListRankMF` and `UserKNN` on Goodreads and Yelp, and `SVD++` on MovieLens. A similar pattern is also observed for other user groups, further demonstrating the advantage of percentile values in aligning recommendations with users' popularity preferences.

6.2 Embedding Analysis

To further scrutinize the behavior of recommendation models when trained on rating values vs. percentile values, we analyze the user and item embeddings learned after training. Specifically, we examine the similarity of users and items to the popularity region in the embedding space. Let $Q^H \in \mathbb{R}^{m^H \times d}$ denote the embeddings of m^H popular items in a d-dimensional vector space. We define the centroid of the popularity region, $\bar{q}^H \in \mathbb{R}^d$, as the mean of the embeddings in Q^H. Subsequently, we compute the similarity between each user and item embedding and $\bar{q}^P$. Higher similarity to the popularity region indicates stronger alignment with popularity-driven signals.

Figure 12 presents the density plots of similarity scores between user and item embeddings and the popularity region for `BiasedMF`, trained separately on rating values and percentile values across four datasets.

In the original rating-based model, the embedding distributions are largely centered and overlapping, suggesting that popularity is not represented as a distinct geometric factor but is instead entangled within the latent space. In contrast, the model trained on percentile values exhibits a clear structural shift: popular items form a tighter cluster and align more strongly with a latent popularity direction, while user embeddings become more dispersed and shift away from this axis. This indicates that the percentile transformation does not remove popularity, but rather disentangles it from user preferences, allowing the model to separate popularity-driven signals from personalized representations. These patterns are particularly pronounced in the Goodreads

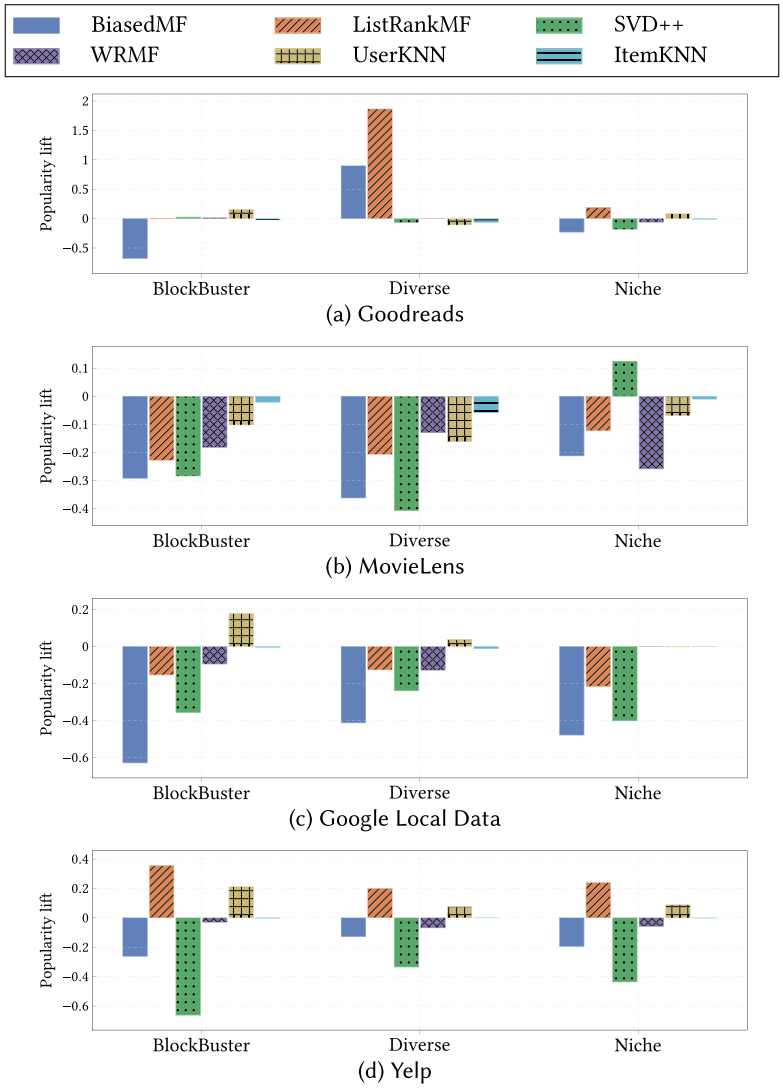
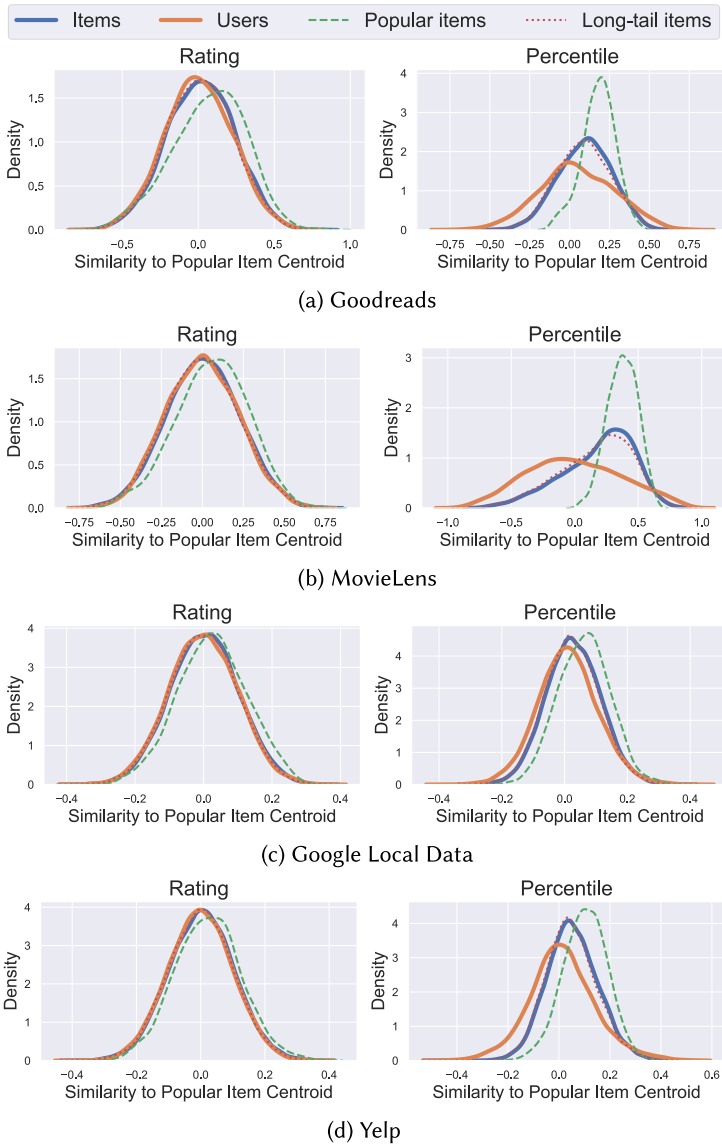


Fig. 11. User popularity deviation of recommendation models with both original and percentile input types across four public datasets, evaluated for different user groups: (i) *Blockbuster*, i.e., users primarily interested in popular items, (ii) *Niche*, i.e., users primarily interested in less popular items, and (iii) *Diverse*, i.e., users with mixed preferences.

and MovieLens datasets, where popular items show a stronger deviation from other items when percentile transformation is applied.

These results highlight an important property of the percentile transformation method in addressing multifactorial bias. As discussed earlier, the method does not aim to eliminate popularity bias itself, which is consistent with its design. Instead, it mitigates the effect of positivity bias on popular items, as also illustrated in Figure 6. The embedding analysis demonstrates that this objective is achieved by disentangling popularity-related signals in the embedding space, thereby enabling a clearer separation between popularity and user preference.



(a) Goodreads

(b) MovieLens

(c) Google Local Data

(d) Yelp

Fig. 12. Analysis of the learned embedding space for the `BiasedMF` model across four public datasets. The plots illustrate the similarity of user and item embeddings, including popular and tail items, to the centroid of the popularity region (i.e., the centroid of popular items) in the embedding space.

7 Comparing with Baselines and Improving Existing Unfairness Mitigation Pipelines

In this section, we further investigate how our proposed pre-processing method can enhance the performance of existing fairness mitigation strategies. As demonstrated in Section 6, our percentile-based pre-processing method significantly improves the fairness of recommendation models while preserving their accuracy. Next, we extend that analysis by examining the effectiveness of our pre-processing method when combined with existing fairness mitigation approaches, particularly within the broader recommendation pipeline.

As discussed in Section 2, existing fairness mitigation techniques are commonly categorized into three groups: pre-processing, in-processing, and post-processing. Among these, post-processing methods, also known as rerankers, have received considerable attention due to two key advantages: (i) they are model-agnostic and thus applicable across various recommendation models and (ii) they have demonstrated strong performance in fairness enhancement, owing to their flexibility in optimizing fairness criteria.

In a typical post-processing approach, an initial (often long) recommendation list is generated for each user by a base recommendation model. The reranker then refines this list to produce a shorter and fairer top-K recommendation output. However, it has been shown that the size of the initial recommendation list directly affects both fairness outcomes and computational efficiency. Longer initial lists generally lead to higher fairness but require greater computational resources and time. Shorter initial lists are more efficient but often result in reduced fairness [8, 44, 87].

Our central hypothesis is that if a recommendation model can produce sufficiently fair and accurate initial recommendation lists, even with a smaller size, then the reranker has a better foundation to work with, potentially achieving strong fairness outcomes at reduced computational cost. To demonstrate this, we evaluate the effectiveness of post-processing methods when applied to initial recommendation lists generated using models trained on percentile values (i.e., after applying our pre-processing method).

For this analysis, we focus exclusively on our percentile-based transformation, not alternative pre-processing methods, for two reasons. First, to the best of our knowledge, no existing pre-processing fairness method operates by modifying rating values to mitigate positivity bias. While several pre-processing fairness approaches exist [36, 56, 64, 74, 80], none are designed to adjust the rating distribution itself, which is essential for addressing multifactorial bias. Second, introducing a new pre-processing algorithm aimed at optimally improving fairness is not the primary objective of this work. Rather, our goal is to demonstrate the role that multifactorial bias [28] plays in shaping item-side exposure fairness and to show that even a simple, data-driven transformation can meaningfully reduce this bias in the input data. The percentile transformation serves as a clear and effective example of how mitigating multifactorial bias at the data level can lead to improved item exposure fairness in downstream recommendation outputs.

In our experiments, we use `BiasedMF` as the base recommendation model. We evaluate the following four reranking algorithms: **Discrepancy Minimization (DM)** [8], FA*IR [87], xQuAD [2], and FairMatch [44]. These rerankers are described in Section 2. Additionally, we include two naïve baselines for comparison: (i) Random, which randomly selects K items from the initial list and (ii) Reverse, which selects the bottom-K items based on their original relevance scores (i.e., sorted in the ascending order).

We test three initial recommendation list sizes: 20, 50,100, from which we generate a final top-10 recommendation list using the rerankers. We evaluate four experimental settings:

—Original ratings only: $rating \rightarrow$ BiasedMF
—Percentile values only: $percentile \rightarrow$ BiasedMF
—Reranking applied on original ratings: $rating \rightarrow$ BiasedMF $\rightarrow$ reranker
—Reranking applied on percentile values: $percentile \rightarrow$ BiasedMF $\rightarrow$ reranker

Figures 13 and 14 present results across four datasets using six rerankers, plotted in terms of nDCG vs. $IA(\alpha = 1)$ and nDCG vs. EE, respectively.[5] Several interesting patterns can be observed in these plots.

[5]Similar trends were observed across other metrics as well.

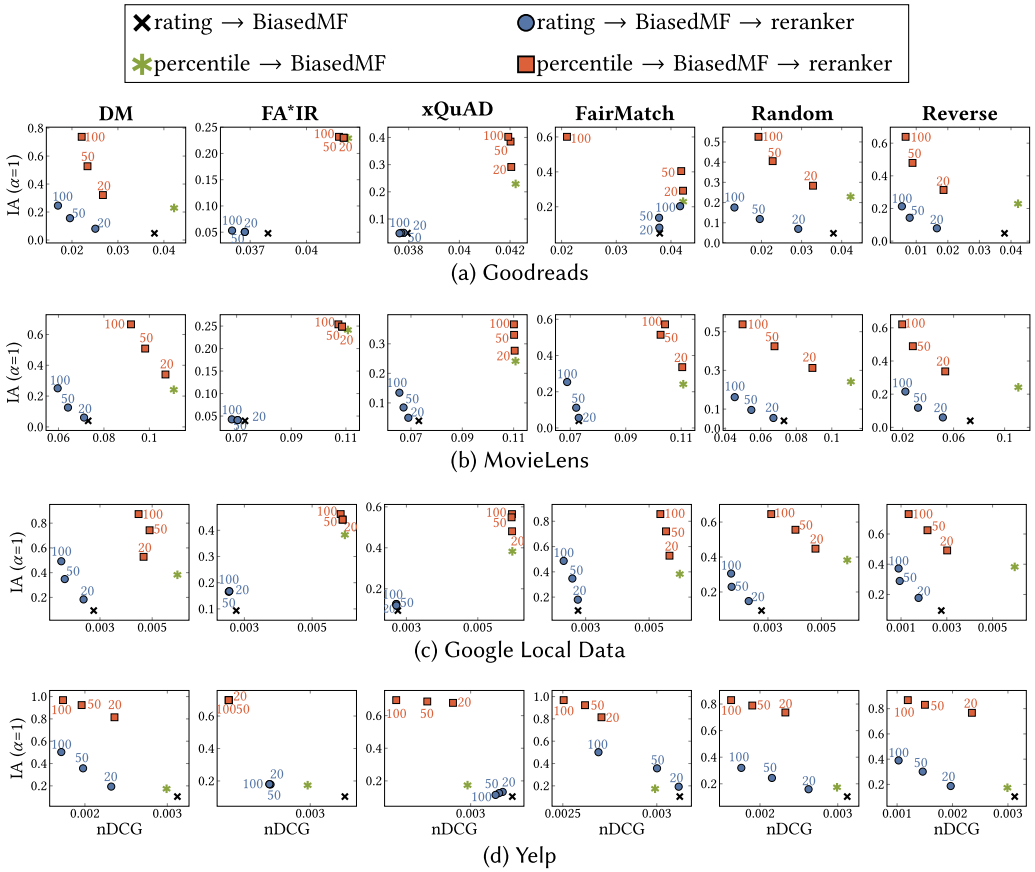


Fig. 13. Performance comparison of six rerankers with varying sizes of long recommendation lists $N \in \{20, 50, 100\}$ and two types of input data (original rating vs. percentile values) in terms of tradeoff between nDCG and $IA(\alpha = 1)$ for $K = 10$. For both metrics, a higher value is better. The legends present settings of the recommendation process: the black cross and green star show the results of `BiasedMF` with the original ratings and percentile values as input data, respectively, while the blue circle and red square show the reranker results on long recommendation lists of varying size generated by `BiasedMF` trained on the original ratings and percentile values, respectively.

First, we observe a clear tradeoff between the size of the initial recommendation list and the resulting performance. Larger initial lists (e.g., size 100) consistently lead to greater fairness, but at the cost of lower accuracy. Smaller initial lists (e.g., size 20) maintain higher accuracy but deliver less fairness. This tradeoff is well documented in prior work [23, 54]. Our results align with these findings.

Second, reranking methods require increased processing time as the initial list grows. Figure 15 shows the runtime of DM across different initial list sizes and datasets. For example, on the MovieLens dataset, DM takes approximately 3.7 times longer to process an initial list of size 100 compared to size 20, regardless of whether ratings or percentile values are used as input. Ideally, we aim to achieve high fairness and low accuracy loss without the need for large initial lists, which leads to the next observation.[6]

[6]The increased runtime of DM with percentile transformation is primarily due to the higher item diversity in the generated recommendation lists. As shown in Section 6, percentile transformation leads to substantially higher aggregate diversity,

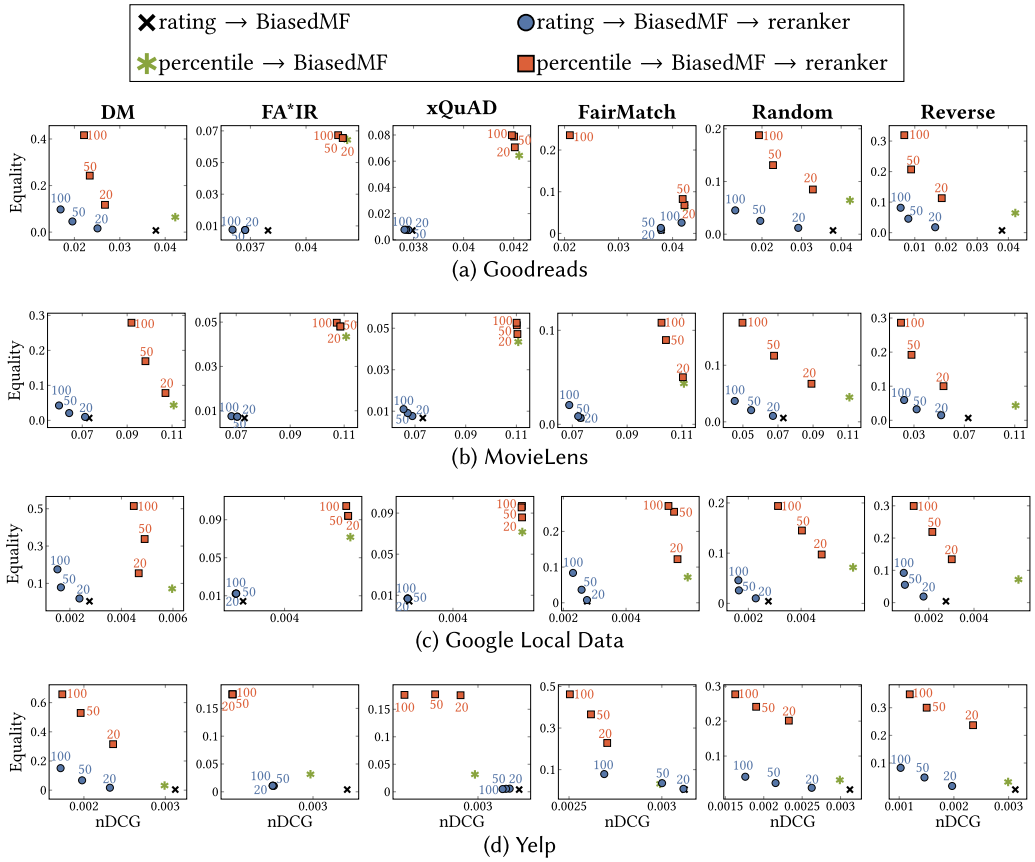
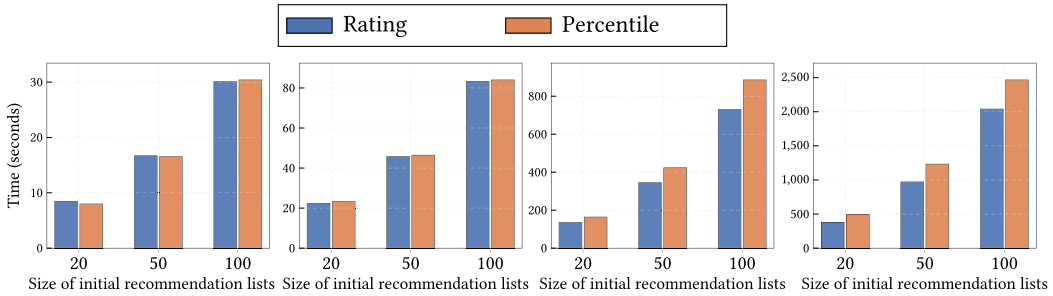


Fig. 14. Performance comparison of six rerankers with varying size of long recommendation lists $N \in \{20, 50, 100\}$ and two different input data (original rating vs. percentile values) in terms of tradeoff between nDCG and *Equality* for $K = 10$. For both metrics, a higher value is better. The legend presents settings of the recommendation process: black cross and green star show the results of `BiasedMF` with original ratings and percentile values as input data, respectively, while blue circle and red square show the reranker results on long recommendation lists of varying size generated by `BiasedMF` trained on original ratings and percentile values, respectively.

Third, our percentile transformation method clearly meets the ideal of achieving high fairness with minimal accuracy loss and reduced computational overhead. Across all datasets, rerankers, and evaluation metrics, using percentile-transformed input consistently results in: (i) higher fairness scores, (ii) comparable or improved accuracy, and (iii) notably strong performance even with smaller initial recommendation lists.

For example, as shown in Figure 13 for the MovieLens dataset, the FairMatch reranker applied to percentile-transformed input with an initial list size of 20 achieved an $IA(\alpha = 1)$ of 0.335 and an nDCG of 0.11, substantially outperforming the same reranker using an initial list of size 100 trained on the original ratings, which yielded $IA(\alpha = 1)$ of 0.253 and nDCG of 0.068.

meaning that more unique items appear in the recommendation lists. Since DM is a graph-based method that constructs a bipartite graph over users and recommended items, a larger and more diverse item set increases the graph size and, consequently, the computational cost, especially for larger list sizes (e.g., 100).



(a) Goodreads (b) MovieLens (c) Google Local Data (d) Yelp

Fig. 15. Running time of DM reranker with both original and percentile input types on four datasets.

Table 2. Performance Gain of DM Reranker with Initial Recommendation Lists of Size 20 Generated by Percentile Values Compared to Initial Recommendation List of Size 100 Generated by Rating Values on Four Datasets

Dataset	nDCG	IA ($\alpha = 1$)	EE	Running time
Goodreads	57.69%	30.64%	20.25%	73.49%
MovieLens	79.52%	35.79%	83.78%	71.97%
Google Local Data	207.84%	7.30%	−11.91%	77.57%
Yelp	37.65%	61.98%	109.11%	75.87%

Similarly, in Figure 14, the same reranker with percentile-transformed input achieved an *EE* score of 0.05 on an initial list of size 20, compared to just 0.006 when using original rating input with a larger list size of 100.

To further understand the performance boost provided by our method, we compare the utility of DM reranker when applied to: (i) a shorter list (size 20) generated using percentile values, and (ii) a longer list (size 100) generated using original rating values. We compute the relative gain as:

$$gain^{\mathcal{M}} = \frac{Utility^{\mathcal{M},20}_{Percentile} - Utility^{\mathcal{M},100}_{Rating}}{Utility^{\mathcal{M},100}_{Rating}} \times 100, \tag{9}$$

where $Utility^{\mathcal{M},20}_{Percentile}$ denotes the performance score (e.g., nDCG, IA, *EE*) of the final recommendation list produced by DM using percentile input and a 20-item initial list. $Utility^{\mathcal{M},100}_{Rating}$ represents the same score using original ratings and a 100-item initial list. This analysis allows us to assess whether our method can match or exceed the performance of traditional pipelines while using far fewer resources.

Table 2 presents the gain values for each dataset and metric. The results strongly favor our method across nearly all cases. For instance, we observe a 207.84% nDCG gain on the Google Local Data and a substantial *EE* gain on the Yelp dataset (109.11%). An exception is found only in the *EE* metric on the Google Local Data, where the performance is comparable. These results demonstrate that using percentile values as input enables recommendation models to generate fairer initial lists, which in turn allows rerankers to achieve better final fairness and accuracy outcomes, even with shorter input lists, leading to a more effective and efficient fairness-aware recommendation pipeline.

8 Conclusion

In this article, we have investigated the combined effect of popularity bias and positivity bias, to which we refer as multifactorial bias, on the fairness of recommender systems. Our analysis revealed that positivity bias disproportionately affects popular items, intensifying their over-exposure and leading to unfair treatment of less popular items in recommendation results. We demonstrated through simulation and empirical studies that multifactorial bias plays a significant role in amplifying exposure bias, a critical form of item-side unfairness.

To mitigate this issue, we adapted a percentile-based rating transformation as a pre-processing method aimed at reducing multifactorial bias in the input data. Our extensive experiments across multiple datasets and recommendation algorithms showed that this approach improves exposure fairness without compromising accuracy. Moreover, we showed that incorporating this method into existing post-processing fairness pipelines can enhance both their effectiveness and efficiency, enabling fair outcomes with shorter initial recommendation lists and lower computational cost.

Overall, these findings highlight the need to explicitly address multifactorial bias in recommender systems and demonstrate that simple, data-driven pre-processing techniques can play a powerful role in building fairer and more efficient recommendation pipelines.

Despite these contributions, the percentile-based transformation method has two key limitations. First, its reliability depends on the availability of a sufficient number of rating values; for example, applying a percentile transformation to a rating vector such as $\langle 3, 4 \rangle$ would be unstable. Second, the transformation assumes diversity in rating values; a uniform vector such as $\langle 3, 3, \ldots, 3 \rangle$ would yield uninformative percentile scores. In this work, we focused on dense datasets to avoid these issues, but a natural direction for future research is to examine how percentile-based transformations behave under sparse data conditions. Given that data sparsity constitutes a fundamental challenge in recommender systems, future research should examine the robustness and generalizability of our approach in sparse data environments and cold-start scenarios.

Additionally, future work will also explore whether combining existing mitigation strategies for popularity and positivity biases can effectively address multifactorial bias when applied together. We further aim to extend the application of the proposed percentile transformation to implicit feedback data (e.g., clicks, views, or purchases), investigating its effectiveness across a broader class of recommendation models.

Acknowledgments

Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of their respective employers, funders and/or granting authorities.

Resources

To facilitate the reproducibility of experiments in this article, we make the code and datasets available at: https://github.com/masoudmansoury/Unfairness_MultifactorialBias.

References

[1] Himan Abdollahpouri and Robin Burke. 2019. Multi-stakeholder recommendation and its connection to multi-sided fairness. arXiv:1907.13158. Retrieved from https://arxiv.org/abs/1907.13158

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing popularity bias in recommender systems with personalized re-ranking. arXiv:1901.07555. Retrieved from https://arxiv.org/abs/1901.07555

[3] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2019. The unfairness of popularity bias in recommendation. In *Proceedings of the RecSys Workshop on Recommendation in Multistakeholder Environments (RMSE)*.

[4] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2020. The connection between popularity bias, calibration, and fairness in recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*, 726–731.

[5] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, Bamshad Mobasher, and Edward Malthouse. 2021. User-centered evaluation of popularity bias in recommender systems. In *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization*, 119–129.

[6] Gediminas Adomavicius and YoungOk Kwon. 2011. Maximizing aggregate recommendation diversity: A graph-theoretic approach. In *Proceedings of the 1st International Workshop on Novelty and Diversity in Recommender Systems (DiveRS '11)*. Citeseer, 3–10.

[7] Thomas Allard, Lea H. Dunn, and Katherine White. 2020. Negative reviews, positive impact: Consumer empathetic responding to unfair word of mouth. *Journal of Marketing* 84, 4 (2020), 86–108.

[8] Arda Antikacioglu and R. Ravi. 2017. Post processing recommender systems for diversity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 707–716.

[9] Ricardo Baeza-Yates. 2018. Bias on the web. *Communications of the ACM* 61, 6 (2018), 54–61.

[10] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed. H. Chi, et al. 2019. Fairness in recommendation ranking through pairwise comparisons. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2212–2220.

[11] Avishek Bose and William Hamilton. 2019. Compositional fairness constraints for graph embeddings. In *Proceedings of the International Conference on Machine Learning*. PMLR, 715–724.

[12] Robin Burke, Nasim Sonboli, and Aldo Ordonez-Gauger. 2018. Balanced neighborhoods for multi-sided fairness in recommendation. In *Proceedings of the Conference on Fairness, Accountability and Transparency*. PMLR, 202–214.

[13] Miaomiao Cai, Lei Chen, Yifan Wang, Haoyue Bai, Peijie Sun, Le Wu, Min Zhang, and Meng Wang. 2024. Popularity-aware alignment and contrast for mitigating popularity bias. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 187–198.

[14] Rocío Cañamares and Pablo Castells. 2018. Should I follow the crowd? A probabilistic analysis of the effectiveness of popularity in recommender systems. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, 415–424.

[15] Alessandro Castelnovo, Riccardo Crupi, Greta Greco, Daniele Regoli, Ilaria Giuseppina Penco, and Andrea Claudio Cosentini. 2022. A clarification of the nuances in the fairness metrics landscape. *Scientific Reports* 12, 1 (2022), 4209.

[16] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems* 41, 3 (2023), 1–39.

[17] Jiajia Chen, Jiancan Wu, Jiawei Chen, Xin Xin, Yong Li, and Xiangnan He. 2024. How graph convolutions amplify popularity bias for recommendation? *Frontiers of Computer Science* 18, 5 (2024), 185603.

[18] Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Bias and unfairness in information retrieval systems: New challenges in the LLM era. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 6437–6447.

[19] Bora Edizel, Francesco Bonchi, Sara Hajian, André Panisson, and Tamir Tassa. 2020. FaiRecSys: Mitigating algorithmic bias in recommender systems. *International Journal of Data Science and Analytics* 9, 2 (2020), 197–213.

[20] Michael D. Ekstrand, Mucun Tian, Ion Madrazo Azpiazu, Jennifer D. Ekstrand, Oghenemaro Anuyah, David McNeill, and Maria Soledad Pera. 2018. All the cool kids, how do they fit in? Popularity and demographic biases in recommender evaluation and effectiveness. In *Conference on Fairness, Accountability and Transparency*. PMLR, 172–186.

[21] Chen Gao, Yu Zheng, Wenjie Wang, Fuli Feng, Xiangnan He, and Yong Li. 2024. Causal inference in recommender systems: A survey and future directions. *ACM Transactions on Information Systems* 42, 4 (2024), 1–32.

[22] Yingqiang Ge, Shuchang Liu, Ruoyuan Gao, Yikun Xian, Yunqi Li, Xiangyu Zhao, Changhua Pei, Fei Sun, Junfeng Ge, Wenwu Ou, et al. 2021. Towards long-term fairness in recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 445–453.

[23] Yingqiang Ge, Xiaoting Zhao, Lucia Yu, Saurabh Paul, Diane Hu, Chu-Cheng Hsieh, and Yongfeng Zhang. 2022. Toward pareto efficient fairness-utility trade-off in recommendation through reinforcement learning. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, 316–324.

[24] F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 1–19.

[25] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 8th IEEE International Conference on Data Mining*. IEEE, 263–272.

[26] Jin Huang, Harrie Oosterhuis, and Maarten de Rijke. 2022. It is different when items are older: Debiasing recommendations when selection bias and user preferences are dynamic. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, 381–389.

[27] Jin Huang, Harrie Oosterhuis, Maarten de Rijke, and Herke Van Hoof. 2020. Keeping dataset biases out of the simulation: A debiased simulator for reinforcement learning based recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, 190–199.

[28] Jin Huang, Harrie Oosterhuis, Masoud Mansoury, Herke Van Hoof, and Maarten de Rijke. 2024. Going beyond popularity and positivity bias: Correcting for multifactorial bias in recommender systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 416–426.

[29] Di Jin, Luzhi Wang, He Zhang, Yizhen Zheng, Weiping Ding, Feng Xia, and Shirui Pan. 2023. A survey on fairness-aware recommender systems. *Information Fusion* 100 (2023), 101906.

[30] Anastasiia Klimashevskaia, Dietmar Jannach, Mehdi Elahi, and Christoph Trattner. 2024. A survey on popularity bias in recommender systems. *User Modeling and User-Adapted Interaction* 34 (2024), 1777–1834.

[31] Norman Knyazev and Harrie Oosterhuis. 2022. The bandwagon effect: Not just another bias. In *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval*, 243–253.

[32] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 426–434.

[33] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[34] Dominik Kowald, Markus Schedl, and Elisabeth Lex. 2020. The unfairness of popularity bias in music recommendation: A reproducibility study. In *Proceedings of the European Conference on Information Retrieval*. Springer, 35–42.

[35] Sanjay Krishnan, Jay Patel, Michael J. Franklin, and Ken Goldberg. 2014. A methodology for learning, analyzing, and mitigating social influence bias in recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, 137–144.

[36] Preethi Lahoti, Krishna P. Gummadi, and Gerhard Weikum. 2019. iFair: Learning individually fair data representations for algorithmic decision making. In *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1334–1345.

[37] Jiacheng Li, Jingbo Shang, and Julian McAuley. 2022. UCTopic: Unsupervised contrastive learning for phrase representations and topic mining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6159–6169.

[38] Yunqi Li, Hanxiong Chen, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2021. User-oriented fairness in recommendation. In *Proceedings of the Web Conference 2021*, 624–632.

[39] Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, Juntao Tan, Shuchang Liu, and Yongfeng Zhang. 2023. Fairness in recommendation: Foundations, methods, and applications. *ACM Transactions on Intelligent Systems and Technology* 14, 5 (2023), 1–48.

[40] Siyi Lin, Chongming Gao, Jiawei Chen, Sheng Zhou, Binbin Hu, Yan Feng, Chun Chen, and Can Wang. 2025. How do recommendation models amplify popularity bias? An analysis from the spectral perspective. In *Proceedings of the 18th ACM International Conference on Web Search and Data Mining*, 659–668.

[41] Xu Liu, Tong Yu, Kaige Xie, Junda Wu, and Shuai Li. 2024. Interact with the explanations: Causal debiased explainable recommendation system. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 472–481.

[42] Sijin Lu, Zhibo Man, Fangyuan Luo, and Jun Wu. 2025. Dual debiasing in LLM-based recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2685–2689.

[43] Masoud Mansoury. 2022. Understanding and mitigating multi-sided exposure bias in recommender systems. *ACM SIGWEB Newsletter* 2022 (2022), 1–4.

[44] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. 2020. FairMatch: A graph-based approach for improving aggregate diversity in recommender systems. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, 154–162.

[45] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. 2020. Feedback loop and bias amplification in recommender systems. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, 2145–2148.

[46] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. 2021. A graph-based approach for mitigating multi-sided exposure bias in recommender systems. *ACM Transactions on Information Systems* 40, 2 (2021), 1–31.

[47] Masoud Mansoury, Himan Abdollahpouri, Jessie Smith, Arman Dehpanah, Mykola Pechenizkiy, and Bamshad Mobasher. 2020. Investigating potential factors associated with gender discrimination in collaborative recommender systems. In *Proceedings of the 33rd International FLAIRS Conference*, 193–196.

[48] Masoud Mansoury and Robin Burke. 2019. Algorithm selection with Librec-Auto. In *Interdisciplinary Workshop on Algorithm Selection and Meta-Learning in Information Retrieval, Held in Conjunction with the European Conference on Information Retrieval (ECIR)*, 11–17.

[49] Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2021. Flatter is better: Percentile transformations for recommender systems. *ACM Transactions on Intelligent Systems and Technology* 12, 2 (2021), 1–16.

[50] Masoud Mansoury, Robin Burke, Aldo Ordonez-Gauger, and Xavier Sepulveda. 2018. Automating recommender systems experimentation with Librec-Auto. In *Proceedings of the 12th ACM Conference on Recommender Systems*, 500–501.

[51] Masoud Mansoury, Bamshad Mobasher, and Herke van Hoof. 2024. Mitigating exposure bias in online learning to rank recommendation: A novel reward model for cascading bandits. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 1638–1648.

[52] Benjamin M. Marlin and Richard S. Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, 5–12.

[53] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Computing Surveys* 54, 6 (2021), 1–35.

[54] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. 2018. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2243–2251.

[55] Mohammadmehdi Naghiaei, Hossein A. Rahmani, and Mahdi Dehghan. 2022. The unfairness of popularity bias in book recommendation. In *Proceedings of the International Workshop on Algorithmic Bias in Search and Recommendation*. Springer, 69–81.

[56] Brodie Oldfield, Ziqi Xu, and Sevvandi Kandanaarachchi. 2025. Revisiting pre-processing group fairness: A modular benchmarking framework. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, 6498–6502.

[57] Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, Kathryn Vasilaky, and Elena Zheleva. 2020. Correcting for selection bias in learning-to-rank systems. In *Proceedings of the Web Conference 2020*, 1863–1873.

[58] Umberto Panniello, Shawndra Hill, and Michele Gorgoglione. 2016. The impact of profit incentives on the relevance of online recommendations. *Electronic Commerce Research and Applications* 20 (2016), 87–104.

[59] Bruno Pradel, Nicolas Usunier, and Patrick Gallinari. 2012. Ranking with non-random missing ratings: Influence of popularity and positivity on evaluation metrics. In *Proceedings of the 6th ACM Conference on Recommender Systems*, 147–154.

[60] Hossein A. Rahmani, Yashar Deldjoo, Ali Tourani, and Mohammadmehdi Naghiaei. 2022. The unfairness of active users and popularity bias in point-of-interest recommendation. In *Proceedings of the International Workshop on Algorithmic Bias in Search and Recommendation*. Springer, 56–68.

[61] Bashir Rastegarpanah, Krishna P. Gummadi, and Mark Crovella. 2019. Fighting fire with fire: Using antidote data to improve polarization and fairness of recommender systems. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, 231–239.

[62] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 452–461.

[63] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, 175–186.

[64] Clara Rus, Andrew Yates, and Maarten de Rijke. 2024. A study of pre-processing fairness intervention methods for ranking people. In *Proceedings of the European Conference on Information Retrieval*. Springer, 336–350.

[65] Robert Sanders. 1987. The Pareto principle: Its use and abuse. *Journal of Services Marketing* 1, 2 (1987), 37–40.

[66] Fatemeh Sarvi, Ali Vardasbi, Mohammad Aliannejadi, Sebastian Schelter, and Maarten de Rijke. 2023. On the impact of outlier bias on user clicks. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*. ACM, 18–27.

[67] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, 285–295.

[68] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1670–1679.

[69] Yue Shi, Martha Larson, and Alan Hanjalic. 2010. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*, 269–272.

[70] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2219–2228.

[71] Ashudeep Singh and Thorsten Joachims. 2019. Policy learning for fairness in ranking. In *Advances in Neural Information Processing Systems*, Vol. 32.

[72] Ayan Sinha, David F. Gleich, and Karthik Ramani. 2016. Deconvolving feedback loops in recommender systems. In *Advances in Neural Information Processing Systems*, Vol. 29.

[73] Nasim Sonboli, Farzad Eskandanian, Robin Burke, Weiwen Liu, and Bamshad Mobasher. 2020. Opportunistic multi-aspect fairness through personalized re-ranking. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, 239–247.

[74] Ryosuke Sonoda. 2021. A pre-processing method for fairness in ranking. arXiv:2110.15503. Retrieved from https://arxiv.org/abs/2110.15503

[75] Harald Steck. 2011. Item popularity and recommendation accuracy. In *Proceedings of the 5th ACM Conference on Recommender Systems*, 125–132.

[76] Harald Steck. 2018. Calibrated recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, 154–162.

[77] Saúl Vargas and Pablo Castells. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the 5th ACM Conference on Recommender Systems*, 109–116.

[78] Bohao Wang, Feng Liu, Changwang Zhang, Jiawei Chen, Yudi Wu, Sheng Zhou, Xingyu Lou, Jun Wang, Yan Feng, Chun Chen, et al. 2025. LLM4DSR: Leveraging large language model for denoising sequential recommendation. *ACM Transactions on Information Systems* 44, 1 (2025), 1–32.

[79] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. 2023. A survey on the fairness of recommender systems. *ACM Transactions on Information Systems* 41, 3 (2023), 1–43.

[80] Madiha Waris, Muhammad Zaman Fakhar, Mert Gulsoy, Emre Yalcin, and Alper Bilge. 2024. A novel pre-processing technique to combat popularity bias in personality-aware recommender systems. *IEEE Access* 12 (2024), 183230–183251.

[81] Le Wu, Lei Chen, Pengyang Shao, Richang Hong, Xiting Wang, and Meng Wang. 2021. Learning fair representations for recommendation: A graph-based perspective. In *Proceedings of the Web Conference 2021*, 2198–2208.

[82] Xinwei Wu, Hechang Chen, Jiashu Zhao, Li He, Dawei Yin, and Yi Chang. 2021. Unbiased learning to rank in feeds recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 490–498.

[83] Emre Yalcin and Alper Bilge. 2022. Evaluating unfairness of popularity bias in recommender systems: A comprehensive user-centric analysis. *Information Processing and Management* 59, 6 (2022), 103100.

[84] Mengyue Yang, Jun Wang, and Jean-Francois Ton. 2023. Rectifying unfairness in recommendation feedback loop. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 28–37.

[85] Sirui Yao and Bert Huang. 2017. Beyond parity: Fairness objectives for collaborative filtering. In *Advances in Neural Information Processing Systems*, Vol. 30.

[86] Hyunsik Yoo, Zhichen Zeng, Jian Kang, Ruizhong Qiu, David Zhou, Zhining Liu, Fei Wang, Charlie Xu, Eunice Chan, and Hanghang Tong. 2024. Ensuring user-side fairness in dynamic recommender systems. In *Proceedings of the ACM on Web Conference 2024*, 3667–3678.

[87] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. 2017. FA*IR: A fair top-K ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1569–1578.

[88] Meike Zehlike and Carlos Castillo. 2020. Reducing disparate exposure in ranking: A learning to rank approach. In *Proceedings of the Web Conference 2020*, 2849–2855.

[89] Jingsen Zhang, Xu Chen, Jiakai Tang, Weiqi Shao, Quanyu Dai, Zhenhua Dong, and Rui Zhang. 2023. Recommendation with causality enhanced natural language explanations. In *Proceedings of the ACM Web Conference 2023*, 876–886.

[90] Yuying Zhao, Yu Wang, Yunchao Liu, Xueqi Cheng, Charu C. Aggarwal, and Tyler Derr. 2025. Fairness and diversity in recommender systems: A survey. *ACM Transactions on Intelligent Systems and Technology* 16, 1 (2025), 1–28.

[91] Zihao Zhao, Jiawei Chen, Sheng Zhou, Xiangnan He, Xuezhi Cao, Fuzheng Zhang, and Wei Wu. 2022. Popularity bias is not always evil: Disentangling benign and harmful bias for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 35, 10 (2022), 9920–9931.

[92] Zhi Zheng, Zhaopeng Qiu, Tong Xu, Xian Wu, Xiangyu Zhao, Enhong Chen, and Hui Xiong. 2022. CBR: Context bias aware recommendation for debiasing user modeling and click prediction. In *Proceedings of the ACM Web Conference 2022*, 2268–2276.

[93] Ziwei Zhu, Xia Hu, and James Caverlee. 2018. Fairness-aware tensor-based recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1153–1162.

[94] Honglei Zhuang, Zhen Qin, Xuanhui Wang, Michael Bendersky, Xinyu Qian, Po Hu, and Dan Chary Chen. 2021. Cross-positional attention for debiasing clicks. In *Proceedings of the Web Conference 2021*, 788–797.

Appendix

A.1 Best-Performing Hyperparameter Settings

As noted in Section 4.4, we perform a grid-search over the hyperparameter space for finding the best-performing model. We use nDCG of the recommendation model as a performance criteria for finding the best-performing model. Tables A1–A6 present the hyperparameter settings for all recommendation models across all datasets.

Table A1. Best-Performing Hyperparameter Values for `BiasedMF`

Dataset	Input	#Iterations	#Factors	Learning rate
Goodreads	Rating	50	20	0.0001
	Percentile	100	200	0.0001
MovieLens	Rating	30	100	0.0001
	Percentile	30	100	0.0001
Google Local Data	Rating	100	100	0.0001
	Percentile	200	200	0.001
Yelp	Rating	30	100	0.0001
	Percentile	100	200	0.0001

Table A2. Best-Performing Hyperparameter Values for `ListRankMF`

Dataset	Input	#Iterations	#Factors	Learning rate
Goodreads	Rating	30	200	0.01
	Percentile	30	30	0.001
MovieLens	Rating	30	100	0.001
	Percentile	30	100	0.001
Google Local Data	Rating	200	200	0.0001
	Percentile	200	200	0.0001
Yelp	Rating	150	30	0.001
	Percentile	30	50	0.01

Table A3. Best-Performing Hyperparameter Values for `SVD++`

Dataset	Input	#Iterations	#Factors	Learning rate
Goodreads	Rating	30	100	0.01
	Percentile	10	300	0.001
MovieLens	Rating	50	30	0.001
	Percentile	30	100	0.0001
Google Local Data	Rating	30	100	0.001
	Percentile	30	100	0.001
Yelp	Rating	50	100	0.0001
	Percentile	30	500	0.001

Table A4. Best-Performing Hyperparameter Values for WRMF

Dataset	Input	#Iterations	#Factors	Weight coef.
Goodreads	Rating	50	50	1
	Percentile	50	50	1
MovieLens	Rating	50	50	1
	Percentile	50	50	1
Google Local Data	Rating	100	200	5
	Percentile	100	200	10
Yelp	Rating	30	200	2
	Percentile	30	200	2

Table A5. Best-Performing Hyperparameter Values for UserKNN

Dataset	Input	#Neighbors (k)	Shrinkage
Goodreads	Rating	30	100
	Percentile	100	200
MovieLens	Rating	200	200
	Percentile	50	200
Google Local Data	Rating	30	200
	Percentile	30	200
Yelp	Rating	30	200
	Percentile	50	200

Table A6. Best-Performing Hyperparameter Values for ItemKNN

Dataset	Input	#Neighbors (k)	Shrinkage
Goodreads	Rating	10	50
	Percentile	10	50
MovieLens	Rating	200	100
	Percentile	200	100
Google Local Data	Rating	200	50
	Percentile	200	50
Yelp	Rating	200	50
	Percentile	200	50

Received 2 December 2025; revised 21 April 2026; accepted 31 July 2026