Fairness among New Items in Cold Start Recommender Systems

Ziwei Zhu[*]
Texas A&M University
zhuziwei@tamu.edu

Jingu Kim
Netflix
jinguk@netflix.com

Trung Nguyen
Netflix
trungn@netflix.com

Aish Fenton
Netflix
afenton@netflix.com

James Caverlee
Texas A&M University
caverlee@tamu.edu

ABSTRACT

This paper investigates recommendation fairness among new items. While previous efforts have studied fairness in recommender systems and shown success in improving fairness, they mainly focus on scenarios where unfairness arises due to biased prior user-feedback history (like clicks or views). Yet, it is unknown whether new items without any feedback history can be recommended fairly, and if unfairness does exist, how can we provide fair recommendations among these new items in such a cold-start scenario. In detail, we first formalize fairness among new items with the well-known concepts of *equal opportunity* and *Rawlsian Max-Min fairness*. We empirically show the prevalence of unfairness in cold start recommender systems. Then we propose a novel *learnable post-processing framework* as a model blueprint for enhancing fairness, with which we propose two concrete models: a *joint-learning generative model*, and a *score scaling model*. Extensive experiments over four public datasets show the effectiveness of the proposed models for enhancing fairness while also preserving recommendation utility.

CCS CONCEPTS

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

KEYWORDS

fairness; cold start recommendation

ACM Reference Format:
Ziwei Zhu, Jingu Kim, Trung Nguyen, Aish Fenton, and James Caverlee. 2021. Fairness among New Items in Cold Start Recommender Systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462948

1 INTRODUCTION

Recommender systems are important for connecting users to the right items. But are items recommended fairly? For example, in a recruiting recommender that recommends job candidates (the items here), are candidates of different genders treated equally? In a news recommender, are news stories with different political ideologies recommended fairly? And even for product recommenders, are products from big companies favored over products from new entrants? The danger of unfair recommendations for items has been recognized in the literature [5, 7, 10, 41, 43], with potential negative impacts on item providers, user satisfaction, the recommendation platform itself, and ultimately social good.

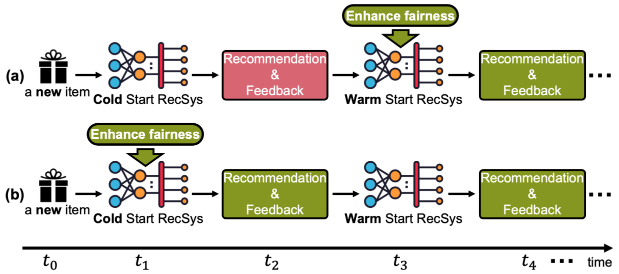


Figure 1: (a) Existing works consider fairness during warm start recommendation period; (b) we study fairness among new items, i.e., the cold start recommendation period.

Previous works have revealed that widely used recommendation algorithms can indeed produce unfair recommendations toward different items (like job candidates of different genders), e.g., [5, 10, 14, 21, 43]. However, these existing works consider fairness only during the middle of the life cycle of an item, that is in the *warm start* recommender scenario. In this scenario, prior works show that the main driver of unfairness is the data bias in historical feedback (like clicks or views), and recommendation algorithms unaware of this bias can inherit and amplify this bias to produce unfair recommendations. But what if there is no historical feedback? Can we fairly recommend new items (we use 'new items' and 'cold start items' interchangeably) in such a *cold start* scenario?

To illustrate, Figure 1a shows the life cycle of an item: the item first appears in the system at t_0; in the absence of historical feedback, a cold start recommendation algorithm recommends this item to users and receives the first collection of feedback at t_2; then, a warm start recommender can be trained at t_3 by this first collection of feedback, further recommending the item to users and collecting new feedback at t_4; the system continues this loop of collecting new feedback and training a new warm start model until the item leaves the system. While methods in existing works can introduce fairness at time t_3 and later, is the item treated fairly before then? In this work, we show that the data bias can be transferred from warm start items to new items through item content features by machine

[*] Part of this work performed while interning at Netflix.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462948

learning based cold start recommendation algorithms, inducing unfair recommendations among these new items.

This *fairness gap* can be especially problematic since unfairness introduced by cold start recommenders will be perpetuated and accumulated through the entire life cycle of an item, resulting in growing difficulty for mitigating unfairness as the life cycle goes on. Instead, providing fair recommendations among new items could give rise to a virtuous circle of collecting (relatively) unbiased feedback and training fairer models later in the life cycle. Hence, as shown in Figure 1b, we propose to investigate fairness in cold start recommender systems and aim to enhance fairness among new items at the beginning of their life cycles.

Fairness goal. In this work, we follow two well-known concepts – equal opportunity [12] and Rawlsian Max-Min fairness principle of distributive justice [26] – to introduce the *Max-Min Opportunity Fairness* in the context of cold start scenarios. The fairness goal is to provide recommendations that maximize the true positive rate (that is, the probability of being accurately recommended to matched users who will like the item during testing) of the worst-off items so that no item is under-served by the recommendation model. The advantages of this fairness goal are twofold: i) by following equal opportunity to measure fairness by the true positive rate, the fairness is directly aligned with the feedback or economic gain items receive as well as user satisfaction; and ii) by following Rawlsian Max-Min fairness to accept inequalities, the fairness does not require decreasing utility for the better-served items and thus can better preserve the overall utility. Thus, by improving the Max-Min Opportunity Fairness, we aim to improve item fairness without decreasing overall satisfaction.

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

i) We introduce the problem of fairness among new items in cold start scenarios, and we conduct a comprehensive data-driven study to demonstrate the prevalence of unfairness among new items in cold start recommender systems.

ii) To mitigate unfairness among new items, we propose a novel learnable post-processing framework as a solution blueprint, which promises better practical feasibility than existing strategies. Based on this blueprint, we demonstrate two concrete approaches for enhancing item fairness: a score scaling model based on existing work for addressing popularity bias [34] and a novel joint-learning generative model that can effectively improve fairness.

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

2 RELATED WORK

2.1 Item Fairness In Recommender Systems

Unfair recommendations for items can bring harmful impacts to item providers, users, the platform owner, and society [5, 7, 10, 41, 43]. Hence, growing efforts have been dedicated to the study of item fairness in recommender systems. Early studies mainly focus on rating prediction tasks and investigate item fairness by measuring the difference of predicted rating distributions across item groups [14–17, 41]. To improve this score/rating based concept of item fairness, regularization based [14–17] and latent factor manipulation based [41] methods have been proposed. Later, with the prevalence of ranking based recommender systems, new formulations [5, 10, 21, 25, 43] to directly study item fairness on ranking results instead of on the intermediate predicted scores/ratings have been proposed. Another line of research [6, 30, 31] studies fairness in terms of equality of exposure, which investigates whether the amortized exposure of items in recommendations are proportional to the amortized relevance of items. Further, inspired by fairness-aware classification [12], equal opportunity based fairness that requires an equal true positive rate across item groups has been proposed [5, 10, 25, 43]. To improve such ranking based equal opportunity fairness, many new algorithms utilizing regularization [5, 25], re-ranking [10], and adversarial learning [43] have also been proposed.

All of these previous works study item fairness in a warm start setting. Yet, it is equally crucial to ensure new items are fairly recommended at the first moment when they join a system. Thus, we propose to investigate fairness among new items in cold start recommender systems.

2.2 Cold Start Recommender Systems

In almost all real-world applications, recommending new items without any historical feedback from users (formally called the cold start recommendation task), is heavily in demand and challenging. Over the years, many approaches have been proposed including heuristic non-parametric algorithms like KNN [29], though now there is an emphasis on optimization based machine learning algorithms. These methods can be categorized into two types [42]: separate-training methods and joint-training methods.

Separate-training methods [4, 9, 27, 32, 35] separately learn two models: a collaborative filtering (CF) model learns CF embeddings of warm start items; and a content model learns how to transform the item content features to the learned collaborative embeddings using warm start items. During inference, the content model is first applied on content features of new items to generate CF embeddings and then recommendations for these new items are provided by these embeddings. A typical example is DeepMusic [35], which utilizes a multi-layer perceptron (MLP) to transform item content features to CF embeddings learned by a pretrained matrix factorization model. Joint-training methods [20, 28, 37] combine the CF model and content model together and train both of them through a single backpropagation process. A typical example is DropoutNet [37], which has an MLP based content component to first transform item content features, followed by a dot-product based neural CF component to provide recommendations. Moreover, Heater in [42] mixes separate-training and joint-training methods, which delivers the state-of-the-art performance. Yet, there is no notion of fairness in these cold start recommenders. Hence, we aim to investigate the fairness of these representative models and propose novel methods to enhance fairness in the cold start scenario. Specifically, we investigate four typical cold start recommenders: Heater, DropoutNet, DeepMusic, and KNN.

3 FAIRNESS AMONG NEW ITEMS

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

3.1 Cold Start Recommendation

We focus on the cold start item recommendation task [37], where all users are warm start during training and testing, but new items never seen during training are to be recommended during testing. Assume we have N users $\mathcal{U} = \{1, 2, \ldots, N\}$ and M_w warm start items $\mathcal{I}_w = \{1, 2, \ldots, M_w\}$, where each item has at least one historical interaction record in the training data. We denote $O_{tr} = \{(u, i)\}$ as the training set, where $u \in \mathcal{U}$ indexes one user, and $i \in \mathcal{I}_w$ indexes one warm start item. We also have M_c cold start (new) items $\mathcal{I}_c = \{1, 2, \ldots, M_c\}$, none of which are included in the training set O_{tr}. We denote $O_{te} = \{(u, i)\}$ as the test set, where $u \in \mathcal{U}$ and $i \in \mathcal{I}_c$. For each item i, we have a subset of users $\mathcal{U}_i^+$ to indicate the matched users (users who have already interacted the item during training or will interact with the item during testing) for this item: for a warm start item $i \in \mathcal{I}_w$, $\mathcal{U}_i^+$ are matched users in the training set O_{tr}; for a new item $i \in \mathcal{I}_c$, $\mathcal{U}_i^+$ are matched users in the test set O_{te}. *The goal of a cold start recommender [29, 35, 37, 42] is to provide a ranked list of cold start items to each user as recommendations.*

To provide cold start recommendations, typical machine learning based methods [4, 9, 20, 27–29, 32, 35, 37, 42] need to utilize user-item interactions O_{tr} of existing warm start users and items, and content features of both warm and cold start items. These content features – such as item descriptions, reviews, or from other sources – are often readily available even for new items. The main idea of these cold start recommendation algorithms is to learn a transformation from item content features of warm start items to user-item interactions between warm start users and items during training, and then apply this learned transformation process to the content features of new items to predict possible interactions between users and new items as recommendations during testing. Note that item content features of warm start items and new items share the same feature space. As a result, bias inherent in training data of warm users and items collected from an existing fairness-unaware recommender system will be transferred through the item content features to recommendations for new items, generating unfair recommendations among these new items.

3.2 Formalizing Fairness

A natural following question is how to determine if recommendations are fair or not among new items? In this work, we follow two well-known concepts to formalize fairness: equal opportunity [12] and Rawlsian Max-Min fairness principle of distributive justice [26].

In a classification task, **equal opportunity** requires a model to produce the same true positive rate (TPR) for all individuals or groups. Equal opportunity fairness has already been recognized as unquestionably important in recommender systems by previous works [5, 10, 25, 43]. The goal is to ensure that items from different groups can be equally recommended to matched users during testing (the same true positive rate): for example, candidates of different genders are equally recommended to job openings that they are qualified for. In contrast, demographic parity fairness [14, 43] only focuses on the difference in the amount of exposure to users without considering the ground-truth of user-item matching. However, because only the exposure to matched users (as considered by equal opportunity fairness) can influence the feedback or economic gain of items, in recommendation tasks, equal opportunity is better aligned than demographic parity fairness.

Rawlsian Max-Min fairness requires a model to maximize the minimum utility of individuals or groups so that no subject is underserved by the model. Unlike equality (or parity) based notions of fairness [5, 10, 14–17, 25, 43] aiming to eliminate difference among individuals or groups but neglecting a decrease of utility for better-served subjects, Rawlsian Max-Min fairness accepts inequalities and thus does not requires decreasing utility of better-served subjects. So, Rawlsian Max-Min fairness is preferred in applications where perfect equality is not necessary, such as recommendation tasks, and it can also better preserve the overall model utility.

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

Definition 3.1 (Max-Min Opportunity Fairness). Suppose $\mathcal{H}$ is a set of models, $TPR(i)$ is the expected true positive rate a new item i gets from a model h, then the model h^* is said to satisfy Max-Min Opportunity Fairness if it maximizes the true positive rate of the worst-off item:

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

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

Then, the next question is how to calculate the true positive rate of an item? We calculate the true positive rate for a new item by averaging a scoring function of ranking positions[1] across all matched users in the test set. Concretely, we propose the true positive rate metric Mean Discounted Gain (MDG) for an item i:

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

where $\mathcal{U}_i^+$ is the set of matched users for the new item i in the test set; $\widehat{z}_{u,i}$ is the ranking position of i (among all new items $\mathcal{I}_c$, ranging from 1 to M_c) for user u by a cold start recommendation model; $\delta(x)$ returns 1 if x is true, otherwise 0. That is to say, we only consider the discounted gain $1/log(1 + \widehat{z}_{u,i})$ for ranking positions within the top-k and assign 0 for positions after k (we fix $k = 100$ in this work) so that MDG is aligned with the well-known metric NDCG@k [22]. $MDG_i = 0$ means that item i is never recommended

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

Table 1: Characteristics of the four public datasets.

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

to matched users who like it during testing; $MDG_i = 1$ means that i is ranked at the top position to all matched users during testing.

With the introduced metric MDG, we measure the Max-Min Opportunity Fairness in Definition 3.1 for a cold start recommender by calculating the average MDG of the $t\%$ worst-off items, which are the $t\%$ items with the lowest MDG among all cold start items during testing. In the empirical study, we report results with $t\% = 10\%$ and 20%, denoted as **MDG-min10%** and **MDG-min20%**. Higher values indicate the evaluated system is fairer. We also report the average MDG for the 10% best-served items for comparison, which are the 10% items with the highest MDG, denoted as **MDG-max10%**.

3.3 Measuring Utility for Items

The typical way to evaluate the quality of a recommender system, such as with NDCG@k, is to first calculate the recommendation utility for each user and then average across users as the measured utility for a recommendation algorithm. Thus, this metric represents the expectation of recommendation utility a random user can receive from an algorithm, which is essentially from the view of users. We call these conventional metrics like NDCG@k as **user-view utility**. In contrast, we can also evaluate the recommendation utility from the view of items to show how well items are generally served by a recommendation algorithm. Because we study item fairness in this work, it is natural to consider this **item-view utility** as one evaluation aspect in empirical studies. In this work, given the true positive rate metric MDG introduced in Section 3.2, we report **MDG-all** = $\sum_{i \in \mathcal{I}_c} MDG_i / |\mathcal{I}_c|$ as the item-view utility metric, which shows the expectation of recommendation utility a random item can get from an algorithm.

3.4 Data-Driven Study

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

Datasets. We adopt four widely used datasets for cold start recommendation: **ML1M** [13], **ML20M** [13], **CiteULike** [38], and **XING** [3]. ML1M and ML20M are movie rating datasets, where we consider all ratings as positive signals. Both datasets contain tag genome scores [36] (showing relevance of an item to a fixed set of tags) as the item content features. CiteULike is a dataset recording user preferences toward scientific articles, and following [37], we use the abstracts of these articles as item content features. XING is a user-view-job dataset, and it includes career level, tags, and other related information as the item content features. For ML1M, ML20M, and XING, we randomly select 10% items and 30% items as the cold start (new) items, with all the user-item interactions of these items, to be the validation set and test set respectively. For CiteULike, we directly adopt the dataset splitting from [37]. The detailed statistics of the four datasets are shown in Table 1.

Table 2: Empirical results of four algorithms on ML1M (DN stands for DropoutNet, DM stands for DeepMusic).

		Heater	DN	DM	KNN	Optimal	Random
User utility	NDCG@15	.5516	.5488	.5312	.4402	1.000	.0550
	NDCG@30	.5332	.5316	.5167	.4226	1.000	.0586
Item utility	MDG-all	.0525	.0552	.0572	.0646	.1932	.0236
Fairness	MDG-min10%	.0000	.0000	.0000	.0001	.1388	.0118
	MDG-min20%	.0000	.0000	.0001	.0020	.1498	.0145
	MDG-max10%	.2272	.2294	.2323	.2091	.2471	.0386

Cold start recommendation models. There are many different cold start recommendation models in the literature, and it is impossible to test all of them. Generally, existing algorithms can be categorized into joint-training, separate-training, combined, and heuristic non-parametric methods as introduced in Section 2.2. Thus, in this work, we pick four representative algorithms from each category: a typical joint-training method **DropoutNet** [37]; a typical separate-training method **DeepMusic** [35]; a combination of joint-training and separate-training method **Heater** [42]; and a heuristic non-parametric method **KNN** [29].

Experiment protocol. Following these cold start recommendation works [29, 35, 37, 42], during testing, we evaluate recommendations for only new items without mixing warm start items in, which allows us to focus on the behavior of cold start recommenders and deepen our understandings of fairness among new items.

Empirical results. First, we report the results of four cold start recommendation algorithms on the ML1M dataset in Table 2. Besides, we also show two special cases: i) the **Optimal** case we can achieve, for which we directly get access to the positive user-item pairs in the test set, and rank the matched items to the most top positions with a random order for each user; and ii) the **Random** case, for which we randomly rank the items for each user. The first three rows in Table 2 show the results of user-view utility (NDCG@15 and NDCG@30) and item-view utility (MDG-all). *User-view utility and item-view utility reveal different aspects of a recommender system, which usually show opposite patterns.*[2] For example, in Table 2, Heater achieves the best user-view utility, however, it has the lowest item-view utility (while KNN is the opposite). Since items are treated unfairly by Heater, a small subset of items (potential popular items) with large numbers of test samples receive very high MDG and the majority of items get very low MDG, leading to high user-view utility but low item-view utility. In other words, the different patterns of user-view and item-view utility is due to the unfairness in recommendations.

Hence, we next analyze the fairness as shown in the last three rows in Table 2, where we present MDG-min10%, MDG-min20% to show the average MDG of the worst-off items; and we also present MDG-max10% to show the average MDG of the best-served items for comparison. We can observe that Heater and DropoutNet produce zero MDG for the 10% and 20% worst-off items, which means these items are never exposed to matched users who will like them during testing. DeepMusic and KNN perform slightly better,

[2] Although in practice, user-view and item-view utility often show opposite patterns, indeed, they are not opposite to each other. Theoretically, an optimal model can achieve the best user-view and item-view utility at the same time, and achieve fairness among new items as well. One example of such an optimal model is the 'Optimal' in Table 2.

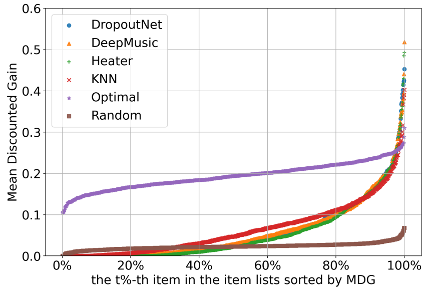


Figure 2: For ML1M and each model, sort items by MDG in ascending order and plot their corresponding MDG.

but the MDG-min10% and MDG-min20% are still very low. However, MDG-max10% values are very high for all four algorithms. *The large difference between MDG-min and MDG-max illustrates the unfairness among new items in these four cold start recommendation models.* For the Optimal case, because the recommendation is optimal for every user and item, the fairness is also guaranteed. Moreover, we can also observe that MDG-min10% and MDG-min20% in the Random case are higher than the personalized cold start recommendation models, which on one hand shows the unfairness in these cold start models, and on the other hand, demonstrates one possible way to enhance the fairness by introducing randomness to recommendations.

To further understand the unfairness issue in these algorithms, we plot the MDG of each item by different algorithms in Figure 2, where for each algorithm, items are sorted based on their MDG in ascending order. Each dot in Figure 2 represents one item for one algorithm; the y-axis shows the MDG an item receives from one of the six algorithms; and the x-axis shows the position of an item in a sorted item list of an algorithm (e.g., the dot corresponding to 100% represents the item with the largest MDG for one algorithm, 0% represents the item with the lowest MDG). From the figure, we can see that all four cold start recommendation algorithms produce skewed distributions of MDG across recommended new items: most items receive low or even zero MDG, and only a few items receive extremely high MDG, confirming the existence of unfairness. The Optimal shows the best result we can achieve on the given test set, where we can observe that the overall distribution is much more flat, with higher MDG for worst-off items but lower MDG for the best-served items compared with the other four algorithms. *The goal of fairness enhancement is to generate a distribution as close as possible to the optimal case.*

Last, we report results for the best-performing model Heater on all four datasets in Table 3. From the table, we can see that for all four datasets, MDG-min10% and MDG-min20% are very small (or even zero) compared with MDG-max10%, demonstrating that *the unfairness among new items is prevalent across datasets from different domains and with different characteristics.* Besides, comparing results of different datasets, we find that *fairness is highly related to the density of training data and quality of item content features*: training data with high density or with high-quality item content features can lead to more unfair recommendations, such as ML1M and ML20M which are very dense and have high-quality item content features (informative tag genome scores [36]). When the training data is dense or item content features are informative, a cold start recommendation model can more effectively learn information including data bias in training data and hence deliver unfairer recommendations.

Table 3: Empirical results of Heater on four datasets.

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

4 FAIRNESS ENHANCEMENT APPROACHES

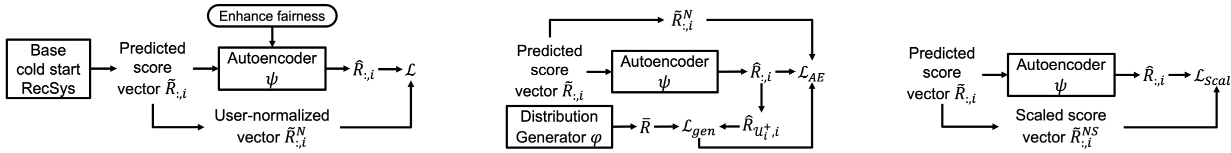
Given the observation of unfairness, we study in this section how to enhance the fairness for new items. We first propose a novel learnable post-processing framework to enhance the fairness for new items, which is not a concrete model but a high-level solution blueprint. Then, based on this blueprint, we propose two concrete models: a novel joint-learning generative method; and a score scaling model, which adapts a previous work for popularity bias [34] into the proposed framework, as a baseline for comparison.

4.1 Learnable Post-processing Framework

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

Main structure of the framework. Most existing works improve item fairness [5, 10, 39, 43], popularity bias [2, 8, 34], or diversity [24, 40] for warm start recommender systems by either in-processing methods or non-parametric post-processing methods. In-processing methods [2, 5, 39] modify the original recommendation models to achieve fairness or other goals. The major drawback of this type of method is that it requires re-training the whole recommender system with all training data, which is highly resource intensive and not practically feasible in real-world systems. Another widely investigated class is the non-parametric post-processing method [10, 21, 33], which keeps the original recommender systems unchanged, but conducts a heuristic re-ranking to the output of the original model to achieve fairness or other goals. This type of approach is more practically feasible because it does not require re-training the base model. But limitations are that it usually produces inferior performance than learning based in-processing methods and can be difficult to adapt to equal opportunity based fairness. To overcome the disadvantages of these two types of approaches, we propose a learnable post-processing framework to enhance fairness.

The proposed learnable post-processing framework is shown in Figure 3a, which has three main components: i) Similar to the non-parametric post-processing method, we keep the original base model unchanged, which can be any existing cold start recommendation model, such as Heater, DropoutNet, DeepMusic, or KNN used in Section 3.4. ii) Instead of the heuristic re-ranking in the non-parametric post-processing method, we learn an autoencoder (denoted as ψ) to conduct the 're-ranking'. We input the predicted score vector of an item i from the base model to the autoencoder ψ, where the score vector is denoted as $\widetilde{R}_{:,i} \in \mathbb{R}^N$ and the size of the vector is the total number of users N. Autoencoder ψ outputs the reconstructed score vector $\widehat{R}_{:,i} \in \mathbb{R}^N$ for final rankings. iii) By introducing fairness to the training process of the autoencoder ψ, we enable ψ to produce fairer results for new items.


Enhance fairness
Base cold start RecSys
Predicted score vector $\widetilde{R}_{:,i}$
Autoencoder ψ
$\widehat{R}_{:,i}$
$\mathcal{L}$
User-normalized vector $\widetilde{R}^N_{:,i}$
$\widetilde{R}^N_{:,i}$
Predicted score vector $\widetilde{R}_{:,i}$
Autoencoder ψ
$\widehat{R}_{:,i}$
$\mathcal{L}_{AE}$
Distribution Generator φ
$\overline{R}$
$\mathcal{L}_{gen}$
$\widehat{R}_{\mathcal{U}_i^+,i}$
Predicted score vector $\widetilde{R}_{:,i}$
Autoencoder ψ
$\widehat{R}_{:,i}$
$\mathcal{L}_{Scale}$
Scaled score vector $\widetilde{R}^{NS}_{:,i}$

(a) The learnable post-processing framework. **(b) The Gen method.** **(c) The Scale method.**

Figure 3: The structures of the proposed learnable post-processing framework and two concrete models.

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

Enhancing fairness in this framework. The next question is how can we enhance the fairness among new items in such a framework? The ultimate goal of fairness is to promote the true positive rate for worst-off items so that they can receive similar true positive rate as best-served items. From the formulation of true positive rate (such as Equation 1), we know that if we can improve the expectation of ranking position to matched users for under-served items during testing, then fairness can be enhanced. However, ranking position is hard to control in a recommendation model. So we need to further align the ranking positions with scores predicted by models, then ensure the fairness by increasing the expectation of predicted scores to matched users for under-served items. Therefore, to achieve fairness, we need to accomplish two requirements:

- **Requirement-1**: promote under-served items so that their distributions of matched-user predicted scores, denoted as $P(\widehat{R}_{\mathcal{U}_i^+,i})$, are as close as possible to the distributions of best-served items.
- **Requirement-2**: for every user, the predicted scores, denoted as $\widehat{R}_{u,:}$, follow the same distribution.

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

To achieve these two requirements for new items during testing, we need to train the autoencoder ψ so that ψ achieves these requirements for warm start items during training. Requirement-2 is easy to accomplish. First, before training ψ, we normalize the score distributions of users in the outputs of the base model to a standard normal distribution (score distributions for users are usually considered as normal distributions [23, 43]). And we use the normalized scores as the ground truth to train ψ. By this, ψ will learn to output scores normalized for users. Thus, during training, we have the predicted score matrix for warm start items $\widetilde{R} \in \mathbb{R}^{N \times M_w}$ from the base model, and we normalize it for each user:

$$\widetilde{R}^N_{u,:} = (\widetilde{R}_{u,:} - Mean(\widetilde{R}_{u,:}))/Std(\widetilde{R}_{u,:}), \tag{2}$$

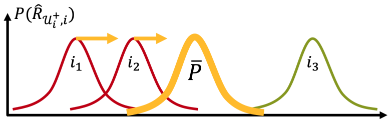


Figure 4: In each training epoch, update ψ to push $P(\widehat{R}_{\mathcal{U}_i^+,i})$ of under-estimated items (i_1 and i_2) as close as possible to the target $\overline{P}$ generated by φ.

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

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

4.2 The Joint-learning Generative Method

The overall framework of the joint-learning generative method is shown in Figure 3b where there are two main components: an autoencoder ψ and a distribution generator φ. The intuition of Gen is: in each training epoch, the distribution generator φ first generates a target distribution $\overline{P}$; then we update the autoencoder ψ so that it promotes items that are under-estimated in prior epochs by pushing their matched-user score distributions $P(\widehat{R}_{\mathcal{U}_i^+,i})$ as close as possible to the target distribution $\overline{P}$, and at the same time, have ψ preserve the recommendation utility as much as possible; last we update the distribution generator φ to generate a new target distribution $\overline{P}$ as the average of all items' matched-user score distributions $P(\widehat{R}_{\mathcal{U}_i^+,i})$. We show an example in Figure 4, where there are 3 items i_1, i_2 and i_3 and we show their matched-user score distributions at current training epoch. i_1 and i_2 have worse distributions (lower expectations of matched-user scores) than i_3, and we have the target distribution $\overline{P}$ which is the average of all three items. So, in this epoch, we need to update ψ so that $P(\widehat{R}_{\mathcal{U}_i^+,i})$ of i_1 and i_2 are promoted to be close to $\overline{P}$. After this, we also need to update the $\overline{P}$ to be the average of the new $P(\widehat{R}_{\mathcal{U}_i^+,i})$ of these three items. By jointly learning these two components, we can push the matched-user score distribution $P(\widehat{R}_{\mathcal{U}_i^+,i})$ of under-estimated items (like i_1 and i_2) to be closer and closer to best-served items (like i_3) epoch by epoch, and eventually achieve a fair status.

Concretely, the autoencoder ψ is the same as the one in the framework in Section 4.1, which takes predicted score vectors $\widetilde{R}_{:,i}$ from a base model as input and outputs $\widehat{R}_{:,i}$, with user-normalized score vectors $\widetilde{R}^N_{:,i}$ as training ground-truth.

The distribution generator φ is to generate a target distribution $\overline{P}$ (the average distribution of $P(\widehat{R}_{\mathcal{U}_i^+,i})$ over all items from last epoch) for under-estimated items to be promoted to. φ is a multi-layer perceptron (MLP) with 1-dimension input and output layers, which takes S random seeds from a standard normal distribution as inputs and outputs S samples $\overline{R} \in \mathbb{R}^S$ to represent the target distribution $\overline{P}$. To generate such a $\overline{R}$, at the end of each training epoch, we first retrieve the matched-user entries $\widehat{R}_{\mathcal{U}_i^+,i}$ in the output vectors $\widetilde{R}_{:,i}^N$ from the autoencoder ψ for all warm items $\mathcal{I}_w$. Then, for each item i, we update φ to minimize the sum of distribution distances between generated samples $\overline{R}$ from φ and matched-user scores $\widehat{R}_{\mathcal{U}_i^+,i}$ so that the underlying distribution of $\overline{R}$ has the minimum sum of distances to all items. For example, in Figure 4, the generated $\overline{P}$ has the minimum sum of distances to these three items, that is, $\overline{P}$ is the average distribution of these items. Here, we adopt the Maximum Mean Discrepancy (MMD) [11], an effective kernel based statistic test method, to calculate the distribution distance by samples from two distributions as the loss for the distribution generator:

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

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

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

$$\begin{aligned}\min_{\psi} \mathcal{L}_{AE} = \sum_{i \in \mathcal{I}_w} (&\|\widetilde{R}_{:,i}^N - \widehat{R}_{:,i}\|_{\mathrm{F}} \\ &+ \alpha(MMD(\overline{R}, \widehat{R}_{\mathcal{U}_i^+,i}) \cdot \delta(i \in \mathcal{I}_{UE}))) + \lambda\|\psi\|_{\mathrm{F}},\end{aligned}$$

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

4.3 The Score Scaling Method

In addition to Gen, we adapt an existing work for addressing popularity bias [34] into the proposed learnable post-processing framework to enhance fairness among new items. This score scaling method can also serve as a baseline method.

To counteract popularity bias, the work [34] re-scales the training data based on item popularity: it up-scales ratings for unpopular items and down-scales ratings for popular items of high popularity, and then it trains a recommendation model on the scaled data to deliver debiased recommendations. Similarly, we can scale the user-normalized predicted score vectors $\widetilde{R}_{:,i}^N$ to be fairer ground-truth to train a fair autoencoder ψ in the proposed post-processing framework. In detail, for the user-normalized score vector $\widetilde{R}_{:,i}^N$ of each warm start item $i \in \mathcal{I}_w$, we scale the matched-user entries:

$$\widetilde{R}_{\mathcal{U}_i^+,i}^{NS} = \widetilde{R}_{\mathcal{U}_i^+,i}^{N} \times \frac{Max(\{Mean(\widetilde{R}_{\mathcal{U}_j^+,j}^N)^\beta | j \in \mathcal{I}_w\})}{Mean(\widetilde{R}_{\mathcal{U}_i^+,i}^N)^\beta}, \tag{3}$$

where $\widetilde{R}_{\mathcal{U}_i^+,i}^N$ are the matched-user entries in $\widetilde{R}_{:,i}^N$; $Mean(\widetilde{R}_{\mathcal{U}_i^+,i}^N)$ calculates the mean value of matched-user entries; $Max(\cdot)$ returns the maximum value in a sequence, and we use $Max(\{Mean(\widetilde{R}_{\mathcal{U}_j^+,j}^N)^\beta | j \in \mathcal{I}_w\})$ as the numerator so that no item is down-scaled but only under-estimated items are up-scaled during the scaling; β is the fairness-strength weight – the larger the more strength to enhance fairness; $\widetilde{R}_{\mathcal{U}_i^+,i}^{NS}$ denotes the scaled entries for matched users, and we write them back to $\widetilde{R}_{:,i}^N$ to have the final score vector $\widetilde{R}_{:,i}^{NS}$ for i as the ground-truth for training the autoencoder ψ. The overall framework of the proposed Scale is shown in Figure 3c.

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

$$\min_{\psi} \mathcal{L}_{Scale} = \sum_{i \in \mathcal{I}_w} \|\widetilde{R}_{:,i}^{NS} - \widehat{R}_{:,i}\|_{\mathrm{F}} + \lambda\|\psi\|_{\mathrm{F}}. \tag{4}$$

5 EXPERIMENTS

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

5.1 Experiment Setup

Data and Metrics. Similar to the data-driven study in Section 3.4, we use the same four datasets (ML1M, ML20M, CiteULike, and XING) and same metrics: we report NDCG@k with k=15 and 30 for evaluating user-view utility; MDG-all for item-view utility; MDG-min10% and MDG-min20% (the larger the fairer a model is) for fairness; and we also report MDG-max10% for comparison.

Baselines. We use the same four cold start recommendation base models (Heater, DropoutNet, DeepMusic, and KNN) as introduced in Section 3.4. And we investigate the fairness-enhancement performance of the proposed Gen and Scale. We consider the Scale method, which adapts a previous work for addressing popularity bias into the proposed learnable post-processing framework, as one baseline to compare with Gen. Besides, we saw in Section 3.4 that random rankings can also improve the true positive rates of worst-off items compared with personalized cold start recommendation algorithms. Hence, we also consider a **Noise** method which adds random noise to the output of base cold start recommendation

Table 4: Empirical results on ML1M dataset for all models.

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

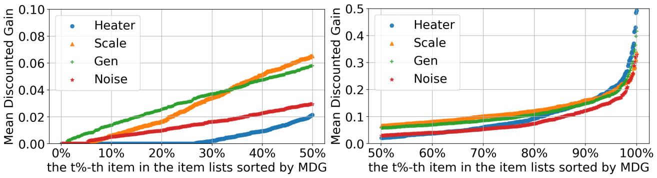
models as another baseline, in which there is a fairness-strength weight γ to control the amount of noise added.

Reproducibility. All models are implemented by Tensorflow [1] and optimized by Adam algorithm [18]. For the four cold start recommendation base models, we follow the hyper-parameter tuning strategies in [42]. For the two proposed fairness-enhancement models, we assign the autoencoder ψ a single hidden layer of dimension 100 with a linear activation function. For Gen, we assign the distribution generator φ two hidden layers of dimension 50 with a *tanh* activation function. We tune the fairness-strength weight α, β, γ in Gen, Scale, and Noise models by grid search on validation sets. Because there is a trade-off between the user-view utility and fairness, when tuning the fairness-strength weights α and β, we try to preserve a relatively high NDCG@k for both methods and analyze the fairness performance of them. *All code, data, and settings will be available at https://github.com/Zziwei/Fairness-in-Cold-Start-Recommendation.*

5.2 RQ1: Fairness-Enhancement Performance

First, we investigate how do the two proposed methods perform in terms of improving fairness and preserving recommendation utility. We first apply the two proposed methods and baseline Noise to each of the four cold start recommendation models. Results on the Ml1M dataset are reported in Table 4, where we show results of the four cold start recommendation base models. The three rows following each of the cold start recommendation base models are results of Noise, Scale, and Gen with the given cold start recommendation model as the base model (forming a four-row group).

From Table 4, comparing the results before and after applying the two proposed methods Scale and Gen, we have four major observations: i) for all cold start recommendation methods, after applying the proposed models, the user-view utility decreases but with small percentages; ii) after applying Scale and Gen, the fairness among new items (evaluated by MDG-min10% and MDG-min20%) is significantly improved; iii) after applying proposed methods, the utility for the best-served items (measured by MDG-max10%) decreases, but with a limited percentage as well; and iv) the item-view utility (evaluated by MDG-all) significantly increases after applying Scale and Gen. This item-view utility improvement is due to that items originally under-served by base models receive more utility from the two proposed models, leading to the improvement of overall item-view utility even though the best-served items receive lower utility. Based on these observations, we can conclude that Scale and Gen can significantly enhance the fairness among new items; and they can also effectively preserve the user-view utility and improve the item-view utility.



(a) 0~50%-th items.

(b) 50~100%-th items.

Figure 5: Heater as base, MDG of items by different models.

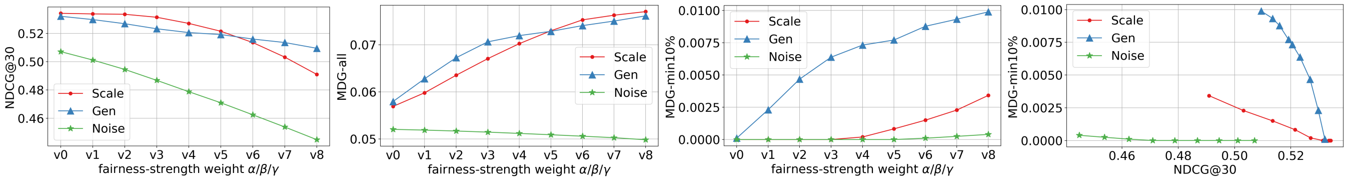
Next, we compare the performance between Noise, Scale, and Gen in Table 4. We can find that Scale and Gen promote fairness to a greater extent with higher utility (both user-view and item view) preserved than Noise, showing the effectiveness of the proposed framework and methods. Then, comparing Scale and Gen, under the circumstance that they produce similar user-view utility: i) Scale provides slightly higher item-view utility than Gen; but ii) Gen delivers better fairness-enhancement performance than Scale: Gen outperforms Scale for 295.68% for MDG-min10% and 80.95% for MDG-min20%. As a result, we conclude that Gen can enhance fairness for new items more effectively than Scale.

To better understand the effects of the proposed Scale and Gen, on the ML1M dataset, we sort the recommended new items by MDG in ascending order and plot them in Figure 5 for Heater and three fairness-enhancement methods with Heater as the base model. Figure 5a shows the MDG for items belonging to the first half of the sorted list (from 0%-th to 50%-th items), and Figure 5b shows the MDG for items belonging to the left half of the sorted list (from 50%-th to 100%-th items). From these two figures, we see that compared to the base model Heater: both Scale and Gen significantly increase MDG (better than Noise) for most items (around from 0%-th to 90%-th in the sorted item list), which are originally under-served by Heater; and MDG for items that are best-served by Heater (around 90%-th to 100%-th) are only slightly decreased by Scale and Gen. Moreover, another interesting observation is that Gen improves the MDG of the worst-served items better than Scale as shown in Figure 5a, while Scale promotes the MDG for items between the worst-served and best-served (around 40%-th to 90%-th) more than Gen. This is the reason why Gen outperforms Scale for fairness but Scale performs better for item-view utility as in Table 4.

Last, in Table 5, we report the results of Heater and the three fairness-enhancement methods with Heater as base model on all datasets. Similar conclusions can be drawn from these results: for all different datasets, the proposed Scale and Gen can more effectively enhance the fairness for new items and preserve the utility.

5.3 RQ2: Impact of Hyper-parameters

We next turn to study the impact of the fairness-strength weights α in Gen, β in Scale, and γ in Noise. Recall that larger α, β, and γ lead to more strength for enhancing fairness. We run experiments for Gen with α varying from 40 to 360 with step 40, Scale with β



(a) (b) (c) (d)

Figure 6: Investigate impact of α, β, and γ on ML1M dataset: (a) shows the impact on NDCG@30; (b) shows the impact on MDG-all; and (c) shows the impact on NDCG@30 and MDG-max10% together.

Table 5: Results on 4 datasets for Heater as base model.

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

varying from 1 to 5 with step 0.5, and Noise with γ varying from 0.4 to 0.8 with step 0.5. In Figure 6a, we show the results of NDCG@30 on the ML1M dataset, where v_0 to v_8 correspond to different values of α, β, and γ in ascending order. The figure shows that with larger fairness-strength weights, the user-view utility gets smaller for all methods. Then, we present how the item-view utility (measured by MDG-all) changes when we increase the fairness-strength weights in Figure 6b. We can observe that with weights increasing, MDG-all keeps increasing for both Scale and Gen, but slowly decreases for Noise. Next, we show how fairness (MDG-min10%) changes with weights increasing in Figure 6c, which demonstrates that all models improve fairness when the weights get larger.

For Figure 6a, Figure 6b, and Figure 6c, note that we cannot directly compare the NDCG@30, MDG-all, and MDG-min10% between Scale, Gen, and Noise because the x-axis (values of α, β, and γ) is not the same for these two methods. Therefore, to further compare these three methods, we plot Figure 6d: the y-axis is MDG-min10%; the x-axis is NDCG@30; each dot represents an experiment result of a model with a specific fairness-strength weight; and weights are in decreasing order from left to right (for example, the leftmost dot for Gen corresponds to the experiment with $\alpha = 360$). Now, we can conclude from this figure that with the same user-view utility preserved, Gen enhances fairness more effectively than Scale and Noise for different fairness-strength weights.

5.4 RQ3: Impact on Group-level Fairness

The fairness we discussed so far is for individual items: we consider the difference among individual items as unfairness. Another widely investigated concept is group-level fairness: consider the difference among different items groups determined by item attributes as unfairness [5, 10, 43]. Here, we want to study how do our proposed methods impact group-level fairness in cold start recommendation.

To be consistent to the measurement of fairness in this paper, we evaluate the group-level fairness by calculating the average MDG for the worst-off item group (the item group with the lowest average MDG), which is also similar to the measurement of group-level fairness in a classification task [19]. We denote the group-level fairness metric as **MDG-min-group**. For the ML1M dataset, we show the group-level fairness results of all four cold start recommendation base models and their corresponding fairness-enhanced results by the three methods in Figure 7, where we group items by the movie genres provided by ML1M dataset [13], and the worst-off movie genre for all methods is 'Documentary'. From Figure 7, we see that after applying the two proposed methods and baseline Noise, the group-level fairness is significantly improved, and Gen performs better than Scale and Noise. This result is reasonable and expected, because these methods improve MDG for all under-served items, resulting in under-served groups consisting of under-served items being promoted in general. This property is very helpful when group-level fairness is required but no group attribute is accessible.

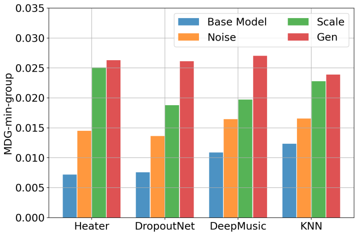


Figure 7: Investigate group-level fairness on ML1M.

6 CONCLUSION AND FUTURE WORK

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

ACKNOWLEDGMENTS

Co-author James Caverlee is partially supported by NSF #IIS-1939716.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*.

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*.

[3] Fabian Abel, Yashar Deldjoo, Mehdi Elahi, and Daniel Kohlsdorf. 2017. Recsys challenge 2017: Offline and online evaluation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 372–373.

[4] Iman Barjasteh, Rana Forsati, Farzan Masrour, Abdol-Hossein Esfahanian, and Hayder Radha. 2015. Cold-start item and user recommendation with decoupled completion and transduction. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 91–98.

[5] 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 & Data Mining*. 2212–2220.

[6] Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. 2018. Equity of attention: Amortizing individual fairness in rankings. In *The 41st international acm sigir conference on research & development in information retrieval*. 405–414.

[7] Robin Burke. 2017. Multisided fairness for recommendation. *arXiv preprint arXiv:1707.00093* (2017).

[8] Rocío Cañamares and Pablo Castells. 2018. Should i follow the crowd?: A probabilistic analysis of the effectiveness of popularity in recommender systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM.

[9] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning Attribute-to-Feature Mappings for Cold-Start Recommendations.. In *ICDM*, Vol. 10. Citeseer, 176–185.

[10] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. 2019. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2221–2231.

[11] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. 2007. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*. 513–520.

[12] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*. 3315–3323.

[13] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[14] Toshihiro Kamishima and Shotaro Akaho. 2017. Considerations on Recommendation Independence for a Find-Good-Items Task. (2017).

[15] T Kamishima, S Akaho, H Asoh, and J Sakuma. 2013. Efficiency Improvement of Neutrality-Enhanced Recommendation.. In *Decisions@RecSys*.

[16] Toshihiro Kamishima, S Akaho, H Asoh, and J Sakuma. 2018. Recommendation Independence. In *Conference on Fairness, Accountability and Transparency*.

[17] T Kamishima, S Akaho, H Asoh, and I Sato. [n.d.]. Model-based approaches for independence-enhanced recommendation. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed H Chi. 2020. Fairness without Demographics through Adversarially Reweighted Learning. *arXiv preprint arXiv:2006.13114* (2020).

[20] Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. From Zero-Shot Learning to Cold-Start Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[21] Weiwen Liu and Robin Burke. 2018. Personalizing fairness-aware re-ranking. *arXiv preprint arXiv:1809.02921* (2018).

[22] Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Cambridge university press.

[23] Andriy Mnih and Russ R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[24] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. 2014. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*. ACM, 677–686.

[25] Flavien Prost, Hai Qian, Qiuwen Chen, Ed H Chi, Jilin Chen, and Alex Beutel. 2019. Toward a better trade-off between performance and fairness with kernel-based distribution matching. *arXiv preprint arXiv:1910.11779* (2019).

[26] John Rawls. 2001. *Justice as fairness: A restatement*. Harvard University Press.

[27] Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 89–96.

[28] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, Lexing Xie, and Darius Braziunas. 2017. Low-rank linear cold-start recommendation from social data. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[29] Suvash Sedhain, Scott Sanner, Darius Braziunas, Lexing Xie, and Jordan Christensen. 2014. Social collaborative filtering for cold-start recommendations. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 345–348.

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

[31] Ashudeep Singh and Thorsten Joachims. 2019. Policy learning for fairness in ranking. *arXiv preprint arXiv:1902.04056* (2019).

[32] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 650–658.

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

[34] Harald Steck. 2019. Collaborative filtering via high-dimensional regression. *arXiv preprint arXiv:1904.13033* (2019).

[35] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*. 2643–2651.

[36] Jesse Vig, Shilad Sen, and John Riedl. 2012. The tag genome: Encoding community knowledge to support novel interaction. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 2, 3 (2012), 1–44.

[37] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. In *Advances in Neural Information Processing Systems*. 4957–4966.

[38] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 448–456.

[39] Sirui Yao and Bert Huang. 2017. Beyond parity: Fairness objectives for collaborative filtering. In *Advances in Neural Information Processing Systems*. 2921–2930.

[40] Mi Zhang and Neil Hurley. 2008. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems*.

[41] 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.

[42] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1121–1130.

[43] Ziwei Zhu, Jianling Wang, and James Caverlee. 2020. Measuring and Mitigating Item Under-Recommendation Bias in Personalized Ranking Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 449–458.