Fighting Mainstream Bias in Recommender Systems via Local Fine Tuning

Ziwei Zhu, James Caverlee
Department of Computer Science and Engineering, Texas A&M University
College Station, TX, USA
zhuziwei,caverlee@tamu.edu

ABSTRACT

In collaborative filtering, the quality of recommendations critically relies on how easily a model can find similar users for a target user. Hence, a niche user who prefers items out of the mainstream may receive poor recommendations, while a mainstream user sharing interests with many others will likely receive recommendations of higher quality. In this work, we study this mainstream bias centering around three key thrusts. First, to distinguish mainstream and niche users, we explore four approaches based on outlier detection techniques to identify a mainstream score indicating the mainstream level for each user. Second, we empirically show that severe mainstream bias is produced by conventional recommendation models. Last, we explore both global and local methods to mitigate the bias. Concretely, we propose two global models: Distribution Calibration (DC) and Weighted Loss (WL) methods; and one local method: Local Fine Tuning (LFT) method. Extensive experiments show the effectiveness of the proposed methods to improve utility for niche users and also show that the proposed LFT can improve the utility for mainstream users at the same time.

CCS CONCEPTS

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

KEYWORDS

recommender systems; mainstream bias; local models

ACM Reference Format:
Ziwei Zhu, James Caverlee. 2022. Fighting Mainstream Bias in Recommender Systems via Local Fine Tuning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3488560.3498427

1 INTRODUCTION

Recommender systems play an increasingly important role in connecting users to interesting items to alleviate the information overload issue. Most recommendation systems, including those based on classic linear models [9, 17, 27] and recent neural-network models [15, 23, 34], predict user preference and provide recommendations based on Collaborative Filtering (CF). The main idea is to estimate the preference from a user to an item depending on the attitudes from other similar users to the item. By finding other users with similar interests as the target user, these CF approaches have demonstrated strong recommendation performance.

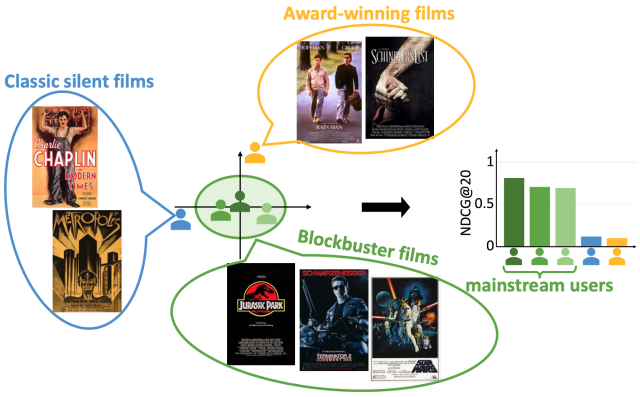

Figure 1: Mainstream vs. niche users from MovieLens data.

Naturally, the quality of recommendations critically relies on how easily the model can find similar users for a target user. A *niche (or "indie") user* who prefers items that are out of the mainstream may have few if any nearby users, resulting in poor recommendations. In contrast, a *mainstream user* who shares interests with many other users will likely receive many high-quality recommendations. To illustrate, Figure 1 shows three mainstream users and two niche users from the MovieLens dataset [14], who are identified based on a method introduced in this paper. All three mainstream users share similar preferences for blockbuster films: the recommendations from a recent variational autoencoder (VAE) model [23] result in high NDCG@20. In contrast, we see that for the two niche users – one of whom prefers classic silent films, while one prefers award-winning films of the late 80s/early 90s – the resulting recommendation quality is quite poor. Indeed, we find similar patterns for mainstream vs. niche users across multiple datasets (including Yelp [1] and Epinions [33]) and for different models (including matrix factorization [17], BPR [27], and local collaborative autoencoders [7]).

This *mainstream bias* – **the tendency for recommendation models to favor mainstream users over niche users** – is a critical challenge for the ongoing success of recommendation systems. But how do we identify mainstream users vs. niche ones? What impact does the degree of mainstream-ness have on recommendation utility? And can we develop methods to ameliorate this mainstream bias? Can we improve the recommendation utility for users of low

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498427

mainstream levels while preserving or even increasing the utility for mainstream users at the same time? Toward answering these research questions, this paper is organized around three key thrusts:

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

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

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

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

2 RELATED WORK

Many issues related to bias and fairness in recommender systems have been studied in recent years. Some prominent examples include exposure bias, popularity bias, and item fairness, among others. In *exposure bias*, the training data to train a new model is usually collected from an existing recommender system where items are recommended to users with different probabilities. Hence, the new model cannot learn true user-item relevance from the data but will follow the behavior of the existing system [20, 29, 30, 41]. With respect to *popularity bias*, many studies have identified how recommendation models tend to over-recommend popular items but overlook long-tail items [2, 3, 35, 38, 39, 42]. *Item fairness* refers to the situation where different item groups (often determined by item content features) are treated differently by recommendation models with some groups being overly exposed to users while others being rarely recommended [5, 13, 24, 37, 43].

These prior works mainly focus on the item perspective. Yet, how users are treated is an equally important topic. The majority of research works investigating the bias on users aim to analyze the utility difference among different user groups determined by user demographic attributes, such as age or gender [10, 12, 22, 31, 40]. For example, Schedl et al. [31] study the music preference difference among different user age groups and shows that the recommendation performance for these age groups are also different. Ekstrand et al. [10] empirically study multiple types of recommendation models and demonstrate that all of the investigated models produce a utility difference across user demographic groups. To address this problem, Fu et al. [12] propose to take advantage of the rich information from knowledge graphs, and Li et al. [22] create a re-ranking algorithm to reduce the utility gap among user groups.

Different from the aforementioned works studying bias based on demographic groups, we aim to recognize the mainstream and niche users and study the utility difference between them. Also note that user demographic attributes may not necessarily explain the interests and behaviors of a user. A similar work to this paper is [21], whose goal is to improve the utility for niche users. Nevertheless, the major differences are: in [21], the mainstream and niche users are determined purely based on recommendation utility they receive rather than based on the true user preference reflected by historical feedback; and the algorithm proposed in [21] requires additional auxiliary information of users and items (such as the review text users give to items), while we aim to debias relying merely on feedback from users.

In this work, we find that local recommendation models [7, 8, 18, 19], although not designed for this purpose, can mitigate the bias to some degree by improving the utility for niche users. The main idea of these methods is to use different local models to serve different types of users. Among existing methods, the recently proposed local collaborative autoencoder (LOCA) [7] produces the state-of-the-art performance, which uses multiple variational autoencoders (VAE) [23] as local models to capture the special patterns of different sub-communities. Hence, in our experiments, we follow LOCA to consider VAE as the base model for our proposed local method, and we empirically compare our proposed methods with LOCA.

3 ANALYZING MAINSTREAM BIAS

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

Table 1: NDCG@20 of different subgroups determined by different mainstream level evaluation approaches.

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

3.1 Formalizing the Recommendation Task

Formally, we have a set of N users as $\mathcal{U} = \{1, 2, \ldots, N\}$ and a set of M items as $\mathcal{I} = \{1, 2, \ldots, M\}$. We denote the set of implicit feedback from users to items as $\mathcal{O} = \{(u, i)\}$ where $u \in \mathcal{U}$ indexes one user, and $i \in \mathcal{I}$ indexes one item. We use this feedback set as the training data to train a recommendation model and provide recommendations to users. For a user u, we use a binary vector of size M, denoted as $\mathbf{O}_u \in \{0, 1\}^M$, to represent the feedback record vector of user u, with 1 representing u likes the corresponding item. During evaluation, the trained model provides a ranked list of items for each user as recommendations, and we evaluate the recommendation list based on the ranking positions of positive items in a testing set for every user. Many ranking evaluation metrics can be used, such as NDCG@k and recall@k [23], which are typically averaged over all users.

3.2 Evaluating Mainstream Level of Users

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

The problem of assessing a user's mainstream level can be easily turned to an outlier detection problem: we consider niche users who have different preferences from the majority as the outlier samples to detect. So, inspired by various outlier detection techniques [4], we explore four different approaches and want to determine which approach performs the best for assessing user mainstream level.

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

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

Density-based approach. The next approach we propose is based on the density-based outlier detection method, which determines whether one sample is an outlier by investigating the density of the sample's neighbors. In this work, we propose to directly apply the well-known local outlier factor (LOF) algorithm [6] to the user feedback records to identify niche users. The LOF algorithm outputs the local outlier factor value for each user, which indicates an outlier if its value is large. Thus, we add a negative sign to the local outlier factor value as the mainstream score for a user u:

$$MS_u^{den} = -LOF(u). \tag{2}$$

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

$$MS_u^{dis} = cos(\mathbf{O}_u, \mathbf{d}). \tag{3}$$

DeepSVDD-based approach. Last, we apply the recent deep learning based outlier detection algorithm – deep support vector data description (DeepSVDD) [28] – to identify niche users. DeepSVDD attempts to map most of the data samples (belonging to one class) into a hypersphere by neural networks and considers the samples far from the center of the hypersphere as outliers. Moreover, since in a recommender system, there can be more than one mainstream preference, resulting in more than one user class in terms of preference. Hence, we further replace the multi-layer perceptron in the mapping component in the original DeepSVDD to a mixture-of-experts structure [32], so that the model can have different mapping functions for different classes to handle the multi-class situation more effectively. In our experiments, we set a 2-layer perceptron of size (400, 300) as one expert component and adopt 10 experts in total. After the mapping, we have a vector $\mathbf{c}$ in the new hyper-space representing the center of the hypersphere covering the majority of users, and we also have a vector $DeepSVDD(\mathbf{O}_u)$ to represent user u in the mapped hyper-space. For a user u, we use the negative distance from $DeepSVDD(\mathbf{O}_u)$ to center $\mathbf{c}$ as the score:

$$MS_u^{deep} = -\|DeepSVDD(\mathbf{O}_u) - \mathbf{c}\|_{\mathrm{F}}. \tag{4}$$

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

Table 2: Niche users classifying accuracy of four approaches.

	Similarity	Density	Distribution	DeepSVDD
Accuracy	0.66	0.31	0.68	0.73

3.3 Empirical Studies

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

3.3.1 Recommendation models produce mainstream bias. To answer the first question, we conduct experiments with real-world datasets and state-of-the-art recommendation models. More specifically, we first run a VAE [23] on the MovieLens 1M dataset [14]. Then, we apply the introduced four approaches to calculate mainstream scores for all users. Last, we sort users based on calculated mainstream scores in non-descending order and divide them into five subgroups with equal size. Note that we also run experiments with other models including MF [17], BPR [27], and LOCA [7], and other datasets including Yelp [1] and Epinions [33]. These experiments show similar patterns. Code and data can be found at https://github.com/Zziwei/Measuring-Mitigating-Mainstream-Bias.

The average NDCG@20 for subgroups corresponding to different mainstream-ness measuring ways are shown in Table 1, where we denote the first 20% of users with lowest mainstream scores as users of 'low' mainstream level, the subgroup of 20%-40% users as users of 'med-low' mainstream level, and so on for 40%-60% ('medium'), 60%-80% ('med-high'), and 80%-100% ('high'). From the table, we can observe that all four proposed approaches show a similar pattern – users with larger mainstream scores receive higher NDCG@20. For example, for all four bias measuring cases, the average NDCG@20 of 'high' mainstream level users is more than twice larger than those of 'low' mainstream level users. This result reveals that all proposed approaches are able to identify niche users who are underserved by the recommendation model and severe mainstream bias is produced by the recommendation model.

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

Given this setup, we run the four proposed approaches to identify niche users in this dataset. Here, we can formalize a binary classification task, where we consider the 400 users with lowest mainstream scores from each approach as the predicted niche users, and the 400 users from the last two generated user groups are the ground-truth labels. The classification accuracy is shown in Table 2, from which we can observe that with the help of deep learning techniques, the DeepSVDD-based approach performs the best. The next best approaches are similarity-based and distribution-based approaches, which perform similarly because both of them rely on the similarity calculation between users by their feedback records. Density-based approach performs the worst, which may be because the LOF algorithm cannot work effectively for high-dimensional and sparse data. As a result, we adopt the DeepSVDD-based approach as the best choice to analyze mainstream bias.

4 MITIGATING MAINSTREAM BIAS

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

4.1 Global Methods

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

4.1.1 Distribution Calibration Method (DC). This first method is a data augmentation based approach, whose main intuition is to generate synthetic users similar to existing niche users so that these niche users become mainstream in the training dataset. To achieve this, we adapt the Distribution Calibration method [36] for few-shot learning to the recommendation task. In the original paper [36], the distributions of few-shot classes are calibrated by transferring statistics from similar classes with abundant data. Then, synthetic examples of few-shot classes can be sampled based on the calibrated distributions to augment the training data. In a recommendation task, we can consider each niche user as a single class. Then, in a similar way, we can calibrate the distribution for each niche user by transferring statistics from other similar users and generate synthetic users based on the calibrated distribution.

Specifically, we first identify niche users by any of the proposed approaches in Section 3.2. For example, we consider the 50% users with lowest mainstream scores from DeepSVDD-based approach as niche users. Then, for one niche user u, we fetch similar users to u, and have the calibrated distribution vector $\mathbf{p}_u$ of u:

$$\mathbf{p}_u = \alpha \mathbf{O}_u + (1-\alpha)\frac{1}{|\mathcal{N}_u|}\sum_{v\in\mathcal{N}_u}\mathbf{O}_v, \tag{5}$$

where $\mathcal{N}_u$ is the set of similar users to u in terms of Jaccard similarity on feedback records; and $0 \le \alpha \le 1$ is a hyper-parameter to control the importance of original feedback of u in the resulting distribution.

Last, we sample synthetic users based on $\mathbf{p}_u$ for u. Given a budget for synthetic users (we use the total number of real users in this work), the number of synthetic users for each niche user is proportional to the reciprocal mainstream score of the user. At the end, a model trained by such an augmented dataset can mitigate the mainstream bias and improve utility for niche users.

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

$$\mathcal{L}_{WL} = \sum_{u \in \mathcal{U}} w_u \cdot \mathcal{L}_{VAE}(u), \quad w_u \propto (\frac{1}{MS_u})^{\beta}, \tag{6}$$

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

4.2 Local Method

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

Local recommendation methods have been studied in prior works [7, 8, 18, 19]. The main idea is to first select anchor users and train specialized anchor models for each of the anchor users. Then, during inference, given a user, we can customize a local model for this user by ensembling anchor models based on the relationship between the target user and anchor users. Although these local recommendation algorithms are not specifically designed for addressing mainstream bias, we empirically find that they can improve utility for niche users. Hence, in this section, we move further based on these local recommendation models to propose a Local Fine Tuning (LFT) method to effectively mitigate the mainstream bias, whose goal is to increase the utility for niche users with the utility for mainstream users preserved or even increased.

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

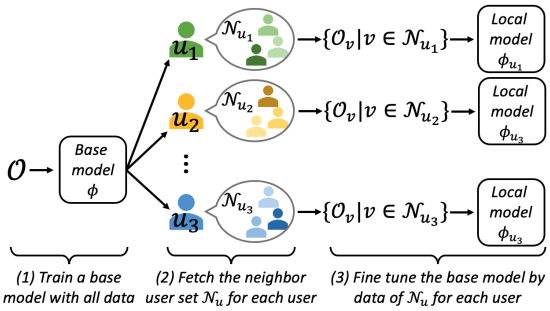

Figure 2: The proposed Local Fine Tuning method.

The proposed LFT is illustrated in Figure 2. Concretely, we first assume we have a global base model ϕ which is trained by the entire dataset as the step (1) in Figure 2, such as an ordinary VAE model in Section 3.3.1. Then, demonstrated as the step (2): for a target user u, we fetch the neighbor users $\mathcal{N}_u$ (including u herself) that are similar to u in terms of preference, and create a sub-dataset $\mathcal{O}_{\mathcal{N}_u} = \{\mathcal{O}_v | v \in \mathcal{N}_u\}$ only containing feedback data of neighbor users. Last, during inference, for a target user u, we further train the base model ϕ by the sub-dataset $\mathcal{O}_{\mathcal{N}_u}$ to fine tune the model as step (3) in Figure 2, so that it can provide accurate prediction for u without influence from irrelevant users. We denote the local model after fine tuning for u as ϕ_u. In this way, niche users can receive better utility by the local models since influence from mainstream users and other niche users with different preference is eliminated. Furthermore, mainstream users also benefit from their local models since they also suffer from the influence of niche users and other mainstream users with different preferences.

Now, the key question is: how to find the neighbor users $\mathcal{N}_u$ of a user u so that the feedback data of them $\mathcal{O}_{\mathcal{N}_u}$ can help improve the fine tuning effectiveness and eliminate the influence from irrelevant users? A naive way is to fetch the users with highest similarity (e.g., Jaccard or Cosine similarity) based on feedback records. However, the limitation is that the similarity between users based on discrete and sparse feedback records does not consider the latent relationship between users. For example, if a user only likes item A and another user only likes item B, the similarity between them will be 0 based on their feedback records. However, if item A and B are very similar, then the ground truth similarity between them should be high. Therefore, the naive neighbor user searching method could omit important neighbor users. Instead, we propose to fetch the neighbor users based on the similarity between the calibrated distributions of users introduced in Section 4.1.1. More specifically, we calculate the Cosine similarity between users by their calibrated distributions $\mathbf{p}_u$ calculated by Equation 5. For a target user u, we regard users with similarity over a threshold t as her neighbor users: $\mathcal{N}_u = \{v | cos(\mathbf{p}_u, \mathbf{p}_v) > t\}$. Because the calibrated distribution of a user tries to approximate the preference probability of the user toward items, it can help to capture the latent relationship between users that naive method cannot achieve.[1]

4.2.2 Choice of Base Model. Another key factor that can influence the performance of the proposed LFT is the choice of base model

[1] Because conventional recommendation models, such as MF [17], BPR [27], or VAE [23], are vulnerable to various bias including the mainstream bias, it is not appropriate to use the generated embeddings from these regular models to fetch neighbor users.

ϕ. To allow the base model to be fine tuned effectively, the base model should not be overly optimized for specific users and neglect other users, i.e., the base model should produce low mainstream bias. Otherwise, even if local fine tuning is applied, the final prediction will still be biased and in low quality for users overlooked by the base model. Hence, we propose to use the global debiasing model DC or WL in Section 4.1 as the base model. Besides, another desirable property of the base model is to adapt quickly to a specific user to provide accurate prediction for this user after few epochs of fine tuning training. Thus, we also consider meta-learning techniques [11, 25] to train a base model that can be easily fine tuned to serve specific users. For these meta-learning approaches, we regard every user as an independent learning task and use the same way in Section 4.2.1 to get the sub-dataset O_u as the training data for each user u. In Section 5, we will show the empirical comparison of these difference choices of base model, where we find that with WL as the base model, LFT performs the best. Hence, in the rest of this paper, we consider WL as the default choice of the base model.

4.2.3 Ensemble Model. Since the proposed LFT requires additional fine-tuning training every time a user visits the recommendation platform, it requires more computational resources than conventional inference paradigm without additional fine-tuning training. Although we can control the consuming of time and computational resources by choosing appropriate fine-tuning epoch number and the size of neighbor user set, it may still not be feasible for platforms with limited computational resources and high concurrency of user visits. Hence, we also provide an ensemble version of the propose LFT, which finishes all the model training and stores the model during the training phase, and provides predictions without additional training during inference. Similar to existing local recommendation models [7, 18, 19], during training, we select anchor users and train anchor models for them by the proposed LFT. During inference, for each target user, we ensemble anchor models based on the relationship between the target user and anchor users. So, the key is: how to select anchor users so that they can cover as diverse user preference as possible?

Prior local recommendation models either randomly select anchor users [18] or select mainstream users to maximize the neighbor user coverage by anchor users [7], which tends to select mainstream users as anchor users. Both of these are not ideal for addressing mainstream bias. Hence, in this work, we propose a *similar-dissimilar anchor user selection algorithm* to adequately cover mainstream and non-mainstream preference. The proposed similar-dissimilar algorithm is a greedy algorithm, whose core idea is to select the user who is most similar to unselected users and dissimilar to already selected users in each iteration. Concretely, we define an anchor user set $\mathcal{A}$ beginning as an empty set. Then, we iteratively add users into $\mathcal{A}$ until reach a pre-defined set size. In each iteration, we select the user by:

$$\arg\min_{u \in \mathcal{U}-\mathcal{A}} \frac{1}{|\mathcal{U}-\mathcal{A}|} \sum_{v \in \mathcal{U}-\mathcal{A}} cos(\mathbf{p}_u, \mathbf{p}_v) - \lambda \frac{1}{|\mathcal{A}|} \sum_{v \in \mathcal{A}} cos(\mathbf{p}_u, \mathbf{p}_v), \tag{7}$$

where we calculate the Cosine similar between users by their calibrated distributions from Section 4.1.1; and λ controls the balance between similarity to unselected users and dissimilarity to selected users during the current anchor selection.

Table 3: Characteristics of three datasets.

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

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

$$\widehat{\mathbf{R}}_u = \frac{\sum_{v \in \mathcal{A}} cos(\mathbf{p}_u, \mathbf{p}_v)\phi_v(u)}{\sum_{v \in \mathcal{A}} cos(\mathbf{p}_u, \mathbf{p}_v)}. \tag{8}$$

We denote the ensemble version of LFT as EnLFT. Compared with the state-of-the-art local recommendation model LOCA [7], EnLFT has two major improvements. First, EnLFT adopts the more effective similar-dissimilar algorithm to maximize the coverage of different user preference for anchor user selection. Second, EnLFT can train more effective anchor models by fine tuning a global base model with precise neighbor users following LFT.

5 DEBIASING EXPERIMENTS

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

5.1 Experimental Setup

5.1.1 Data and Metric. We use three public datasets for the experiments: **ML1M** [14], **Yelp** [1], and **Epinions** [33]. For all datasets, we consider the ratings or reviews as positive feedback from users to items. Then, we randomly split each dataset into 70%, 10%, and 20% for training, validation, and testing. The details of these datasets are shown in Table 3. Since in Section 3.3.2 we show that DeepSVDD-based bias measuring approach is more effective than other approaches, in this section, we only report the results based on DeepSVDD-based approach for evaluating the mainstream bias. Concretely, we apply DeepSVDD-based approach to all three datasets to compute mainstream scores for users. Then, we sort users in non-descending order based on mainstream scores and divide them into five subgroups evenly. Last we report the average NDCG@20 for each subgroup to show the mainstream bias. **The goal of debiasing is to improve the average NDCG@20 for subgroups with low mainstream scores while preserving or even increasing the utility for subgroups with high mainstream scores at the same time.**

5.1.2 Methods. In the experiments, we adopt the variational autoencoder (**VAE**) [23] as the base and develop different debiasing models based on the VAE. By focusing on the same base model, we can isolate and directly analyze the effects of different debiasing algorithms. VAE is also a baseline method representing the vanilla recommendation model without any debiasing. For debiasing, we consider both global and local methods as introduced in Section 4. For global methods, we have our proposed Distribution Calibration (**DC**) and Weighted Loss (**WL**) methods. For local methods, we include the state-of-the-art Local Collaborative Autoencoders (**LOCA**) model [7] as a strong local method baseline that has demonstrated superior performance over other local recommendation models like LLORMA [18, 19] and GLSVD [8]. LOCA adopts VAE as its local

Table 4: Comparing overall utility and utility for different subgroups across methods and datasets.

	ML1M						Yelp						Epinion					
	NDCG	Subgroups of mainstream levels					NDCG	Subgroups of mainstream levels					NDCG	Subgroups of mainstream levels				
	@20	low	med-low	medium	med-high	high	@20	low	med-low	medium	med-high	high	@20	low	med-low	medium	med-high	high
VAE	.3153	.2092	.2642	.2832	.3368	.4831	.0893	.0586	.0711	.0812	.0966	.1387	.0823	.0601	.0712	.0758	.0908	.1136
DC	.3170	.2223	.2655	.2824	.3347	.4798	.0901	.0632	.0744	.0827	.0935	.1368	.0817	.0634	.0710	.0745	.0892	.1102
WL	.3190	.2315	.2724	.2838	.3318	.4755	.0903	.0632	.0763	.0820	.0938	.1365	.0820	.0657	.0738	.0752	.0880	.1076
LOCA	.3230	.2415	.2763	.2861	.3350	.4762	.0921	.0647	.0786	.0833	.0964	.1374	.0842	.0654	.0727	.0784	.0917	.1126
EnLFT	.3328	.2521	.2847	.2963	.3434	.4876	.0959	.0688	.0840	.0872	.0994	.1400	.0859	.0671	.0758	.0802	.0926	.1139
LFT	**.3372**	**.2549**	**.2876**	**.2982**	**.3492**	**.4963**	**.0984**	**.0706**	**.0860**	**.0917**	**.1035**	**.1403**	**.0876**	**.0697**	**.0779**	**.0825**	**.0930**	**.1150**
$\Delta_{LOCA}(\%)$	4.40**	5.55**	4.09**	4.23**	4.24**	4.22**	6.84**	9.12**	9.41**	10.08*	7.37*	2.11*	4.04**	6.57**	7.15**	5.23**	1.42	2.13*

component. Further, we consider our proposed Local Fine Tuning method with WL as the base model (**LFT**), and we also have the ensemble version of the LFT model (**EnLFT**).

5.1.3 Reproducibility. All models are implemented in PyTorch [26] and optimized by Adam algorithm [16]. For the baseline VAE and the VAE component in other models, we set one hidden layer of size 100. For all methods involving the Distribution Calibration step, including the DC model, LFT, and EnLFT, we set $\alpha = 0.7$. For the WL model, we set $\beta = 1.5$ for ML1M, $\beta = 3$ for Yelp and Epinions. For both LFT and EnLFT: we set the fine tuning epoch number as 30 for ML1M and Yelp, as 5 for Epinions; and we set the similarity threshold t for fetching neighbor users as 0.2 for ML1M, 0.01 for Yelp, and 0.05 for Epinions. For both ensemble models LOCA and EnLFT, we set the number of anchor models as 100. And we set $\lambda = 1.5$ in EnLFT. *All code and data can be found at https://github.com/Zziwei/Measuring-Mitigating-Mainstream-Bias.*

5.2 Compare Debiasing Performance

First, we compare different methods and answer three research questions: i) which method performs the best in terms of overall utility and bias mitigating? ii) how do the two proposed global methods perform compared to each other? and iii) how do the proposed local method and its ensemble version perform compared with the state-of-the-art local recommendation baseline? To answer these, we present the overall NDCG@20 and average NDCG@20 for 5 user subgroups of different mainstream levels for all methods and datasets in Table 4, where the best results of all metrics for each dataset are marked in bold, and the improvement rate from proposed LFT over the best baseline LOCA is exhibited as well (results are significant judged by paired t-test). The 'low' row represents the 20% users with lowest mainstream score, 'med-low' represents 20% to 40% users in the sorted user sequence, and so on for 40%-60% ('medium'), 60%-80% ('med-high'), and 80%-100% ('high').

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

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

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

5.3 Ablation Study

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

5.3.1 Selecting Neighbor users. In the proposed LFT, when we want to fine tune a local model for one user u, we need to fetch the neighbor users for u. The naive way is to directly calculate similarity on raw feedback records of users, such as Jaccard and Cosine similarity. We compare the performance of LFT by the naive way (by Jaccard and Cosine similarity) and the proposed method of calculating similarity on calibrated distribution of users (denoted as DC), and show the results in Table 5. For each dataset, the best results are marked in bold. We can see that LFT with the proposed neighbor user selecting method performs the best for all methods in terms of both overall utility and utility for each subgroup. The superior

Table 5: Compare different neighbor user selection methods.

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

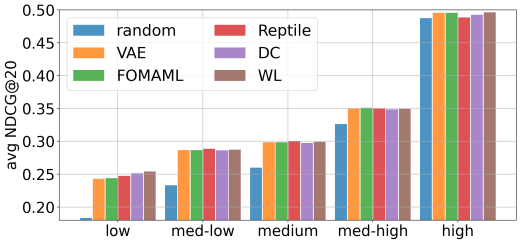

Figure 3: Compare different base model choices.

performance of the proposed method is because that calibrated distribution of users can help to capture the latent relationship between users. Due to this, the improvement for niche users is larger than for mainstream users because identifying neighbor users for niche users heavily relies on latent relationships.

5.3.2 Choosing base model. Next, we study the impact of different choices of base model in LFT. As discussed in Section 4.2.2, there are many different choices of base model, including: an ordinary recommendation model without debiasing, such as a VAE; a global debiasing model, such as the proposed DC or WL; and a meta-learning model which is supposed to be easily adapted to a specialized local model to predict accurately for a specific user, for which we adopt the FOMAML model from [11] and Reptile model from [25]. To train these meta-learning models, we consider every user as an independent task and use the same way in Section 4.2.1 to get the sub-dataset O_u for each user u as the training data for this task. Besides, we also include a random model as a baseline to show the importance of having a good base model. For ML1M dataset, the overall NDCG@20 for choices of random, VAE, FOMAML, Reptile, DC, and WL are 0.2979, 0.3347, 0.3351, 0.3349, 0.3353, and 0.3372, where base model of WL performs the best. Then, we show the average NDCG@20 for user subgroups by different choices of base model in Figure 3, from which we observe that base model of WL performs the best for niche users, while FOMAML and Reptile perform similarly as WL for subgroups with higher mainstream level. Hence, we can conclude that choosing WL as the base model produces the best result, and adopting meta-learning techniques does not significantly help to improve the performance. Moreover, we find salient difference between results of random model and other choices, showing that choosing a well-trained base model is important, especially for niche users.

5.3.3 Selecting anchor users in ensemble model. Last, we investigate the impact of different anchor user selection methods in EnLFT. In the baseline LOCA [7], mainstream users are selected as anchor users to maximize the neighbor user coverage, which has limited coverage for niche users. Instead, we propose the similar-dissimilar method to cover both niche and mainstream users to maximize the coverage of user preference. Besides, we also include the random method to randomly select anchor users. We compare these methods with other settings the same for EnLFT, and results are listed in Table 6, where 'LOCA' represents EnLFT with anchor user selection method from LOCA, 'EnLFT' represents EnLFT with proposed similar-dissimilar method, and 'random' represents EnLFT with random selection method. This table shows that with the proposed similar-dissimilar method, EnLFT performs the best. Compared with LOCA method, we find that the major improvements are from the niche users (the first three subgroups with lowest mainstream levels), and utility for mainstream users are very similar for these two methods. This is because the proposed similar-dissimilar method improve the coverage for niche users compared with the method from LOCA. Besides, we see that utility for mainstream users by EnLFT with anchor user selection method from LOCA (the 'LOCA' rows in Table 6) is higher than the original LOCA (the 'LOCA' columns in Table 4), which validates the effectiveness of proposed LFT to learn powerful anchor models.

Table 6: Compare different anchor user selection methods.

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

5.4 Hyper-parameter Study

Last, we study how two hyper-parameters in LFT – the number of training epochs for local fine tuning; and the similarity threshold for fetching neighbor users – influence the performance. The detailed results are referred to Appendix A.

6 CONCLUSION AND FUTURE WORK

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

ACKNOWLEDGMENTS

This work is in part supported by NSF grant IIS-1939716 and Amazon Research Awards.

REFERENCES

[1] 2021. Yelp dataset. https://www.yelp.com/dataset

[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*. 42–46.

[3] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing popularity bias in recommender systems with personalized re-ranking. In *The thirty-second international flairs conference*.

[4] Irad Ben-Gal. 2005. Outlier detection. In *Data mining and knowledge discovery handbook*. Springer, 131–146.

[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] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 93–104.

[7] Minjin Choi, Yoonki Jeong, Joonseok Lee, and Jongwuk Lee. 2021. Local Collaborative Autoencoders. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 734–742.

[8] Evangelia Christakopoulou and George Karypis. 2018. Local latent space models for top-n recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1235–1243.

[9] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. 39–46.

[10] 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.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. PMLR, 1126–1135.

[12] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. 2020. Fairness-aware explainable recommendation over knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 69–78.

[13] 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.

[14] 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.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

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

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

[18] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local low-rank matrix approximation. In *International conference on machine learning*. PMLR, 82–90.

[19] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. 2016. LLORMA: Local low-rank matrix approximation. (2016).

[20] Jae-woong Lee, Seongmin Park, and Jongwuk Lee. 2021. Dual Unbiased Recommender Learning for Implicit Feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1647–1651.

[21] Roger Zhe Li, Julián Urbano, and Alan Hanjalic. 2021. Leave No User Behind: Towards Improving the Utility of Recommender Systems for Non-mainstream Users. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 103–111.

[22] 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.

[23] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.

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

[25] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

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

[28] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *International conference on machine learning*. PMLR, 4393–4402.

[29] Yuta Saito. 2020. Unbiased Pairwise Learning from Biased Implicit Feedback. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*. 5–12.

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

[31] Markus Schedl and Christine Bauer. 2019. Online music listening culture of kids and adolescents: Listening analysis and music recommendation tailored to the young. *arXiv preprint arXiv:1912.11564* (2019).

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[33] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: discerning multi-faceted trust in a connected world. In *Proceedings of the 5th WSDM*.

[34] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[35] Tianxin Wei, Fuli Feng, Jiawei Chen, Chufeng Shi, Ziwei Wu, Jinfeng Yi, and Xiangnan He. 2020. Model-Agnostic Counterfactual Reasoning for Eliminating Popularity Bias in Recommender System. *arXiv preprint arXiv:2010.15363* (2020).

[36] Shuo Yang, Lu Liu, and Min Xu. 2021. Free lunch for few-shot learning: Distribution calibration. *arXiv preprint arXiv:2101.06395* (2021).

[37] Sirui Yao and Bert Huang. 2017. Beyond parity: Fairness objectives for collaborative filtering. *arXiv preprint arXiv:1705.08804* (2017).

[38] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H Chi. 2021. A Model of Two Tales: Dual Transfer Learning Framework for Improved Long-tail Item Recommendation. In *Proceedings of the Web Conference 2021*. 2220–2231.

[39] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. *arXiv preprint arXiv:2105.06067* (2021).

[40] Xing Zhao, Ziwei Zhu, Majid Alfifi, and James Caverlee. 2020. Addressing the Target Customer Distortion Problem in Recommender Systems. In *Proceedings of The Web Conference 2020*. 2969–2975.

[41] Ziwei Zhu, Yun He, Yin Zhang, and James Caverlee. 2020. Unbiased Implicit Recommendation and Propensity Estimation via Combinational Joint Learning. In *Fourteenth ACM Conference on Recommender Systems*. 551–556.

[42] Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee. 2021. Popularity-Opportunity Bias in Collaborative Filtering. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 85–93.

[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.

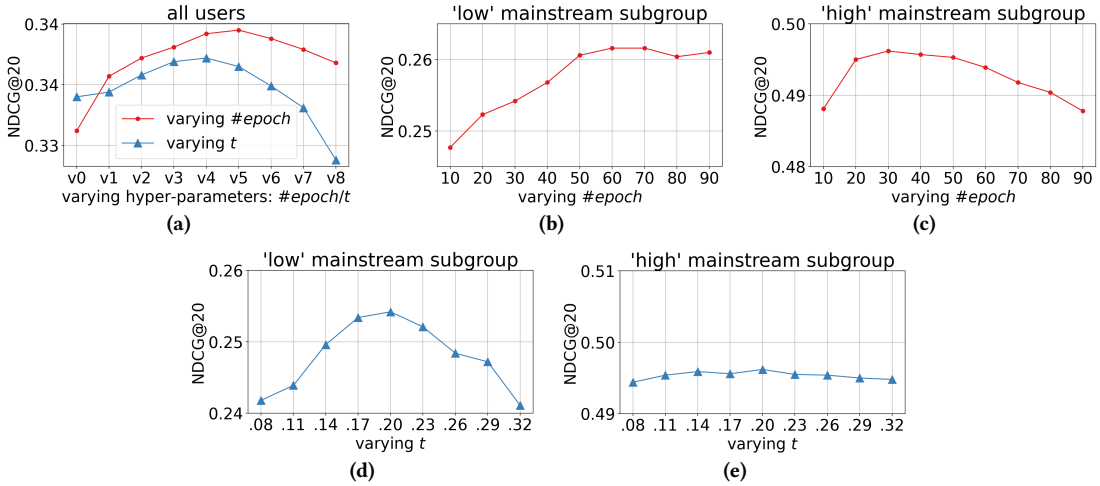

Figure 4: Hyper-parameter study: (a) how NDCG@20 changes with varying *#epoch* and t; (b) and (c) how NDCG@20 for users of 'low' and 'high' mainstream levels changes with varying *#epoch*; (d) and (e) how NDCG@20 for users of 'low' and 'high' mainstream levels changes with varying t.

A HYPER-PARAMETER STUDY

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

Number of epochs

Here, we run LFT on ML1M dataset with the number of epochs for local fine tuning (denoted as *#epoch*) varying in {10, 20, 30, 40, 60, 70, 80, 90} and other settings the same as in Section 5.2. The overall NDCG@20 are shown as the red line in Figure 4a, where v0 to v8 represent 10 to 90. We can observe that the utility first increases then decreases with increasing epochs. Then, we also show how the average NDCG@20 changes for the user subgroup of 'low' mainstream level in Figure 4b and for the subgroup of 'high' mainstream level in Figure 4c. It shows that with increasing training epochs, utility for niche users first increases and then converges, which may decrease with more epochs. However, the utility for mainstream users first increases and turns to decrease quickly, which is because the base model already provides high accuracy and fewer local fine tuning epochs are needed for mainstream users. Hence, the next step of the local fine tuning research is to personalize the training epochs for different users.

Similarity threshold

Then, we study the impact of the similarity threshold t when we fetch neighbor users. Lower t leads to more neighbor users are selected in LFT. We vary t from 0.08 to 0.32 with step 0.03 and show how the overall NDCG@20 changes as the blue line in Figure 4a. In this figure, v0 to v8 represent 0.08 to 0.32. We can see that with increasing t, NDCG@20 first increases then decreases and reaches peak at 0.20. We also show how the average NDCG@20 changes for the user subgroup of 'low' mainstream level in Figure 4d and for the subgroup of 'high' mainstream level in Figure 4e. From these two figures, we find that with increasing t, NDCG@20 for niche users first increases then decreases, while for mainstream users, the utility does not change notably. It is because niche users have small numbers of neighbor users, and it is more challenging to find these neighbor users. So, utility for niche users is more sensitive to the choice of t. But for mainstream users, there are a large number of similar users with high similarity, thus LFT produces strong performance for mainstream users and is not sensitive to t within certain value range.