Popularity Bias in Dynamic Recommendation

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

ABSTRACT

Popularity bias is a long-standing challenge in recommender systems: popular items are overly recommended at the expense of less popular items that users may be interested in being under-recommended. Such a bias exerts detrimental impact on both users and item providers, and many efforts have been dedicated to studying and solving such a bias. However, most existing works situate the popularity bias in a static setting, where the bias is analyzed only for a single round of recommendation with logged data. These works fail to take account of the dynamic nature of real-world recommendation process, leaving several important research questions unanswered: how does the popularity bias evolve in a dynamic scenario? what are the impacts of unique factors in a dynamic recommendation process on the bias? and how to debias in this long-term dynamic process? In this work, we investigate the popularity bias in dynamic recommendation and aim to tackle these research gaps. Concretely, we conduct an empirical study by simulation experiments to analyze popularity bias in the dynamic scenario and propose a dynamic debiasing strategy and a novel False Positive Correction method utilizing false positive signals to debias, which show effective performance in extensive experiments.

CCS CONCEPTS

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

KEYWORDS

popularity bias; dynamic recommendation

ACM Reference Format:
Ziwei Zhu, Yun He, Xing Zhao, James Caverlee. 2021. Popularity Bias in Dynamic Recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467376

1 INTRODUCTION

Popularity bias is a long-standing challenge in recommender systems [4, 19, 22, 23, 28, 32]. In essence, *popularity bias* means that popular items are overly exposed in recommendations at the expense of less popular items that users may find interesting. This bias can hurt both users and items. Users are worse off since the system can only learn a biased view of their true preferences. Popular (though not necessarily "better") items can become even more popular, while less popular items lose their deserved feedback (via clicks or views) and economic gains due to this bias.

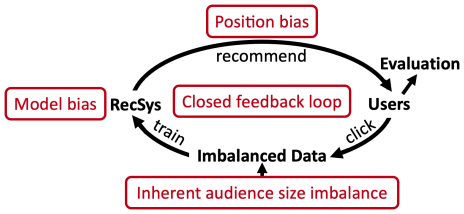

Figure 1: Dynamic recommendation: a closed loop where users generate feedback; this feedback train models; models recommend items to users; and then the loop continues.

Most existing efforts to study popularity bias adopt a static setting [4, 19, 22, 23, 28, 32]. That is, a recommendation model is trained over an offline dataset, and popularity bias is analyzed by conducting a single round of recommendation. While these studies have highlighted the prevalence of popularity bias, there is a significant research gap in our understanding of the dynamics of this bias, the factors impacting popularity bias and its evolution, and the efficacy of methods to mitigate this bias under real-world assumptions of system evolution. Hence, this paper proposes a framework for the study of popularity bias in dynamic recommendation.

Dynamic recommendation [7, 13, 17, 24] can be viewed as a closed loop illustrated in Figure 1. Users interact with the system through a set of actions (e.g., clicks, views, ratings); this user-feedback data is then used to train a recommendation model; the trained model is used to recommend new items to users; and then the loop continues. While there are many opportunities for bias to affect this dynamic recommendation process, we identify four key factors that may impact popularity bias and its evolution: (i) *inherent audience size imbalance*: users may like some items more than others (even with a purely bias-free random recommender), meaning that a few items may have very large audience sizes while the majority have small ones; (ii) *model bias*: the recommendation model itself may amplify any imbalances in the data it ingests for training; (iii) *position bias*: once the model makes recommendations, the top-ranked items are more likely to be examined by users; and (iv) *closed feedback loop*: since the cycle repeats, the feedback data collected from recommendations made by the current model will impact the training of future versions of the model, potentially accumulating the bias.

With these factors in mind, we investigate popularity bias through the following questions: First, how does popularity bias evolve in dynamic recommendation? Second, what impact do these four factors have on the bias? Are some more critical than others for the bias? And if so, the last question is how can we mitigate the popularity bias by counteracting the critical factors? To the best of

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467376

our knowledge, this is the first work to comprehensively explore popularity bias in dynamic recommendation.

In this paper, we follow the recently introduced popularity-opportunity bias [32], a formalization of popularity bias based on the concept of equal opportunity. Compared to conventional notions of popularity bias based on statistical parity [4, 22, 28] – which compares the number of times of being recommended across popular and unpopular items – popularity-opportunity bias measures *whether popular and unpopular items receive clicks (or other engagement metrics) proportional to their true audience sizes*. In other words, do popular and unpopular items receive similar *true positive rates*? By comparing engagement rather than just counts of recommendation, this popularity-opportunity bias is directly aligned with user satisfaction and economic gains of item providers.

From this perspective, we undertake a three-part study:

i) First, we conduct a comprehensive empirical study by simulation experiments to investigate how the popularity bias evolves in dynamic recommendation, and how the four factors impact the bias. We find that inherent audience size imbalance and model bias are the main drivers of popularity bias, while position bias and the closed feedback loop further exacerbate the bias. Besides, we also compare two different negative sampling strategies to show how the careful design of negative sampling can benefit recommendation in the context of popularity bias.

ii) Second, we explore methods to mitigate popularity bias in dynamic recommendation. We show how to adapt existing debiasing methods proposed in a static setting to the dynamic scenario. We further propose a model-agnostic False Positive Correction (FPC) method for debiasing, which can be integrated with other debiasing methods for further performance improvements.

iii) Finally, we report on extensive experiments to show the effectiveness of the proposed dynamic debiasing method compared with the static strategy, and we also illustrate the salient performance improvement brought by the proposed FPC. With similar debiasing effect achieved, on average, 15% more clicks are generated after applying the proposed FPC. Moreover, in contrast to prior studies that find a trade-off between popularity bias and recommendation utility [23, 32], we show that higher recommendation utility can be achieved in concert with reductions in popularity bias.

2 RELATED WORK

Popularity bias is a long-standing problem and has been widely studied. Some methods adopt an in-processing strategy to mitigate bias by modifying the model itself [2, 22, 28], while others adopt a post-processing strategy to mitigate bias by modifying the predictions of the model [3, 23, 32]. One of the most typical approaches is to debias by assigning weights inversely proportional to item popularity in the loss of a model [16, 22]. By this, popular and unpopular items can be balanced during training and more even recommendations can be generated. Similar to this idea, Steck [23] proposes to directly re-scale the predicted scores based on popularity to promote unpopular items and prevent popular items from over-recommendation. The scaling weights are also inversely proportional to item popularity. Besides, a recent work [28] investigates the bias from the perspective of causal inference and propose a counterfactual reasoning method to debias. Note that all of these works evaluate popularity bias by comparing how often items are recommended without regard for the ground truth of user-item matching. To address this gap, [32] proposes the concept of popularity-opportunity bias which compares the true positive rate of items to evaluate the bias, and a popularity compensation method is proposed, which explicitly considers user-item matching.

All of these existing studies of popularity bias are based on the static recommendation task [4, 19, 22, 23, 28, 32], which is a simplified sub-component of the real-world dynamic recommendation process [7, 13, 17, 24]. It is an open question how popularity bias evolves, what are the impacts of different factors in dynamic recommendation on the bias, and how to mitigate the bias in the dynamic scenario. Hence, in this work, we aim to investigate the popularity bias in dynamic recommendation to tackle this research gap.

Although to our best knowledge, there is no existing work studying popularity bias in dynamic recommendation, there indeed are some works investigating fairness [17], diversity [5, 13], or algorithm confounding [7, 11, 21, 24] in a dynamic manner (sharing a similar philosophy as this work). For example, Morik at al. [17] study item group fairness in a dynamic recommendation process and propose a fairness enhancement algorithm FairCo to improve fairness over time. This FairCo method can also work to reduce the popularity bias we are studying in this work if we consider each item as a group, and so we include FairCo as a baseline in our debiasing experiments in Section 6.

Algorithm 1: Dynamic Recommendation Process

```
Bootstrap: Randomly show K items to each user and
  collect initial clicks D and train the first model ψ by D;
for t = 1 : T do
    Recommend K items to the current user u_t by ψ;
    Collect new clicks and add them to D;
    if t%L == 0 then
        Retrain ψ by D;
```

3 PROBLEM FORMALIZATION

In this section, we formalize dynamic recommendation and popularity bias, and also introduce four major factors in dynamic recommendation inducing this bias.

3.1 Formalizing Dynamic Recommendation

Suppose we have an online platform that provides recommendations, such as recommending movies, jobs, or songs. The dynamic recommendation process is: i) every time a user visits the platform, we present a ranked list of items based on a personalized recommendation model that is learned based on the user's historical feedback; and (ii) periodically, we update the personalized recommendation model with the newly collected user feedback. To bootstrap a new (cold-start) user, we learn the user's preference through some non-personalized approaches, such as randomly showing some items to the user and collecting feedback.

Concretely, we assume there are a set of users $\mathcal{U} = \{1, 2, \ldots, N\}$ and a set of items $\mathcal{I} = \{1, 2, \ldots, M\}$ in the system. Every user has a subset of items the user likes (unknown to the system), and we

define the total number of matched users who like the item i as the audience size of i, denoted as A_i. At the beginning (a bootstrap step), for each user, the system randomly exposes K items to bootstrap the user and thus collects initial user-item clicks $\mathcal{D}$. Based on the initial data $\mathcal{D}$, the first recommendation model ψ, such as a matrix factorization (MF) [15], is trained. Then, as users coming to the system one by one, the system uses the up-to-date model to provide K ranked items as recommendations and collect new user-item clicks. After every L user visits, the system retrains the recommendation model with all clicks collected up to now. This dynamic recommendation process is summarized in Algorithm 1.

3.2 Formalizing Popularity Bias

Then, a key question is: how to formalize popularity bias? Many previous works view popularity bias [4, 22, 28] from the perspective of statistical parity [8, 30]. That is, they consider the difference of how many times items are recommended as bias, without considering the ground-truth of user-item matching. In contrast, we adopt the recently introduced popularity-opportunity bias [32], a formalization of popularity bias based on the concept of equal opportunity. Popularity-opportunity bias evaluates *whether popular and unpopular items receive clicks (or other engagement metrics) proportional to their true audience sizes*. In other words, do popular and unpopular items receive similar *true positive rates*? Compared to statistical parity, this formalization of popularity bias is directly aligned with user satisfaction and economic gains of item providers [1].

At iteration t in the dynamic recommendation process, to quantify the popularity bias, we need to first calculate the true positive rate for each item. Suppose item i has received C_i^t clicks in total from the beginning to iteration t, the true positive rate for i is $TPR_i = C_i^t / A_i$. Then, we can use the Gini Coefficient [6, 26] to measure the inequality in true positive rates corresponding to item popularity at iteration t:

$$Gini_t = \frac{\sum_{i \in \mathcal{I}} (2i - M - 1) TPR_i}{M \sum_{i \in \mathcal{I}} TPR_i}, \tag{1}$$

where items are indexed from 1 to M in audience size non-descending order ($A_i \leq A_{(i+1)}$). We use $-1 \leq Gini_t \leq 1$ to quantify the popularity bias [2]: a small $|Gini_t|$ indicates a low bias; $Gini_t > 0$ represents that true positive rate is positively correlated to item audience size; and $Gini_t < 0$ represents that the true positive rate is negatively correlated to audience size (reversed popularity bias).

3.3 Factors Impacting Popularity Bias

One of the goals in this paper is to deepen our understanding of factors that may produce and worsen this bias. As introduced in Figure 1, we focus on four major factors:

1. Inherent audience size imbalance. Items inherently have different audience sizes, and this imbalance can potentially lead to popularity bias. It has been observed that the audience size for items usually follows a long-tail distribution [19], meaning that a few items have a very large audience size while the majority have small ones. This inherent imbalance will result in imbalanced engagement data (like clicks), even if every item is equally recommended by a bias-free random recommender.

2. Model bias. A recommendation model tends to rank an item with more clicks in the training data higher than an item with fewer clicks, even if the ground truth is that the user equally likes both of them [32]. This is a common deficiency of collaborative filtering based algorithms and directly leads to popularity bias if the training data is imbalanced.

3. Position bias. This is a well-known issue in ranking scenarios: the probability to examine items at top ranking positions is higher than at lower positions [12, 17]. If inherent audience size imbalance and model bias exist, position bias can further exacerbate the popularity bias because a matched popular item being recommended and ranked at a top position is more likely to receive a click than a matched but unpopular item also being recommended but ranked at a lower position.

4. Closed feedback loop. Finally, we consider the phenomenon that future models are trained by the click data collected from the recommendations by previous models [11, 21, 24]. In this way, the popularity bias generated in the past can accumulate, leading to more bias in subsequent models as the feedback loop continues.

All of these four factors can potentially play important roles in generating and exacerbating the bias in dynamic recommendation. But which factor is the most critical for the bias? Are some the main drivers of the bias, and some less essential? Our hypothesis is that the inherent audience size imbalance and model bias are the main sources of the popularity bias, while position bias and closed feedback loop exacerbate the bias only when the other two factors exist. We empirically examine this in the next section.

Table 1: Dataset statistics.

	#user	#item	density	*Gini*(audience size)
ML1M	1,000	3,406	0.0657	0.6394
Ciao	1,000	2,410	0.0696	0.4444

4 EMPIRICAL STUDY

In this section, we conduct an empirical study to study how the popularity bias evolves in dynamic recommendation; the impacts of the four discussed bias factors on the bias; moreover, we also compare two different negative sampling strategies to show how the careful design of negative sampling can benefit the recommendation.

4.1 Setup

Due to the challenges of running repeatable experiments over live platforms, we follow the widely-adopted approach [5, 7, 11, 13, 17] of conducting experiments to simulate the dynamic recommendation process in Section 3.1. There are two key challenges for this experiment: i) how to obtain complete ground truth of user-item relevance? and ii) how to simulate the user click behavior given recommendations?

To tackle the first challenge, we follow [13, 17] to generate semi-synthetic data based on real-world user-item interaction datasets. Concretely, we adopt MovieLens 1M (**ML1M**) [9] and **Ciao** [25] as base datasets and randomly keep 1,000 users in each dataset for the

[1] Comparing the popularity-opportunity bias and conventional notion of popularity bias, the main difference is how to measure the bias. The debiasing methods for conventional popularity bias can still be applied to reduce popularity-opportunity bias.
[2] In this paper, we conduct simulation experiments with semi-synthetic data to study the popularity bias in dynamic recommendation, in which audience size of items are known. In practice, we need to estimate the audience size based on observed clicks, such as inverse propensity scoring based methods from [17, 29].

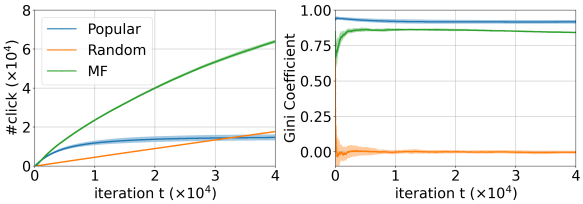

Figure 2: Results of three methods on ML1M.

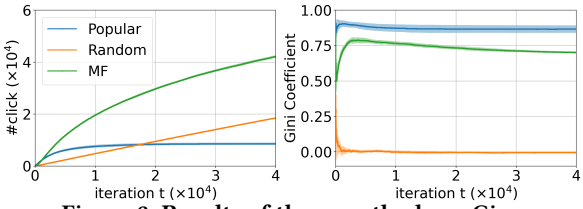

Figure 3: Results of three methods on Ciao.

experimental efficiency. Then, we run the matrix factorization (MF) model [15] to complete the original datasets to provide the ground truth of user-item relevance. The detailed statistics of the semi-synthetic datasets are shown in Table 1, where we also calculate the Gini Coefficient of the item audience size in each dataset to quantify the inherent audience size imbalance. Furthermore, by modifying the data generation process, for each base dataset, we also generate 4 variants with different inherent audience size imbalance levels to investigate the impact of inherent audience size imbalance.

Then, we conduct experiment to simulate the process in Algorithm 1. Concretely, in this empirical study, we recommend $K = 20$ items to users at each iteration; run the simulation for $T = 40,000$ iterations; and retrain the recommendation model after every $L = 50$ iterations. For the second challenge of modeling user click behavior, we follow [17] and model the click behavior based on the position bias of $\delta_k = 1/log_2(1+k)$ to determine whether user u will examine item i at position k. We observe a click only if the user examines and likes the recommended item. More details about the experimental setup can be found in Appendix A.

In this empirical study, we implement multiple different recommendation methods to study the impact of different factors (details are introduced in following sections). After every $L = 50$ iterations, we retrain the recommendation models for 15 epochs. A special operation of these models is that *the negative samples are drawn from items being recommended but unclicked for each user*. By this, we can achieve higher recommendation utility and lower popularity bias compared with the vanilla negative sampling strategy that draws negative samples from all unclicked items. We justify this choice in Section 4.4. All experiments are repeated for 10 times.

4.2 Evolution of Popularity Bias

The first question to investigate is: how does popularity bias evolve in dynamic recommendation? Here, we use the basic MF as the recommendation model, and the dynamic recommendation process involves all four bias factors introduced in Section 3.3. Results for ML1M and Ciao are shown in Figure 2 and Figure 3 respectively, where for comparison, we also include a **Popular** method to rank items only based on the number of observed clicks so far, and a **Random** method to randomly rank items. At iteration t, we report the number of cumulative clicks up to now as the metric evaluating

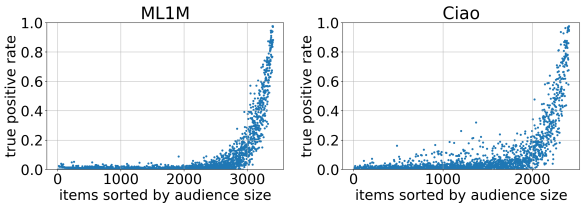

Figure 4: After all 40,000 iteration, the true positive rates of items for ML1M (left) and Ciao (right).

recommendation utility, and we report *Gini* defined in Equation 1 for measuring the popularity bias.

First, we observe in the left figures in Figure 2 and 3 that MF produces significantly higher recommendation utilities than the Popular and Random methods. Moreover, the number of cumulative clicks first increases then converges for the Popular method, and after some iterations the Random method can even outperform the Popular method on both datasets, which illustrates the harm of popularity bias. Second, we observe in the right figures that: i) the Random method produces near zero *Gini* because random ranking has no bias for either popular or unpopular items; ii) the Popular method results in high *Gini* values throughout the whole experiment because the provided recommendations directly follow popularity; and iii) MF first results in a rapid increase in *Gini* and then maintains this high *Gini* value to the end of the experiment.

Moreover, we also visualize the bias by plotting the true positive rates (the number of received clicks divided by the audience size) of items after 40,000 iterations in Figure 4, where each dot represents one item, and items are sorted in non-descending order by their audience sizes along x-axis. We can see that for both datasets, the true positive rate follows a long-tail distribution: only a few items with largest audience size have high true positive rates while majority items receive near 0 true positive rates. That is, most items have extremely low chances of being recommended to users who would like to click them.

While it is not surprising that we observe popularity bias in dynamic recommendation, it is surprising that a traditional MF (which is also the foundation of many more advanced models [10, 18, 27]) boosts the bias so fast, and the produced bias nearly equals that in a heavily-biased Popular method. Beyond static studies [32] of popularity bias that have observed its prevalence, we observe that this bias grows rapidly and maintains at a high level, indicating the need for special interventions to mitigate this issue.

4.3 Impacts of Four Bias Factors

After showing the evolving pattern of the popularity bias in dynamic recommendation, we next investigate the impacts of the four factors introduced in Section 3.3.

4.3.1 Impact of Position Bias. First, we study the position bias by comparing two dynamic recommendation experiments: one using the same vanilla MF (denoted as **w/ PB**) as the one adopted in Section 4.2, which does not explicitly counteract position bias; and another using an unbiased MF with position bias removed (denoted as **w/o PB**). For the unbiased MF, we adopt the inverse propensity scoring based loss from [20], where we use $p_k = 1/log_2(1+k)$ as the propensity estimation for a click observed at position k. All other experiment settings are the same as in Section 4.2.

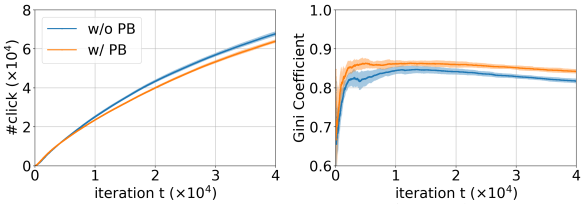

Figure 5: Compare experiments with (w/ PB) and without position bias (w/o PB) on ML1M.

Table 2: Utility (#click) and popularity bias (*Gini*) of different recommendation methods after 40,000 iterations.

	ML1M			Ciao		
	w/ PB (BetterNS)	w/o PB	VanillaNS	w/ PB (BetterNS)	w/o PB	VanillaNS
#click	63967	67815	20723	42134	41984	13461
Gini	0.8425	0.8177	0.9419	0.7416	0.7043	0.9226

Results for ML1M are shown in Figure 5, from which we can see that with the position bias removed (the w/o PB): more clicks are observed; lower popularity bias is measured; and *Gini* reaches its peak value slower. This implies that position bias does exacerbate popularity bias. We also list click counts and *Gini* after 40,000 iterations for both methods and datasets in Table 2. It shows that with position bias removed, more clicks and lower *Gini* are observed for both datasets, demonstrating *the positive impact of position bias on intensifying popularity bias*. Due to this, in the following experiments, we will use the unbiased MF [20] with inverse propensity scoring based loss as the default recommendation model.

4.3.2 Impact of Closed Feedback Loop. Then, we conduct a new experiment that removes the closed feedback loop by: i) not using the clicks collected from personalized recommendation (by MF) as training data; ii) after every L personalized recommendation iterations, adding a random recommendation step to generate random rankings to L randomly selected users and collect random-recommendation clicks; and iii) only using the random-recommendation clicks to train the personalized MF model. In this way, the MF is trained by data purely from random recommendations and will not be influenced by previous personalized recommendation models, i.e., breaking the closed feedback loop. We evaluate the popularity bias only for personalized recommendations by MF. We denote this experiment setup as **w/o CFL**, and denote the experiment with closed feedback loop (the same as w/o PB in Section 4.3.1) as **w/ CFL**.

Because the MF in w/o CFL is trained by click data from random recommendations, whose data size is much smaller than that in w/ CFL. Hence, it is unfair and not informative to compare the utility between w/ CFL and w/o CFL, and we only show the popularity bias comparison in Figure 6. From the figures we can see that compared to w/ CFL, in w/o CFL, the popularity bias also keeps increasing but at a much slower speed. This indicates that *the closed feedback loop does exacerbate the popularity bias*. Without the closed feedback loop, the popularity bias is only from the current recommendation model, and there is no accumulated bias from previous models. Notice that *Gini* in w/o CFL still keeps increasing. This is because the training data gets increasingly denser, making the model bias increases as we will justify in the following section.

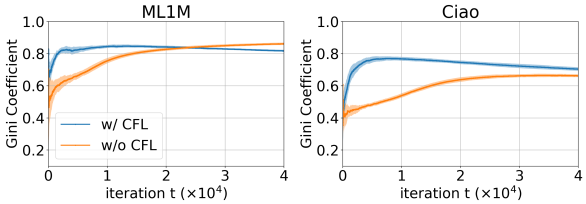

Figure 6: Compare popularity bias in experiments with (w/ CFL) and without closed feedback loop (w/o CFL).

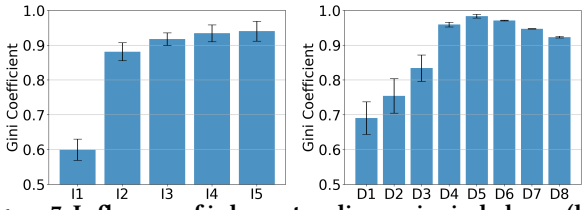

Figure 7: Influence of inherent audience size imbalance (left) and training data density (right) on model bias.

4.3.3 Impact of Model Bias. Next, we study the impact of model bias on popularity bias. In Figure 6, the w/o CFL experiment does not contain position bias nor the closed feedback loop, but we can still observe the bias. This is because *the model bias and inherent audience size imbalance are the main sources of popularity bias*. As long as these two factors exist, popularity bias will be generated. To better understand the impact of model bias, in this section, we conduct a series of static recommendation experiments without position bias or the closed feedback loop.

First, we aim to study how the inherent audience size imbalance influences model bias. Beside the semi-synthetic dataset ML1M we already used, we also generate 4 variants with different levels of inherent audience size imbalance by modifying the data generation process. Hence, now we have 5 datasets with increasing levels of inherent audience size imbalance, denoted as I_1, I_2, I_3, I_4, I_5, and the corresponding *Gini* of audience size are 0.37, 0.45, 0.51, 0.57, 0.64 (higher value means severer imbalance). Then, for each of the 5 datasets, we uniformly and randomly select positive user-item pairs to be a training dataset of density 0.2% and leave remaining user-item pairs as the testing dataset. We train conventional MF by training datasets, and evaluate the bias on testing datasets. Result is shown in the left of Figure 7, where we see that with severer imbalance, the model bias increases.

Next, we aim to study how training data density influences the model bias. In this case, we use the same ML1M dataset with *Gini* of audience size 0.64, but generate 8 training datasets with different densities by uniformly and randomly selecting positive user-item pairs with different probabilities in ML1M. The 8 training datasets has increasing densities, denoted as $D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8$, and the corresponding densities are 0.01%, 0.05%, 0.1%, 0.2%, 0.4%, 0.8%, 1.6%, 3.2%. We train MF by training datasets, and evaluate the bias on remaining data. Results are presented in the right of Figure 7, where we can see that with training datasets getting denser, the model bias first increases but then deceases. This may be because with denser data, both model bias and ability to learn user-item relevance increase. And after a threshold, the ability to learn user-item relevance surpasses the effect of model bias, leading to lower popularity bias observed. However in practice, dense training data is rare and the model bias usually plays a major role.

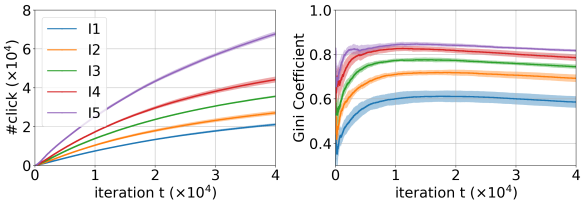

Figure 8: Utility (left) and popularity bias (right) for datasets with different inherent audience size imbalance.

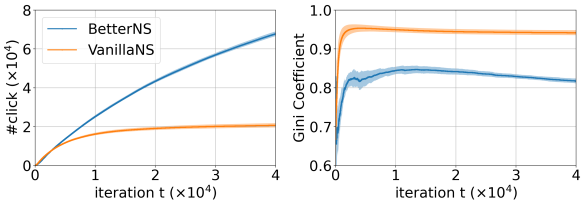

Figure 9: Comparison between two different negative sampling strategies on ML1M.

4.3.4 Impact of Inherent Audience Size Imbalance. The inherent audience size imbalance exerts its influence on popularity bias mainly through model bias, which we already exhibit by static experiments in Figure 7. But how does inherent audience size imbalance impact dynamic recommendation? To answer this, we run dynamic recommendation experiments for the 5 datasets with different levels of inherent audience size imbalance, where all other experiment settings are the same as the w/o PB experiment in Section 4.3.1. Results are presented in Figure 8. The left figure demonstrates that with severer inherent audience size imbalance, a system can receive more user clicks. This is because popular items can be more easily recognized and correctly recommended to matched users to receive large amounts of clicks in imbalanced datasets. On the other hand, the right part in Figure 8 shows that with a severer inherent audience size imbalance, higher popularity bias is generated.

4.3.5 Summary. In sum, we find that the inherent audience size imbalance and model bias are the main sources of popularity bias, which can produce the bias without existence of other factors; while position bias and closed feedback loop can intensify the bias when inherent audience size imbalance and model bias exist. Moreover, we also find that higher training data density and greater imbalance can increase the effect of model bias. Then, the following challenge is how to effectively reduce the bias given these observations? We investigate this in Section 5.

4.4 Compare Two Negative Sampling Strategies

Last, we compare two different negative sampling strategies to show how a careful design of negative sampling benefits the recommendation. The vanilla negative sampling strategy (denoted as **VanillaNS**) adopted in most works [10, 18, 27] considers all unclicked items as candidate negative samples, and uniformly and randomly selects negative training data from them. However, for a user, her preference towards most of the unclicked items are indeed unknown because these items are not exposed to her before. So, a better way is to only consider the recommended but unclicked items as potential negative samples (denoted as **BetterNS**) as we do in this empirical study. Because recommended but unclicked items are more likely the items users dislike, two advantages of BetterNS are: on one hand, with more precise negative samples, higher utility is expected to be achieved; on the other hand, with negative samples drawn from recommended items, popular items being mistakenly over-recommended to unmatched users previously can be corrected by involving those recommended but unclicked popular items as negative samples during model training, leading to bias reduced.

To verify the benefits of BetterNS, we conduct dynamic recommendation experiments with these two negative sampling strategies and show results on ML1M in Figure 9. From the results, we can see that experiment with BetterNS has significantly more clicks and lower popularity bias compared with the one with VanillaNS. We also list the click counts and *Gini* after 40,000 iterations for both methods and both datasets in Table 2, showing the same results and supporting our conclusion that negative sampling from recommended but unclicked items can bring higher utility and lower popularity bias. Moreover, because BetterNS has the ability to correct the overestimation towards popular items being over-recommended previously, after certain number of iterations, the popularity bias can gradually decrease as shown in Figure 2, 3, 5, 6, 8, and 9. This is also why after certain iteration, the w/ CFL experiment has lower *Gini* than w/o CFL experiment in Figure 6.

5 DEBIASING APPROACHES

While we have demonstrated the evolution of popularity bias in dynamic recommendation, how can we begin to counteract it? As we discussed in Section 3.3 and empirically studied in Section 4.3, model bias and inherent audience size imbalance are the two most essential factors. Of the two, practitioners can directly impact the degree of model bias, whereas the other factor is often an inherent aspect of a system. And indeed, by reducing model bias we have seen the Random method in Figure 2 and 3 that there will be no or low popularity bias even if the other three factors exist. Thus, in this section, we focus on how to mitigate the popularity bias in dynamic recommendation by reducing model bias.

5.1 By Dynamically Reducing Model Bias

First, we show how to adapt existing debiasing methods proposed in a static setting to the dynamic scenario. The key idea is to gradually increase the debiasing strength weight of existing models.

Most existing works reduce popularity bias in a static setting by reducing model bias [22, 23, 28, 32]. For example, [23] proposes a re-scaling method (denoted as **Scale**) to reduce the bias by re-scaling the outputs of recommendation models as a post-processing step. Concretely, the re-scaled score for a user-item pair (u, i) is:

$$\widehat{r}_{u,i}^{(scaled)} = \widehat{r}_{u,i}^{(model)} / (C_i)^{\alpha}, \tag{2}$$

where $\widehat{r}_{u,i}^{(model)}$ is the output predicted score from a recommendation model; C_i is the number of clicks the item has in training data; α is the hyper-parameter to control the debiasing strength, higher α means more strength for debiasing; and $\widehat{r}_{u,i}^{(scaled)}$ is the re-scaled score used for final ranking.

In static recommendation, this debiasing strength hyper-parameter α is a constant. However, as we see in Section 4.3.3, model bias is proportional to training data density and imbalance. Hence, it is not feasible to set a constant α for the entire lifetime of dynamic recommendation. Instead, we propose to gradually increase α from 0 with

an increasing step Δ through the dynamic recommendation process. That is, starting from 0, we increase α by Δ for each iteration. By this, we can gradually increase the debiasing strength so that the model can perform effectively when the density and imbalance of the training data grow. Beyond the specific Scale method [23], most existing popularity debiasing methods [22, 23, 28, 32] include such a debiasing strength weight α, meaning that we can apply them dynamically in the same way by involving the increasing step Δ.

5.2 By False Positive Correction

We further propose a model-agnostic False Positive Correction (FPC) method for debiasing, which can be integrated with other debiasing methods for further performance improvements.

In our experiments we have found that naively adapting existing static methods for mitigating popularity bias to dynamic recommendation can lead to large drops in recommendation utility. One reason is that these methods only utilize the true positive signals (i.e. the clicks) to debias without considering the false positive signals (i.e. items being recommended but unclicked). Because in a high popularity bias case, popular items can be incorrectly over-recommended to unmatched users (generating false positive signals), the false positive signal is correlated with the popularity bias. If we could correct the recommendations based on these false positive signals, we could lower the popularity bias. This is also why in Section 4.4, using the recommended but unclicked items (i.e., false positive signals) as negative samples for training the model can lead to higher utility and also lower popularity bias.

We propose the False Positive Correction (denoted as **FPC**) method to correct the predicted scores based on false positive signals in a probabilistic way. More specifically, suppose we are going to predict the relevance $\widehat{r}_{u,i}$ between user u and item i for determining ranking, and we already have a predicted score $\widehat{r}_{u,i}^{(model)}$ from a recommendation model. Assume that item i has been recommended to user u for F times before and has never been clicked, and we record the ranking positions of these F times of recommendation as $\{k_1, k_2, \ldots, k_F\}$. So, the false positive signals can be denoted as $\{c_{k_1} = 0, c_{k_2} = 0, \ldots, c_{k_F} = 0\}$, where c_k represents whether user u clicks the item i ranked at position k. We further denote the probability that u likes i as $\theta_{u,i}$, i.e., $P(r_{u,i} = 1) = \theta_{u,i}$; and denote the probability of examining an item at ranking position k as δ_k, i.e., $P(e_k = 1) = \delta_k$. Then, we can calculate the conditional probability that u likes i given the false positive signals as:

$$
\begin{aligned}
&P(r_{u,i} = 1 | c_{k_1} = 0, \ldots, c_{k_F} = 0) \\
&\quad = 1 - \frac{P(r_{u,i} = 0, c_{k_1} = 0, \ldots, c_{k_F} = 0)}{P(c_{k_1} = 0, \ldots, c_{k_F} = 0)} \\
&\quad = 1 - \frac{P(r_{u,i} = 0)}{\prod_{f=1}^{F} P(c_{k_f} = 0)} \\
&\quad = 1 - \frac{P(r_{u,i} = 0)}{\prod_{f=1}^{F} (P(e_{k_f} = 0) + P(e_{k_f} = 1, r_{u,i} = 0))} \\
&\quad = 1 - \frac{1 - \theta_{u,i}}{\prod_{f=1}^{F} (1 - \delta_{k_f} \theta_{u,i})},
\end{aligned}
\tag{3}
$$

where line 3 substitutes $P(r_{u,i} = 0) = P(r_{u,i} = 0, c_{k_1} = 0, \ldots, c_{k_F} = 0)$ in the numerator because if $r_{u,i} = 0$ no click can be observed, and substitutes $\prod_{f=1}^{F} P(c_{k_f} = 0) = P(c_{k_1} = 0, \ldots, c_{k_F} = 0)$ in the denominator because the click observations are independent between recommendation iterations; line 4 decomposes $P(c_{k_f} = 0)$ into $P(e_{k_f} = 0) + P(e_{k_f} = 1, r_{u,i} = 0)$ in the denominator because a recommended but unclicked item can happen when the user does not examine the item or the item is examined but the user dislikes it. $\theta_{u,i}$ is unknown and needs to be estimated. And we can use the prediction $\widehat{r}_{u,i}^{(model)}$ from a recommendation model as $\theta_{u,i}$. Therefore, every time when we need a relevance score of (u, i) for ranking, we use Equation 3 with $\theta_{u,i} = \widehat{r}_{u,i}^{(model)}$ and $\delta_{k_f} = 1/log_2(1 + k_f)$ (as how we model the position bias) to correct predictions from a recommendation model by false positive signals.

Yet, one disadvantage of the proposed FPC is that if we use the prediction $\widehat{r}_{u,i}^{(model)}$ from a biased recommendation model, such as an MF, as $\theta_{u,i}$, FPC is still vulnerable to the model bias. Thus, we propose to use the predictions from a debiased model, such as the Scale method introduced in Section 5.1 or other popularity debiasing models introduced in Section 2, to be $\theta_{u,i}$ in Equation 3. In this case, we can take full advantage of both true positive signals and false positive signals to counteract the popularity bias.

6 DEBIASING EXPERIMENTS

In this section, we conduct experiments to show how the popularity bias is mitigated in dynamic recommendation by dynamically reducing model bias and the proposed FPC method.

6.1 Setup

The basic experiment setup is the same as Section 4.1. To validate the effectiveness of our proposed FPC, for the recommendation models, we include many different types of debiasing methods for comparison.

6.1.1 Recommendation Models. The basic recommendation model is the **MF** with an inverse propensity scoring based loss to counteract position bias as used in Section 4. For the debiasing models, first, we consider existing static debiasing methods, including: **Scale**[23] as introduced in Equation 2; **Weight** [22] which assigns weights to items in the loss; **MACR** [28] which removes the causal effect of item and user popularity; and **PC** [32] which adds compensation to the predicted scores of unpopular items.

As discussed in Section 5.1, to debias in dynamic recommendation, we need to apply these debiasing methods dynamically. Hence, we have the dynamic versions of these debiasing methods by gradually increasing the debiasing strength hyper-parameter (such as the α of Scale as shown in Equation 2) from 0 by a tune-able increasing step Δ as experiments continue. We denote the dynamic versions of these four models as **DScale**, **DWeight**, **DMACR**, and **DPC**.

Besides, **FairCo** proposed in [17] is a debiasing method designed for the dynamic scenario, which calculates the true positive rate gap between each item to the best served item and adds the calculated gap to the predicted scores to reduce the imbalance.

Last, we have the proposed **FPC** method to debias based on false positive signals. And we also combine the proposed FPC with other debiasing methods to reduce the popularity bias utilizing both true positive and false positive signals, denoted as **FPC-DScale**, **FPC-DWeight**, **FPC-DMACR**, **FPC-DPC**, and **FPC-FairCo**.

Table 3: #click and *Gini* results of static and corresponding dynamic debiasing models.

	ML1M		Ciao	
	#click	*Gini*	#click	*Gini*
MF	67816	0.8177	41984	0.7043
Scale	66630	0.6378	44146	0.4544
Δ(DScale)	+2015	-0.0368	+3267	-0.0137
Weight	62044	0.6099	42385	0.4557
Δ(DWeight)	+1843	-0.0129	+1103	-0.0210
MACR	52161	0.6079	31037	0.4375
Δ(DMACR)	+1286	-0.0047	+1425	-0.0154
PC	58917	0.6131	41591	0.4306
Δ(DPC)	+2001	-0.0008	+1668	-0.0096

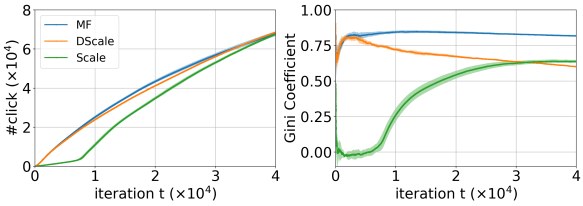

Figure 10: Compare the static debiasing method Scale and its dynamic version DScale.

Furthermore, Multi-Armed Bandit (MAB) algorithms which are designed to balance exploration-exploitation in dynamic recommendation can also improve the recommendation chances for unpopular items (mainly through exploration). Hence, we also conduct experiments to study their effects in Appendix C and show that MAB algorithms cannot effectively address the studied popularity bias.

6.1.2 Reproducibility. Experiments are implemented by Tensorflow [1] and optimized by Adam algorithm [14]. The foundation for all models is an MF with the inverse propensity scoring based loss to counteract position bias [20]. All models draw negative samples from recommended but unclicked items. For every debiasing method (except FPC), there is a debiasing strength weight or the increasing step Δ, we tune these hyper-parameters so that all methods achieve similar bias level, and we compare the click counts to compare the performance. All experiments are repeated for 10 times. Code is available at https://github.com/Zziwei/Popularity-Bias-in-Dynamic-Recommendation.

6.2 Empirical Results

In the following experiments, we study the effect of dynamic debiasing compared with static ones; the effect of the proposed FPC; and the effect of integrating FPC with other debiasing methods.

6.2.1 How do dynamic debiasing methods perform compared with static ones? To show the advantage of dynamic debiasing over static approaches, we conduct experiments on ML1M and Ciao with four existing static popularity debiasing models – Scale, Weight, MACR, and PC, compared with their dynamic versions – DScale, DWeight, DMACR, and DPC. We tune all models so that similar popularity bias level is achieved after 40,000 iterations of experiment, and compare the number of clicks in Table 3. Because similar level of bias is generated, the more clicks are observed the more effective the model is. From the table, we can see that the dynamic versions of debiasing models can produce more clicks than their static versions, demonstrating the effectiveness of the dynamic strategy.

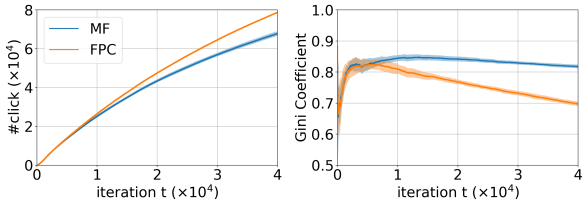

Figure 11: Compare the proposed FPC with MF on ML1M.

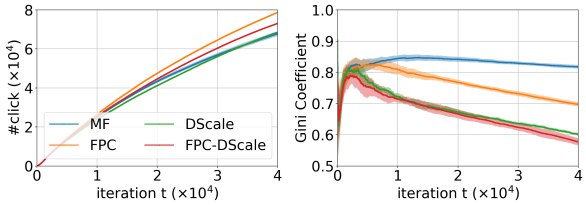

Figure 12: Compare MF, FPC, DScale, and FPC-DScale on ML1M. (Medium debiasing level)

We also plot how the utility and bias change for the basic MF, the static Scale, and the dynamic DScale in Figure 10 (other pairs of methods show similar patterns). The right figure shows that comparing to MF, both Scale and DScale reduce the bias. However, they show very different patterns: DScale increases the bias at the beginning then keeps decreasing the bias; while Scale keeps increasing the bias and eventually surpasses DScale. This is because as the experiment continues, density and imbalance in training data increases, resulting in higher model bias and more debiasing strength needed. So, dynamically increasing the debiasing strength following the increasing bias can produce better results.

6.2.2 What is the effect of FPC alone? Then, we plot the results of MF and FPC on ML1M in Figure 11. The right figure shows that the proposed FPC increases the bias at the beginning, but then keeps decreasing the popularity bias. Compared to MF, the reduction of bias metric *Gini* is significant. On the other hand, the left figure shows that FPC can even increase the number of clicks during the experiment compared with MF. This is because by mitigating the popularity bias, popular items are prevented to be over-recommended to unmatched users and more unpopular items can be accurately recommended to matched users to receive clicks. Hence, it is a win-win scenario that both users and item providers can benefit from. Moreover, after all 40,000 iterations, we found that: for ML1M dataset, FPC gets 78,763 clicks and 0.6973 *Gini*; for Ciao, FPC gets 54,813 clicks and 0.5665 *Gini*. Compared with results of MF in Table 3, it shows the effect of FPC that more clicks and lower bias are generated.

6.2.3 What is the effect of integrating FPC with other debiasing methods? Although, the proposed FPC can reduce the bias and improve the utility, as discussed in Section 5.2, FPC only utilizes the false positive signals without considering the true positive signals as existing debiasing models do. Hence, combining FPC with other debiasing methods is expected to achieve even better performance. To justify this, we conduct experiments to compare the dynamic debiasing models with their variants integrated with FPC. Results of MF, FPC, DScale, and FPC-DScale are shown in Figure 12. The right figure demonstrates that DScale and FPC-DScale are able to reduce the bias lower than FPC (there is no tune-able hyper-parameter

Table 4: Difference between results before and after integrating FPC to five debiasing methods.

	ML1M						Ciao					
Debiasing Level	Low($Gini$≈0.66)		Medium($Gini$≈0.60)		High($Gini$≈0.53)		Low($Gini$≈0.46)		Medium($Gini$≈0.42)		High($Gini$≈0.38)	
	#click	*Gini*	#click	*Gini*	#click	*Gini*	#click	*Gini*	#click	*Gini*	#click	*Gini*
Δ(FPC-DScale)	+6973 (+10%)	-.020	+4500 (+7%)	-.023	+1781 (+3%)	-.013	+5159 (+11%)	-.006	+4623 (+10%)	-.005	+2056 (+4%)	-.033
Δ(FPC-DWeight)	+5167 (+7%)	-.013	+3847 (+6%)	-.002	+747 (+12%)	-.017	+2982 (+7%)	-.007	+2824 (+6%)	-.001	+2625 (+6%)	+.002
Δ(FPC-DMACR)	+8876 (+16%)	-.023	+9963 (+19%)	-.007	+8399 (+16%)	-.008	+6241 (+19%)	-.004	+6531 (+20%)	+.003	+7858 (+26%)	-.008
Δ(FPC-DPC)	+13311 (+21%)	-.016	+12945 (+21%)	-.018	+11887 (+21%)	-.021	+9331 (+21%)	-.002	+8187 (+19%)	-.008	+7921 (+18%)	-.006
Δ(FPC-FairCo)	+13307 (+22%)	-.001	+14137 (+25%)	+.005	+11729 (+21%)	-.026	+9255 (+23%)	-.009	+8816 (+22%)	-.007	+9745 (+26%)	+.005

in FPC to adjust the debiasing strength). And we see that DScale and FPC-DScale produce similar level of bias, however, the left figure shows that FPC-DScale generates significantly more clicks, illustrating the advantage of integrating FPC with DScale.

It shows similar results when we combine FPC with other debiasing methods, and we list the comparison results before and after integrating FPC after all 40,000 iterations in Table 4. In this table, we show the changes of #click and *Gini* after integrating FPC compared with original debiasing methods. For example, Δ(FPC-DScale) shows changes after integrating FPC to DScale. And we include five dynamic debiasing methods – DScale, DWeight, DMACR, DPC, FairCo. Moreover, for each method, we tune the model to conduct three experiments with different debiasing levels. For the same debiasing level, all methods are tuned to produce similar bias and the average *Gini* of each debiasing level is shown in Table 4. From the table, we find that with the same level of debiasing achieved, the FPC integrated models can generate significantly more clicks than original models: after combining FPC, 14.33% and 15.88% more clicks are received on average for ML1M and Ciao respectively.

Detailed results of all methods are shown in Appendix B. Comparing debiasing methods with MF, an interesting observation is: when we set the debiasing level as low or medium, more clicks can be generated by debiasing models than MF which produces high bias; however, when we set a high debiasing strength, some of debiasing methods produce fewer clicks than MF. This is because when high debiasing is enforced, unpopular items are promoted but popular items are overly depressed, leading to fewer clicks from users to these popular items. Thus, it is still important to balance between utility and debiasing in practice.

7 CONCLUSION

In this work, we investigate popularity bias in dynamic recommendation. We first conduct an empirical study by simulation experiments to show how the bias evolves in the dynamic process and the impacts of four bias factors on the bias. Then, we propose to dynamically debias and also propose the FPC method to debias utilizing false positive signals. Last, by extensive experiments, we empirically validate the effectiveness of the proposed dynamic debiasing strategy and the proposed FPC algorithm.

ACKNOWLEDGMENTS

This work is in part supported by NSF grant 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 *RecSys*.
[3] H Abdollahpouri, R Burke, and B Mobasher. 2019. Managing popularity bias in recommender systems with personalized re-ranking. *arXiv:1901.07555* (2019).
[4] H Abdollahpouri, M Mansoury, R Burke, and B Mobasher. 2019. The unfairness of popularity bias in recommendation. *arXiv:1907.13286* (2019).
[5] Guy Aridor, Duarte Goncalves, and Shan Sikdar. 2020. Deconstructing the Filter Bubble: User Decision-Making and Recommender Systems. In *RecSys*.
[6] Malcolm C Brown. 1994. Using Gini-style indices to evaluate the spatial patterns of health practitioners: theoretical considerations and an application based on Alberta data. *Social science & medicine* (1994).
[7] A JB Chaney, B M Stewart, and B E Engelhardt. 2018. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *RecSys*.
[8] Moritz Hardt, Eric Price, and Nathan Srebro. 2016. Equality of opportunity in supervised learning. In *NeurIPS*.
[9] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *TIIS* (2015).
[10] X He, L Liao, H Zhang, L Nie, X Hu, and T Chua. 2017. Neural collaborative filtering. In *WWW*.
[11] Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli. 2019. Degenerate feedback loops in recommender systems. In *AAAI/ACM Conference on AI, Ethics, and Society*.
[12] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *WSDM*.
[13] Sami Khenissi, Boujelbene Mariem, and Olfa Nasraoui. 2020. Theoretical Modeling of the Iterative Properties of User Discovery in a Collaborative Filtering Recommender System. In *RecSys*.
[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).
[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* (2009).
[16] Dawen Liang, Laurent Charlin, and David M Blei. 2016. Causal inference for recommendation. In *Causation: Foundation to Application, Workshop at UAI. AUAI*.
[17] Marco Morik, Ashudeep Singh, Jessica Hong, and Thorsten Joachims. 2020. Controlling fairness and bias in dynamic learning-to-rank. In *SIGIR*.
[18] Wei Niu, James Caverlee, and Haokai Lu. 2018. Neural personalized ranking for image recommendation. In *WSDM*.
[19] Yoon-Joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *RecSys*.
[20] Y Saito, S Yaginuma, Y Nishino, H Sakata, and K Nakata. 2020. Unbiased recommender learning from missing-not-at-random implicit feedback. In *WSDM*.
[21] Ayan Sinha, David F Gleich, and Karthik Ramani. 2016. Deconvolving feedback loops in recommender systems. In *NeurIPS*.
[22] Harald Steck. 2011. Item popularity and recommendation accuracy. In *RecSys*.
[23] Harald Steck. 2019. Collaborative filtering via high-dimensional regression. *arXiv:1904.13033* (2019).
[24] Wenlong Sun, Sami Khenissi, Olfa Nasraoui, and Patrick Shafto. 2019. Debiasing the human-recommender system feedback loop in collaborative filtering. In *Companion Proceedings of World Wide Web Conference*.
[25] J Tang, H Gao, and H Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In *WSDM*.
[26] Adam Wagstaff, Pierella Paci, and Eddy Van Doorslaer. 1991. On the measurement of inequalities in health. *Social science & medicine* (1991).
[27] X Wang, X He, M Wang, F Feng, and T Chua. 2019. Neural graph collaborative filtering. In *SIGIR*.
[28] 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:2010.15363* (2020).
[29] L Yang, Y Cui, Y Xuan, C Wang, S Belongie, and D Estrin. 2018. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In *RecSys*.
[30] Brian Hu Zhang, B Lemoine, and M Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *AAAI/ACM Conference on AI, Ethics, and Society*.
[31] X Zhao, W Zhang, and J Wang. 2013. Interactive collaborative filtering. In *CIKM*.
[32] Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee. 2021. Popularity-Opportunity Bias in Collaborative Filtering. In *WSDM*.

A EXPERIMENTAL SETUP FOR DYNAMIC RECOMMENDATION

Due to the challenges of running repeatable experiments over live platforms, we follow the widely-adopted approach [5, 7, 11, 13, 17] of conducting experiments to simulate the dynamic recommendation process. There are two key challenges for this experiment: i) how to obtain complete ground truth of user-item relevance? and ii) how to simulate the user click behavior given recommendations?

To tackle the first challenge, we follow [13, 17] to generate semi-synthetic data based on real-world user-item interaction datasets. Concretely, we use MovieLens 1M (**ML1M**) [9] and **Ciao** [25] as base datasets and regard all ratings as positive signals. We randomly keep 1,000 users in each dataset for the experimental efficiency. Then, we run a matrix factorization (MF) model [15] with 100 latent dimensions and cross entropy loss to generate a relevance probabilistic matrix for each dataset. Next, we obtain a binary relevance matrix $\mathbf{R}$ by drawing a Bernoulli sample for each user-item pair given the generated relevance probability matrix. We consider the generated binary matrices as the ground-truth user-item relevance. The detailed statistics of the semi-synthetic datasets are shown in Table 1, where we also calculate the Gini Coefficient of the item audience size in each dataset to quantify the inherent audience size imbalance. Furthermore, by modifying the Bernoulli sampling for different items, we also generate 4 variants with different inherent audience size imbalance levels for each dataset to investigate the impact of inherent audience size imbalance.

Then, following Algorithm 1, we first randomly recommend K items to each user to collect initial clicks, then simulate the dynamic recommendation process for T iterations. Every L iterations, we retrain the recommendation model with all the clicks collected up to that time. At each iteration t, we randomly pick one user u_t and show the top K items that the user did not click before as recommendations based on the up-to-date recommendation model. Since we aim to recommend new items to users, items already clicked by users before will not be considered as recommendation candidates. This experiment process is a flexible framework that allows different settings. In this empirical study, we set $K = 20$, $T = 40,000$, and $L = 50$.

But how can we model user click behavior? Following [17], given a recommendation list, we iterate it from the top position ($k = 1$) to the lowest position ($k = 20$): for the item i at position k, we draw a Bernoulli sample e_k based on the position bias calculated by $\delta_k = 1/log_2(1+k)$ to determine whether user u will examine item i at position k (i.e., whether $e_k = 1$). We observe a click only if $e_k = 1$ and $\mathbf{R}_{u,i} = 1$. Every time we observe new clicks, we add them to the training dataset $\mathcal{D}$. Initial clicks from the random recommendations follow the same process.

B DETAILED EXPERIMENT RESULTS

In Table 4, we only show the difference between results before and after integrating FPC. So, here, we list the numbers of clicks observed and *Gini* for all methods after 40,000 iterations in Table 5 for ML1M and Table 6 for Ciao. Note that there is no notion of debiasing level for MF and proposed FPC, so the three columns of different debiasing levels are the same for these two methods. And for other methods, they are tuned to achieve similar level of bias under the same debiasing level.

Table 5: Results of different recommendation models with three different debiasing strength levels on ML1M.

	Low debiasing		Medium debiasing		High debiasing	
	#click	*Gini*	#click	*Gini*	#click	*Gini*
MF	67816	0.8177	67816	0.8177	67816	0.8177
FPC	78763	0.6973	78763	0.6973	78763	0.6973
ϵ-greedy	62531	0.6539	58132	0.5994	55595	0.5474
UCB	61711	0.6287	59882	0.6019	56814	0.5442
DScale	71308	0.6722	68645	0.6010	63546	0.5280
FPC-DScale	78281	0.6526	73145	0.5778	65327	0.5153
DWeighted	69209	0.6869	63887	0.5970	60894	0.5514
FPC-DWeight	74376	0.6742	67734	0.5953	61641	0.5344
DMACR	54985	0.6561	53447	0.6032	52387	0.5550
FPC-DMACR	63861	0.6335	63410	0.5963	60786	0.5473
DPC	63308	0.6582	60918	0.6122	57959	0.5302
FPC-DPC	76619	0.6422	73863	0.5946	69846	0.5097
FairCo	61619	0.6546	57208	0.5919	55189	0.5517
FPC-FairCo	74926	0.6536	71345	0.5968	66918	0.5254

Table 6: Results of different recommendation models with three different debiasing strength levels on Ciao.

	Low debiasing		Medium debiasing		High debiasing	
	#click	*Gini*	#click	*Gini*	#click	*Gini*
MF	41984	0.7043	41984	0.7043	41984	0.7043
FPC	54813	0.5665	54813	0.5665	54813	0.5665
ϵ-greedy	41380	0.4654	40239	0.4222	39204	0.3872
UCB	41415	0.4691	39081	0.3781	39990	0.4152
DScale	48218	0.4717	47413	0.4307	46807	0.4096
FPC-DScale	53377	0.4662	52036	0.4261	48863	0.3769
DWeighted	44492	0.4596	43488	0.4347	42113	0.3897
FPC-DWeight	47474	0.4524	46312	0.4336	44738	0.3904
DMACR	33031	0.4567	32462	0.4221	30695	0.4123
FPC-DMACR	39272	0.4527	38993	0.4252	38553	0.4040
DPC	43814	0.4516	43259	0.4210	43055	0.3932
FPC-DPC	53145	0.4539	51446	0.4127	50976	0.3876
FairCo	40932	0.4608	40225	0.4376	37318	0.3820
FPC-FairCo	50187	0.4522	49041	0.4308	47063	0.3869

C EXPERIMENTS OF MAB MODELS

In addition to the debiasing models we have conducted experiments with in Section 6, another category of algorithms that need to be considered for dynamic recommendation is the Multi-Armed Bandit (MAB) model [31]. MAB models are designed to balance exploration and exploitation in dynamic recommendation process. Exploitation means the decision of recommendations follows the up-to-date knowledge about the system, such as following the predictions from a learned MF model, which may be biased due to the fact that the up-to-date knowledge only reveals partial information about the system. Exploration is to generate recommendations to obtain unexplored knowledge about the system, such as recommending less recommended items, which can help reduce the bias. Thus, MAB algorithms are supposed to reduce the popularity bias (mainly

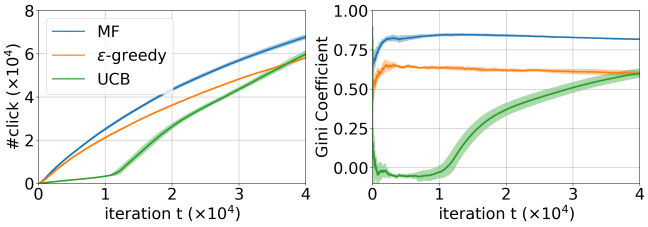

Figure 13: Results of ϵ-greedy and UCB on ML1M. (Medium debiasing level)

through the exploration). Hence, in this work, we also experiment with two MAB algorithms for comparison and study whether MAB algorithms can mitigate the investigated popularity bias in dynamic recommendation.

Concretely, we adopt the **ϵ-greedy** [31] and **UCB** [31] as the MAB baselines. There is a exploration-exploitation trade-off hyper-parameter in ϵ-greedy and UCB. We tune them to achieve three different levels of debiasing effects, and we show the numbers of clicks observed and *Gini* after 40,000 iterations in Table 5 for ML1M and Table 6 for Ciao. From these results, we can see that ϵ-greedy and UCB do reduce the popularity bias compared with MF, while the numbers of clicks are decreased as well. Besides, we also find that with similar levels of debiasing effect achieved, ϵ-greedy and UCB generate fewer clicks than most of other debiasing methods.

To further understand the effect of these two methods, we also plot the results on ML1M in Figure 13. From the right figure, we can see that neither of ϵ-greedy and UCB generates decreasing curve as other debiasing methods in Figure 12. Because ϵ-greedy exposes random items to users with a probability ϵ, it can only lower the overall popularity bias due to the exploration by random recommendations, but it cannot keep reducing the bias over time. On the other hand, because UCB tends to recommend items with high uncertainty, it tries to balance all items (low bias) at the first few iterations. However, after the uncertain period, UCB tends to follow the popularity bias again and increase the bias. As a result, although they help to mitigate the issue, MAB algorithms are not good choices to address the popularity bias.