Information Retrieval Research 1 (2025) 3–27 Submitted 10/24; Revised 12/24; Published 02/25

On the challenges of studying bias in Recommender Systems: The effect of data characteristics and algorithm configuration

Savvina Daniil S.DANIIL@CWI.NL
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands

Manel Slokom M.SLOKOM@CWI.NL
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands

Mirjam Cuper MIRJAM.CUPER@KB.NL
National Library of the Netherlands
The Hague, The Netherlands

Cynthia C.S. Liem C.C.S.LIEM@TUDELFT.NL
Delft University of Technology
Delft, The Netherlands

Jacco van Ossenbruggen JACCO.VAN.OSSENBRUGGEN@VU.NL
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands

Laura Hollink L.HOLLINK@CWI.NL
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands

Editor: Shangsong Liang, Djoerd Hiemstra

Abstract

Statements on the propagation of bias by recommender systems are often hard to verify or falsify. Research on bias tends to draw from a small pool of publicly available datasets and is therefore bound by their specific properties. Additionally, implementation choices are often not explicitly described or motivated in research, while they may have an effect on bias propagation. In this paper, we explore the challenges of measuring and reporting popularity bias. We showcase the impact of data properties and algorithm configurations on popularity bias by combining real and synthetic data with well known recommender systems frameworks. First, we identify data characteristics that might impact popularity bias, and explore their presence in a set of available online datasets. Accordingly, we generate various datasets that combine these characteristics. Second, we locate algorithm configurations that vary across implementations in literature. We evaluate popularity bias for a number of datasets, three real and five synthetic, and configurations, and offer insights on their joint effect. We find that, depending on the data characteristics, various configurations of the algorithms examined can lead to different conclusions regarding the propagation of popularity bias. These results motivate the need for explicitly addressing algorithmic configuration and data properties when reporting and interpreting bias in recommender systems.

Keywords: Recommender Systems, Bias, Data Synthesis, Reproducibility

 WWW: https://irrj.org DOI: 10.54195/irrj.19607

1 Introduction

Recommender systems are commonly used as a tool to encode taste based on the information available, be it user history or metadata. The wide use of recommender systems necessitates critical reflection on the issues that may arise when we allow automation to dictate our exposure to information. Specifically, bias in recommender systems is a topic of interest within the scholarly community. Bias is a complex term that can refer to various types of biases associated with interactions between users and items in a given system (Chen et al., 2023).

Many studies have focused on measuring the phenomenon of *popularity bias* in collaborative filtering systems (Klimashevskaia et al., 2024; Ahanger et al., 2022; Elahi et al., 2021; Yalcin, 2021; Abdollahpouri, 2020; Zhao et al., 2022). Despite this large research effort to track and mitigate popularity bias, there is no univocal message regarding why and when it occurs. In previous work, we found that studies that measure popularity bias propagated by commonly used algorithms on benchmark datasets report varying, sometimes contradicting results (Daniil et al., 2024). This observation raises questions; is popularity bias sensitive to properties of the system that do not receive sufficient attention? Why is a seemingly simple phenomenon so hard to study? Additionally to the evaluation strategy which was the focus of our previous work, we hypothesize that two factors that also complicate bias measuring and reporting are data characteristics and algorithm configuration.

Data characteristics Benchmark datasets are useful for academic research, as they allow researchers to evaluate their hypotheses and compare their proposed debiasing methods. However, their consistent use raises concerns that relate to the dependence on the domain and source they were constructed from, and the potentiality for blind spots that stem from outdated rating behaviour. Most importantly, by reporting on types of bias on only a small set of publicly available datasets, researchers are restricted by their specific characteristics. This specificity limits the scope of research, and obfuscates the process of examining causality. In other words, it is not trivial to conclude whether the propagation of bias or lack thereof is a result of the respective algorithm's functionality, or of certain intricate details of the user-item interactions within these datasets. Data synthesis based on assumptions can potentially assist with shedding light on the aforementioned blind spots and gaining new insights into bias in recommender systems.

Algorithm configuration Insufficient reporting of algorithm configuration leads to a reproducibility problem within research on recommender systems. Studies have shown that papers published in big conferences often do not disclose sufficient information for replication and verification (Ferrari Dacrema et al., 2021). This issue is also relevant in the bias discussion. Even relatively simple algorithmic approaches, such as neighbour-based ones, are constructed using hyperparameters and implementation choices that might affect whether bias propagation is observed. The RecSys community proposes a set of evaluation frameworks to promote reproducibility[1], but there are important configuration differences between them that often go unmentioned (Bellogín and Said, 2021). Testing the effect of algorithm configuration can be a means of reporting on bias in a comprehensive manner.

1. `https://github.com/ACMRecSys/recsys-evaluation-frameworks`

In this paper, we experiment with data characteristics and algorithm configurations and observe the effect on popularity bias. First, we look into *data characteristics* that might have an impact on popularity bias given a rating prediction and top-10 recommendation task, a common setup among recent studies on popularity bias (Naghiaei et al., 2022; Kowald et al., 2020; Kowald and Lacic, 2022). Specifically, we delve into the relation between popularity and rating, as well as the preferences of users with large profiles. We analyze these characteristics for three real datasets: a subset of Book-Crossing (Ziegler et al., 2005) constructed by Naghiaei et al. (2022), MovieLens1M (Harper and Konstan, 2015) and Epinion (Massa and Avesani, 2007). Additionally, we form a set of data scenarios by tweaking and combining these characteristics. For each scenario, we generate a corresponding synthetic dataset of ratings, based on the interactions from the subset of Book-Crossing. Second, for a set of algorithms we identify *configuration choices* that may impact whether or not popularity bias is observed. We study three widely used algorithms, UserKNN, Biased Matrix Factorisation and Deep Matrix Factorization, implemented in three frameworks recommended by ACM RecSys, LensKit (Ekstrand, 2020), Cornac (Salah et al., 2020), and Elliot (Anelli et al., 2021). We perform the recommendation process for the subset of Book-Crossing, MovieLens1M and Epinion, as well as each synthetic dataset with varied algorithm configuration choices. We apply commonly used popularity bias metrics to evaluate the recommended lists, as well as RMSE and NDCG@10 to estimate the performance of the algorithms when it comes to rating prediction and ranking.

Our results show that whether popularity bias is observed, and to what extent, depends on much more than the algorithm that was used, or the domain in which a study is carried out: all algorithms that we tested were seen to strongly propagate popularity bias in some experimental settings, while not propagating popularity bias in other settings. The same is true for datasets: all datasets led to bias in some settings and not in others. The results further clarify that whether or not popularity bias is observed depends on, firstly, specific (often unreported) configuration and implementation details of algorithms. The implication of that is that the choice for a certain framework largely impacts the outcome of a study. It also depends, secondly, on characteristics of the dataset that is studied; specifically, the relationship between rating and popularity, as well as the preferences of users with large profiles are crucial when it comes to popularity bias propagation. Finally, it is especially the interplay between algorithm configuration and dataset characteristics that determines whether popularity bias will be observed or not.

The contributions of this paper are as follows:

- a systematic investigation into the effect of data characteristics on popularity bias, by comparing results on three commonly used datasets as well as five synthetic datasets for which we control the properties.

- a systematic investigation into the effect of implementation differences, by comparing results of algorithm configurations as well as non-configurable implementation differences in well known frameworks.

- for the more interpretable algorithms, we highlight notable results among the many, to give insights into why certain combinations of dataset characteristics and algorithm configurations lead to popularity bias.

With this work, we wish to contribute to the field by highlighting and disentangling the challenges in studying popularity bias in recommender systems.

2 Related Work

In this section, we provide a brief overview of existing work on bias in recommender systems and datasets and reproducibility.

2.1 Bias in Recommender Systems

Recommender systems are not immune to bias, even when only user consumption history is fed to the model and not other information about the users or items. Edizel et al. (2019) discuss that a model might learn sensitive information like the gender of the user in the latent space, and produce recommendations that are gender-dependent, even more so than the interactions observed in the training set itself. In a survey on the topic of bias and debias in recommender systems research, Chen et al. (2023) identify three factors that contribute to bias: user behavior's dependence on the exposure mechanism of the system, imbalanced presence of items (and users) in the data, and the effect of feedback loops. One type of bias that arises from the interaction between an algorithm and imbalanced data is popularity bias.

Popularity bias is the phenomenon where popular items (i.e., items that are frequently interacted with in the dataset) are recommended even more frequently than their popularity would warrant (Abdollahpouri and Mansoury, 2020). It is commonly believed to be caused by the long-tail distribution that often characterizes user-item interactions: most items have been rated by only a few users, and a few items have been rated by many users (Brynjolfsson et al., 2006). Various studies have reported that frequently used recommender systems algorithms are prone to propagating popularity bias existing in the dataset they were trained on (Abdollahpouri et al., 2019b; Kowald et al., 2020; Naghiaei et al., 2022). Different metrics have been proposed to quantify popularity bias (Abdollahpouri et al., 2019b, 2017). Despite the extensive literature, our understanding of why certain algorithms and datasets are more or less prone to popularity bias is limited. In a systematic work, Deldjoo et al. (2021) used regression-based modeling to explain accuracy and fairness exhibited by a set of collaborative filtering algorithms through the lens of data characteristics such as rating and popularity distribution. In this paper, we describe scenarios of data-algorithm interaction and report the results of different metrics associated specifically with popularity bias.

2.2 Datasets and Reproducibility

The way that recommender systems researchers usually test their hypotheses, novel algorithms, and metrics is by conducting experiments on one or more publicly available datasets of user-item interactions. Surveys on the topic of recommender systems research show that the pool of datasets used is small (Bobadilla et al., 2013); user behavior data from real-world applications such as media platforms is often proprietary and therefore cannot be used for benchmarking (Khusro et al., 2016). In popularity bias studies, the use of different versions of MovieLens is exceedingly common (Wang et al., 2023), which leads us to wonder whether studies can be conclusive when they are carried out solely on a few datasets. In

our approach, we include a data synthesis step that allows us to experiment with different data distributions and observe the result. Synthetic data serves multiple purposes, each with its own specific requirements and evaluation setup (Slokom and Larson, 2021). Data synthesis is a much-discussed topic in recommender systems research, often explored in the context of privacy (Slokom, 2018; Tso and Schmidt-Thieme, 2006). Additionally, studies have employed data simulation to measure bias in recommender systems under varying data properties (Bellogín et al., 2017).

The existence of datasets for training and testing is valuable for the recommender systems community; research on publicly available data is necessary in order to ensure reproducibility (Said and Bellogín, 2014). However, as noted by Cremonesi and Jannach (2021), sharing the used data is not always sufficient to ensure basic reproducibility. Studies showed that in most cases recommender systems papers presented at top-tier conferences did not provide code for their data preprocessing or hyperparameter tuning (Ferrari Dacrema et al., 2021, 2019). This is also the case in popularity bias research; studies are often not accompanied by code, and sometimes do not describe the data filtering or hyperparameter setting (Abdollahpouri et al., 2019b, 2020). Therefore, concluding that an algorithm or a dataset is prone to popularity bias becomes challenging, as it is not possible to verify or falsify the claims (Cremonesi and Jannach, 2021). Research has shown that the choice of hyperaparameters can highly impact quality metrics, including average popularity of the items recommended (Jannach et al., 2015). At the same time, popular frameworks seem to have important differences that translate into different performances for the same datasets given somewhat different implementations of the same algorithm (Bellogín and Said, 2021). To further explore this issue, we experiment with algorithms implemented in different libraries and with different parameter configurations and evaluate popularity bias for each of them.

3 Identifying Data Characteristics and Algorithm Configurations

In this section, we identify data characteristics and algorithm configurations that can influence popularity bias propagation in the context of a rating prediction and top-10 recommendation task. First, we locate data characteristics that can have an effect on whether popularity bias is propagated, partly inspired by the functionality of UserKNN. UserKNN is a relatively simple algorithmic approach that simulates a 'word-of-mouth' setting and has lower dependence on non-intuitive parameters that impact optimization (e.g., learning rate). Accordingly, we form a set of data scenarios that combine the located characteristics. Second, we inspect UserKNN and two matrix factorization algorithms, one traditional and one neural network-based, and locate configurations that can be potentially impactful for popularity bias propagation.

3.1 Data Characteristics

Whether or not popularity bias is propagated depends on how popularity manifests in the dataset at hand. We discuss the relation between rating and popularity and the preferences of influential users.

Relation Between Rating and Popularity In the context of a rating prediction task, an algorithm aims to predict a future rating for every user of every item they have not

already consumed. Given that this is done by considering the other users' ratings, it may be that items with high average rating will be prioritized by the system. Popularity bias studies often do not disclose whether the popular items in the dataset also have high ratings, but instead assume that their frequent recommendation is solely due to their popularity. Previous work has shown the impact of high ratings on popularity bias (Yalcin, 2021).

Influential Users In the context of UserKNN, certain users may be influential because they neighbour with many other users. For example, if only two users have rated an item and they have not rated any other items, then this item will not be recommended to anyone, because the two users are not influential at all within the system. Consequently, it is interesting to investigate the notion of user influence and whether the result is dominated by the preferences of users who, because of their large profile size, are more likely to have many neighbours.

3.1.1 Real Datasets

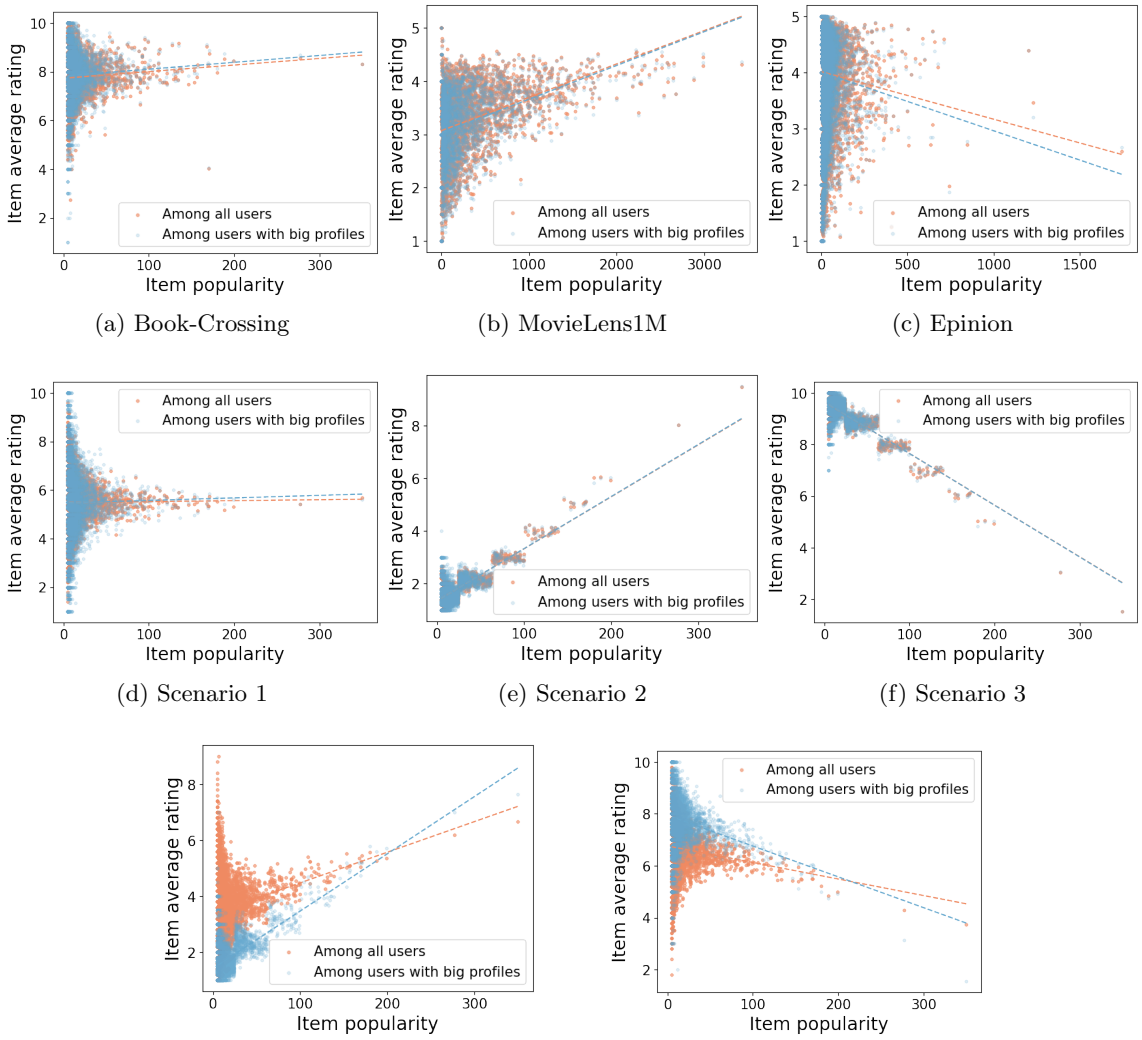
Figures 1a, 1b and 1c show the aforementioned characteristics for Book-Crossing, MovieLens1M and Epinion, respectively. Specifically, they show the correlation between item average rating and item popularity among all users, as well as among the users with the 20% largest profiles. We see that for Book-Crossing and MovieLens1M there is a slight positive correlation between rating and popularity, and a negative one for Epinion. Additionally, for the first two datasets there is no obvious difference between the tendencies of all users and the ones with large profiles, while the users with large profiles in Epinion are less favourable towards the popular items than the entire set of users. Table 1 shows the number of users, items, and interactions, and sparsity for each of the datasets.

Table 1: Basic characteristics of the real datasets. The synthetic datasets share the same characteristics as Book-Crossing.

Dataset	**#users**	**#items**	**#ratings**	**Sparsity**
Book-Crossing (subset)	6,358	6,921	88,552	99.80%
MovieLens1M	6,040	3,706	1,000,209	95.53%
Epinion	22,164	296,277	912,441	99.99%

3.1.2 Data Scenarios

We synthesize data that follows a long-tail distribution for items and users, as it is discussed as a prerequisite for popularity bias to occur (Brynjolfsson et al., 2006; Celma and Cano, 2008). Specifically, we choose the interactions in a subset of the Book-Crossing dataset (Naghiaei et al., 2022; Ziegler et al., 2005) as a baseline, but remove the rating values. To reflect on the observations above, we form a set of scenarios around the relationship between popularity, rating and user influence to assign a synthesized rating to each interaction. This approach allows us to simulate a real-world scenario where consumption is long-tail, while still experimenting with data properties relevant for popularity bias. We recognize that the scenarios are not necessarily realistic. User tendencies are likely to be more subtle in real world situations. However, we believe that experimenting with extreme behaviors can



(a) Book-Crossing (b) MovieLens1M (c) Epinion

(d) Scenario 1 (e) Scenario 2 (f) Scenario 3

(g) Scenario 4 (h) Scenario 5

Figure 1: Relation between item average rating and item popularity among all users and among users with the 20% largest profiles, given three real and five synthetic datasets.

help us showcase the effect that we are investigating, and lead the way for more nuanced experimentation.

The scenarios, as well as the process we followed to generate each of them are as follows:

1. **Scenario 1: There is no relation between popularity and rating**: For each interaction, draw a rating value between 1 and 10 uniformly at random. In this case, the popularity of the item is not taken into account when a rating is generated.

2. **Scenario 2: Popular items are generally rated higher by the users**: For each interaction, draw a rating value between 1 and 10 from a normal distribution, where the mean is the popularity of the item normalized between 1 and 10. Since the mean of the rating distribution is the item's normalized popularity, more popular items tend to receive higher ratings.

3. **Scenario 3: Popular items are generally rated lower by the users**: For each interaction, draw a rating value between 1 and 10 from a normal distribution, where the mean is the opposite number of the popularity of the item normalized between 1 and 10. Since the mean of the rating distribution is the opposite number of the item's normalized popularity, more popular items tend to receive lower ratings.

4. **Scenario 4: Only users with large profiles rate popular items higher**: For each interaction, draw a rating value between 1 and 10 uniformly at random. For the users with the 20% largest profiles, replace by drawing from a Poisson distribution where the mean is the popularity of the item normalized between 1 and 10. While most users rate at random, users with large profiles tend to rate popular items higher.

5. **Scenario 5: Only users with large profiles rate popular items lower**: For each interaction, draw a rating value between 1 and 10 uniformly at random. For the users with the 20% largest profiles, replace by drawing from a Poisson distribution where the mean is the opposite of the popularity of the item normalized between 1 and 10. While most users rate at random, users with large profiles tend to rate popular items lower.

Figures 1d to 1h show the correlation between item average rating and item popularity within the five synthetic datasets. The effects are much more pronounced than for the real datasets. Scenario 1 shows no relation between average rating and popularity, as the ratings were drawn uniformly at random. Scenarios 2 and 3 showcase a very positive and a very negative correlation, respectively. For scenario 4, we see a positive correlation, which is higher for users with large profiles, and for scenario 5 a negative correlation, which is even lower for users with large profiles.

3.2 Algorithm Configurations

In this section, we describe the algorithm configurations examined for UserKNN and two matrix factorization algorithms, one traditional and one neural network based.

3.2.1 UserKNN

Despite UserKNN's simplicity, there are configuration choices that can potentially greatly influence the result. We identified the following: minimum similarity, the items considered for similarity, and minimum neighbours.

Minimum Similarity It is common in UserKNN implementations that not all users who rated an item are considered (Desrosiers and Karypis, 2010). Instead, the notion of neighbourhood is introduced; only the users most similar to the target user are taken into account when producing a predicted score. The filtering can be done by introducing a cut off value of minimum similarity for consideration, among other techniques.

Items for Similarity Given a similarity metric (e.g., cosine similarity), a design choice still has to be made on whether similarity between two users will be calculated for their full rating vectors, or only for ratings on items these two users have in common. See (Aggarwal, 2016) for clarification.

Minimum Neighbours When the neighbourhood is constructed for a given user, then a score is predicted for each item in a list of candidates. In some implementations, the predicted score is not calculated for all potential items. Instead, the algorithm focuses on items that have been rated by at least a minimum number of neighbours of the current user.

It is worth noting that LensKit and Cornac differ when it comes to these choices as seen in Table 2. Two of the parameters tested are not configurable in Cornac, while the third is not configurable in either framework and is set to a different value in each of them.

Table 2: Configuration choices related to UserKNN made by LensKit for Python and Cornac.

Configuration choice	**LensKit for Python**	**Cornac**
Minimum similarity	Configurable (default: 0)	Fixed to -1
Items for similarity	Fixed to all items	Fixed to common items
Minimum neighbours	Configurable (default: 1)	Fixed to 1

3.2.2 Traditional Matrix Factorization

In order to experimentally inspect the effect of implementation, we focus on a matrix factorization algorithm that is implemented by both LensKit and Cornac. Namely, we look into Biased Matrix Factorization (BMF).

BMF is a type of matrix factorization used for explicit rating prediction, and it is differently implemented in LensKit and Cornac. LensKit provides an alternating least squares implementation of BMF, and the default solver is coordinate descent (Takács et al., 2011) with weighted regularization (Zhou et al., 2008). Cornac implements a stochastic gradient descent solver for BMF (Koren et al., 2009). In both libraries, the Boolean parameter of bias is included to signify whether user and item bias are used to predict the ratings, and is by default set to True. Bias is an interesting parameter to investigate in the context of popularity bias propagation.

3.2.3 Deep Matrix Factorization

Along with BMF, we perform preliminary analysis on the popularity bias propagated by a neural network based matrix factorization algorithm, namely Deep Matrix Factorization (DMF). DMF uses a multi-layer perceptron to project users and items into a latent structured space (Xue et al., 2017). It takes into account explicit ratings and implicit interactions to compute the low-dimensional vectors via a neural network architecture, and then estimates the relevance of an item to a user with cosine similarity between the vectors. DMF is not available in either Cornac or LensKit. The Elliot framework (Anelli et al., 2021) includes an implementation of DMF that comes with a set of hyperparameters that can be tweaked. DMF's hyperparameters consist of latent factors, which represent the number of units in the final MLP layer for both users and items. The regularization term controls overfitting by penalizing large weights in the model. The learning rate determines the step size during optimization. Finally, the similarity measure computes the relevance between user and item embeddings using cosine similarity.

Neural network based approaches are generally understood to be less interpretable. Therefore, predicting or even explaining the effect of their parameters on popularity bias is nontrivial. For the purposes of this study, we experiment with the number of latent factors in the final layer of the network, since they can affect the underfitting/overfitting of the model, which can have an interplay with the data characteristics.

4 Experimental Setup

In this section, we describe the experiments that we run in order to determine the effect of data characteristics and algorithm configuration on the propagation of popularity bias. We run all experiments on a Fedora Linux 40 machine with a 24-core AMD EPYC 7401 Processor and 1TiB RAM. The code we wrote to run the experiments[2,3] has been made open source. For the Book-Crossing subset, the datasets MovieLens1M and Epinion, as well as every synthetic data scenario, we perform a recommendation process given every version of every algorithm. For UserKNN, we test the following versions given the configurations discussed in section 3.2.1:

- Min. similarity 0, over all items, 1 min. neighbour.
- Min. similarity 0, over all items, 2 min. neighbours.
- Min. similarity -1, over all items, 1 min. neighbour.
- Min. similarity -1, over all items, 2 min. neighbours.
- Min. similarity -1, over common items, 1 min. neighbour.

For BMF, we test the LensKit and Cornac version, along with the bias parameter. For DMF, we test the Elliot version, along with the number of factors in the final layer. For each algorithm version and each dataset, we perform optimization based on RMSE by splitting the dataset into 80-20% sets to find the best values for some of the non-fixed hyperparameters

2. `https://github.com/SavvinaDaniil/DiagnosingBiasRecSys`
3. `https://github.com/SavvinaDaniil/Elliot`

of the respective version, except for DMF that we based the optimization on NDCG@10 since DMF does not predict rating but relevance score. The resulting hyperparameters can be seen in our repositories. Afterwards, we divide the users into training and test users in a 5-fold cross validated way. We make sure to use the same splits for all algorithms and all versions. For every test user, we use 80% of their ratings for training and the remaining 20% for testing, which is an option in LensKit. We train the model on the training set. For each user in the test set, we predict a rating for every item they have not rated in the training set (see TrainItems in section 3.1.3 of Said and Bellogín (2014)), rank the items based on the predicted score and recommend the top-10 items, in line with recent studies on popularity bias.

We report on RMSE and NDCG@10 where applicable to estimate the effectiveness of the rating prediction and ranking, respectively. We also calculate the following widely used metrics on the recommended lists to estimate popularity bias propagation:

1. **Popularity Correlation (PopCorr)**: The correlation between popularity in training set and recommendation frequency for every item (Kowald and Lacic, 2022).
2. **Average Recommendation Popularity (ARP)**: The average popularity of the items in the recommended lists (Yin et al., 2012; Abdollahpouri et al., 2019a).
3. **Popularity Lift (PL)**: The average relative difference in popularity between the recommended items and the items in the users' profiles (Abdollahpouri et al., 2020).

Finally, for every dataset and algorithm we perform a Mann–Whitney U test to observe whether there is a significant difference among configurations for ARP and PL, and include the result in the respective tables.

5 Results

In this section, we provide insights into how algorithm configurations impact popularity bias and performance for the different datasets by presenting the results across the set of metrics listed in section 4. For each metric, we embolden the highest value among configurations. For ARP and PL, we use the asterisk (*) to signify which values are significantly lower than the highest one according to a Mann–Whitney U test with $p < 0.005$. We abbreviate Book-Crossing as B-C and MovieLens1M as ML1M.

5.1 Popularity bias by UserKNN

5.1.1 Real Data

Table 3 shows the results when combining different UserKNN configuration choices with the Book-Crossing subset, MovieLens1M, and Epinion.

The extent to which popularity bias propagates in the recommendation varies across the datasets. For Book-Crossing, the algorithm performance is best when popularity bias is high, based on both RMSE and NDCG@10. When minimum neighbours are set to 1, there is no notable popularity bias according to all metrics. On the other hand, when minimum neighbours are set to 2, the PopCorr and PL metrics indicate strong popularity

Table 3: Popularity bias and performance of different UserKNN configurations given Book-Crossing, MovieLens1M, and Epinion. OverCommon set to True corresponds to the Cornac implementation, and set to False to the LensKit implementation. For each metric, we embolden the highest value among configurations. For ARP and PL, we use the asterisk (*) to signify which values are significantly lower than the highest one according to a Mann–Whitney U test with $p < 0.005$.

Dataset	Min Sim	Items for Similarity	Min Nbrs	Pop Corr↑	ARP↑	PL↑	RMSE↓	NDCG @10↑
B-C	-1	Common	1	0.010	0.002*	-32.843*	1.739	0.001
		All	1	-0.000	0.002*	-38.583*	1.860	0.001
			2	0.282	0.003*	15.018*	1.758	0.003
	0		1	0.071	0.003*	-17.740*	1.816	0.001
			2	**0.446**	**0.005**	**67.356**	**1.704**	**0.006**
ML1M	-1	Common	1	-0.093	0.000*	-99.722*	0.910	0.000
		All	1	-0.082	0.000*	-99.782*	0.906	0.000
			2	-0.053	0.017*	-86.270*	0.904	0.004
			10	**0.247**	**0.175**	**26.134**	**0.894**	**0.055**
	0		1	-0.050	0.010*	-91.918*	0.900	0.003
			2	-0.003	0.041*	-67.794*	0.894	0.013
			10	0.176	0.170*	21.997	0.898	0.048
Epinion	-1	Common	1	0.020	0.000*	20.789*	1.154	0.000
		All	1	0.023	0.000*	36.538*	1.212	0.000
			2	**0.173**	0.001*	**176.224**	1.168	0.000
	0		1	0.043	0.001*	65.527*	1.148	0.000
			2	0.153	**0.001**	165.929	**1.108**	**0.001**

bias. Additionally, minimum similarity is impactful: increasing it from -1 to 0 increases popularity bias across all metrics. Finally, which items to consider for similarity has a small effect on all metrics, but without a clear direction.

For MovieLens1M, according to NDCG@10, ranking performance is higher than for Book-Crossing and Epinion. For MovieLens1M, the minimum neighbours hyperparameter of UserKNN impacts popularity bias largely, with bias increasing when more minimum neighbours are required. In contrast to Book-Crossing, popularity bias is the highest for minimum similarity of -1, based on all metrics. Which items to consider for similarity has no discernible effect on popularity bias.

On Epinion, popularity bias is present for all versions of the algorithms according to the PopCorr and PL metrics. Again, increasing minimum neighbours increases popularity bias across metrics. The parameters of minimum similarity and which items to consider for similarity affect popularity bias, but not largely.

It is immediately observable that popularity bias varies across the different configurations of UserKNN with the parameter of minimum neighbours, one that is not configurable in one of the frameworks, being especially influential. This is true for every dataset, though the degree of popularity bias also differs between the datasets. The results indicate that popularity bias is not unavoidable when the data follows a long tail distribution, and depends on other characteristics of the datasets as well as the configuration of the algorithms.

5.1.2 Synthetic Data

To investigate which data characteristics could impact popularity bias propagation given each version of UserKNN, we present the results for the synthetic datasets in table 4.

Performance varies across the data scenarios. RMSE specifically is lower for scenarios 2 and 3 compared to the other three. In these two scenarios, users tend to agree between them on whether they like popular items or not, which facilitates the rating prediction task. NDCG@10 is the highest for scenario 2, where popular items are highly rated by the users. In this case, the rating prediction and ranking tasks are linked, since the highest ranked (i.e., popular items) are also highly rated.

Popularity bias also varies across the data scenarios, and the effect depends on the algorithm configuration. In the following paragraphs, we describe and reflect on the most impactful effects of the interaction between data and configuration.

For scenario 1 where ratings are uniformly at random generated, there is no notable popularity bias propagation observed when minimum neighbours are set to 1, while there is bias when minimum neighbours is set to 2. This observation is in line with what we noted for the real datasets, where increasing minimum neighbours results in higher popularity bias for all datasets and metrics.

In scenario 3 where all users agree that popular items are bad, popularity bias is not propagated when minimum similarity is set to 0. However, when setting minimum similarity to -1, we can observe popularity bias propagation across all metrics. The reason is that users with completely different opinions are considered and their opinions count negatively. Therefore, popular items still get recommended since everyone's "negative" neighbours dislike them, and we can observe popularity bias propagation across all metrics.

Table 4: Popularity bias and performance of different UserKNN configurations given synthetic data based on different data scenarios. OverCommon set to True corresponds to the Cornac implementation, and set to False to the LensKit implementation. For each metric, we embolden the highest value among configurations. For ARP and PL, we use the asterisk (*) to signify which values are significantly lower than the highest one according to a Mann–Whitney U test with $p < 0.005$.

Data Scenario	Min Sim	Items for Similarity	Min Nbrs	Pop Corr↑	ARP↑	PL↑	RMSE↓	NDCG @10↑
Scenario 1	-1	Common	1	0.004	0.002*	-35.746*	**3.337**	0.001
		All	1	0.018	0.002*	-32.285*	3.502	0.001
			2	0.418	0.004*	21.252*	3.352	0.003
	0		1	0.101	0.003*	-12.827*	3.624	0.002
			2	**0.615**	**0.005**	**65.440**	3.464	**0.005**
Scenario 2	-1	Common	1	0.604	0.015*	305.197*	1.150	0.013
		All	1	0.596	0.021*	426.621*	1.188	0.019
			2	**0.614**	0.022*	447.618*	1.190	0.021
	0		1	0.552	**0.027**	**632.300**	1.040	0.023
			2	0.562	0.027	591.966	**1.026**	**0.025**
Scenario 3	-1	Common	1	0.522	0.006*	151.686*	1.151	0.001
		All	1	0.559	0.008*	187.197*	1.182	0.002
			2	**0.728**	**0.008**	**192.127**	1.182	0.002
	0		1	0.025	0.002*	-35.765*	1.044	0.001
			2	0.161	0.003*	-13.100*	**1.034**	**0.004**
Scenario 4	-1	Common	1	0.184	0.003*	8.669*	2.458	0.001
		All	1	0.253	0.003*	23.063*	2.502	0.001
			2	**0.772**	0.006*	97.490*	2.404	0.004
	0		1	0.588	0.008*	164.549*	2.500	0.004
			2	0.701	**0.014**	**297.047**	**2.386**	**0.010**
Scenario 5	-1	Common	1	0.057	0.002*	-16.243*	2.783	0.001
		All	1	0.087	0.003*	-7.924*	2.880	0.001
			2	**0.623**	0.005*	**57.969**	**2.776**	0.003
	0		1	0.136	0.003*	-16.122*	2.914	0.003
			2	0.612	**0.005**	42.849	2.794	**0.006**

When considering only common items to calculate similarity, users with smaller profiles have a larger influence. This is relevant in scenario 4 where users with large profiles like popular items. Table 4 shows that even though scenario 4 still leads to popularity bias, considering only common items reduces it across all metrics. Therefore, this implementation choice can have a big impact on whether popularity bias is propagated and to what extent.

Finally, the value for minimum neighbours largely influences popularity bias. Across almost all scenarios and metrics, increasing minimum neighbours from 1 to 2 leads to increased popularity bias. By setting a higher neighbour barrier for considering an item for recommendation, it follows that less popular items will be disadvantaged. This result is particularly relevant given that the parameter of minimum neighbours could only be tweaked in one of the considered frameworks, so studies that use Cornac or LensKit might reach different conclusions on the extent of popularity bias propagated by UserKNN.

Popularity bias manifests differently in different datasets; an explanation for this can be found in data characteristics, specifically the relation between item ratings, item popularity, and the influence of users with large profiles. All three configuration choices affect the observed bias, as they influence the weight of each user's preference.

5.2 Popularity bias by Matrix Factorization algorithms

5.2.1 Real Data

Table 5 shows the results for BMF, trained on Book-Crossing, MovieLens1M and Epinion.

Table 5: Popularity bias and performance on the Book-Crossing, MovieLens1M and Epinion datasets given different BMF implementations. For each metric, we embolden the highest value among configurations. For ARP and PL, we use the asterisk (*) to signify which values are significantly lower than the highest one according to a Mann–Whitney U test with $p < 0.005$.

Dataset	Framework	Bias parameter	Pop Corr↑	ARP↑	PL↑	RMSE↓	NDCG @10↑
B-C	Cornac	False	-0.015	0.001*	-68.639*	1.587	0.001
		True	0.003	0.002*	-40.168*	**1.535**	0.001
	Lenskit	False	-0.013	0.002*	-58.518*	1.762	**0.004**
		True	**0.108**	**0.005**	**46.533**	1.560	**0.004**
ML1M	Cornac	False	**0.266**	**0.193**	35.186	**0.856**	**0.064**
		True	0.234	0.192	**35.859**	0.856	0.059
	Lenskit	False	0.151	0.125*	-14.392*	0.860	0.038
		True	0.183	0.155*	7.987*	0.866	0.040
Epinion	Cornac	False	0.001	0.000*	-53.845*	1.152	0.000
		True	0.009	0.001*	19.814*	**1.029**	0.000
	Lenskit	False	0.020	0.001*	-54.107*	1.240	0.000
		True	**0.126**	**0.003**	**402.394**	1.030	**0.001**

BMF tends to have a better performance than UserKNN overall. Popularity bias varies across the LensKit and Cornac implementations of BMF. Also very notable is the effect of the bias parameter.

For Book-Crossing, in both implementations the bias parameter increases popularity bias. For example, PopCorr on Book-Crossing with LensKit is -0.013 with bias set to False and 0.108 with bias set to True, as seen in Table 5. Specifically, the LensKit implementation with the bias parameter set to True is the only version where there is a positive popularity correlation, as well as positive popularity lift, with the difference being significant.

For MovieLens1M, there is higher popularity bias across metrics when the Cornac implementation is used. Additionally, the bias parameter changes PL from negative to positive when the LensKit implementation is used.

For Epinion, there is higher popularity bias across metrics when the LensKit implementation is used. The bias parameter is also very influential, as setting it to True results in higher popularity bias across all metrics, given both implementations, as was the case for Book-Crossing.

We see that the choice for a framework - LensKit or Cornac - largely determines whether popularity bias is observed when using a Matrix Factorization algorithm on widely used datasets, and the effect differs per dataset. In addition, setting the bias parameter of these algorithms impacts to what extent popularity bias is observed.

5.2.2 Synthetic Data

Additionally to the conclusions drawn from the results on the real datasets, to observe whether the data scenarios influence popularity bias propagated by BMF and the impact of the bias parameter, we present the results on the synthetic datasets in table 6.

BMF has a better performance than UserKNN when trained on the synthetic datasets. Similarly to UserKNN, BMF also results in low RMSE for scenarios 2 and 3, and high NDCG@10 for scenario 2. Experimenting with different synthetic datasets helps explore the observation from the previous section that BMF propagates popularity in some cases. Popularity bias can be observed for scenarios 2 and 4, while scenario 5 where users with large profiles do not like popular items results in no popularity bias across metrics. This indicates that the preferences of users with large profiles are influential for the system, and thus, for popularity bias.

In the previous section, we saw that the LensKit and Cornac implementations lead to varying results with respect to popularity bias. On the synthetic datasets, the effect is more apparent. The bias parameter has opposite effects between the two implementations in scenario 4 as seen in Table 6. In Cornac, using bias increases popularity bias across all metrics. On the contrary, the bias parameter decreases popularity bias in the LensKit implementation. This can be due to the use of different optimization methods by Cornac and LensKit. Further investigation is needed to better understand the impact of the bias parameter on popularity bias.

We conclude that, even though the data synthesis was based on the functionality of UserKNN, the differences between the synthesized datasets largely impact popularity bias propagated by BMF. Furthermore, the results confirm that for both BMF implementations,

Table 6: Popularity bias and performance given different BMF implementations and synthetic data based on different data scenarios. For each metric, we embolden the highest value among configurations. For ARP and PL, we use the asterisk (*) to signify which values are significantly lower than the highest one according to a Mann–Whitney U test with $p < 0.005$.

DataScenario	Framework	Bias parameter	Pop Corr↑	ARP↑	PL↑	RMSE↓	NDCG @10↑
Scenario 1	Cornac	False	-0.013	0.001*	-67.527*	3.191	0.001
		True	**0.079**	**0.006**	**74.548**	**2.872**	**0.003**
	Lenskit	False	-0.015	0.002*	-52.395*	3.400	0.002
		True	-0.008	0.002*	-49.725*	3.028	0.001
Scenario 2	Cornac	False	0.475	0.030	743.388	0.938	**0.026**
		True	0.467	**0.030**	**743.446**	**0.802**	0.025
	Lenskit	False	**0.507**	0.030*	740.064	0.940	0.025
		True	0.498	0.030*	742.366	0.818	0.025
Scenario 3	Cornac	False	-0.022	0.001*	-76.210*	0.850	0.001
		True	0.010	**0.002**	**-31.473**	**0.796**	0.001
	Lenskit	False	-0.091	0.001*	-73.688*	1.032	0.001
		True	**0.017**	0.002*	-37.387*	0.810	0.001
Scenario 4	Cornac	False	0.036	0.004*	-20.286*	2.510	0.005
		True	0.424	**0.027**	**647.074**	**2.297**	**0.019**
	Lenskit	False	**0.462**	0.010*	149.807*	2.746	0.012
		True	0.219	0.008*	129.271*	2.390	0.008
Scenario 5	Cornac	False	-0.018	0.001*	-73.474*	2.671	0.001
		True	**-0.006**	0.002*	-53.320*	**2.500**	0.001
	Lenskit	False	-0.045	0.001*	-59.393*	2.890	**0.002**
		True	-0.013	**0.002**	**-50.113**	2.582	0.001

the bias parameter affects popularity bias. The effect is different - sometimes even opposite - for different combinations of dataset and implementation.

5.3 Popularity bias by Deep Matrix Factorization

The goal of this section is to see whether the lessons learned from the detailed analysis presented in the previous sections hold for a neural network-based method.

5.3.1 Real Data

Table 7 shows the results when combining different DMF configuration choices with the Book-Crossing subset and MovieLens1M. Training DMF on Epinion was not possible due to memory allocation limitations.

The number of latent factors in the final layer of the neural network has a clear impact on the results. Interestingly, the impact differs between the two datasets. Specifically, when the number of factors is set to 64, popularity bias increases for Book-Crossing, with the difference being significant. In the case of MovieLens1M, the same configuration results in significantly lower values across metrics.

We observe that the combination of data and configuration also affects popularity bias propagation by neural architectures.

Table 7: Popularity bias and performance on the Book-Crossing, MovieLens1M and Epinion datasets given different DMF versions. For each metric, we embolden the highest value among configurations. For ARP and PL, we use the asterisk (*) to signify which values are significantly lower than the highest one according to a Mann–Whitney U test with $p < 0.005$.

Dataset	Factors	Pop Corr↑	ARP↑	PL↑	NDCG @10↑
B-C	32	0.012	0.002*	-38.610*	0.002
	64	**0.125**	**0.005**	**35.582**	**0.003**
ML1M	32	**0.046**	**0.064**	**-54.777**	**0.010**
	64	0.032	0.057*	-59.654*	0.008

5.3.2 Synthetic Data

To confirm and expand upon the observations of the previous section, we report on performance and popularity bias of DMF on the synthetic data in table 8.

Popularity bias and performance fluctuate among scenarios. In line with the results from previous sections, popularity bias is the highest for scenarios 2 and 4, and the low for the other scenarios. The parameter tested also has an effect, which aligns with the effect on the real data. For all scenarios, increasing the number of factors in the final layer increases popularity bias, in most cases significantly, which aligns with the results on Book-Crossing.

Note that the results presented in this section are preliminary. Further research is needed to explore the effect of data characteristics and algorithm configuration on popularity

Table 8: Popularity bias and performance given different DMF versions and synthetic data based on different data scenarios. For each metric, we embolden the highest value among configurations. For ARP and PL, we use the asterisk (*) to signify which values are significantly lower than the highest one according to a Mann–Whitney U test with $p < 0.005$.

DataScenario	Factors	Pop Corr↑	ARP↑	PL↑	NDCG @10↑
Scenario 1	32	0.057	**0.003**	**-8.529**	0.003
	64	**0.059**	0.003	-10.247	0.003
Scenario 2	32	0.183	**0.005**	**42.030**	0.004
	64	**0.720**	0.004*	-13.542	**0.017**
Scenario 3	32	0.039	0.003*	-19.041*	0.002
	64	**0.074**	**0.004**	**4.613**	**0.003**
Scenario 4	32	0.009	0.002*	-41.003*	0.002
	64	**0.393**	**0.003**	**-34.005**	**0.006**
Scenario 5	32	0.004	0.002*	-43.433*	0.002
	64	**0.079**	**0.003**	**-10.451**	**0.003**

bias propagated by neural network based algorithms. A future study could align with recent advancements in recommender systems by formulating a ranking prediction task and studying popularity bias propagation by the neural network based algorithms available in commonly used open source frameworks.

The preliminary results indicate that the conclusions drawn above also hold for neural network based approaches: data characteristics determine whether popularity bias occurs; popularity bias is not unavoidable even when the data follows a long tail distribution; parameters that can be set at implementation time have a large effect on the propagation of popularity bias.

6 Discussion

6.1 Implications of the present study

Our research shows that multiple data and configuration factors can have an effect on whether bias is propagated. Relying on frameworks readily available to researchers is convenient and a concrete step towards reproducibility, but requires being aware and detailed about the limitations. When simple parameters, such as minimum neighbours in UserKNN, are so influential, it raises questions on how generalizable research on recommender systems bias can be. Our results indicate that bias studies can only draw conclusions within the limits of their specific research and not further than that.

It follows that being explicit about the context within which a type of bias is studied is crucial, both in terms of data characteristics and implementation. It is a known issue in recommender systems literature that implementation details are often not disclosed by studies. Even in cases where they are, guessing the effect of different hyperparameters that are not present in an implementation or experimented with is not trivial. Bias reporting is

definitely not complete if it is not accompanied by clarity around the characteristics, goals and limitations of the system that is being studied.

6.2 Recommendations of the present study

Based on the results of the present study, we put forward two recommendations towards researchers who study bias in recommender systems.

First, researchers should analyze and report on the dataset characteristics that might impact the type of bias they are concerned with. For UserKNN, the relationship between rating and popularity, as well as the preferences of users with large profiles impact popularity bias and should be taken into account in relevant studies. For other algorithms and types of bias, there could be other relevant characteristics. Such analysis will help the reader understand the extent to which the results are a result of the dataset characteristics.

Second, researchers should test multiple algorithm configurations when measuring bias propagation. In a similar way that the community expects x-fold cross validation, since presenting the results of only one run may not be reliable, we could expect to see results on multiple algorithm configurations as well. If the conclusions are only valid for one specific configuration of the algorithm at hand, then that should be clear in the limitations of the study.

6.3 Limitations of the present study and future work

Despite our extensive testing, the results are potentially sensitive to our own experimental design, such as the method for train-test splitting or randomness in the data generation process. Similarly, instantiating the different implementations with the exact same configuration choices is not always possible due to some of the parameters not being configurable. As a result, there might be implementation differences between the frameworks that we are not aware of and cause part of the variation in results, irrespectively of the configurations tested. On this note, we noticed that our reported accuracy does not always coincide with the conclusions of other papers that study the same algorithms and/or datasets. For example, the developers of DMF reported high NDCG@10 results for MovieLens1M (Xue et al., 2017). However, their evaluation strategy was very different from ours: for every user, they only held out one item consumed by them for testing, and only ranked 100 random items instead of all the items not present in the training set like we did. It is reasonable that their performance is better than the one we report. This discrepancy further supports our claim that generalizing conclusions beyond the boundaries of a specific study is not always possible and should be done carefully, which aligns with our previous work where we emphasized the impact of evaluation strategy (Daniil et al., 2024). These observations highlight the importance of our line of research instead of hindering it, since they hint that data and implementation dependence might be present more often than we think.

We recognize that the scholar community has generally moved on from explicit user preferences and rating prediction. We do not focus on implicit feedback given that recent studies on popularity bias are often performed on datasets with explicit ratings in the context of rating prediction tasks (Naghiaei et al., 2022; Kowald et al., 2020; Kowald and Lacic, 2022). A process similar to ours can be followed in the context of implicit feedback: instead of tweaking ratings, researchers can form scenarios around the distribution of popularity in

the dataset and study distributions with varying levels of long-tail. Future work can also focus on components that we chose not to investigate in this study, such as more advanced algorithms, other open libraries for implementation, and the vulnerability of other types of bias to data and algorithms. For example, researchers can create data scenarios based on assumptions regarding the relationship between an item's rating and the demographic characteristics of its creator. Further nuance can be introduced in the data synthesis part, by allowing for more complex relationships between popularity, rating and user influence. Future work could further investigate the impact of data characteristics and configurations on the complex relationship between system performance and popularity bias. Our findings could also be of use in the domain of bias mitigation: certain algorithm configurations appear to have a clear effect on popularity bias (e.g., increasing the minimum neighbours in UserKNN consistently increases it). It is, therefore, relevant to investigate whether appropriately configuring certain parameters assists with popularity bias mitigation.

7 Conclusion

In this study, we reflected on the need for fundamental understanding of the relationship between data, algorithms and bias in recommender systems. We focused on reporting on popularity bias, and tracked algorithm configurations and data characteristics that are of importance in its propagation. Accordingly, we generated a set of synthetic datasets, experimented with performing a recommendation process on real and synthetic datasets using different configurations of the algorithms at hand, and evaluated popularity bias using well-known metrics. We found that even when the distribution of popularity in the dataset is long-tail, popularity bias is not unavoidable. We showed that the relationship between popularity and rating, as well as the preferences of users with large profiles have an impact on bias. We highlighted the sensitivity of bias propagation to algorithm configuration and, by extension, framework implementation. Our observations point to methodology and reproducibility issues that extend further than a specific use case, to the recommender systems field at large.

Recommender systems are widely used in our online lives, and bias propagation by such systems can have serious societal impact. With this work, we hope to have called attention to the ambiguity in bias reporting and motivated researchers to strive for reproducibility and highlight specificity when appropriate.

Acknowledgments and Disclosure of Funding

This research was funded through a public-private partnership between CWI and the KB National Library of the Netherlands.

References

Himan Abdollahpouri. *Popularity bias in recommendation: a multi-stakeholder perspective.* PhD thesis, University of Colorado at Boulder, 2020.

Himan Abdollahpouri and Masoud Mansoury. Multi-sided exposure bias in recommendation, 2020. URL https://arxiv.org/abs/2006.15772.

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the 11th ACM conference on recommender systems*, pages 42–46, 2017.

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking, 2019a. URL https://arxiv.org/abs/1901.07555.

Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The unfairness of popularity bias in recommendation. In *Recommendation in Multi-stakeholder Environments (RMSE), in conjunction with the 13th ACM Conference on Recommender Systems*, 2019b.

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

Charu C Aggarwal. Neighborhood-based collaborative filtering. *Recommender Systems: The Textbook*, pages 29–70, 2016.

Abdul Basit Ahanger, Syed Wajid Aalam, Muzafar Rasool Bhat, and Assif Assad. Popularity bias in recommender systems-a review. In *International Conference on Emerging Technologies in Computer Engineering*, pages 431–444. Springer, 2022.

Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 2405–2414, 2021.

Alejandro Bellogín and Alan Said. Improving accountability in recommender systems research through reproducibility. *User Modeling and User-Adapted Interaction*, 31(5):941–977, 2021.

Alejandro Bellogín, Pablo Castells, and Iván Cantador. Statistical biases in information retrieval metrics for recommender systems. *Information Retrieval Journal*, 20:606–634, 2017.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

Erik Brynjolfsson, Yu Jeffrey Hu, and Michael D Smith. From niches to riches: Anatomy of the long tail. *Sloan management review*, 47(4):67–71, 2006.

Òscar Celma and Pedro Cano. From hits to niches? or how popular artists can bias music recommendation and discovery. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–8, 2008.

Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *ACM Transaction on Information System*, 41(3), 2023. ISSN 1046-8188.

Paolo Cremonesi and Dietmar Jannach. Progress in recommender systems research: Crisis? what crisis? *AI Magazine*, 42(3):43–54, 2021.

Savvina Daniil, Mirjam Cuper, Cynthia CS Liem, Jacco van Ossenbruggen, and Laura Hollink. Reproducing popularity bias in recommendation: The effect of evaluation strategies. *ACM Transactions on Recommender Systems*, 2(1):1–39, 2024.

Yashar Deldjoo, Alejandro Bellogin, and Tommaso Di Noia. Explaining recommender systems fairness and accuracy through the lens of data characteristics. *Information Processing & Management*, 58(5):102662, 2021.

Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, pages 107–144, 2010.

Bora Edizel, Francesco Bonchi, Sara Hajian, André Panisson, and Tamir Tassa. Fairecsys: Mitigating algorithmic bias in recommender systems. *International Journal of Data Science and Analytics*, 9(2):197–213, 2019.

Michael D Ekstrand. Lenskit for python: Next-generation software for recommender systems experiments. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 2999–3006, 2020.

Mehdi Elahi, Danial Khosh Kholgh, Mohammad Sina Kiarostami, Sorush Saghari, Shiva Parsa Rad, and Marko Tkalčič. Investigating the impact of recommender systems on user-based and item-based popularity bias. *Information Processing & Management*, 58(5):102655, 2021.

Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM conference on recommender systems*, pages 101–109, 2019.

Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems (TOIS)*, 39(2):1–49, 2021.

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

Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction*, 25:427–491, 2015.

Shah Khusro, Zafar Ali, and Irfan Ullah. Recommender systems: issues, challenges, and research opportunities. In *Information science and applications*, pages 1179–1189. Springer, 2016.

Anastasiia Klimashevskaia, Dietmar Jannach, Mehdi Elahi, and Christoph Trattner. A survey on popularity bias in recommender systems. *User Modeling and User-Adapted Interaction*, pages 1–58, 2024.

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

Dominik Kowald and Emanuel Lacic. Popularity bias in collaborative filtering-based multimedia recommender systems. In *International Workshop on Algorithmic Bias in Search and Recommendation*, pages 1–11. Springer, 2022.

Dominik Kowald, Markus Schedl, and Elisabeth Lex. The unfairness of popularity bias in music recommendation: A reproducibility study. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR, Lisbon, Portugal, Proceedings, Part II 42*, pages 35–42. Springer, 2020.

Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24, 2007.

Mohammadmehdi Naghiaei, Hossein A Rahmani, and Mahdi Dehghan. The unfairness of popularity bias in book recommendation. In *Advances in Bias and Fairness in Information Retrieval: Third International Workshop, BIAS 2022, Stavanger, Norway, Revised Selected Papers*, pages 69–81. Springer, 2022.

Alan Said and Alejandro Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 129–136, 2014.

Aghiles Salah, Quoc-Tuan Truong, and Hady W Lauw. Cornac: A comparative framework for multimodal recommender systems. *The Journal of Machine Learning Research*, 21 (1):3803–3807, 2020.

Manel Slokom. Comparing recommender systems using synthetic data. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 548–552, 2018.

Manel Slokom and Martha Larson. Doing data right: How lessons learned working with conventional data should inform the future of synthetic data for recommender systems. *arXiv preprint arXiv:2110.03275*, 2021.

Gábor Takács, István Pilászy, and Domonkos Tikk. Applications of the conjugate gradient method for implicit feedback collaborative filtering. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 297–300, 2011.

Karen HL Tso and Lars Schmidt-Thieme. Empirical analysis of attribute-aware recommender system algorithms using synthetic data. *Journal of Computers*, 1(4):18–29, 2006.

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

Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *IJCAI*, volume 17, pages 3203–3209. Melbourne, Australia, 2017.

Emre Yalcin. Blockbuster: A new perspective on popularity-bias in recommender systems. In *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pages 107–112. IEEE, 2021.

Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. Challenging the long tail recommendation. *Proceedings of the 38th International Conference on Very Large Data Bases (VLDB Endowment)*, 5(9):896–907, 2012.

Zihao Zhao, Jiawei Chen, Sheng Zhou, Xiangnan He, Xuezhi Cao, Fuzheng Zhang, and Wei Wu. Popularity bias is not always evil: Disentangling benign and harmful bias for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In Rudolf Fleischer and Jinhui Xu, editors, *Algorithmic Aspects in Information and Management*, pages 337–348, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-68880-8.

Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32, 2005.