Can Fairness Get Users Out of the Recommendation Echo Chamber?

Piyush Mishra **Jessie J. Smith** **Abigail Zimmerman**

Abstract

Can fairness aware re-ranking get users out of the feedback loop in recommendation systems? This work explores the relationship between recommendation fairness and unwanted echo chambers. Since fairness aware re-ranking seeks to improve the diversity of recommendations for users, this could lead to greater representation of items in the recommendation training set. More representation and diversity in user-item ratings can lower unwanted echo chambers that naturally emerge in recommender systems–such as popularity bias. Through simulations of user interactions on re-ranked recommendation lists as compared to a baseline recommender, we explore the impact that added diversity may have on a biased system.

1 Introduction

Recommender systems are increasingly common in the online world with applications ranging from music and movies to healthcare and jobs. The task of creating recommendations is interdisciplinary and often involves multiple stakeholders. In many scenarios, recommender systems have stakeholders who provide the items to be recommended (providers) and stakeholders that consume those recommendations (consumers). Multistakeholder recommender systems need to balance the interests and needs of each unique set of stakeholders to ensure equal accuracy for consumers and equal representation for providers (Abdollahpouri et al., 2020). For this reason, many researchers have recently begun to explore the inherent trade-off between accuracy and provider-side fairness in recommendation.

One fundamental aspect of recommender systems are feedback loops, or echo chambers. Users provide feedback to the system by way of clicks, likes, and ratings, and the system adjusts its recommendations to match the user's interests; the process continually adjusts in this manner. Although this process is desirable, feedback loops can become degenerate in certain cases (Jiang et al., 2019). In such echo chambers, users only see and consume items based on the system's influence, and not their personal interests. In multistakeholder recommendation, degenerate feedback loops can cause problems like *popularity bias*, where popular providers on a platform are consistently overexposed to users, and unpopular providers are underexposed. A system that reinforces these kinds of biases is susceptible to issues of fairness for providers. One method for creating recommender systems that are more fair to providers is *fairness-aware reranking*. In fairness-aware reranking, the list of recommendations provided to a consumer are re-ranked to give greater exposure to providers with protected attributes (such as race or gender), and ultimately give more diverse recommendations to users. In this paper, we explore the relationship between provider-fairness and degenerate feedback loops. First, we will provide a background and an overview of relevant work. Second, we introduce our methods. Finally, we present and interpret our results.

Our paper offers the following contributions:

1. An introduction to fairness aware re-ranking in a simulated system where re-ranked recommendations can be compared to a baseline over time.

2. An analysis of the relationship between protected item selection for re-rankers and the distribution of popularity amongst recommendations.

3. An open source github repository with all of the code and data from these experiments, provided so that others can validate this work and

contribute in the future.[1]

2 Background and Relevant Work

Previous work has called for research on how multistakeholder recommender systems might balance fairness with the needs of all involved parties (Abdollahpouri et al., 2020). Chaney et al. recognized that degenerate feedback loops or echo chambers can arise in such systems, and noted that degenerate feedback loops homogenize user behavior without increasing utility (Chaney et al., 2018). Jiang et al. further discussed the social implications of degenerate feedback loops, and offered practical solutions to avoiding system degeneracy (Jiang et al., 2019).

Our work explores one method by which systems might avoid degenerate feedback loops, i.e. through Fairness Aware Re-ranking (FAR). This approach was proposed by Liu et al. to improve borrower-side fairness in a microlending environment. They showed that such algorithms can promote fairness without reducing accuracy (Liu et al., 2019).

Yao et al. showed that recommender effects, including popularity bias, can be isolated by simulating simple user models over time. User models as simple as random selection or stochastic preference for a single attribute are effective (Yao et al., 2020). We take this approach in order to isolate fairness and feedback loops.

3 Methods

In order to investigate the relationship between fairness aware re-ranking algorithms, provider-side fairness and the presence of degenerate feedback loops, we conducted the following experiment. In this experiment we simulate the performance of a recommender system with a fairness-aware re-ranking algorithm over time leveraging the implementations provided in Librec Auto (Mansoury et al., 2018). We track the accuracy and fairness (as measured through the statistical parity metric) over time, and test for the presence of feedback loops. In the following sections, we will describe in detail the components of the simulation and our evaluation metrics.

3.1 Dataset

We ran the experiments for this project using the user, item, rating data from the MovieLens 26M dataset (Harper and Konstan, 2015). In order to efficiently run the algorithms and simulations for the context of this course, we selected a sub-set of this large dataset that contained: 559,070 user-item ratings, which included 6,000 users and 14,623 items. For the intial experiment, we selected protected features based off of movies that belonged to the bottom 10% of under-represented genres and production countries, the data for this was obtained by cross referencing the MovieLens dataset with the IMDB ratings dataset (Maas et al., 2011).

[1]The GitHub repository for these experiments can be found at: github.com/jesmith14/FairnessFeedback

3.2 Base Recommender

We use a Librec Auto implementation of Non-negative Matrix Factorization (NMF) as a base recommender. NMF approximates factors for a user-item matrix, from which we can glean item ratings for each user. Given user-item matrix $\mathcal{V} = \mathcal{H} \cdot \mathcal{W}$, Librec Auto approximates non-negative factor matrices $\mathcal{H}$ and $\mathcal{W}$ (Mansoury et al., 2018). These resulting matrices represent, respectively, the users' correlations with an unlabeled group and items' correlation with an unlabeled group. The approximation is done through minimizing the following equation:

$$||\mathcal{H} \cdot \mathcal{W} - \mathcal{V}||^2 \tag{1}$$

This minimization can be done using either stochastic gradient descent or alternating least squares (kor, 2009).

3.3 Fairness aware re-ranking algorithm

For our fair re-ranking algorithm, we use the Librec Auto implementation of FAR. This algorithm works by promoting items in the protected class for re-ranking using the following formula. For any user $u \in \mathcal{U}$, where $\mathcal{U}$ is the set of users, $v \in \mathcal{V}$ is the set of all items, and $\mathcal{V}_c$ is a set of items that belong to protected classes (Liu et al., 2019), we take the following maximum:

$$\max_{v \in R(u)} (1-\lambda)P(v|u) + \lambda \sum_c P(\mathcal{V}_c) \mathbb{1}_{\{v \in \mathcal{V}_c\}} \prod_{i \in S(u)} \mathbb{1}_{\{i \notin \mathcal{V}_c\}} \tag{2}$$

Here, the first term represents personalization, the second term represents fairness, and λ represents a hyperparameter that controls the tradeoff between personalization and fairness (Liu et al., 2019). In the personalization term, $P(v|u)$ is a personalization score calculated from the output of the base recommender. In the fairness term, $\mathbb{1}_A$

is an indicator function for some boolean expression A that returns true if the boolean expression is true (i.e. if a given v is or is not a member of the protected class $\mathcal{V}_c$)(Liu et al., 2019).

3.3.1 Choosing the protected class

In order to effectively run the re-ranking algorithm, certain features (and consequently the items that have those features) must be selected as ”protected”. Protected items are often chosen to include groups that are typically under-served, and in this case under-recommended. For the context of this project, we selected protected items in two different ways for two rounds of experimentation. In the first experiment, we selected protected items based off of protected features: genres and production countries that were historically under-represented in the dataset as shown by belonging to the bottom 10% of user-item ratings. The items that had features in this protected class were labeled as protected items. This experiment was done with 100 rounds of simulations as an intial test experiment.

For the final experiment, we selected the protected items based off of those that only had 1 rating in the training dataset, in other words: the least-represented items. This included 4094 items, making our protected class about 28% of the user-item ratings in the training dataset. This final experiment was done with 600 rounds of simulations.

3.3.2 Evaluating for Fairness

In order to evaluate how *fair* the recommendation lists were, we utilized the **Statistical Parity** metric. This metric indicates the probability that a protected item will be represented in the recommendation lists for users as compared to a non-protected item.

$$f = \frac{\sum_{l \in L} \sum_{i \in l} \mathbb{I}_{G_p}}{\sum_{l \in L} \sum_{i \in l} \mathbb{I}_{G_u}} \tag{3}$$

Where L is the list of all recommendations, G represents all of the items, G_p is the items that have been selected as protected, and G_u are the items that are unprotected. I is an indicator function that is 0 if G_x is false and 1 if it is true.

Statistical parity will be less than 1 if the non-protected items are more likely to be recommended, more than 1 if the protected items are more likely to be recommended, and exactly 1 if the recommendations have reached perfect fairness, which in this case represents perfect ”parity”.

3.4 Simulations

To simulate the user selection and rating, an environment was designed that was responsible for interacting with the recommendation system and returning a new set of user-item (movie) selections and ratings for a new round of simulation.

For simulating user selection, three types of strategies were used.

1. **Best**: This approach assumes that the user is going to choose the top recommended item every simulation. It signifies the user's choice to rely strongly on the system's recommendation.

2. **Random**: This approach selects one arbitrary item every simulation. Prior selections aren't taken into account while making a choice.

3. **Explore**: This approach takes previous selections into account and chooses an item using a probability distribution. The distribution is generated using the frequency of an item selected with the items present in the recommendation list for that simulation.

3.4.1 Rating

At each step, we simulate the rating a given user would have assigned to their chosen item. We used a simple model to generate ratings similar to the approach Yao et al. described for modeling choices (Yao et al., 2020). For an item selected by the user, we calculate the average of all previous ratings of this item. Then, we draw from a normal distribution centered around this mean with σ of 0.5.

Note that while collaborative filtering is a generally accepted approach to generating these weights, this would not greatly influence our measurement of the difference between the base and fair recommender systems over time, and would instead add unnecessary execution overhead.

3.5 Gini Coefficient and Popularity Bias

In order to measure the prevalence of degenerate feedback loops–otherwise known as unwanted echo chambers–we chose to utilize the metric formally known as popularity bias.

Popularity can be defined as the the ratio of the number of times an item was rated and the total number of ratings.

$$p = \frac{\# \ of \ times \ an \ item \ has \ been \ rated}{\# \ of \ total \ ratings} \tag{4}$$

Gini Coefficient is a measure of inequality of a distribution. It is defined as a ratio with values between 0 and 1. Ratio approaching 1 presents higher inequality in distribution and 0 signifies greater equality.

$$G = 1 - \frac{\sum_{i=1}^{n} f(y_i)(S_{i-1} + S_i)}{S_n} \tag{5}$$

where:

$$S_i = \sum_{j=1}^{i} f(y_j) y_j \text{ and } S_0 = 0 \tag{6}$$

Popularity bias is then measured by interpreting the Gini coefficient. A higher Gini coefficient signifies higher popularity bias. Graphically, a long tailed distribution of ”item” vs. ”number of ratings” also portrays a higher bias towards more popular items.

4 Results, Analysis, and Discussion

In the following section we present the results from the experimentation on the three simulator-types over the course of 600 rounds of simulation for both the base recommender and the FAR fairness-aware re-ranker. Some of the results from the 600 simulations are compared to an earlier experiment done with 100 simulations on a different protected feature set.

4.1 Accuracy

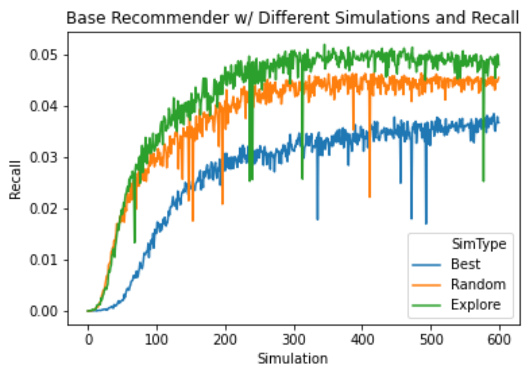

Figure 1: Recall evaluation of base recommender for all simulation types

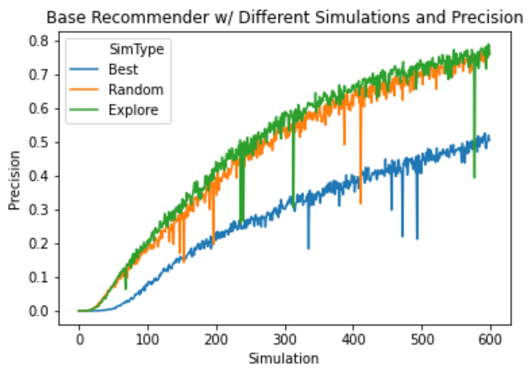

Figure 2: Precision evaluation of base recommender for all simulation types

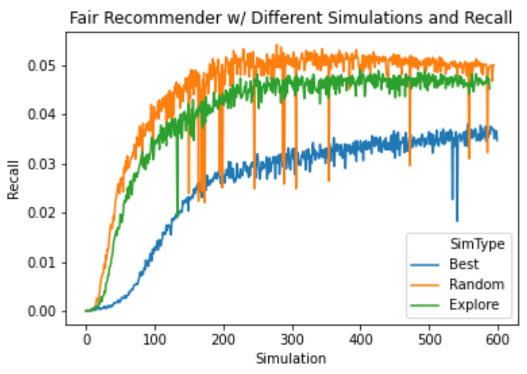

Figure 3: Recall evaluation of fair recommender for all simulation types

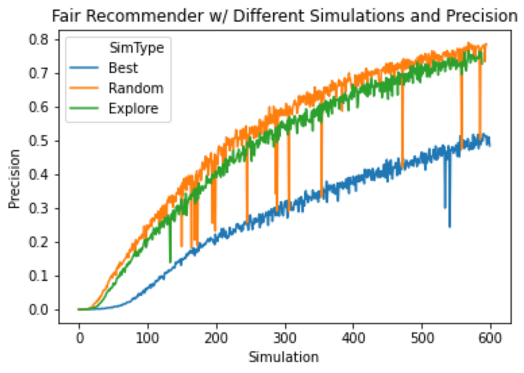

Figure 4: Precision evaluation of fair recommender for all simulation types

It was expected that the overall accuracy of both the base recommender and the fair re-ranker would get better over time, because every simulation iteration added 5500 more data points to the training data for the next simulation, which represented about 1% of the existing data–especially given that the new data points were relevant to every user and their preferences. As shown in *Figures 1 - 4*, the simulation titled “best” ironically had the worst performance, while the simulation titled “Explore” consistently beat the ”best”. The differences in simulation types happened because the ”best” simulation only selected the top ranked recommendation for a user, while the explore simulator took into account the probability distribution over all the selections and ratings that were made from users for items, which meant that a more diverse set of items was being added to the ratings each simulation. For random, because each item selection was arbitrary, the representation of items was also diverse, which caused the recommenders in that simulation-type to trend towards higher accuracy as well.

4.2 Baseline Recommender

One important observation from the results was that the base recommender trended towards promoting ”protected” items at an increasing rate, even without re-ranking implemented. One possible reason for this can be seen in *Figure 6*, which shows that the base recommender had a statistical parity above 1 at the start of the simulations. This shows that although the ”protected” items were chosen from the least-popular items in the ratings dataset, they still had a greater likelihood of being recommended than ”non-protected” items from the start. Our speculation for why this might have happened is (1) due to the size of our dataset, and (2) due to the average rating of the un-popular items.

4.2.1 Size of the dataset

Our datset for this project was a subset of the Movilens 26M dataset and only included about 559,070 user-item ratings, 6,000 users and 14,623 items. This means that by selecting 28% of the items to be classified as ”protected”, many of those items would likely show up again in recommendation lists, even with a high popularity bias. We measured the gini index of the distribution of users in our sub-dataset and found that it was about 67%. This could perhaps be an indication that many users who did not have many item-ratings in their own profiles were recommended these ”protected” and under-represented items a large amount, further increasing the likelihood of them appearing in recommendation lists, which would have raised the amount of ”protected” items in the recommendation lists, and thus raised the overall statistical parity of the base recommender far above 1 as we saw in the results.

4.2.2 Average rating of un-popular items

Though the terms ”most-popular” and ”least-popular” might imply the most highly rated and lowest rated movies in this dataset, for our project - these terms actually mean the quantity of ratings that existed for a particular item. An ”un-popular” item was one that only had 1 user-rating in our training dataset (the bottom 28% of popularity). The ”popular items” were those that had more than 1 rating. After checking the rating information for these two groups, we found that the *popular* items had an average rating of 3.5, while the *un-popular* items had an average rating of 3.2. This could be one of the causes for un-popular items becoming recommended more frequently throughout the simulations, because since low-popularity did not indicate low-user-preference, many of them were rated highly enough to be included on a future recommendation list.

4.3 Statistical Parity

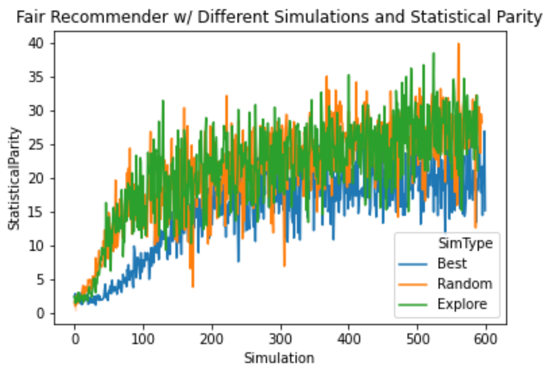

Figure 5: Statistical parity evaluation of fair recommender for all simulation types

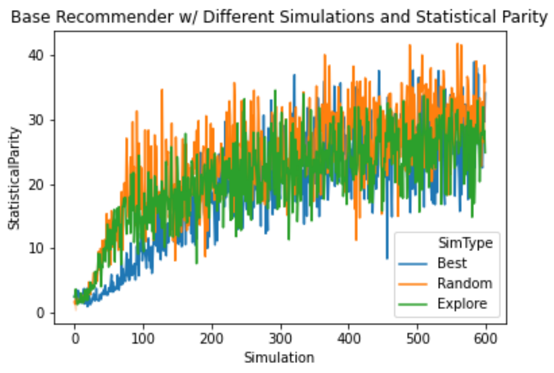

Figure 6: Statistical parity evaluation of base recommender for all simulation types

An interesting observation from the 600 simulations was the change in statistical parity over time. As shown in *Figure 5*, the statistical parity increased consistently (with variations in noise) over the course of the simulations. This was expected for the fair re-ranker, as more items from the protected classes were intentionally being added to user's recommendation lists. But, this was less expected for the base recommender, as this system did not take any protected items into account while curating recommendations.

Through a more in-depth analysis of the results shown in *Figure 6*, it is clear that even the base recommender began the simulations with a statistical parity above 1. This indicates that the base recommender was more likely to recommend protected items over non-protected items before the simulations began. One possible reason for this is due to the method we used to select the protected items. As mentioned in section 5.1.2, by choosing to use the least-represented items (those with only 1 user-item rating in the training data) as the protected items, the items that were chosen as protected may have inevitably been some of the higher-rated items, and thus chosen to be recommended anyways, even without re-ranking in place.

4.4 Popularity Bias

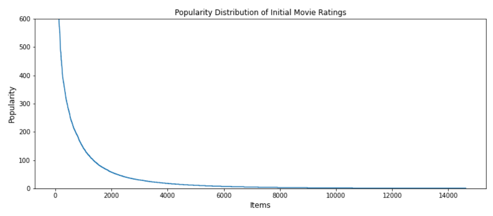

Figure 7: Popularity distribution of original training dataset. Gini Coefficient = 0.824

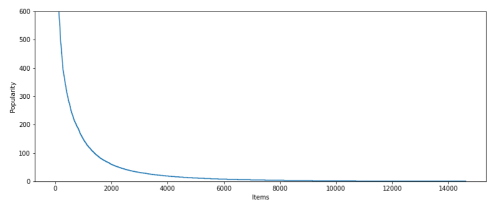

Figure 8: Popularity distribution of final training dataset for the 600 simulations. Gini Coefficient = 0.875

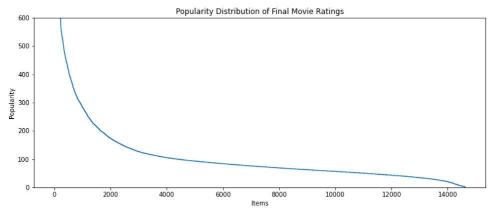

Figure 9: Popularity distribution of final training dataset for the 100 simulations. Gini Coefficient = 0.467

The original popularity bias in the recommender system–as measured from the training dataset–was 0.824. As shown in *Figure 7*, the long-tail popularity distribution and the high gini coefficient of the initial user-item rating dataset indicates that the popularity bias was very high. After running the 600 recommendations, we measured that one fourth of the items that were initially in the bottom 28% of popularity had landed in the top 28% of the training dataset. So to some degree- the experiments were attempting to increase fairness by making the protected items more popular (and consequently attempting to make the initially least-represented items more popular). But, once the protected items began to be recommended more frequently, the re-ranker continued to promote these "now-popular" items even more–resulting in an arguably "less-fair" system. This resulted in a gini index that showed a *higher* popularity bias of 0.875 than before the simulations were ran, as shown in *Figure 8*. This leads to the notion that it may be more productive to choose protected features rather than protected items, as we did in our first experiment. Choosing protected features introduced greater diversity into the recommendations over time, which led to a much lower gini coefficient: 0.467 after 100 simulations, as shown in *Figure 9*.

4.5 Choice of Protected Items

Throughout the analysis of the results of the 600 simulations, it became clear that the choice of protected features (and consequently the protected items) made a significant impact on the results of the popularity bias and the statistical parity. When the protected items were more heavily correlated with less-represented items in the user-item rating dataset, the statistical parity and popularity bias increased over the course of the simulations. This raises an important question - what is the best way to decide which features and items to choose as protected in recommendation re-ranking? Clearly this choice will require making a value judgement that could result in tradeoffs for certain groups of people, and thus must be taken seriously. This question has not been answered in the world of recommender systems yet, and is an important topic to consider for future work in this space.

5 Limitations and Future Work

This project utilized a small subset of the MovieLens 26M dataset (Harper and Konstan, 2015), which allowed for the simulations and experiments to be run more quickly (for reference: the 600 simulation experiment took 3 full days to run). However, given access to machines with greater processing power and more time, an improvement to this study would be to perform the same experiments on the full MovieLens 26M dataset. This could potentially solve problems that arose in this experimentation, particularly in relation to protected items being recommended for users with a small profile who might have only initially rated one item in the dataset.

Additionally, the biggest obstacle with this work was choosing the best way to optimize for "fairness". As we saw in this experimentation, choosing protected items that related to lower popularity in terms of representation in the training dataset was not an optimal solution. The choice for which features to protect is a choice rooted in social science and needs to take into account historically underrepresented groups within the MovieLens dataset.

In the future, it will be important to research not only the variations in choice of protected items, but also the variations in choice of recommendation algorithms, datasets, and simulation design. For example, for this project we only experimented with the NMF algorithm, but there are other relevant latent factor models that would be interesting to run these simulations on (e.g., BiasedMF, Weighted Regularized Matrix Factorization (WRMF) and Probabalistic Matrix Factorization (PMF)) as well as neigbhorhood-based algorithms (e.g., SLIM, item KNN, and user KNN).

While our simulations took into account a variety of different kinds of users, including those who only click on the top recommended item and those who do not act on their earlier movie preferences, a good addition to this project would be to incorporate collaborative filtering on top of the probability distribution in the "explore" simulation in order to more accurately guess what rating a user might give a recommended movie during a selection simulation.

As well, during these simulations we had to make many assumptions about user behavior that may or may not reflect user decisions to select and rate movies in the real world. A much more accurate testing bed for these experiments would involve online data with real-time user decisions and interactions with recommendations, as opposed to the offline data that was used in this project.

6 Conclusion

This work explored the relationship between fairness re-ranking and unwanted feedback loops in recommendation systems. Initially, the goal of this project was to see if adding fairness as a design choice for recommendations could help get users out of the echo chamber of popularity bias. It was discovered that if the protected features and items that are chosen aren't representative of real-world items that are high-quality but low-representation, then re-ranking to include more of these protected items in user recommendation lists can be counterintuitive. Though our design choices did help some of the protected (and in this case under-represented) items rise to the list of the top-most popular items–it did so by creating a more biased system that favored protected items much more than unprotected items.

This is all to say that *design decisions matter* in machine learning systems, especially ones that are considering fairness goals and are impacting human lives. In the same way that heavily debated fairness measures like affirmative action have helped some while harming others–if adding fairness into a system increases fairness for some but decreases fairness for others, is it really a *fair* system? These are all considerations that recommender system designers must take into account at every phase of the design process, from dataset curation and protected feature selection to the metrics we choose to evaluate *fairness* at all–design decisions matter.

References

2009. Matrix factorization techniqures for recommender systems. *IEEE Computer*, 42:30–37.

Himan Abdollahpouri, Gediminas Adomavicius, Robin Burke, Ido Guy, Dietmar Jannach, Toshihiro Kamishima, Jan Krasnodebski, and Luiz Pizzato. 2020. Multistakeholder recommendation: Survey and research directions. *User Modeling and User-Adapted Interaction*, 30(1):127–158.

Allison J. B. Chaney, Brandon M. Stewart, and Barbara E. Engelhardt. 2018. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 224–232, New York, NY, USA. Association for Computing Machinery.

F. Maxwell Harper and Joseph A. Konstan. 2015. Movielens 20m dataset. *ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015) 19 pages*.

Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli. 2019. Degenerate feedback loops in recommender systems. *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*.

Weiwen Liu, Jun Guo, Nasim Sonboli, Robin Burke, and Shengyu Zhang. 2019. Personalized fairness-aware re-ranking for microlending. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 467–471, New York, NY, USA. Association for Computing Machinery.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Masoud Mansoury, Robin Burke, Aldo Ordonez-Gauger, and Xavier Sepulveda. 2018. Automating recommender systems experimentation with librec-auto. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 500–501, New York, NY, USA. Association for Computing Machinery.

Sirui Yao, Alex Beutel, Jilin Chen, Yoni Halpem, Nithum Thain, Kang Lee, Flavien Prost, and Ed H. Chi. 2020. Measuring recommender system effects with simulated users. In *In proceedings of FATES 2020*. Association for Computing Machinery.