Measuring Popularity Bias and Visibility of Providers and Items in Multisided Recommender Networks

Jiwoo Cheon

August 12, 2022

1 Introduction

Recommender systems are prevalent in many domains in our society, such as media, commerce, and news. While recommender systems can help users find items that fit their needs or preferences, many forms of bias may be present in the system and disadvantage certain entities participating in the system. One well-known algorithmic bias in recommender systems is popularity bias: the tendency to recommend popular items too frequently at the expense of under-recommending less popular "long-tail" items. Popularity bias may be caused by bias present in the initial data or through user bias entering the system through feedback (ratings). Collaborative filtering methods especially suffer from popularity bias as they rely on user ratings to make recommendations.

Several previous works examine the effects of popularity bias on item recommendation, showing that popularity bias is detrimental to both items and users. One perspective not considered in these works is that of the provider who creates or otherwise supplies items in the system. The providers may be the artists, production companies, content creators, or other entities that generate items. By including the providers along with the items and users, we consider the multisided nature of recommender systems. On one side, providers supply items to the system, and on the other side, the system recommends items to users who then interact with those items.

To analyze popularity bias in the multisided recommender setting, we measure the disparate visibility of both providers and items. Dividing the two groups reflected by the disparate visibility according to popularity allows disparate visibility to naturally represent popularity bias. We present an experiment on a popular movie recommendation dataset where we compare the disparate visibility resulting from two standard collaborative filtering methods against a random predictor and a baseline estimate. The results show that collaborative filtering methods do suffer from popularity bias for providers although to a lesser degree than items.

In the next section, we review relevant literature motivating this project. Next, we define the multisided recommender system as a network and define visibility in this setting. Then, we present the experimental set up and results and end with a conclusion.

2 Related Works

Popularity bias for items is widely recognized and studied by many works on recommender systems. By definition, popularity bias overly promotes a handful of items, hindering new, possibly innovative items from reaching many users. Ultimately, popularity bias can lead the market to become homogenous. Moreover, overly recommending popular items make it more difficult for users to discover new items and may conflict with preferences of users with niche interests [2]. Some works also propose methods to mitigate the effects of popularity bias. One method is a post-processing step that re-ranks the items to increase representation of long-tail items in recommendations [1].

A relatively new approach in analyzing bias in recommender system is treating the recommender system as a graph and using network-based methods to measure and mitigate bias. This approach can be taken in several directions, depending what components of the recommender system the nodes and edges of the graph represent. Fabbri et al. model "What-to-watch-next" recommenders as an item to item graph where nodes correspond to videos (or other types of content) and directed edges represent recommendation from one video to another [4]. On this graph, they model the browsing activity of a user as a random walk, and they measure how easily the user may get trapped in a radicalisation pathway from a certain node by the expected length of a random walk from that node to a "neutral" node. They also propose a rewiring operation to reduce the length of such radicalization pathways while maintaining the quality of the recommendations. On the other hand, Mansoury et al. present FairMatch, a graph-based algorithm that improves the aggregate diversity of recommended items [7]. They model a item recommender as a bipartite graph where the left nodes represent items, the right nodes represent users, and left-to-right edges represent recommendations. From this graph, a network is created by adding source and sink nodes and weighing each edge from an item to a user is weighted by the weighted sum of item visibility and relevance to the user. On this network, the algorithm repeatedly solves the Maximum Flow problem to select items with high quality and low visibility.

A work that directly motivates this projet is the one by Fabbri et al. [3], which studies the concept of visibility in people recommender systems. In particular, they define individual, group, and disparate visibility and conduct experiments on real and synthetic datasets with various levels of homophily, the tendency of similar people to connect, measuring the disprate visibility resulting from various recommendation algorithms to find the effect of homophily on disparate visibility. In section 3 we expand their definitions of visibility for users to items and providers to study popularity bias.

3 Popularity Bias and Visibility in Multisided Recommendation

In this section we define visibility, a graph-based measure that represents the number of users that sees an item or a provider through the set of recommendations they receive. This clearly relates to popularity bias, and we further establish this relation by comparing the visibilities of two groups that are divided on the basis of popularity. Since we are interested in the multisided setting, we first give a graph definition of a multisided recommender system before providing definitions of visibility.

3.1 Multisided Recommendation Graph

Item recommender systems are inherently multisided. On one side, providers (suppliers) supply items to the system. On the other side, the system recommends items to consumers (users). The users interact with items, providing explicit or implicit feedback, which enters the system as input for future recommendations.

We represent the multisided recommendation system as a graph $G = (V, E)$. The vertex set $V = P \cup I \cup C$ consist of nodes representing a provider (P), item (I), or consumer (C). The sets P, I, C partition V. $E = \{E_S \cup E_R\} \subset V \times V$ is the set of directed edges such that there is an edge $(v_p, v_i) \in E_S$ for $v_p \in P$ and $v_i \in I$ if provider v_p supplies item v_i and an edge $(v_i, u) \in E_R$ for $v_i \in I$ and $u \in C$ if item v_i is recommended to user u.

We are interested in comparing the number of times the system recommends items supplied by popular providers to the number of times it recommends items supplied by non-popular providers, so we divide the providers each into two groups according to its initial popularity. For each provider v_p we find the sum of the number of ratings received by an item over all items supplied by v_p. We order the providers in decreasing order by this sum, and we put providers in the α_p-th percentile in P_1 for a given threshold α_p. The remaining providers are put in P_2. We also want to observe any difference, if any, between popularity bias in providers and in items, so we also count the number of times the system recommends popular items and the number of times it recommends nonpopular items. Similarly to the two provider groups, we order the items in decreasing order by the number of ratings it received, and we put items in the α_i-th percentile in I_1 for a given threshold α_i. The remaining providers are put in I_2.

We consider the item recommender represented by a function $\ell : (I \times C) \to [0, 5]$ which associates to each edge $(v_i, u) \in E_R$ a score $\ell(v_i, u) \in [0, 5]$ (as in a five star rating). In each round of recommendation the system recommends a set of items $R(u)$ to each user $u \in C$ where $|R(u)| = k$ for a given parameter $k \in \mathbb{N}^+$. $R(u)$ consists of the k items v_i with the highest values of $\ell(v_i, u)$.

3.2 Defining Visibility

In the multisided recommendation setting, we're interested in bias regards to both the items and the providers of those items. To that end we extend the notion of visibility defined for users in a social network [3] to items and providers. We consider one round of recommendation and focus on how many times each item and each provider appears in the recommendation sets of each user.

We call the number of times each item v_i appears in the recommendation sets of each user u the *item visibility* of v_i, and we denote it by

$$\phi(v_i) = |\{u \in C | v_i \in R(u)\}|.$$

For providers, we call the number of times each provider v_p appears in the recommendation sets of each user u the *provider visibility* of v_p, and we denote it by

$$\phi(v_p) = \sum_{(v_p, v_i) \in E_S} \phi(v_i).$$

We're particularly interested in comparing what portion of recommendations the popular nodes compare to the remaining nodes. For two groups of items I_1 and I_2, the visibility of a specific group j can be expressed as

$$\mathcal{V}(I_j) = \frac{1}{k|C|} \sum_{v_i \in I_j} \phi(v_i)$$

For two groups of providers P_1 and P_2, the visibility of a specific group j can be expressed as

$$\mathcal{V}(P_j) = \frac{1}{k|I|} \sum_{v_p \in P_j} \phi(v_p)$$

Since P_1 consist of the α_p-th percentile of the providers, the relative sizes of P_1 and P_2 are $1 - \alpha_p$ and α_p respectively. Similarly for items, the relative sizes of I_1 and I_2 are $1 - \alpha_i$ and α_i. We find the difference in the visibility of two groups with respect to the size of each group by normalizing each group's visibility by the relative group size. Following the social network setting in [3] we call this measure *disparate visibility*

$$\Delta(S) = \frac{S_1}{|s_1|} - \frac{S_2}{|s_2|},$$

where S represents either the providers P or the items I. This measure is close to 0 when the visibility received by the two groups is proportional to their relative sizes. A large positive value indicates that the system favors group 1 (the popular group), while a large negative value indicates that the system favors group 2 (the non-popular group).

4 Observations

We measure the disparate visibility of both providers and items for several distinct prediction algorithms to compare their effect on popularity bias. This experimentation was made difficult by the lack of datasets that include any information about the provider of the items.

4.1 Datasets Considered

We examined several item rating datasets as candidates for this project and narrowed it down to two sets. The first is the **Book-Crossing** dataset, collected from BookCrossing, a free online book club [8]. The dataset contains 278,858 users (with demographic information) providing 1,149,780 ratings about 271,379 books. The ratings consist of explicit scores that are on a scale of 1-10 (higher score is preferred) and implicit rating 0 that indicates that the book have not been rated by that user. Book-Crossing includes content-based information about the books, including the author and publisher which both can be reasonably interpreted as the provider of the books. Although Book-Crossing suits our needs, we found many typos, duplicates, and missing entries for both authors and publishers that made it difficult to accurately represent the supplier-item relationship in our multisided network. For the authors, the dataset contains 99,347 distinct entries. After removing extra white spaces, standardizing capitalization and punctuation, and grouping null values into one entry we reduced the number of distinct author entries to 96,942. However, there are still distinct entries that represent the same author, such as "Richard Barber" and "Richard W. Barber". Moreover, the experiments took too much time to run on this dataset due to the large size of the user-item matrix.

Below in describing the experimental setup we present the movie rating dataset we chose for this project.

4.2 Experimental setup

We analyze the ml-latest-small dataset (**ml-small**) collected from the MovieLens web site by GroupLens [5]. MovieLens is a movie recommendation service, and the dataset consists of 100,836 ratings of 9,742 movies by 610 users. The ratings are given a five star scale in half star increments (0.5 - 5.0). We chose ml-small over many of the larger MovieLens datasets available from GroupLens because ml-small provides the id of each movie in a movie database that we need to obtain the provider of each movie. The MovieLens 25M dataset, which also includes the id of each movie in a movie database, required more time to run than available.

For ml-small, the natural choice for items and consumers are the movies and the users of MovieLens. To analyze movie recommendation in the multisided recommender setting we also need information about the supplier of each movie. We choose the production company of each movie as the supplier because the production company (or a list of production companies if there are multiple) for a given movie is easily obtainable from ml-small. The original dataset contains the TMDB id for all but 8 movies. TMDB is a popular online database for movies and TV show, and by calling the TMDB api, we can fetch information, including a list of production companies, about a movie specified by its TMDB id. Using the TMDB api with the TMDB id for 9,734 movies inl we colected 8675 production companies that produced those movies.

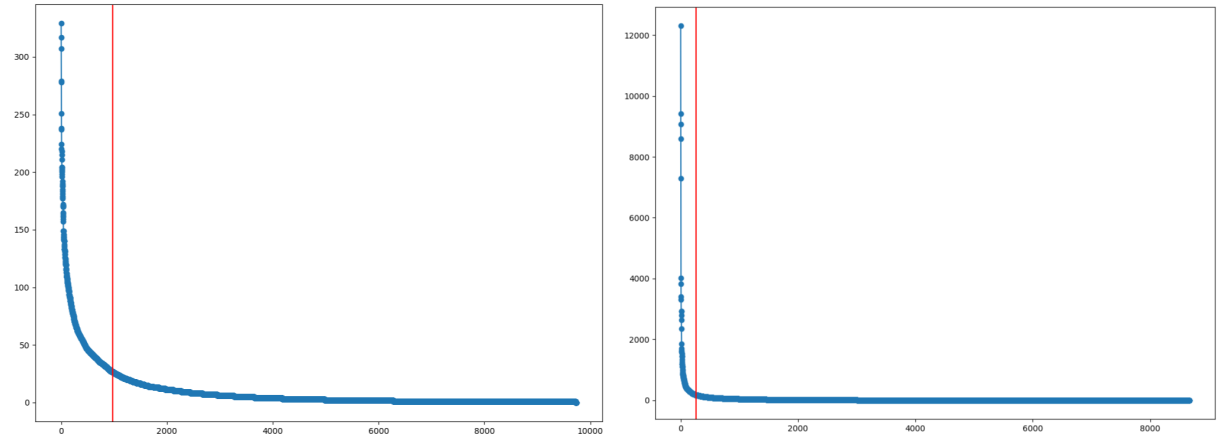
We consider three standard collaborative filtering methods **SVD**, **itemKNN**, and **userKNN** as well as a **random** predictor and a **baseline** estimate. Each algorithm calculates a prediction $\hat{r}_{ui}$ for a user u and an item i. We use the algorithms as implemented in Surprise [6] (for the KNN algorithms, we use the KNNBaseline implementation).

4.3 Splitting into groups by popularity

We divide the items and the providers each into two groups by putting "popular" items or providers into one group (I_1 and P_1) and putting the remaining items or providers into the other (I_2 and P_2). We start by plotting the number of ratings for each item (figure 1a) and the sum of the number of ratings for each item supplied by each provider (figure 1b). The red line on each figure indicate the respective threshold α. We select $\alpha_i = .9$ and $\alpha_p = .97$. Notice that the provider popularity distribution shows a longer tail, as reflected in higher threshold value chosen.

4.4 Results

Figure 2 shows the ratings predicted by the algorithms (y) against the actual ratings in the dataset (x) for each user and item. The left most plot (blue) shows no correlation between the predicted and actual ratings for the random algorithm, as expected since the random predictor should compute predicted ratings without



(a) Items (b) Providers

Figure 1: The "popularity" distribution of items and providers is plotted as the number of ratings each item or provider received in ml-small in decreasing order. The red line indicates α, the threshold that separates popular and nonpopular items/providers.

any regard for the actual ratings. For baseline, we observe that the predicted ratings tend to be higher than the actual ratings. The distribution of the predicted ratings with regards to the actual ratings appear to be similar for the SVD and the KNN algorithms, all with positive correlation between the predicted and actual ratings and variability in the predicted score that is greater for lower actual ratings (0.5-3) and less for higher actual ratings (3.5-5.0).

Table 1 shows the group visibility for popular and nonpopular provider groups as well as the disparate visibilities for each of the four prediction algorithms. Table 2 shows the same measures for item groups. We see the same overall trend between the providers and items. Namely, the disparate visibility is low for the random predictor, high for the baseline estimate, and the disparate visibility for the SVD algorithms. One notable difference between the provider and item visibilities is that the provider disparate visibility result for ItemKNN is lower than that for the random baseline for providers, while the item disparate visibility result for ItemKNN falls between the random predictor and the baseline estimate. For both providers and items, the UserKNN unexpectedly results in the lowest disparate visibility, even lower than the random predictor. The ItemKNN result for providers and the UserKNN results are surprising because we expect the random predictor to show the least amount of bias. We observe that the differences in disprate visibility among the four algorithms are smaller for providers than items, which we attribute to "lumping" the items under a smaller number of providers and overlap between providers that co-produce items.

Table 1: Provider Visibility

	$\mathcal{V}(P_1)$	$\mathcal{V}(P_2)$	$\Delta(\mathcal{V}(P))$
Random	0.090	0.077	2.905
Baseline	0.138	0.019	4.570
SVD	0.119	0.048	3.931
ItemKNN	0.080	0.077	2.594
UserKNN	0.032	0.095	0.965

Table 2: Item Visibility

	$\mathcal{V}(P_1)$	$\mathcal{V}(P_2)$	$\Delta(\mathcal{V}(P))$
Random	0.335	0.665	0.842
Baseline	1.000	0.000	5.000
SVD	0.838	0.162	4.209
ItemKNN	0.417	0.583	1.358
UserKNN	0.024	0.975	-1.099

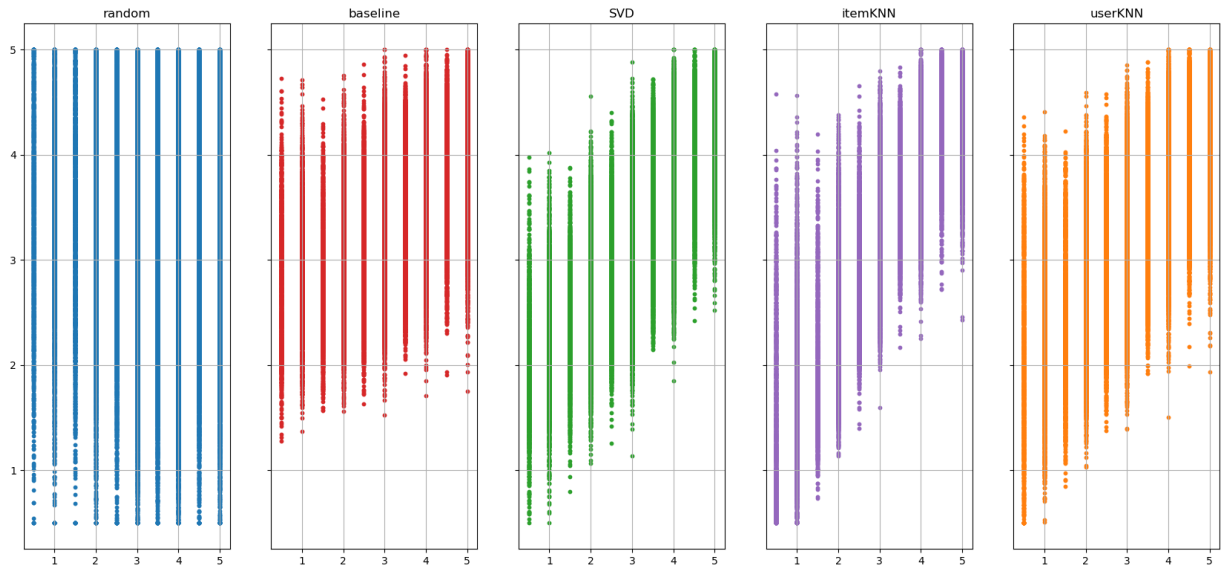
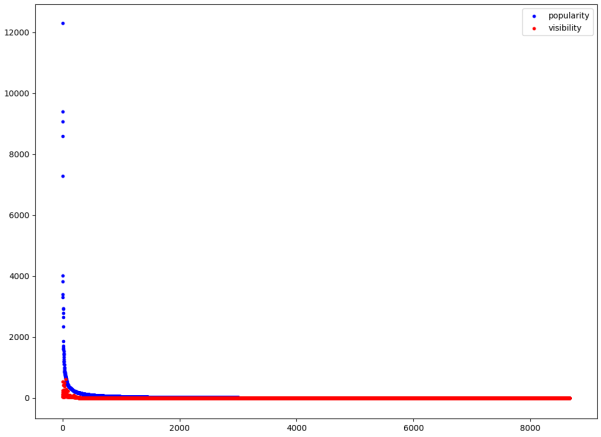


Figure 2: Ratings predicted by each of the algorithms (left to right: Random, Baseline, SVD, ItemKNN, UserKNN) are plotted against the actual ratings in ml-small.

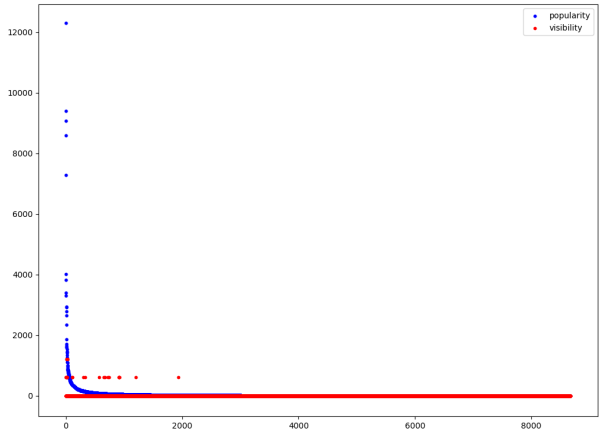
To further examine the unexpected result that the disparate visibility is the lowest for UserKNN we compare the visibility of each provider to its popularity in ml-small. We observe that the distribution of the visibilities of individual providers are generally similar to the distribution of their popularity in the sense that more popular providers tend to have higher visibility across all algorithms. The plot for UserKNN stands out because unlike the other plots there are some providers with very low popularities (towards the right end of the tail) that have higher visibility (red dots) than popularity (blue dots), which may be related to the low disparate visibility for this prediction algorithm.

5 Conclusion

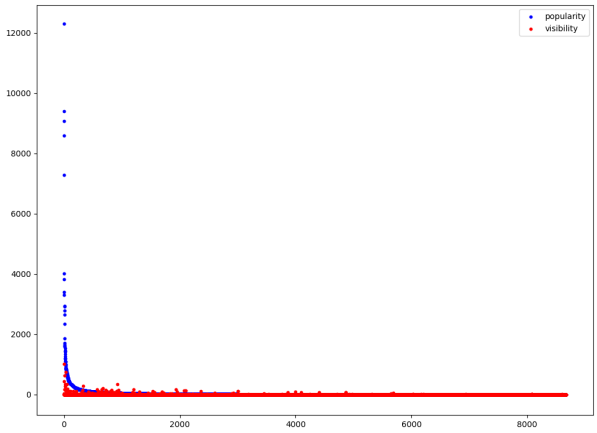
Popularity bias cause many recommendation algorithms to overly recommend popular items over less popular ones. In this work we shift the focus onto the popularity bias experienced by providers. For this purpose, we define disparate visibility for providers and used it as a way of measuring popularity bias resulting from recommendation algorithms. From our experiment on movie recommendations we see that common collaborative filtering methods do show in popularity bias for providers in a similar way as for items, although the effect may be less noticeable. One idea that we did not get to implement in this project is *k-step visibility*, inspired by the application of random walks to model user activity [4]. After getting the initial set of recommendations, we weight edges from an item to another item by probability proportional to its visibility. On this new graph, we simulate multiple rounds of recommendations (steps) as random walks from item to item in order to investigate how visibility is affected by the recommendation algorithm over time. We also want to investigate disparate visibility on groups separated by other characteristics about the provider, such as demographic information.



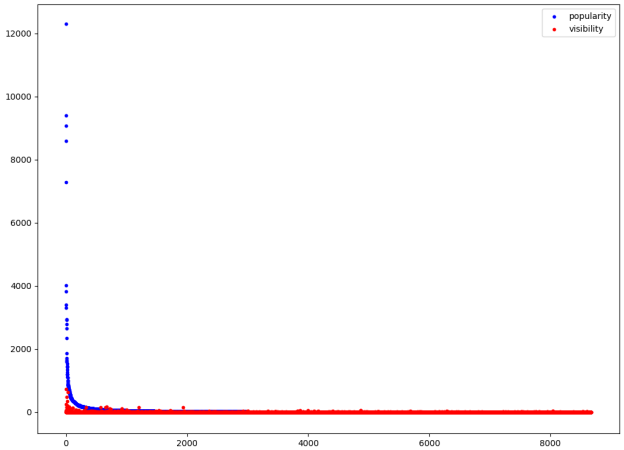
(a) Random



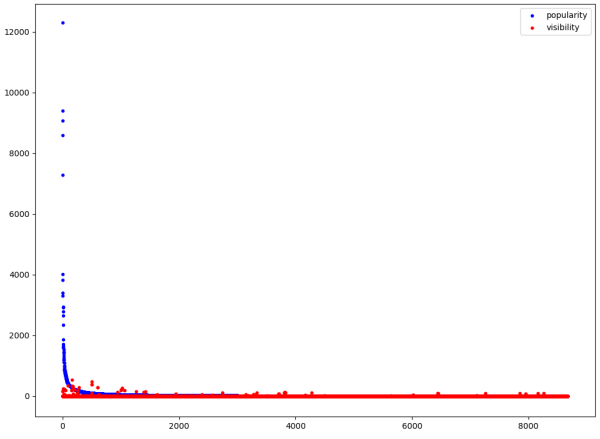
(b) Baseline



(c) SVD



(d) ItemKNN



(e) UserKNN

Figure 3: The blue dots show the number of ratings each provider received in the original dataset while the red dots show the number of recommendations each provider received as a result of each recommendation algorithm

References

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. *CoRR*, abs/1901.07555, 2019.

[2] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The unfairness of popularity bias in recommendation, 2019.

[3] Francesco Fabbri, Francesco Bonchi, Ludovico Boratto, and Carlos Castillo. The effect of homophily on disparate visibility of minorities in people recommender systems. *Proceedings of the International AAAI Conference on Web and Social Media*, 14(1):165–175, May 2020.

[4] Francesco Fabbri, Yanhao Wang, Francesco Bonchi, Carlos Castillo, and Michael Mathioudakis. Rewiring what-to-watch-next recommendations to reduce radicalization pathways, 2022.

[5] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015.

[6] Nicolas Hug. Surprise, a Python library for recommender systems. http://surpriselib.com, 2017.

[7] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. Fairmatch: A graph-based approach for improving aggregate diversity in recommender systems, 2020.

[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*, WWW '05, page 22–32, New York, NY, USA, 2005. Association for Computing Machinery.