How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility

Allison J.B. Chaney
Princeton University
Department of Computer Science
achaney@princeton.edu

Brandon M. Stewart
Princeton University
Department of Sociology
bms4@princeton.edu

Barbara E. Engelhardt
Princeton University
Department of Computer Science
bee@princeton.edu

ABSTRACT

Recommendation systems occupy an expanding role in everyday decision making, from choice of movies and household goods to consequential medical and legal decisions. The data used to train and test these systems is algorithmically confounded in that it is the result of a feedback loop between human choices and an existing algorithmic recommendation system. Using simulations, we demonstrate that algorithmic confounding can disadvantage algorithms in training, bias held-out evaluation, and amplify homogenization of user behavior without gains in utility.

1 INTRODUCTION

Recommendation systems are ubiquitous and impact many domains. One common application of these systems is online platforms for video, music, and product purchases through service providers such as Netflix, Pandora, and Amazon. Recommendation systems have the potential to influence how users perceive the world by filtering access to news, media, and books. Even more gravely, these systems impact crucial decision-making processes, such as loan approvals, criminal profiling, and medical interventions.

In the real world, these systems are updated regularly to incorporate new data and deployed systems are retrained based on observed data that is influenced by the recommendation system itself, forming a feedback loop (figure 1). It seems undesirable to ignore new data entirely, but the effects of *algorithmic confounding* should be understood. In this paper, we expose unintended consequences of algorithmic confounding in recommendation systems so that these effects may be countered.

The findings of this paper are relevant to a variety of individuals. *Recommendation system researchers* need to ensure that their models are evaluated convincingly to prove the efficacy of their systems for broader use and adoption; to do so, they need to consider these confounding factors in the generation of the data and account for them in both training and testing. *Social science researchers* look to online platforms as a rich source of data about human behavior; they should similarly account for algorithmic confounding when possible. *Practitioners* or *platform developers* who wish to increase user satisfaction with recommendations (either as an end, or as a means to an end, e.g., to increase platform engagement or sales) should account for algorithmic confounding when they update their recommendation algorithms. *Platform users* and *policy makers* may be concerned with the impact of recommendation systems on themselves as individuals or on society more broadly. Those interested in understanding the risks of these systems will be able to use the insights we provide to develop and suggest methods for greater transparency and accountability in these platforms.

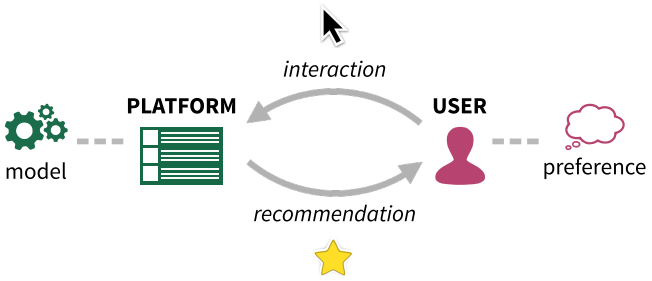

Figure 1: The feedback loop between user behavior and algorithmic recommendation systems. Confounding occurs when the model attempts to capture user preferences without accounting for recommendations. User preferences then influence both recommendations and interactions, obfuscating the causal impact of recommendations on behavior.

We begin with a summary of our claims (section 2). To provide evidence for these claims, we introduce a model for users interacting with recommendations (section 3); this allows us to analyze the impact of algorithmic confounding on simulated communities (section 4). We find that algorithmic confounding disadvantages some algorithms in training (section 4.2), biases held-out evaluation (section 4.3), and amplifies the homogenization of user behavior without gains in utility (section 4.4). We briefly discuss weighting approaches to account for these effects (section 5) and situate this work among related lines of inquiry (section 6) before we conclude (section 7). In an appendix, we outline a general framework for recommendation systems and frame the algorithms we study in this context (appendix A). This general recommendation system framework highlights the commonalities between seemingly distinct recommendation approaches, a contribution in itself.

2 CONSEQUENCES OF THE FEEDBACK LOOP

Real-world recommendation systems are often part of a feedback loop (figure 1): the underlying recommendation model is trained using data that are confounded by algorithmic recommendations from a previously deployed system. We attempt to *characterize* the impact of this feedback loop; we have three core findings.

Utility of Confounded Data (section 4.2). Algorithmically confounded data may be used to train recommendation models; some (but not all) algorithms perform well with confounded data, indicating these data have utility for training, but should be used with care.

Evaluation Using Confounded Data (section 4.3). When a recommendation system is evaluated using confounded held-out data,

results are biased toward recommendation systems similar to the confounding algorithm. This means that the choice of data can considerably impact held-out evaluation and subsequent conclusions.

Homogenization Effects (section 4.4). The recommendation feedback loop causes homogenization of user behavior, which is amplified with more cycles through the loop. Homogenization occurs at both a population level (all users behave more similarly) and at an individual level (each user behaves more like its nearest neighbors). Users with lower relative utility have higher homogenization.

3 INTERACTION MODEL

In order to reason about the feedback dynamics of algorithmic recommendations and user behavior, we need a model of how users engage with recommended items on a platform; we model engagement and not ratings, which is justified in section 6. We draw on a model recently proposed by Schmit and Riquelme [54] that captures the interaction between recommended items and users, and we modify this model to allow for personalized recommendations and multiple interactions for a given user.[1]

Definition 1. *The utility of user u consuming item i at time t is*

$$V_{ui}(t) = P_{ui}(t) + Q_{ui}(t), \tag{1}$$

where $P_{ui}(t)$ and $Q_{ui}(t)$ are the utilities that are known and unknown to the user, respectively. Neither utilities are known to the platform.

When a user considers whether or not they wish to engage with items, they have some notion of their own preferences; these preferences come from any information displayed and external knowledge. The quantity P captures these preferences that are known to the user but unknown to the recommendation system platform. Users rely on P in combination with the ordering of recommended content to select items with which to engage. Users must also have some unknown utility Q, or else they would be omniscient, and recommendation systems would be of no use. The underlying objective of these systems is to estimate the total utility V.

Assumption 1. *The utility of a user interacting with an item is approximately static over time, or $V_{ui}(t) \approx V_{ui}$.*

In the real world, utility fluctuates due to contextual factors such as user mood. However, the variance around utility is likely small and inversely related to the importance of the choice. Moving forward, we will omit the time notation for simplicity.

Assumption 2. *The total utility V_{ui} is beta-distributed,*[2]

$$V_{ui} \sim Beta'(\rho_u \alpha_i^\top), \tag{2}$$

and is parameterized by the dot product of user general preferences ρ_u for user u and item attributes α_i for item i.

This assumption will constrain utility values to be in the range $[0, 1]$; this representation is flexible because any utility with finite support can be rescaled to fall within this range. The use of the dot product to parameterize the utility is likewise a flexible representation; when the underlying vectors ρ and α have a dimensionality of $\min(|\mathcal{U}|, |\mathcal{I}|)$ and either preferences ρ or attributes α use a one-hot representation, then all possible utility values can be captured.

Assumption 3. *General preferences ρ and attributes α are fixed but unknown to the user or the recommendation system. They are drawn from Dirichlet distributions, or*

$$\rho_u \sim Dirichlet(\mu_\rho) \qquad and \qquad \alpha_i \sim Dirichlet(\mu_\alpha), \tag{3}$$

for all users $u \in \mathcal{U}$ and all items $i \in \mathcal{I}$, respectively. Individual preferences are parameterized by a vector of global popularity of preferences μ_ρ over all users. Individual item attributes are similarly parameterized by a global popularity of attributes μ_α over all items.

A draw from the Dirichlet distribution produces a vector that sums to one, allowing for easy interpretation; this distribution is also flexible enough to capture a range of possible preferences and attributes, including the one-hot representation discussed previously. Further, assumption 2 requires $\rho_u \alpha_i^\top \in [0, 1]$ and this guarantees that ρ and α will satisfy this constraint. Most importantly, when aggregated by user (a proxy for activity) or item (popularity), this construction produces a distribution of utility values with a long tail, as seen in real platforms [10].

Assumption 4. *The proportion of the utility known to user u is $\eta_{ui} \sim Beta'(\mu_\eta)$;*[3] *this results in*

$$P_{ui} = \eta_{ui} V_{ui} \quad and \quad Q_{ui} = (1 - \eta_{ui}) V_{ui}. \tag{4}$$

This assumption implies that each user is approximately consistent at assessing utility, but introduces some uncertainty so that the known utility P is a noisy approximation of the true utility V. While each user could theoretically have a different mean proportion μ_η, in practice this is not important because the known utilities P are not compared across users. This representation is simple; however, a more realistic model might vary the known proportion of utility as a function of item attributes or include a time dependency. These additional complexities would likely amplify the effects our findings or have no impact.

Assumption 5. *At every discrete time step t, each user u will interact with exactly one item, $i_u(t)$.*

Users in the real world have varying levels of activity; we argue that the long tail of utility (see the justification for assumption 3) captures the essence of this behavior and that we could adopt different levels of user activity without substantially altering our results.

Definition 2. *To select an item at time t, user u relies on her own preferences P_{ui} and a function f of the rank of the items provided by recommendation system $\Re$.*[4] *The chosen item is*

$$i_u^{\Re}(t) = \arg\max_i \left(f\left(rank_{u,t}^{\Re}(i) \right) \cdot P_{ui}(t) \right), \tag{5}$$

where $rank_{u,t}^{\Re}(i) = n$ such that $r_{u,n}(t) = i$, according to recommender system $\Re$'s ordering of items, as described in appendix A.

[1] Unlike the Schmit and Riquelme model [54], we include no additional noise term because we model utility (or "quality" in the Schmit and Riquelme model) probabilistically instead of holding it fixed.

[2] We use an atypical parameterization of the beta distribution with mean μ and fixed variance σ and distinguish this parameterization as Beta$'$. For our simulations in section 4, we set $\sigma = 10^{-5}$. To convert to the standard parameterization, $\alpha = \left(\frac{1-\mu}{\sigma^2} - \frac{1}{\mu}\right)\mu^2$ and $\beta = \alpha\left(\frac{1}{\mu} - 1\right)$.

[3] Mean proportion $\mu_\eta = 0.98$ in our simulations (section 4).

[4] For our simulations (section 4), we used $f(n) = n^{-0.8}$, which approximates observed effects of rank on click-through rate [27]; our results held for other choices of f.

Users are more likely to click on items presented earlier in a ranked list [27]; the function f captures this effect of rank on the rate of interaction. In keeping with our earlier discussion, to allow for various levels of user activity, one need only add some threshold τ such that if the function of rank and preference $P_{ui}(t)$ inside equation (5) are less than this threshold, then no interaction occurs.

Assumption 6. *Each user u interacts with item i at most once.*

When a user engages with an item repeatedly, utility decreases with each interaction. This is the simplest assumption that captures the notion of decreasing utility; without it, a generally poor recommendation system might ostensibly perform well due to a single item. The interaction model could alternatively decrease utility with multiple interactions, but this would not alter results significantly.

Assumption 7. *New and recommended items are interleaved.*

As in the real world, new items are introduced with each time interval. When no recommendation algorithm is in place (early "start-up" iterations), the system randomly recommends the newest items. Once a recommendation algorithm is active, we interleave new items with recommended items; this interleaving procedure is a proxy for users engaging with items outside of the recommendation system, or elsewhere on the platform. Since this procedure is identical for all systems, it does not impact the comparison across systems.

4 SIMULATED COMMUNITIES

In this section, we explore the performance of various recommendation systems on simulated communities of users and items. We first describe the simulation procedure, and then discuss three claims.

4.1 Simulation Procedure

We consider six recommendation algorithms: *popularity*, *content filtering* ("content"), *matrix factorization* ("MF"), *social filtering* ("social"), *random*, and *ideal*. Appendix A provides further details the first four approaches and describes our general recommendation framework. The core idea of this framework is that each recommendation system provides some underlying score $s_{ui}(t)$ of how much a user u will enjoy item i at time t. These scores are constructed using user preferences $\theta_u(t)$ and item attributes $\beta_i(t)$:

$$s_{ui}(t) = \theta_u(t)\beta_i(t)^\top, \tag{6}$$

and each recommendation approach has a different way of constructing or modeling these preferences and attributes.

For our simulations, all of the six approaches recommend from the set of items that exist in the system at the time of training; *random* recommends these items in random order. *Ideal* recommends items for each user u based on the user's true utility V_{ui} for those items. Comparison with these two approaches minimizes the impact of the interaction model assumptions (section 3) on our results.

In all of our simulations, a community consists of 100 users and is run for 1,000 time intervals with ten new items being introduced at each interval; each simulation is repeated with ten random seeds and all our results are averages over these ten "worlds."

We generate the distributions of user preference and item attribute popularity, as used in equation (3), in $K = 20$ dimensions; we generate uneven user preferences, but approximately even item attributes. The user preference parameter is generated as follows:

$$\tilde{\mu}_\rho \sim \text{Dirichlet}(\mathbf{1}) \qquad \text{and} \qquad \mu_\rho = 10 \cdot \tilde{\mu}_\rho. \tag{7}$$

This mirrors the real world where preferences are unevenly distributed, which allows us to expose properties of the recommendation algorithms. Item attribute popularity is encouraged to be more even in aggregate, but still be sparse for individual items; we draw:

$$\tilde{\mu}_\alpha \sim \text{Dirichlet}(\mathbf{100}) \qquad \text{and} \qquad \mu_\alpha = 0.1 \cdot \tilde{\mu}_\alpha. \tag{8}$$

While item attributes are not evenly distributed in the real world, this ensures that all users will be able to find items that match their preferences. With these settings for user preferences and item attributes, the resulting matrix of true utility is sparse (e.g., figure 2), which matches commonly accepted intuitions about user behavior.

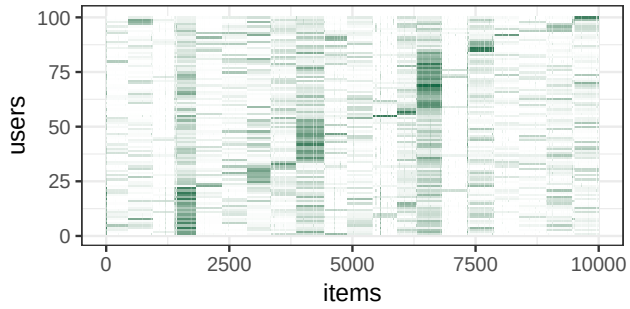

Figure 2: Example true utility matrix V for simulated data; darker is higher utility. The distribution of user preferences is disproportionate, like the real world, and the structure is easily captured with matrix factorization.

We generate social networks using the covariance matrix of user preferences; we impose that each user must have at least one network connection and binarize the covariance matrix using this criteria. This procedure enforces that the network is homophilous, which is generally (but not always) true in the real world.

We consider two cases of observing user interactions with items: a simple case where each recommendation algorithm is trained once, and a more complicated case of repeated training; this allows us to compare a single cycle of the feedback loop (figure 1) to multiple cycles. In the simple paradigm, we run 50 iterations of "start-up" (new items only each iteration), train the algorithms, and then observe 50 iterations of confounded behavior. In the second paradigm, we have ten iterations of "start-up," then train the algorithms every iteration for the remaining 90 iterations using all previous data.

4.2 Utility of Confounded Data

When data is collected from a platform where users are recommended content, the resulting algorithmically confounded user behavior data may be used to train a new recommendation system (or update an old one). We wish to identify which algorithms can use confounded data to increase utility for users.

Because our interaction model (section 3) includes an explicit notion of utility, we can simply observed the utility in our simulations and compare recommendation algorithms to randomly recommended content. If an algorithm trained with confounded

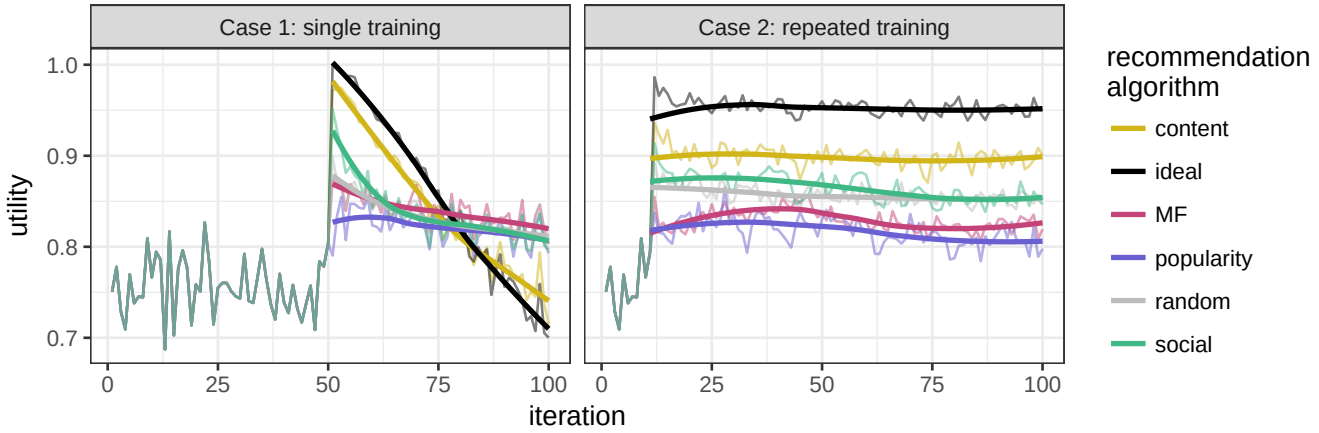

Figure 3: Average utility over all users experienced at each iteration. On the left, utility slowly decreases as the pool of high-utility items are consumed. On the right, repeated training allows users to access new items. Content and social filtering outperform randomly reintroducing old items, whereas matrix factorization and popularity perform worse than random, indicating that the additional social network and item content information make these algorithms more robust to confounding.

data results in higher utility than random recommendations, then confounded data are "useful" in training that algorithm.

We found that content and social filtering made best use of confounded data (figure 3); this is likely because the content information and social network ground the algorithms in more static representations of the world. While matrix factorization cumulatively outperformed random recommendations in the single training case (figure 4), both MF and popularity under-performed random recommendations in the repeated training case, indicating that naively training on confounded data can be disadvantageous in practice.

We conclude that using confounded data for training may still increase the utility of recommendation platforms, but it is not universally beneficial. This means that practitioners can still train models on confounded data and see improvements in user satisfaction (and therefore other metrics like click-through-rate and revenue), but that in some cases performance may be less optimal.

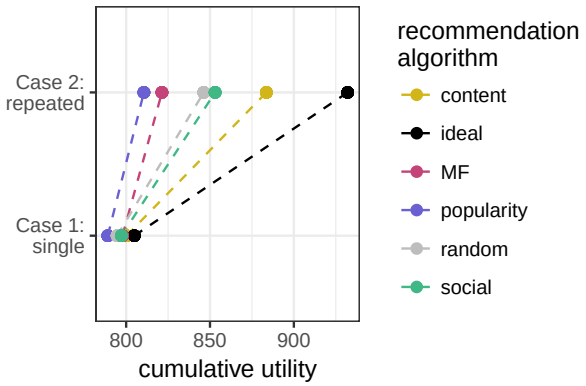

Figure 4: A comparison of the cumulative utility, averaged across all users. The change in random recommendations captures improvement due to the difference in item availability between the two cases. Matrix factorization and popularity increase utility at a lower rate than random, indicating that the confounded data is not useful for training. Content and social filtering increase at the same or higher rate than random, indicating that confounded data is useful or neutral for training these algorithms.

4.3 Evaluation Using Confounded Data

Recommendation system are often evaluated using confounded held-out data. This form of offline evaluation is used by both researchers and practitioners alike, but the choice is not as innocuous as one might expect. When a recommendation system is evaluated using confounded held-out data, results are biased toward recommendation systems similar to the confounding algorithm. Researchers can, intentionally or unintentionally, select data sets that highlight their proposed model, thus overstating its performance.

To illustrate this point, we simulated identical communities interacting with social filtering and matrix factorization recommendations. After generating confounded data, we compared the held-out evaluation of the confounding algorithms using 80% of the data for training and 20% for testing.We used normalized discounted cumulative gain (nDCG), a typical rank-based evaluation metric (see section 6). The unnormalized variant of this metric is

$$\mathrm{DCG}_u = \sum_{n=1}^{|I|} \frac{\mathbb{1}\left[r_{u,n} \in \mathcal{H}_u\right]}{\log_2(n+1)}, \tag{9}$$

where $\mathcal{H}_u$ is the set of held-out items for user u and $r_{u,n}$ is the nth recommended item for user u, as defined in appendix A, equation (14). This is then normalized

$$\mathrm{nDCG}_u = \frac{\mathrm{DCG}_u}{\text{ideal } \mathrm{DCG}_u}. \tag{10}$$

.

We found that held-out evaluation favored the algorithm used to confound the data: social factorization outperformed matrix factorization (MF) when evaluated on data confounded by social recommendations and MF had superior held-out performance when evaluated on data confounded by MF (figure 5). This held for both the simple single-training case and for the case of repeated cycles through the feedback loop. The underlying models of utility in the simulations were identical, so one might naively assume that the training data would be approximately equivalent no matter the confounding algorithm; instead, confounded data yielded conflicting results. These results demonstrate that held-out evaluation using confounded data can be unreliable and that the choice of data matters.

There are specific instances in recommendation system literature where this may be troubling. For example, *MovieLens* is a research website that uses collaborative filtering to make movie recommendations [47]; the researchers behind it have released several public datasets[5] that are used prolifically in evaluation recommendation systems [23]. Recommendation systems based on collaborative filtering will likely perform better on this data.

Douban is a social networking service that recommends popular content to users; there is a publicly available data set from this platform that is used to evaluate many recommendation systems [43]. Because of the popularity features on this website, algorithms that include a notion of popularity will perform better on this data.

Facebook and *Etsy* are among websites that use social networks to recommend content. Many algorithms use social network information to improve recommendations [13, 22, 26, 41, 42, 65] but when training and evaluating these algorithms on data from social platforms, it is not clear if the models capture the true preferences of users, or if they capture the biasing effects of platform features.

These biases are problematic for researchers and platform developers who attempt to evaluate models offline or rely on academic publications that do.

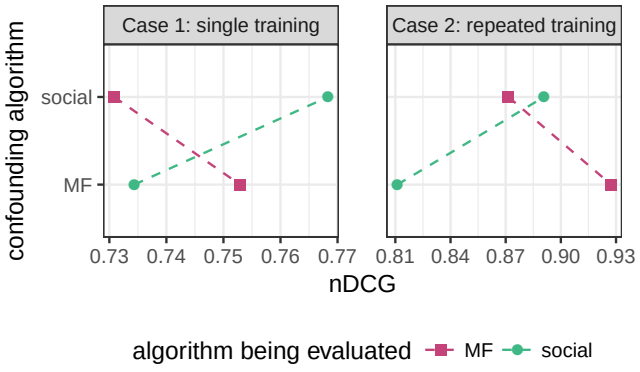

Figure 5: Held-out evaluation (using nDCG; higher is better) favors the algorithm used to confound the data. In both cases, social filtering gave higher true user utility.

4.4 Homogenization Effects

Recommendation systems may not change the underlying preferences of user (especially not when used on short time scales), but they do impact user behavior, or the collection of items with which users interact. Recommendation algorithms encourage similar users to interact with the same set of items, therefore homogenizing their behavior, relative to the same platform without recommended content. For example, *Popularity-based* systems represent all users in the same way; this homogenizes all users, as seen in previous work [11, 62]. *Social* recommendation systems homogenize connected users or within cliques, and *matrix factorization* homogenizes users along learned latent factors; in this latter case, the homogeneity of user interactions will increase as the cosine similarity in their latent representations increases.

Homogenizing effects are not inherently bad as they indicate that the models are learning patterns from the data, as intended. However, homogenization of user behavior does not correspond directly with an increase in utility. There is an optimum amount of homogenization for a given user representation, or we can observe an increase in homogenization without a corresponding increase in utility. This is related to the explore/exploit paradigm, where we wish to exploit the user representation to maximize utility, but not to homogenize users more than necessary. When a representation of users is over-exploited, users are being pushed to be have similar behaviors when, based on their true preferences, they would enjoy a broader range of items. This would indicate that the "tyranny of majority" and niche "echo chamber" effects may both be manifestations of the same problem: over-exploitation of recommendation models.

We measure homogenization of behavior by first pairing each user with their most similar user according to the recommendation system, or user u is partnered with user v that maximizes the cosine similarity of θ_u and θ_v. Next, we compute the Jaccard index of the sets of observed items for these users. If at time t, user u has interacted with a set if items $\mathcal{D}_u(t) = \{i_u(1), \ldots, i_u(t)\}$ the Jaccard index of the two users' interactions can be written as

$$\mathbf{J}_{uv}(t) = \frac{|\mathcal{D}_u(t) \cap \mathcal{D}_v(t)|}{|\mathcal{D}_u(t) \cup \mathcal{D}_v(t)|}. \tag{11}$$

We compared the Jaccard index for paired users against the Jaccard index of that same set of users exposed to ideal recommendations; this difference captures how much the behavior has homogenized relative to ideal. Figure 6 shows these results for both the single training and the repeated training cases. We found that in the single training case, users became slightly homogenized after training, but returned to the ideal homogenization with time. For the repeated training, all recommendation systems (except random), homogenized user behavior beyond what was needed to achieve ideal utility. As the number of cycles in the feedback loop (figure 1) increases, we observe homogenization effects continue to increase, without corresponding increases in utility (figure 3).

We can also consider global homogenization to reveal the impact of the feedback loop at the population level; instead of comparing to paired users based on θ_u, we can compare users matched randomly (figure 7). In this setting, we find that all recommendation systems (except, again, random) increased global homogeneity of

[5] https://grouplens.org/datasets/movielens/

user behavior. The popularity system increased homogeneity the most; after that, matrix factorized and social filtering homogenized users comparably, and content filtering homogenized randomly pair users least of all, but still more than ideal.

We have shown that when practitioners update their models without considering the feedback loop of recommendation and interaction, they encourage users to consume a more narrow range of items, both in terms of local niche behavior and global behavior.

Changes in utility due to these effects are not necessarily born equally across all users. For example, users whose true preferences are not captured well by the low dimensional representation of user preferences may be disproportionately impacted. These minority users may see lesser improvements or even diminishes in utility when homogenization occurs. Figure 8 breaks down the relationship between homogenization and utility by user; for all recommendation algorithms, we find that users who experience lower utility generally have higher homogenization with their nearest neighbor.

Note that we have assumed that each user/item pair has fixed utility (assumption 1). In reality, a collection of similar items is probably less useful than a collection of diverse items [18]. With a more nuanced representation of utility that includes the collection of interactions as a whole, these effects would likely increase in magnitude.

5 ACCOUNTING FOR CONFOUNDING

Researchers and practitioners alike would benefit from methods to address these concerns. Weighting techniques, such as those proposed by Schnabel, et al. [55] and De Myttenaere, et al. [17] for offline evaluation seem promising. We performed a preliminary exploration of weighting techniques and found that in the repeated training case, weighting can simultaneously increase utility and decrease homogenization. These weighting techniques could also be of use when attempting to answer social science questions using algorithmically confounded user behavior data. We leave a full exploration of these methods for future work.

As proposed by Bottou, et al. [8], formal causal inference techniques can assist in the design of deployed learning systems to avoid confounding. This would likely reduce the effects we have described (section 4), but needs to be studied in greater depth. Regardless, practitioners would do well to incorporate measures to avoid confounding, such as these. At the very least, they should log information about the recommendation system in deployment along with observations of behavior; this would be useful in disentangling recommendation system influence from true preference signal as weighting and other techniques are refined.

6 RELATED WORK

Bias, confounding, and estimands. Schnabel, et al. [55] note that users introduce selection bias; this occurs during the interaction component of the feedback loop shown in figure 1. They consider a mechanism for interaction in which users first select an item and then rate it. Other work also considers similar notions of missingness in rating data [44, 66]. However, many platforms exist where users express their preferences implicitly by viewing or reading content, as opposed to explicitly rating it. In the implicit setting, the observations of user behavior are the selections themselves. The quantity of interest (estimand) is no longer the rating of an item, but the probability of the user selecting an item. Thus, we no longer wish to correct for the user preference aspect of selection bias; instead, we wish to predict it.

Recommendation systems introduce confounding factors in this setting; it is difficult to tell which user interactions stem from users' true preferences and which are influenced by recommendations. The core problem is that recommendation algorithms are attempting to model the underlying user preferences, making it difficult to make claims about user behavior (or use the behavior data) without accounting for algorithmic confounding. In this paper, we describe various problems that arise from using data confounded by recommendation systems. Among these problems is offline evaluation using confounded data; De Myttenaere, et al. [17] propose addressing this with weighting techniques and Li, et al. [38] propose a method specifically for reducing this bias in contextual bandit algorithms.

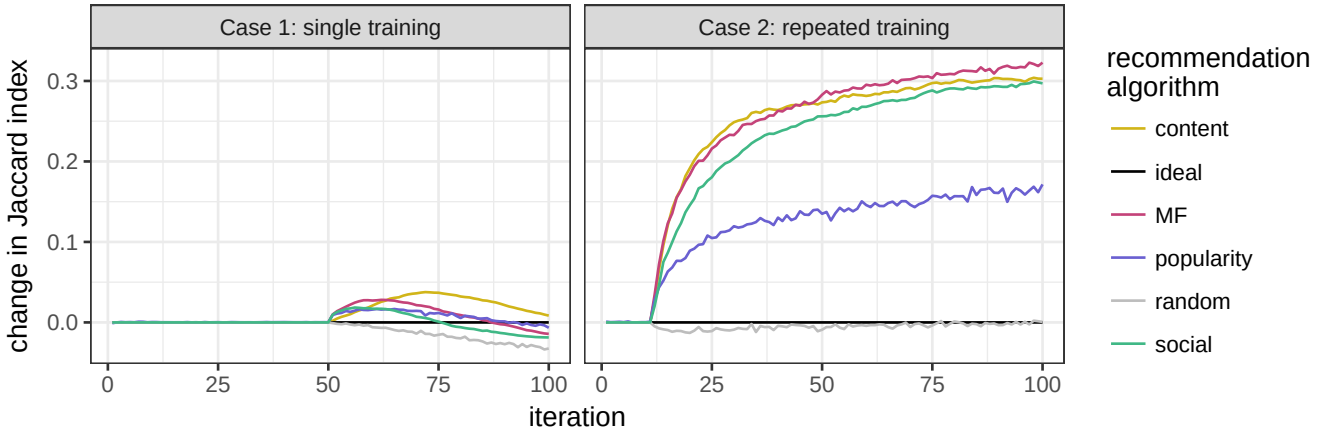

Figure 6: Change in Jaccard index of user behavior relative to ideal behavior; users paired by cosine similarity of θ. On the left, mild homogenization of behavior occurs soon after a single training, but then diminishes. On the right, recommendation systems that include repeated training homogenize user behavior more than is needed for ideal utility (compare figure 3).

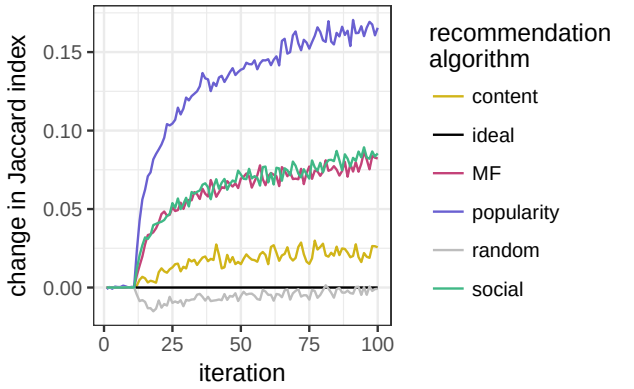

Figure 7: For the repeated training case, change in Jaccard index of user behavior relative to ideal behavior; users paired randomly. Popularity increases homogenization the most globally, but all non-random recommendation algorithms also homogenize users globally.

Previous work has investigated how many algorithms rely on data that is imbued with societal biases and explored how to address this issue [3, 12, 53, 60]. This work is complementary to these efforts as the described feedback effects may amplify societal biases. Due to these and other concerns, regulations are emerging to restrict automated individual decision-making, such as recommendation systems [21]; this work will aid in making such efforts effective.

Evaluating recommendation systems. Rating prediction is a common focus for recommendation algorithms, owing its popularity at least in part to the Netflix challenge [4, 5], which evaluated systems using RMSE on a set of held-out data. In practice, however, recommendation systems are deployed to rank items. Even Netflix has moved away from its original 5-star system in favor of a ranking-friendly thumbs-up/down interaction [49] and now advocates for ranking items as the primary recommendation task, as it considers *all items in a collection* during evaluation instead of *only held-out observed items* [58].

Simply put, top-n accuracy metrics such as precision, recall, and nDCG are better for evaluating the real-world performance of these systems [14]. Accuracy metrics are popular because they are straightforward to understand and easy to implement, but still they do not necessarily capture the usefulness of recommendations; there are a variety of system characteristics that are thought to be important, along with methods for evaluating them [24].

Among these characteristics, diversity is often framed as a counterpart to accuracy [28, 40, 57, 69]. Diversity can be considered at multiple levels: in aggregate, within groups of users, and individually. Many efforts have been made to understand whether or not various recommender systems impact diversity by reinforcing the popularity of already-popular products or by increasing interactions with niche items [2, 11, 15, 19, 20, 46, 48, 50, 62, 67, 68]. These systems also have the potential to create "echo-chambers," which result in polarized behavior [16].

Causality in recommendation systems. Formal causal inference techniques have only recently been applied to recommendation systems. Liang, et al. [39] draw on the language of causal analysis in describing a model of user exposure to items; this is related to distinguishing between user preference and our confidence in an observation [25]. Some work has also been done to understand the causal impact of these systems on behavior by finding natural experiments in observational data [56, 59] (approximating expensive controlled experiments [32]), but it is unclear how well these results generalize. As previously mentioned, Schnabel, et al. [55] use propensity weighting techniques to remove users' selection bias for explicit ratings. Bottou, et al. [8] use ad placement as an example to motivate the use of causal inference techniques in the design of deployed learning systems to avoid confounding; this potentially seminal work does not, however, address the use of already confounded data (e.g., to train and evaluate systems or ask questions about user behavior), which is abundant.

Connections with the explore/exploit trade-off. In considering the impact of recommendation systems, some investigations model temporal dynamics [34] or frame the system in an explore/exploit paradigm [37, 63]. Recommendation systems have natural connections with the explore/exploit trade-off; for example, should a system recommend items that have high probability of being consumed under the current model, or should the system recommend low probability items in order to learn more about a user's preferences? Reinforcement learning models already build in some notion of a feedback loop to maximize reward. One major challenge with this setup, however, is knowing how to construct the reward functions. Usually the reward is based on click-through rate or revenue for companies; we, however, focus on utility for the users of a platform. Our analysis and simulations may be informative for the construction of reward functions in reinforcement-style recommendation systems.

7 CONCLUSION

We have explored the impact of algorithmic confounding on a range of simulated recommendation systems. We found that algorithmic confounding disadvantages some algorithms in training (section 4.2), biases held-out evaluation (section 4.3), and amplifies the homogenization of user behavior without corresponding gains in utility (section 4.4). These findings have implications for any live recommendation platform; those who design these systems need to consider how a system influences its users and how to account for this algorithmic confounding. Researchers who use confounded data to test and evaluate their algorithms should also be aware of these effects, as should researchers who wish to use confounded data to make claims about user behavior from a social science perspective. Platform users and policy makers should take these effects into consideration as they make individual choices or propose policies to guide or govern the use of these algorithms.

A RECOMMENDATION FRAMEWORK

In this appendix, we cast ostensibly disparate recommendation methods into a general mathematical framework that encompasses many standard algorithmic recommendation techniques.

A recommendation system provides some underlying score $s_{ui}(t)$ of how much a user u will enjoy item i at time t. This score is generated from two components: the system's representations at

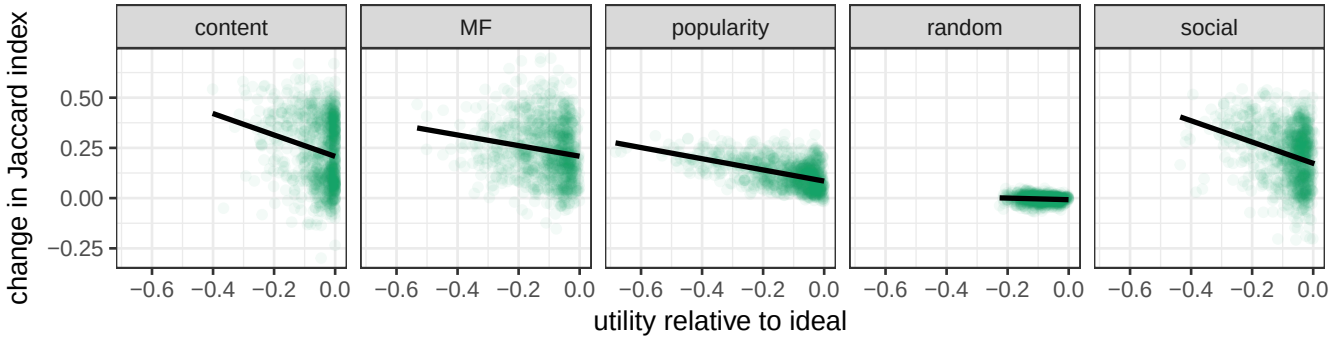

Figure 8: For the repeated training case, change in Jaccard index of user behavior, relative to ideal behavior, and shown as a function of utility relative to the ideal platform; users paired by cosine similarity of θ. Each user is shown as a point, with a linear fit to highlight the general trend that users who experience losses in utility have higher homogenization.

time t of both user preferences $\theta_u(t)$ for user u and item attributes $\beta_i(t)$ for item i. The dot product of these vectors produces the score:

$$s_{ui}(t) = \theta_u(t)\beta_i(t)^\top. \tag{12}$$

This construction is reminiscent of the notation typically used for the matrix factorization approach discussed in appendix A.4, but it also provides us a more general framework.

The scores $s_{ui}(t)$ are not necessarily comparable across recommendation techniques. Recommendation systems that focus on *rating prediction* will provide scores comparable to other rating prediction models. For these systems, the goal is to predict how users will explicitly rate items, most commonly on a five-star scale, and are typically evaluated with prediction error on held-out ratings. Low errors for predicting ratings, however, do not always correspond to high accuracy in rank-based evaluation [14, 58] (e.g., "what should I watch next?"), which is the ultimate goal in many recommendation applications. Additionally, limiting our scope to rating prediction systems would omit models that focus on learning rankings [31] or that otherwise produce rankings directly, such as ordering items by popularity.

Given a collection of scores $s_{ui}(t)$, a recommendation system then produces, for each user, an ordered list (or sequence) of items sorted according to these scores. Formally, we represent these recommendations as a set of sequences

$$\left\{\left(r_{u,n}(t)\right)_{n=1}^{|\mathcal{I}|}\right\}_{u\in\mathcal{U}}, \tag{13}$$

where $\mathcal{I}$ is the set of all items and $\mathcal{U}$ is the set of all users. For each user u, the system provides a sequence, or ranked list of items, where n is the position in the ranked list and $r_{u,n}$ is the recommended item for user u at rank n. This sequence of items for a given user u is defined as all items sorted descendingly by their respective score $s_{ui}(t)$, or

$$\left(r_{u,n}(t)\right)_{n=1}^{|\mathcal{I}|} = \text{descending sort}\big(\forall i \in \mathcal{I}, \text{by} = s_{ui}(t)\big). \tag{14}$$

Our simulated experiments (section 4) revealed that it is important to break ties randomly when performing this sort; if not, the random item recommender baseline receives a performance advantage on early iterations by exposing users to a wider variety of items.

We now cast a collection of standard recommendation systems in this framework by defining the user preferences $\theta_u(t)$ and item attributes $\beta_i(t)$ for each system; this emphasizes the commonalities between the various approaches.

A.1 Popularity

Intuitively, the popularity recommendation system ranks items based on the overall item consumption patterns of a community of users. All users are represented identically, or $\theta_u(t) = 1$ for all $u \in \mathcal{U}$, and thus every user receives the same recommendations at a given time t. Item attributes are based on the interactions of users with items up to time t; these interactions can be structured as a collection of N triplets $\{(u_n, i_n, t_n)\}_{n=1}^N$, where each triplet (u, i, t) indicates that user u interacted with item i at time t.

There are many permutations of the popularity recommendation technique, including windowing or decaying interactions to prioritize recent behavior; this prevents recommendations from stagnating. For our analysis (section 4), we employ the simplest popularity recommendation system; it considers all interactions up to time t, or

$$\beta_i(t) = \sum_{n=1}^{N} \mathbb{1}\left[i_n = i \text{ AND } t_n < t\right].^{6} \tag{15}$$

A.2 Content Filtering

Content-based recommender systems match attributes in a user's profile with attribute tags associated with an item [51, ch. 3]. In the simplest variant, the set of possible attribute tags $\mathcal{A}$ are identical for both users and items. Then, user preferences $\theta_u(t)$ is a vector of length $|\mathcal{A}|$, and the attribute tags for a given user are in the set $\mathcal{A}_u(t)$; this gives us

$$\theta_{ua}(t) = \mathbb{1}\left[a \in \mathcal{A}_u(t)\right]. \tag{16}$$

The item attribute tags can similarly be represented as a vector of length $|\mathcal{A}|$ with values

$$\beta_{ia}(t) = \mathbb{1}\left[a \in \mathcal{A}_i(t)\right], \tag{17}$$

where $\mathcal{A}_i(t)$ is the set of attributes for an item i.

Attributes for both users and items can be input manually (e.g., movie genre), or they can be learned independent of time t with, for example, a topic model [6] for text or from the acoustic structure

[6] The indicator function notation $\mathbb{1}[\text{ EXPRESSION }]$ evaluates to 1 when the internal expression is true and 0 when it is false.

of a song [61] for music; when learned, the attribute tags can be real-valued instead of binary. For our simulations (section 4), we use binary item attributes and learn real-valued user preferences.[7]

A.3 Social Filtering

Social filtering recommendation systems rely on a user's social network to determine what content to suggest. In the simplest variant, the user preferences $\boldsymbol{\theta}(t)$ are a representation of the social network, or a $|\mathcal{U}| \times |\mathcal{U}|$ matrix; for each user u connected to another user v,

$$\theta_{uv}(t) = \mathbb{1}\left[v \in \mathcal{F}_u(t)\right], \tag{18}$$

where $\mathcal{F}_u(t)$ is the set of people u follows (directed network) or with which they are connected as "friends" (undirected network) as of time t. Alternatively, the user preferences $\boldsymbol{\theta}(t)$ can represent the non-binary trust between users, which can be provided explicitly by the user [45] or learned from user behavior [13]; we use the latter in our analysis (section 4).

Item attributes $\boldsymbol{\beta}(t)$ are then a representation of previous interactions, broken down by user, or an $|\mathcal{I}| \times |\mathcal{U}|$ matrix where for each item i and user v,

$$\beta_{iv}(t) = \mathbb{1}\left[v \in \mathcal{U}_i(t)\right], \tag{19}$$

where $\mathcal{U}_i(t)$ is the set of users which have interacted with item i as of time t. The item representation $\boldsymbol{\beta}(t)$ can alternatively be a non-binary matrix, where $\beta_{iv}(t)$ is the number of interactions a user v has with an item i, or user v's rating of item i.

A.4 Collaborative Filtering

Collaborative filtering learns the representation of both users and items based on past user behavior, and is divided into roughly two areas: neighborhood methods and latent factor models.

Neighborhood Methods. The simplicity of neighborhood methods [51, ch. 4] is appealing for both implementation and interpretation. These approaches find similar users, or neighbors, in preference space; alternatively, they can find neighborhoods based on item similarity. In either case, these methods construct similarity measures between users or items and recommend content based on these measures. We outline the user-based neighborhood paradigm, but the item-based approach has a parallel construction.

Users are represented in terms of their similarity to others, or

$$\theta_u(t) = \left[w_{u1}, \ldots, w_{u|\mathcal{U}|}\right], \tag{20}$$

where the weight w_{uv} captures the similarity between users u and v. The similarity between users is typically computed using their ratings or interactions with items, and there are many options for similarity measures, including Pearson's correlation, cosine similarity, and Spearman's rank correlation [1]. These weights can be normalized or limited to the closest K nearest neighbors.

Items are represented with their previous interactions or ratings, just as done for social filtering in equation (19). We can see that these neighborhood methods are very similar to social filtering methods—the biggest distinction is that in social filtering, the users themselves determine the pool of users that contribute to the recommendation system, whereas the collaborative filtering approach determines this collection of users based on similarity of behavior.

While neighborhood-based methods have some advantages, we focus our analysis of collaborative filtering approaches on latent factor methods for two main reasons: first, in the simulated setting, there is little distinction between social filtering and neighborhood-based collaborative filtering. Second, latent factor methods are more frequently used than neighborhood methods.

Latent Factor Methods. Of the latent factor methods, matrix factorization is a successful and popular approach [35]. The core idea behind matrix factorization for recommendation is that user-item interactions form a $|\mathcal{U}| \times |\mathcal{I}|$ matrix (as of time t) $\mathbf{R}(t)$, which can be factorized into two low-rank matrices: a $|\mathcal{U}| \times K$ representation of user preferences $\boldsymbol{\theta}(t)$ and an $|\mathcal{I}| \times K$ representation of item attributes $\boldsymbol{\beta}(t)$. The number of latent features K is usually chosen by the analyst or determined with a nonparametric model. The multiplication of these two low-rank matrices approximates the observed interaction matrix, parallel to equation (12), or

$$\boldsymbol{\theta}(t)\boldsymbol{\beta}(t)^\top \approx \mathbf{R}(t). \tag{21}$$

There are many instantiations of the user preferences and item attributes. Non-negative matrix factorization [36] requires that these representations be non-negative. Probabilistic matrix factorization [52] assumes that each cell in the user preference and item attribute matrices are normally distributed, whereas a probabilistic construction of non-negative matrix factorization [9] assumes that they are gamma-distributed. Under all of these constructions, these latent representations are learned from the data by following a sequence of updates to infer the parameters that best match the observed data.

Other latent factor methods, such as principal component analysis (PCA) [29] and latent Dirichlet allocation (LDA) [7], similarly frame user and item representations in terms of a low dimension K of hidden factors. These factors are then learned algorithmically from the observed interactions. We focus on matrix factorization in our simulations (section 4), for simplicity.[8]

To update a latent factor recommendation system with recently collected data, one has several options. Since these methods are typically presented without regard to time t, one option is to refit the model entirely from scratch, concatenating old data with the new; in these cases, consistent interpretation of latent factors may be challenging. A second option is to hold fixed some latent factors (e.g., all item attributes $\boldsymbol{\beta}$) and update the remaining factors according to the update rules used to originally learn all the latent factors. This approach maintains the ordering of the latent factors, but may break convergence guarantees, even if it works well in practice. This approach does not explicitly address new items or users, often called the "cold-start problem," but can be adapted to account for them.

A.5 Modifications and Hybrid Approaches

The concepts of these systems can be modified and combined to create innumerable permutations. In considering collaborative filtering alone, neighborhood-based methods can be merged with latent factor approaches [33]. Full collaborative filtering system can be supplemented with content information [64] or augmented with

[7] Item attributes are determined by applying a binarizing threshold to α_i in equation (3) such that every item has at least one attribute. User representations are then learned using `scipy.optimize.nnls` [30].

[8] Specifically, we use Gaussian probabilistic matrix factorization with confidence weighting, as described by Wang and Blei [64], with $a = 1$ and $b = 0.001$.

social filtering [13]. Any of these methods can be supplemented with a popularity bias. Under any of these modified or hybrid systems, the changes propagate to the representations of user preferences $\boldsymbol{\theta}(t)$ and item attributes $\boldsymbol{\beta}(t)$, and the general framework for recommendation remains the same.

REFERENCES

[1] Ahn, H. J. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences 178*, 1 (2008), 37–51.

[2] Anderson, C. *The long tail: Why the future of business is selling less of more.* Hachette Books, 2006.

[3] Baeza-Yates, R. Data and algorithmic bias in the web. In *Proceedings of the 8th ACM Conference on Web Science* (2016), ACM, pp. 1–1.

[4] Bell, R. M., and Koren, Y. Lessons from the netflix prize challenge. *Acm Sigkdd Explorations Newsletter 9*, 2 (2007), 75–79.

[5] Bennett, J., Lanning, S., et al. The netflix prize. In *Proceedings of KDD cup and workshop* (2007), vol. 2007, New York, NY, USA, p. 35.

[6] Blei, D. M. Probabilistic topic models. *Communications of the ACM 55*, 4 (2012), 77–84.

[7] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent Dirichlet allocation. *JMLR 3* (Mar. 2003), 993–1022.

[8] Bottou, L., Peters, J., Candela, J. Q., Charles, D. X., Chickering, M., Portugaly, E., Ray, D., Simard, P. Y., and Snelson, E. Counterfactual reasoning and learning systems: the example of computational advertising. *Journal of Machine Learning Research 14*, 1 (2013), 3207–3260.

[9] Canny, J. GaP: a factor model for discrete data. In *SIGIR* (2004), pp. 122–129.

[10] Celma, Ò. The long tail in recommender systems. In *Music Recommendation and Discovery*. Springer, 2010, pp. 87–107.

[11] Celma, Ò., and Cano, P. 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* (2008), ACM, p. 5.

[12] Chander, A. The racist algorithm. *Mich. L. Rev. 115* (2016), 1023.

[13] Chaney, A. J., Blei, D. M., and Eliassi-Rad, T. A probabilistic model for using social networks in personalized item recommendation. In *RecSys* (New York, NY, USA, 2015), RecSys '15, ACM, pp. 43–50.

[14] Cremonesi, P., Koren, Y., and Turrin, R. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems* (2010), ACM, pp. 39–46.

[15] Dan-Dan, Z., An, Z., Ming-Sheng, S., and Jian, G. Long-term effects of recommendation on the evolution of online systems. *Chinese Physics Letters 30*, 11 (2013), 118901.

[16] Dandekar, P., Goel, A., and Lee, D. T. Biased assimilation, homophily, and the dynamics of polarization. *Proceedings of the National Academy of Sciences 110*, 15 (2013), 5791–5796.

[17] De Myttenaere, A., Grand, B. L., Golden, B., and Rossi, F. Reducing offline evaluation bias in recommendation systems. *arXiv preprint arXiv:1407.0822* (2014).

[18] Drosou, M., Jagadish, H., Pitoura, E., and Stoyanovich, J. Diversity in big data: A review. *Big Data 5*, 2 (2017), 73–84.

[19] Fleder, D., and Hosanagar, K. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science 55*, 5 (2009), 697–712.

[20] Fleder, D. M., and Hosanagar, K. Recommender systems and their impact on sales diversity. In *Proceedings of the 8th ACM conference on Electronic commerce* (2007), ACM, pp. 192–199.

[21] Goodman, B., and Flaxman, S. European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813* (2016).

[22] Guo, G., Zhang, J., and Yorke-Smith, N. TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. *AAAI* (2015), 123–129.

[23] Harper, F. M., and Konstan, J. A. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS) 5*, 4 (2016), 19.

[24] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS) 22*, 1 (2004), 5–53.

[25] Hu, Y., Koren, Y., and Volinsky, C. Collaborative filtering for implicit feedback datasets. In *In IEEE International Conference on Data Mining (ICDM 2008* (2008), pp. 263–272.

[26] Jamali, M., and Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys* (2010), pp. 135–142.

[27] Jansen, B. J., Liu, Z., and Simon, Z. The effect of ad rank on the performance of keyword advertising campaigns. *Journal of the american society for Information science and technology 64*, 10 (2013), 2115–2132.

[28] Javari, A., and Jalili, M. A probabilistic model to resolve diversity-accuracy challenge of recommendation systems. *arXiv preprint arXiv:1501.01996* (2015).

[29] Jolliffe, I. *Principal component analysis*. Wiley Online Library, 2002.

[30] Jones, E., Oliphant, T., Peterson, P., et al. SciPy: Open source scientific tools for Python, 2001–.

[31] Karatzoglou, A., Baltrunas, L., and Shi, Y. Learning to rank for recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems* (2013), ACM, pp. 493–494.

[32] Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. M. Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery 18*, 1 (2009), 140–181.

[33] Koren, Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *International Conference on Knowledge Discovery and Data Mining* (2008), KDD '08, pp. 426–434.

[34] Koren, Y. Collaborative filtering with temporal dynamics. *Communications of the ACM 53*, 4 (2010), 89–97.

[35] Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *IEEE Computer 42* (2009), 30–37.

[36] Lee, D. D., and Seung, H. S. Algorithms for non-negative matrix factorization. In *NIPS* (2000), pp. 556–562.

[37] Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web* (2010), ACM, pp. 661–670.

[38] Li, L., Chu, W., Langford, J., and Wang, X. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining* (2011), ACM, pp. 297–306.

[39] Liang, D., Charlin, L., McInerney, J., and Blei, D. M. Modeling user exposure in recommendation. In *Proceedings of the 25th International Conference on World Wide Web* (2016), International World Wide Web Conferences Steering Committee, pp. 951–961.

[40] Liu, J.-G., Shi, K., and Guo, Q. Solving the accuracy-diversity dilemma via directed random walks. *Physical Review E 85*, 1 (2012), 016118.

[41] Ma, H., King, I., and Lyu, M. R. Learning to recommend with social trust ensemble. In *SIGIR* (2009), pp. 203–210.

[42] Ma, H., Yang, H., Lyu, M. R., and King, I. SoRec: Social recommendation using probabilistic matrix factorization. In *CIKM* (2008), pp. 931–940.

[43] Ma, H., Zhou, D., Liu, C., Lyu, M. R., and King, I. Recommender systems with social regularization. In *WSDM* (2011), pp. 287–296.

[44] Marlin, B. M., and Zemel, R. S. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems* (2009), ACM, pp. 5–12.

[45] Massa, P., and Avesani, P. Trust-aware recommender systems. In *RecSys* (2007), pp. 17–24.

[46] Massa, P., and Avesani, P. Trust metrics on controversial users: Balancing between tyranny of the majority and echo chambers. *International Journal on Semantic Web and Information Systems (IJSWIS) 3*, 1 (2007), 39–64.

[47] Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., and Riedl, J. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces* (2003), ACM, pp. 263–266.

[48] Mooney, R. J., and Roy, L. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries* (2000), ACM, pp. 195–204.

[49] Netflix. Introducing thumbs. https://www.youtube.com/watch?v=as4pZhodG5I, April 2017.

[50] Park, Y.-J., and Tuzhilin, A. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM conference on Recommender systems* (2008), ACM, pp. 11–18.

[51] Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., Eds. *Recommender Systems Handbook*. Springer, 2011.

[52] Salakhutdinov, R., and Mnih, A. Probabilistic matrix factorization. In *NIPS* (2007), pp. 1257–1264.

[53] Sandvig, C., Hamilton, K., Karahalios, K., and Langbort, C. Auditing algorithms: Research methods for detecting discrimination on internet platforms. *Data and Discrimination: Converting Critical Concerns into Productive Inquiry* (2014).

[54] Schmit, S., and Riquelme, C. Human interaction with recommendation systems: On bias and exploration. *arXiv preprint arXiv:1703.00535* (2017).

[55] Schnabel, T., Swaminathan, A., Singh, A., Chandak, N., and Joachims, T. Recommendations as treatments: Debiasing learning and evaluation. *CoRR, abs/1602.05352* (2016).

[56] Sharma, A., Hofman, J. M., and Watts, D. J. Estimating the causal impact of recommendation systems from observational data. *EC* (2015).

[57] Shi, X., Shang, M.-S., Luo, X., Khushnood, A., and Li, J. Long-term effects of user preference-oriented recommendation method on the evolution of online system. *Physica A: Statistical Mechanics and its Applications 467* (2017), 490–498.

[58] Steck, H. Evaluation of recommendations: rating-prediction and ranking. In *Proceedings of the 7th ACM conference on Recommender systems* (2013), ACM,

pp. 213–220.

[59] Su, J., Sharma, A., and Goel, S. The effect of recommendations on network structure. In *Proceedings of the 25th International Conference on World Wide Web* (2016), International World Wide Web Conferences Steering Committee, pp. 1157–1167.

[60] Sweeney, L. Discrimination in online ad delivery. *Queue 11*, 3 (2013), 10.

[61] Tingle, D., Kim, Y. E., and Turnbull, D. Exploring automatic music annotation with acoustically-objective tags. In *Proceedings of the international conference on Multimedia information retrieval* (2010), ACM, pp. 55–62.

[62] Treviranus, J., and Hockema, S. The value of the unpopular: Counteracting the popularity echo-chamber on the web. In *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference* (2009), IEEE, pp. 603–608.

[63] Vanchinathan, H. P., Nikolic, I., De Bona, F., and Krause, A. Explore-exploit in top-n recommender systems via gaussian processes. In *Proceedings of the 8th ACM Conference on Recommender systems* (2014), ACM, pp. 225–232.

[64] Wang, C., and Blei, D. M. Collaborative topic modeling for recommending scientific articles. In *International Conference on Knowledge Discovery and Data Mining* (2011), KDD '11, pp. 448–456.

[65] Yang, B., Lei, Y., Liu, D., and Liu, J. Social collaborative filtering by trust. In *IJCAI* (2013), pp. 2747–2753.

[66] Ying, Y., Feinberg, F., and Wedel, M. Leveraging missing ratings to improve online recommendation systems. *Journal of marketing research 43*, 3 (2006), 355–365.

[67] Zeng, A., Yeung, C. H., Medo, M., and Zhang, Y.-C. Modeling mutual feedback between users and recommender systems. *Journal of Statistical Mechanics: Theory and Experiment 2015*, 7 (2015), P07020.

[68] Zeng, A., Yeung, C. H., Shang, M.-S., and Zhang, Y.-C. The reinforcing influence of recommendations on global diversification. *EPL (Europhysics Letters) 97*, 1 (2012), 18005.

[69] Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J. R., and Zhang, Y.-C. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences 107*, 10 (2010), 4511–4515.