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 are ubiquitous and impact many domains; they have the potential to influence product consumption, individuals' perceptions of the world, and life-altering decisions. These systems are often evaluated or trained with data from users already exposed to algorithmic recommendations; this creates a pernicious feedback loop. Using simulations, we demonstrate how using data confounded in this way homogenizes user behavior without increasing utility.

KEYWORDS

Recommendation systems; algorithmic confounding.

1 INTRODUCTION

Recommendation systems are designed to help people make decisions. These systems are commonly used on online platforms for video, music, and product purchases through service providers such as Netflix, Pandora, and Amazon. Live systems are updated or retrained regularly to incorporate new data that was influenced by the recommendation system itself, forming a feedback loop (figure 1). While the broad notion of confounding from the data collection process has been studied extensively, we seek to *characterize* the impact of this feedback loop in the context of recommendation systems, demonstrating the unintended consequences of *algorithmic confounding*. As recommendation systems become increasingly important in decision-making, we have an ethical responsibility to understand the idiosyncrasies of these systems and consider their implications for individual and societal welfare [18].

Individual decisions can aggregate to have broad political and economic consequences. Recommendation systems can influence how users perceive the world by filtering access to media; pushing political dialogue towards extremes [59] or filtering out contrary opinions [25]. Even more gravely, these systems impact crucial decision-making processes, such as loan approvals, criminal profiling, and medical interventions. As recommendation systems shape access to goods and resources, issues of fairness and transparency need to be considered. For example, if a distinct group of minority-preference users exists, a system may deferentially undermine the
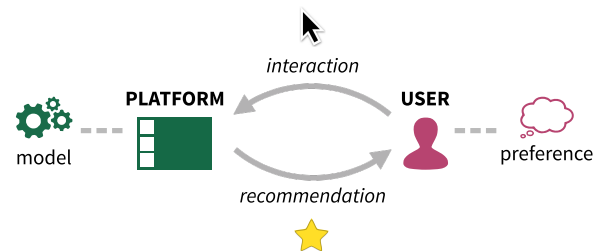
Figure 1: The feedback loop between user behavior and algorithmic recommendation systems. Confounding occurs when a platform attempts to model user behavior without accounting for recommendations. User preferences act as confounding factors, influencing both recommendations (through past interactions) and current interactions.

needs or preferences of the minority in favor of optimizing the utility of the majority group.

Many researchers and practitioners still focus on evaluating recommendation systems in terms of held-out accuracy, which cannot capture the full effects of the feedback loop. Even with accuracy as a primary concern, algorithmic confounding can play a crucial role; for instance, when a recommendation system is evaluated using confounded held-out data, results are biased toward recommendation systems similar to the confounding algorithm. Thus, the choice of data can considerably impact held-out evaluation and subsequent conclusions. Averaged accuracy metrics, however, are only one approach to evaluating recommendation systems, and do not detect disparate impact across users. It is our hope that this work will help motivate researchers and practitioners to 1) actively assess systems with objectives such as diversity, serendipity, novelty, and coverage [31]; 2) apply causal reasoning techniques to counter the effects of algorithmic confounding; and 3) evaluate the distribution of impact across all users, instead of exclusively reporting averages.

We begin with a summary of our claims (section 2) and then situate this work among related lines of inquiry (section 3). To provide evidence for our claims, we introduce a model for users interacting with recommendations (section 4); this allows us to analyze the impact of algorithmic confounding on simulated communities (section 5). We find that algorithmic confounding amplifies the homogenization of user behavior (section 5.2) without corresponding gains in utility (section 5.3) and also amplifies the impact of recommendation systems on the distribution of item consumption (section 5.4). We briefly discuss weighting approaches to counter these effects (section 6) before we conclude (section 7). For our simulations and analysis, we develop a general framework for recommendation systems (appendix A) which highlights commonalities between seemingly distinct approaches.

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 through three claims.

The recommendation feedback loop causes homogenization of user behavior (section 5.2), 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 experience losses in utility due to homogenization effects; these losses are distributed unequally (section 5.3). Across multiple recommendation system types, users with lower relative utility have higher homogenization, indicating that homogenization can be detrimental for subsets of users by encouraging them to make sub-optimal choices.

The feedback loop amplifies the impact of recommendation systems on the distribution of item consumption (section 5.4), irrespective of homogenization effects. Two recommendation systems can produce similar amounts of user homogenization with different distributions of item consumption.

3 RELATED WORK

Bias, confounding, and estimands. Schnabel, et al. [52] 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 [43, 62]. 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 do not wish to correct for user 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. [39] propose a method specifically for reducing this bias in contextual bandit algorithms.

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, 50, 56]. 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 [46] 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* [54]. Simply put, top-*n* correctness metrics such as precision, recall, and nDCG are better for evaluating the real-world performance of these systems [14]. These metrics are popular because they are straightforward to understand and easy to implement, but still they do not necessarily capture the usefulness of recommendations; a variety of system characteristics are thought to be important [24]. Among these characteristics, diversity is often framed as a counterpart to accuracy [28, 41, 64]. 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, 20, 47, 63]. 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 been used extensively in many domains, but have only recently been applied to recommendation systems. Liang, et al. [40] 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 [26]. Some work has also been done to understand the causal impact of these systems on behavior by finding natural experiments in observational data [53, 55] (approximating expensive controlled experiments [33]), but it is unclear how well these results generalize. Schnabel, et al. [52] 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. Some investigators have modeled these systems using temporal dynamics [35] or framed them in an explore/exploit paradigm [38, 60]. 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 low probability items to learn more about a user's preferences? Reinforcement learning models already use notion of a feedback loop to maximize reward. One major challenge with this setup, however, is constructing a reward function. 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 may be informative for the construction of reward functions and may be extended to use the stationary distribution of Markov decision processes (MDPs) to further characterize the impact of the feedback loop.

4 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 3. We draw on a model recently proposed by Schmit and Riquelme [51] 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*

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

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.

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 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.

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 $\mathfrak{R}$.[4] The chosen item is*

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

where $rank_{u,t}^{\mathfrak{R}}(i) = n$, or according to recommender system $\mathfrak{R}$'s ordering of items, as described in appendix A, the nth recommendation r for user u at time t is item i; this can also be written $r_{u,n}(t) = i$.

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.

---

[1]Unlike the Schmit and Riquelme model [51], 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 5, 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 5).

[4]For our simulations (section 5), 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.

5 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.

5.1 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 on 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 $\beta_i(t)$:

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

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 4) 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: $\tilde{\mu}_\rho \sim \text{Dirichlet}(1)$ and $\mu_\rho = 10 \cdot \tilde{\mu}_\rho$. 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: $\tilde{\mu}_\alpha \sim \text{Dirichlet}(100)$ and $\mu_\alpha = 0.1 \cdot \tilde{\mu}_\alpha$. 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. We generate social networks (used by social filtering only) 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 homophily between connected users, 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.
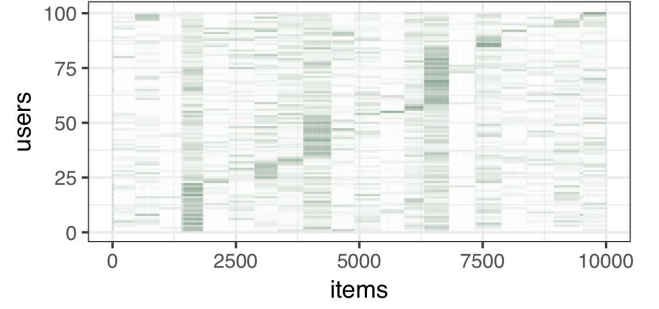
Figure 2: Example true utility matrix V for simulated data; darker is higher utility. The distribution of user preferences is disproportionate, like the real world, and the structure is easily captured with matrix factorization.

5.2 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, 58]. *Social* recommendation systems homogenize connected users or within cliques, and *matrix factorization* homogenizes users along learned latent factors.

Homogenizing effects are not inherently bad as they indicate that the models are learning patterns from the data, as intended; when achieving optimum utility, users will have some degree of homogenization. However, homogenization of user behavior does not correspond directly with an increase in utility: 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 more similar behaviors than their underlying preferences would optimally dictate. This suggests that the "tyranny of majority" and niche "echo chamber" effects may both be manifestations of the same problem: over-exploitation of recommendation models. While concerns about maximizing utility are not new to the recommendation system literature, there are also grave social consequences from homogenization that have received less consideration.

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

$$\mathbf{J}_{uv}(t) = \frac{|\mathcal{D}_u(t) \cap \mathcal{D}_v(t)|}{|\mathcal{D}_u(t) \cup \mathcal{D}_v(t)|}. \qquad (7)$$

We compared the Jaccard index for paired users against the Jaccard index of the same users exposed to ideal recommendations; this difference captures how much the behavior has homogenized relative
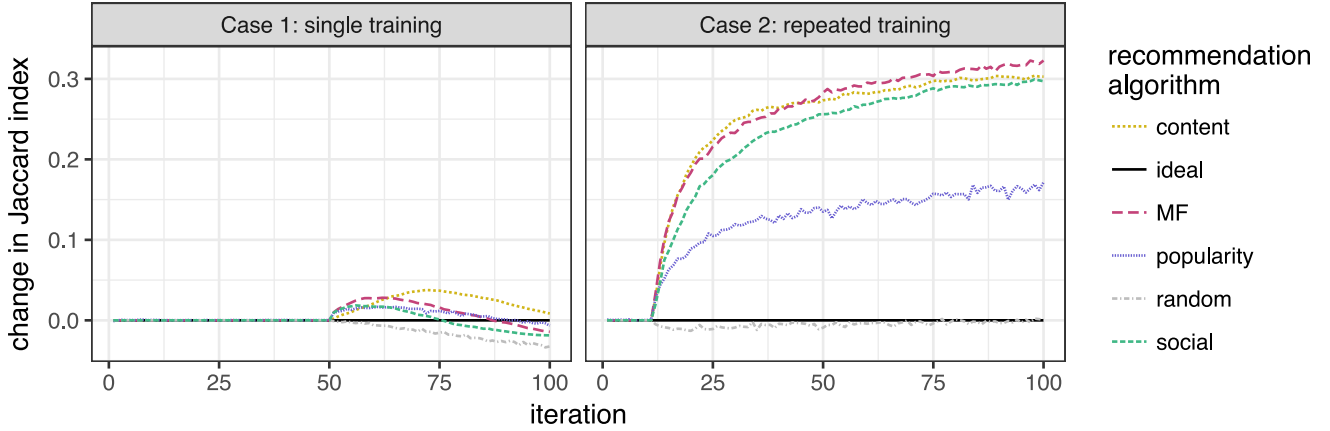
Figure 3: Change in Jaccard index of user behavior relative to ideal behavior; users paired by cosine similarity of θ. On the left, mild homogenization of behavior occurs soon after a single training, but then diminishes. On the right, recommendation systems that include repeated training homogenize user behavior more than is needed for ideal utility.
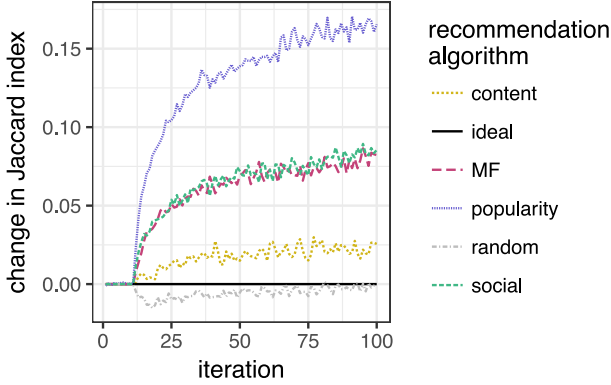
Figure 4: For the repeated training case, change in Jaccard index of user behavior relative to ideal behavior; users paired randomly. Popularity increases homogenization the most globally, but all non-random recommendation algorithms also homogenize users globally.

to ideal. Figure 3 shows these results for both the single training and the repeated training cases. In the single training case, users became slightly homogenized after training, but then returned to the ideal homogenization. With repeated training, all recommendation systems (except random), homogenize 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.

We can 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 compare users matched randomly (figure 4). In this setting, all recommendation systems (except random) increased global homogeneity of user behavior. The popularity system increased homogeneity the most; after that, matrix factorization and social filtering homogenized users comparably, and content filtering homogenized 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.

5.3 Loss of Utility

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 decreases in utility when homogenization occurs. Figure 5 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 [19]. With a more nuanced representation of utility that includes the collection of interactions as a whole, these effects would likely increase in magnitude.

5.4 Shifts in Item Consumption

To explore the impacts of these systems on item consumption, we computed the Gini coefficient on the distribution of consumed items, which measures the inequality of the distribution (0 being all items are consumed at equal rates and 1 indicating maximal inequality between the consumption rate of items). We use the following definition of Gini inequality, which relies on the relative popularity (RP)[5] of each item i at time t:

$$G(t) = \frac{\sum_{i \in I} (2 * \mathrm{RP}(i, t) - |I| - 1) \sum_{u \in \mathcal{U}} \mathbb{1} \left[i \in \mathcal{D}_u(t) \right]}{|I| \sum_{i \in I} \sum_{u \in \mathcal{U}} \mathbb{1} \left[i \in \mathcal{D}_u(t) \right]}. \quad (8)$$

We found that the feedback loop amplifies the impact of recommendation systems on the distribution of item consumption,

---

[5]RP(i, t) is the rank of item i based on how many people have consumed it, relative to all other items (up to time t).
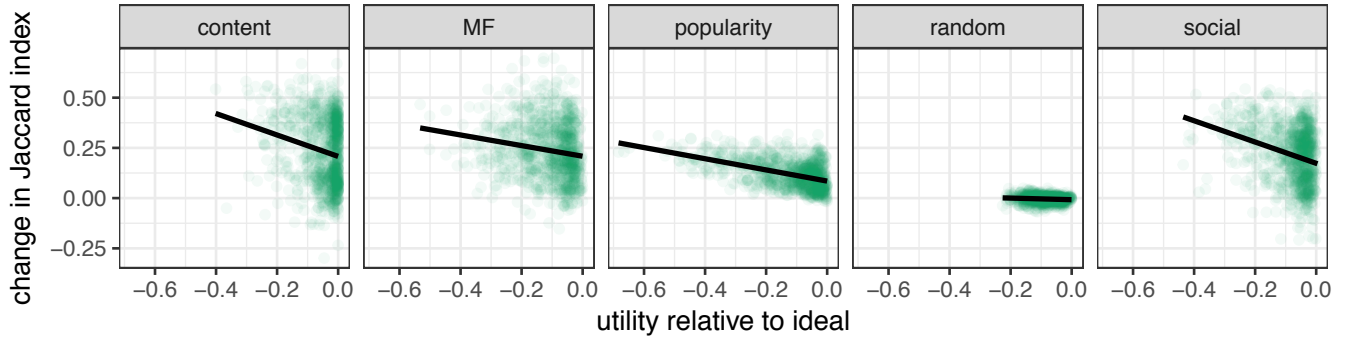
Figure 5: For the repeated training case, change in Jaccard index of user behavior, relative to ideal behavior, and shown as a function of utility relative to the ideal platform; users paired by cosine similarity of θ. Each user is shown as a point, with a linear fit to highlight the general trend that users who experience losses in utility have higher homogenization.
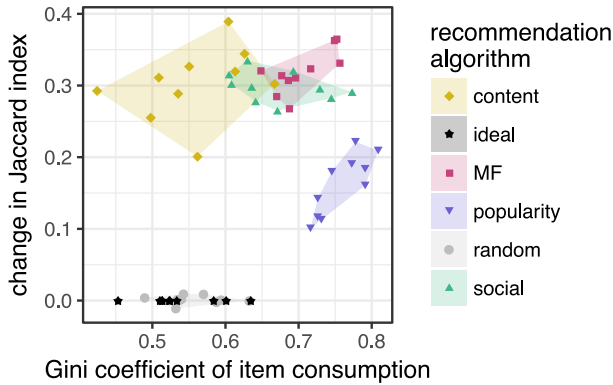

Figure 6: For the repeated training case, change in Jaccard index of user behavior (higher is more homogeneous), relative to the Gini coefficient of the item consumption distribution (higher is more unequal consumption of items). Each point is a single simulation. Similar homogenization can result in different item consumption distributions.

irrespective of homogenization effects. Specifically, two recommendation systems can produce similar amounts of user homogenization with different distributions of item consumption (figure 6). For example, matrix factorization (MF) and content filtering have comparable homogenizing effects, but MF creates a more unequal distribution of item consumption.

As a community, we do not fully understand the ways in which these systems change the popularity of items. Differential item consumption may ultimately change item production from strategic actors, such as companies like Amazon and Netflix which are now producing content based on their consumers' behavior data. Thus, recommendation systems change not only what users see first, but can fundamentally alter the collection of content from which users can choose.

6 ACCOUNTING FOR CONFOUNDING

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.

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 [45]; the researchers behind it have released several public datasets[6] that are used prolifically in evaluation recommendation systems [23]. Recommendation systems based on collaborative filtering will likely perform better on this data. Similarly, the *Douban* service contains popularity features; algorithms that include a notion of popularity should perform better on the publicly available data from this platform [42]. Many algorithms use social network information to improve recommendations [13, 22, 42] 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 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.

Researchers and practitioners alike would benefit from methods to address these concerns. Weighting techniques, such as those proposed by Schnabel, et al. [52] 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 5), but needs to be studied in greater depth. In particular, the distribution of impact across users should be explored, as well as non-accuracy metrics. Regardless, practitioners would do well to incorporate measures to avoid confounding, such as these. At least they should log information about the recommendation system in

---

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

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.

7   CONCLUSION

We have explored the impact of algorithmic confounding on a range of simulated recommendation systems. We found that algorithmic confounding amplifies the homogenization of user behavior (section 5.2) without corresponding gains in utility (section 5.3) and also amplifies the impact of recommendation systems on the distribution of item consumption (section 5.4). These findings have implications for any live recommendation platform; those who design these systems need to consider how a system influences its users and account for algorithmic confounding. Researchers who use confounded data to test and evaluate their algorithms should be aware of these effects, as should researchers who wish to use confounded data to study and make claims about user behavior. 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 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:

$$s_{ui}(t) = \theta_u(t)\beta_i(t)^\top. \qquad (9)$$

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, 54] (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 [32] 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

$$\left\{ \left(r_{u,n}(t) \right)_{n=1}^{|\mathcal{I}|} \right\}_{u \in \mathcal{U}}, \qquad (10)$$

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

$$\left(r_{u,n}(t) \right)_{n=1}^{|\mathcal{I}|} = \text{descending sort}\left(\forall i \in \mathcal{I}, \text{by} = s_{ui}(t) \right). \qquad (11)$$

Our simulated experiments (section 5) 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 5), we employ the simplest popularity recommendation system; it considers all interactions up to time t, or

$$\beta_i(t) = \sum_{n=1}^N \mathbb{1}\left[i_n = i \text{ AND } t_n < t \right].^7 \qquad (12)$$

A.2   Content Filtering

Content-based recommender systems match attributes in a user's profile with attribute tags associated with an item [48, ch. 3]. In the simplest variant, the set of possible attribute tags $\mathcal{A}$ are identical for both users and items. 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 $\theta_{ua}(t) = \mathbb{1}\left[a \in \mathcal{A}_u(t) \right]$. The item attribute tags can similarly be represented as a vector of length $|\mathcal{A}|$ with values $\beta_{ia}(t) = \mathbb{1}\left[a \in \mathcal{A}_i(t) \right]$, 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 of a song [57] for music; when learned, the attribute tags can be

---

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

real-valued instead of binary. For our simulations (section 5), we use binary item attributes and learn real-valued user preferences.[8]

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 $\theta(t)$ are a representation of the social network, or a $|\mathcal{U}| \times |\mathcal{U}|$ matrix; for each user u connected to another user v, $\theta_{uv}(t) = \mathbb{1}\left[v \in \mathcal{F}_u(t)\right]$, 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 $\theta(t)$ can represent the non-binary trust between users, which can be provided explicitly by the user [44] or learned from user behavior [13]; we use the latter in our analysis (section 5).

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, $\beta_{iv}(t) = \mathbb{1}\left[v \in \mathcal{U}_i(t)\right]$, 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 [48, 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

$$\theta_u(t) = \left[w_{u1}, \ldots, w_{u|\mathcal{U}|}\right], \tag{13}$$

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 appendix A.3. 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 [36]. 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 $\theta(t)$ 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 (9), or $\theta(t)\boldsymbol{\beta}(t)^\top \approx \mathbf{R}(t)$.

There are many instantiations of the user preferences and item attributes. Non-negative matrix factorization [37] requires that these representations be non-negative. Probabilistic matrix factorization [49] 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 5), for simplicity.[9]

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 [34]. Full collaborative filtering system can be supplemented with content information [61] or augmented with 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 $\theta(t)$ and item attributes $\boldsymbol{\beta}(t)$, and the general framework for recommendation remains the same.

---

[8]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].

[9]Specifically, we use Gaussian probabilistic matrix factorization with confidence weighting, as described by Wang and Blei [61], with $a = 1$ and $b = 0.001$.

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 *RecSys* (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. *PNAS 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] Diresta, R. Up next: A better recommendation system. https://www.wired.com/story/creating-ethical-recommendation-engines/, March 2018.
[19] Drosou, M., Jagadish, H., Pitoura, E., and Stoyanovich, J. Diversity in big data: A review. *Big Data 5*, 2 (2017), 73–84.
[20] 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.
[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] Hess, A. How to escape your political bubble for a clearer view. https://www.nytimes.com/2017/03/03/arts/the-battle-over-your-political-bubble.html, March 2017.
[26] Hu, Y., Koren, Y., and Volinsky, C. Collaborative filtering for implicit feedback datasets. In *ICDM* (2008), pp. 263–272.
[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] Kaminskas, M., and Bridge, D. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Transactions on Interactive Intelligent Systems (TiiS) 7*, 1 (2017), 2.
[32] Karatzoglou, A., Baltrunas, L., and Shi, Y. Learning to rank for recommender systems. In *RecSys* (2013), ACM, pp. 493–494.
[33] Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. M. Controlled experiments on the web: survey and practical guide. *Data mining and knowledge*

[34] 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.
[35] Koren, Y. Collaborative filtering with temporal dynamics. *Communications of the ACM 53*, 4 (2010), 89–97.
[36] Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *IEEE Computer 42* (2009), 30–37.
[37] Lee, D. D., and Seung, H. S. Algorithms for non-negative matrix factorization. In *NIPS* (2000), pp. 556–562.
[38] Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *WWW* (2010), ACM, pp. 661–670.
[39] 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.
[40] Liang, D., Charlin, L., McInerney, J., and Blei, D. M. Modeling user exposure in recommendation. In *WWW* (2016), pp. 951–961.
[41] Liu, J.-G., Shi, K., and Guo, Q. Solving the accuracy-diversity dilemma via directed random walks. *Physical Review E 85*, 1 (2012), 016118.
[42] Ma, H., Zhou, D., Liu, C., Lyu, M. R., and King, I. Recommender systems with social regularization. In *WSDM* (2011), pp. 287–296.
[43] Marlin, B. M., and Zemel, R. S. Collaborative prediction and ranking with non-random missing data. In *RecSys* (2009), ACM, pp. 5–12.
[44] Massa, P., and Avesani, P. Trust-aware recommender systems. In *RecSys* (2007), pp. 17–24.
[45] 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.
[46] Netflix. Introducing thumbs. https://www.youtube.com/watch?v=as4pZhodG5I, April 2017.
[47] Park, Y.-J., and Tuzhilin, A. The long tail of recommender systems and how to leverage it. In *RecSys* (2008), ACM, pp. 11–18.
[48] Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., Eds. *Recommender Systems Handbook.* Springer, 2011.
[49] Salakhutdinov, R., and Mnih, A. Probabilistic matrix factorization. In *NIPS* (2007), pp. 1257–1264.
[50] 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).
[51] Schmit, S., and Riquelme, C. Human interaction with recommendation systems: On bias and exploration. *arXiv preprint arXiv:1703.00535* (2017).
[52] Schnabel, T., Swaminathan, A., Singh, A., Chandak, N., and Joachims, T. Recommendations as treatments: Debiasing learning and evaluation. *CoRR, abs/1602.05352* (2016).
[53] Sharma, A., Hofman, J. M., and Watts, D. J. Estimating the causal impact of recommendation systems from observational data. *EC* (2015).
[54] Steck, H. Evaluation of recommendations: rating-prediction and ranking. In *RecSys* (2013), ACM, pp. 213–220.
[55] Su, J., Sharma, A., and Goel, S. The effect of recommendations on network structure. In *WWW* (2016), pp. 1157–1167.
[56] Sweeney, L. Discrimination in online ad delivery. *Queue 11*, 3 (2013), 10.
[57] 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.
[58] 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.
[59] Tufekci, Z. Youtube, the great radicalizer. https://www.nytimes.com/2018/03/10/opinion/sunday/youtube-politics-radical.html, March 2018.
[60] Vanchinathan, H. P., Nikolic, I., De Bona, F., and Krause, A. Explore-exploit in top-n recommender systems via gaussian processes. In *RecSys* (2014), ACM, pp. 225–232.
[61] Wang, C., and Blei, D. M. Collaborative topic modeling for recommending scientific articles. In *KDD* (2011), KDD '11, pp. 448–456.
[62] Ying, Y., Feinberg, F., and Wedel, M. Leveraging missing ratings to improve online recommendation systems. *Journal of marketing research 43*, 3 (2006), 355–365.
[63] 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.
[64] 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. *PNAS 107*, 10 (2010), 4511–4515.

discovery 18, 1 (2009), 140–181.