Debiasing Item-to-Item Recommendations With Small Annotated Datasets

Tobias Schnabel
Microsoft
Redmond, WA, USA
tobias.schnabel@microsoft.com

Paul N. Bennett
Microsoft
Redmond, WA, USA
pauben@microsoft.com

ABSTRACT

Item-to-item recommendation (e.g., "People who like this also like…") is a ubiquitous and important type of recommendation in real-world systems. *Observational* data from historical interaction logs abound in these settings. However, since virtually all observational data exhibit biases, such as time-in-inventory or interface biases, it is crucial that recommender algorithms account for these biases. In this paper, we develop a principled approach for item-to-item recommendation based on causal inference and present a practical and highly effective method for estimating the causal parameters from a small annotated dataset. Empirically, we find that our approach substantially improves upon existing methods while requiring only small amounts of annotated data.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; Similarity measures; • **Computing methodologies** → *Learning from implicit feedback*.

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Tobias Schnabel and Paul N. Bennett. 2020. Debiasing Item-to-Item Recommendations With Small Annotated Datasets. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3383313.3412265

1 INTRODUCTION

Item-to-item recommendation, also known as related item recommendation, is a core part of real-world recommender systems. In item-to-item recommendation, the task is to provide users with a list of relevant items around a so-called *seed item*. This enables users to do item-centric exploration of inventories when browsing. Online retailers like Amazon and Ebay, for example, show customers a panel with items under "People who viewed this item also viewed" on product pages [21]. Other prominent examples are job suggestions of "People Also Viewed" when viewing a job opening on LinkedIn, or news websites that show related articles with each story [1]. The other advantage of item-to-item recommendations is that they require no additional user context or history. This is in contrast to the more common blended recommendations task which bases recommendations on an established user profile and thus suffers from cold start problems [6, 38]. Note also that item-to-item recommendations refers to the task of finding related items and is distinct from item-based recommendation techniques that can be used to solve the former.

With engagement signals such as clicks or views being readily available, many existing approaches for item-to-item recommendation leverage collaborative filtering (CF) techniques. CF-based techniques effectively mine observed co-occurrences of items across different user sessions, for example by extracting items that are often bought together. Disappointingly, current CF-based models yield only item-to-item recommendations of mediocre quality as user-centric evaluations show [38]. In this paper, we argue that item-to-item recommendation is fundamentally a counterfactual problem – the question we try to answer is whether a user would have engaged with (e.g., clicked or converted) an item if he or she had been aware of it. However, the latter is confounded by factors such as item popularity [32] or the time the item has been in the inventory [7], leading to biased data. As we will demonstrate in this paper, this causes approaches which do not account for these biases to have suboptimal performance.

To make this more concrete, Table 1 shows an example from the well-known MovieLens 25M dataset where we retrieve the top 10 movies for the animated movie j = "Toy Story (1995)" using the co-occurrences counts count(i, j) of candidate movies. Intuitively, we would expect that at least some of the returned movies would also be animated movies. However, no such movies appears in the list. In fact, as the last column shows, a candidate movie's rank is highly correlated with its overall popularity in the MovieLens dataset. Even worse, all of the returned movies are among the top 25 most frequent movies in the dataset.

This paper's contributions are three-fold. First, we provide a rigorous formulation of item-to-item recommendation as a counterfactual estimation problem inspired by recent work [4, 30]. We then show how to use a propensity scoring model to adjust for biases. Second, we innovate by providing a highly practical method for estimating the propensity model from a small set of annotated data under mild assumptions. This is a departure from previous work assuming access to uniformly sampled and rated per-user data [4] which can be difficult to obtain in practice. Instead, we only require relative judgements of relevance for pairs of seed items and candidate items. Our framework allows us to use this small annotated dataset to debias a much larger observational dataset, yielding

RecSys '20, September 22–26, 2020, Virtual Event, Brazil

ACM ISBN 978-1-4503-7583-2/20/09…$15.00
https://doi.org/10.1145/3383313.3412265

propensity scores that could in principle be re-used for learning more sophisticated models [30]. As such, our estimation framework can be seen complementary to latter approaches, opening up exciting avenues for future research. Finally, we show empirically that our proposed approach outperforms a large variety of baselines on a real-world movie-to-movie recommendation task.

2 ITEM-TO-ITEM RECOMMENDATION

We start by introducing the item-to-item recommendation task in an ideal, full-information setting where we get to observe every user's complete preferences. We later relax this assumption to the more realistic, partial information setting which we will study in the rest of this paper.

2.1 Full information setting

Let $\mathcal{I}$ denote our item inventory and let us assume that we get to observe a sample of N *session vectors* $(S_1, \ldots, S_N)$. These session vectors can be thought of as having been induced by an underlying sample of user contexts $(u_1, \ldots, u_N)$ drawn from a user population $P(U)$. Each session vector has $|\mathcal{I}|$ entries with each entry in i in S_k indicating whether user u_k would engage with item i (for example, purchase, like or share an item). The session vectors themselves can be viewed as having been generated from the user contexts via a deterministic mapping function, i.e.,

$$S_k[i] = \text{engage}(u_k, i) \text{ for } i \in \mathcal{I}.$$

Note that requiring determinism in the mapping function is not an unrealistic assumption to make, since we can always encode additional information in the user context u_k to make this process deterministic. For notational convenience, let us use the equivalent, but more abstract view of each S_k being drawn directly and i.i.d. from an overall distribution of session vectors $P(S)$. We will also use $S[i]$ to denote the i-th entry in S corresponding to user engagement with item i. Note that $S[i]$ is now a single binary random variable whose randomness is due to the sampling of different user contexts.

In item-to-item prediction, our goal is to find the most likely item i in our inventory $\mathcal{I}$ for a user to engage with given a seed item j was engaged with in a session. In other words, we want

$$\arg\max_{i \in \mathcal{I} \setminus j} \mathrm{P}(S[i] = 1 \mid S[j] = 1). \tag{1}$$

Note also that even though we refer to S_k as a *session* vector here, the term *session* is meant to refer to an interesting segment of user behavior in general and not only to short and coherent periods of user activity. Depending on the application and domain, it may for example be defined to be all items in a purchase order or even a user's entire consumption history. Also, the goal of finding only the most likely item can easily be extended to find the top-K most likely items, for example by iteratively applying Equation (1) or scoring all items and sorting them by $\mathrm{P}(S[i] = 1 \mid S[j] = 1)$ in descending order.

2.2 Partial information setting

The full information setting above is clearly unrealistic. In it, we effectively assumed that users give us feedback on all items, meaning that users are aware of all items when making their decisions to engage. In reality, however, it is very unlikely that all items receive full consideration, for example due to cognitive biases like recency effects, as well as presentation limitations arising from the screen or interface [30]. Hence, we adopt the following partial information setting motivated by the potential outcomes framework [15]. Entries in S_k are only observed when a binary and latent observation variable $O_k[i]$ is true:

$$S_k^{\text{obs}}[i] = S_k[i] \cdot O_k[i]. \tag{2}$$

This essentially is a masking process – only entries i in S_k with $O_k[i] = 1$ will be observed. $O_k \in \{0, 1\}^{|\mathcal{I}|}$ is a binary vector drawn from a observation distribution $P(O \mid \mathcal{X}_k)$, where $\mathcal{X}_k$ is the set of variables that the observation probability depends on. Note that $\mathcal{X}_k$ can also include the current session or user context u_k which is why the subscript is used. However, we will make the following mild independence assumption about the observation probabilities throughout this paper. Given the set $\mathcal{X}_k$, the observation probabilities of any pair of items i and j are conditionally independent.

$$\mathrm{P}(O[i] = 1, O[j] = 1 \mid \mathcal{X}_k) = \mathrm{P}(O[i] = 1 \mid \mathcal{X}_k)\,\mathrm{P}(O[j] = 1 \mid \mathcal{X}_k).$$

This, in essence, says that we assume the set $\mathcal{X}_k$ is rich enough to capture any dependencies between the likelihood of getting to observe the true entries $S[i]$ and $S[j]$. We will discuss more details of specifying $\mathcal{X}_k$ in the next section.

3 ESTIMATORS FOR ITEM-TO-ITEM RECOMMENDATION

We start this section by discussing a straight-forward approach to item-to-item recommendation based on counting co-occurrences, an approach resulting from maximum likelihood estimation (MLE) of the conditional probability.

3.1 Estimation via MLE

Applying standard MLE to $\mathrm{P}(S[i] = 1 \mid S[j] = 1)$, we obtain:

$$\begin{aligned}\hat{\mathrm{P}}^{MLE}(S[i] = 1 \mid S[j] = 1) &= \frac{\hat{\mathrm{P}}^{MLE}(S[i] = 1, S[j] = 1)}{\hat{\mathrm{P}}^{MLE}(S[j] = 1)} \\ &= \frac{\cancel{\frac{1}{N}} \sum_k \mathbb{1}[S_k[i] = 1 \wedge S_k[j] = 1]}{\cancel{\frac{1}{N}} \sum_k \mathbb{1}[S_k[j] = 1]}. \end{aligned} \tag{3}$$

In other words, we simply need to count all session contexts in which i and j occur together and divide that by the total number of sessions that item j occurs in. Throughout this paper, we assume that the denominator is non-zero, i.e., $\text{count}(j) = \sum_k \mathbb{1}[S_k^{\text{obs}}[j] = 1] > 0$, which can easily be achieved by eliminating items that were never engaged with.

Note, however, that this estimator assumes that we have access to and use the samples S_k from the full information setting. What could go wrong if we ignored this fact and just used the observational samples S_k^{obs} in the estimator as is typically done [10, 31]?

As Table 1 showed, the ranking was heavily influenced by a movie's overall popularity. Counterfactual inference is able to explain this observation by helping us understand the risk of using data from a partial information setting, often also referred to *observational* data. Since the data we used was observational, we are at risk of over- or under-estimating any effect when there are so-called *confounders* present that govern whether or not we get to observe

Table 1: Most likely movies to co-occur with "Toy Story (1995)" in the MovieLens 25M dataset. Overall popular movies dominate the list of recommended movies.

rank	title	$\hat{P}^{MLE}(S[i] \mid S[j])$	popularity rank
1.	Forrest Gump (1994)	0.634	2
2.	Star Wars: Episode IV - A New Hope (1977)	0.610	6
3.	Shawshank Redemption, The (1994)	0.593	1
4.	Pulp Fiction (1994)	0.578	3
5.	Silence of the Lambs, The (1991)	0.554	4
6.	Matrix, The (1999)	0.554	5
7.	Jurassic Park (1993)	0.537	8
8.	Star Wars: Episode VI - Return of the Jedi (1983)	0.520	16
9.	Star Wars: Episode V - The Empire Strikes Back (1980)	0.506	11
10.	Back to the Future (1985)	0.500	23

an entry of S_k in S_k^{obs}. In our setting above, popularity is a confounder that affects whether or not a user would rate a movie and makes us over-estimate j's effect on i. Other confounders could be time in inventory, promotions or presentation biases. Fortunately, counterfactual inference also provides us with a variety of techniques that help us correct confounding. We will demonstrate how to employ *inverse propensity scoring* (IPS) [15] in the context of our problem in the next section.

3.2 Estimation via Inverse Propensity Scoring

As mentioned in the previous section, inverse propensity scoring offers a way to adjust for bias due to confounding in our estimators. Let $p_{k,i} > 0$ be item i's marginal probability of being observed in session k:

$$p_{k,i} = \mathrm{P}(O_k[i] = 1 \mid \mathcal{X}_k).$$

This is also called the *propensity* of an observation, giving rise to the *inverse propensity* $p_{k,i}^{-1} = 1/p_{k,i}$. Equipped with this, the IPS estimator [15, 22] then gives us the conditional probability as

$$\begin{aligned}\hat{\mathrm{P}}^{IPS}(S[i] = 1 | S[j] = 1) &= \frac{\sum_k p_{k,i}^{-1} p_{k,j}^{-1} \mathbb{1}\left[S^{\text{obs}}[i] = 1 \wedge S_k^{\text{obs}}[j] = 1\right]}{\sum_k p_{k,j}^{-1} \mathbb{1}\left[S^{\text{obs}}[j] = 1\right]} \\ &= Z_j \sum_k p_{k,i}^{-1} p_{k,j}^{-1} \mathbb{1}\left[S^{\text{obs}}[i] = 1 \wedge S_k^{\text{obs}}[j] = 1\right], \qquad (4)\end{aligned}$$

where in the last line we define Z_j as the normalization constant (the denominator). Following the analogous steps of [16], one can easily show that both the numerator as well as the denominator are unbiased and consistent estimators of $\mathrm{P}(S[i] = 1, S[j] = 1)$ and $\mathrm{P}(S[j] = 1)$ respectively. Slutsky's theorem then implies that the ratio estimator in Equation (4) is also consistent estimator of the true conditional probabilities. In other words, given enough data, we are guaranteed to recover the correct value of $\mathrm{P}(S[i] = 1, S[j] = 1)$.

Even though the IPS estimator allows us to address biases correctly, the big challenge in implementing this estimator is estimating the unknown propensities $p_{k,i}$ accurately. Existing approaches either rely on randomization or experimentation [16], both of which are often infeasible in practice, or make strong and unverifiable assumptions about the confounding process. In this paper, we propose a novel approach of inferring the propensities $p_{k,i}$ from a small set of hand-labeled training examples. This allows us to accurately learn propensities in practice and also enables us to use standard validation procedures during training. In what follows, we will parameterize the propensities via a model

$$p_k = f_\theta(\mathcal{X}_k),$$

where θ are the parameters of the model. We will use $\hat{\mathrm{P}}_\theta^{IPS}(S[i] \mid S[j])$ from hereon to denote the IPS estimator that uses a propensity model with parameters θ.

4 LEARNING PROPENSITIES FROM SMALL ANNOTATED DATASETS

Besides formulating item-to-item recommendation as a counterfactual inference problem, the core innovation of this paper is that we provide a method for inferring propensities from only a small set of hand-labeled data. We start by introducing the general framework for learning propensities before demonstrating how it can be instantiated with a specific propensity model.

4.1 General Framework

As mentioned in the last section, the main challenge in making the IPS estimator practical is coming up with accurate estimates for the propensities in it. The main idea of our framework is to learn these propensities from a small amount of labeled data and then using that data to debias the larger observational dataset.

A straight-forward source of labeled data comes from relevance judgements [33, 38], where people are asked to rate how relevant they perceive an item i to be given a seed item j, yielding relevance score $\mathrm{rel}(i \mid j)$. Since it is challenging to assign absolute relevance scores to pairs [5], we only assume that we get a set of pairwise judgements for each seed item j,

$$C_j = \left\{(i_1, i_2) : \mathrm{rel}(i_1 \mid j) > \mathrm{rel}(i_2 \mid j)\right\}.$$

In practice, one can also construct these pairs also from a set of positive judgements with $\mathrm{rel}(i_1 \mid j) > 0$ by sampling from the remaining items in the inventory under the assumption that those are less relevant. Our idea is to relate these pairwise judgements to the true probabilities $\mathrm{P}(S[i] = 1 \mid S[j] = 1)$ by establishing the

following relationships between two candidate items i_1, i_2:

$$\begin{aligned}
&\text{rel}(i_1 \mid j) > \text{rel}(i_2 \mid j) \\
\iff &\text{P}(S[i_1] = 1 \mid S[j] = 1) > \text{P}(S[i_2] = 1 \mid S[j] = 1) \\
\iff &\log \text{P}^{IPS}(S[i_1] = 1 \mid S[j] = 1) \\
&\qquad > \log \text{P}^{IPS}(S[i_2] = 1 \mid S[j] = 1) \\
\iff &\log \text{P}^{IPS}(S[i_1] = 1 \mid S[j] = 1) \\
&\qquad - \log \text{P}^{IPS}(S[i_2] = 1 \mid S[j] = 1) > 0. \qquad (5)
\end{aligned}$$

This says that when an candidate item i_1 is more relevant given j than i_2 is, it also means that this candidate item i_1 should have a higher full information conditional probability. Note also that we used the consistency of the IPS estimator going from the first to the second line. The definition of relevance should try to capture the major factors governing which items people would engage with in an idealized full information setting where they would consider the entire set of items. For example, if the engagement signal is whether or not an item was purchased, one could ask annotaters: "Which of these items would be relevant to a buyer given <item j>?".

The constraints above essentially define a ranking problem that we can optimize to find the parameters θ for the propensity model. Given $|C_j|$ pairs of relevance judgements for an item j, we define the empirical loss for C_j by summing over all pairs and applying a Hinge loss:

$$\begin{aligned}
\ell(C_j; \theta) = \sum_{(i_1, i_2) \in C_j} \max \Big(0, &- \log \hat{\text{P}}_\theta^{IPS}(S[i_1] = 1 \mid S[j] = 1) \\
&+ \log \hat{\text{P}}_\theta^{IPS}(S[i_2] = 1 \mid S[j] = 1)\Big). \qquad (6)
\end{aligned}$$

Again, θ is the set of parameters of our propensity model specifying $p_{k,i}^{-1}$ in the IPS estimator. For this loss to be well-specified, we need to have $\hat{\text{P}}_\theta^{IPS} > 0$ which we ensure by filtering out invalid pairs in C_j in practice. The final empirical risk of an entire dataset of judgements $C = (C_1, \ldots, C_L)$ is then simply the average of over all judgements,

$$\arg\min_\theta \frac{1}{L} \sum_{j=1}^{L} \ell(C_j; \theta). \qquad (7)$$

Optimizing this loss is typically fast in practice as it is both linear in the number of seed items L and their sizes C_j, i.e., is in $O(L \cdot \max |C_j|)$. The the number of seed items L is typically small (e.g., $L < 100$ in our experiments) and sampling can be used to reduce the sizes of each set C_j. After learning, the recommendation step simply involves ranking all candidate items by $\text{P}(S[i_1] = 1 \mid S[j] = 1)$.

4.2 Exponential item-based propensity model

Even though we only detail one specific propensity model here, our approach can be used with virtually all differentiable models. We will discuss alternative models and tradeoffs in Section 7.1. For our experiments, we chose an exponential model with a linear scoring function for the inverse propensity, naturally giving us valid values in the interval of $[0, \infty]$,

$$p_{k,i}^{-1} = \exp\left(\theta^T \cdot \phi(\mathcal{X}_k)\right).$$

Here, $\phi(\mathcal{X}_k)$ is a feature map taking in the current context and mapping it to a representation. Furthermore, we restrict the mapping function ϕ to only use information about the current candidate item i and the seed item j which is why we refer to it as an *item-based* model. This let's us drop the subscript k with a slight abuse of notation:

$$p_{ij}^{-1} = \exp\left(\theta^T \cdot \phi(i, j)\right). \qquad (8)$$

Plugging this propensity model into the IPS estimator then results in the following simplified estimator:

$$\begin{aligned}
\log \hat{\text{P}}_\theta^{IPS}(S[i] = 1 \mid S[j] = 1) = Z_j &+ \theta^T \cdot \phi(i, j) \\
&+ \log \underbrace{\sum_k \mathbb{1}\left[S^{\text{obs}}[i] = 1 \wedge S_k^{\text{obs}}[j] = 1\right]}_{:= \text{count}(i,j)}. \qquad (9)
\end{aligned}$$

Using this with the loss in Equation (6) causes the normalization terms Z_j to cancel gives us

$$\begin{aligned}
\ell(D_j; \theta) = \sum_{(i_1, i_2) \in C_j} \max \Big(0, &-\theta^T \cdot \phi(i_1, j) + \log \text{count}(i_1, j) \\
&+ \theta^T \cdot \phi(i_2, j) + \log \text{count}(i_2, j)\Big). \qquad (10)
\end{aligned}$$

This loss function only involves co-occurrence counts of pairs of items in the data as well as item features; both of which only have to be computed once before learning. After learning, we then score items by their IPS estimate in Equation (9).

4.2.1 ItemKNN as a special case. ItemKNN with cosine similarity is a common and competitive baseline [31] which turns out to be a special case of IPS estimator with the item-based propensity model above.

$$\begin{aligned}
\log \text{cosine}(i, j) &= \log \frac{S[i]^T \cdot S[j]}{\|S[i]\|^\alpha \, \|S[j]\|^{1-\alpha}} \\
&= \underbrace{-(1-\alpha) \log \text{count}(j)}_{=Z_j} \underbrace{- \alpha \log \text{count}(i)}_{=\theta^T \cdot \phi(i,j)} \\
&\quad + \log \text{count}(i, j). \qquad (11)
\end{aligned}$$

Comparing (9) and (11) we can see the equivalence – itemKNN with a generalized cosine similarity corresponds to using the IPS estimator with an item-based model where the feature map only considers item i's popularity. This helps explain why it has been observed to remove some popularity bias in practice.

5 EXPERIMENTAL SETUP

For our experiments, we compare our method against a wide array of competitive baselines on the well-known MovieLens 25M dataset. For the relevance labels, we use similarity judgements that have been collected previously [38] as similarity is a big driver of engagement [39]. To the best of our knowledge, this is only recommendation dataset for which both observational data as well as explicit judgements is available for.

5.1 Data

For the similarity judgements, we used the dataset collected by Yao and Maxwell [38]. For it, the authors chose 100 seed movies j from the MovieLens 25M dataset, balanced by popularity, and generated recommendations by various algorithms for it. MovieLens users who watched these movies were asked to assign a rating from 1-5 to them. We processed the raw ratings as follows. To decrease noise [26], we required judgements to be made by at least two users and binned the judgements by their average score. Following the process we mentioned in Section 4.1, we induced pairwise judgements by treating all samples that had an average score of greater than three as relevant. To ensure that sufficiently many movies had been judged, we only kept seed movies j which had more than four pairs with a positive similarity score. This resulted in 67 seed movies with 10.3 relevant candidates on average which where split into a training, validation and test dataset (34 / 17 / 16 seed movies respectively). For training, we assumed that all other items were less relevant and used this to induce preference pairs. For evaluation, since absolute labels of relevance had to be used with the evaluation metrics and not all positive items had been annotated, we carefully designed our setup as outlined in the next subsection.

For the observational data, we converted the MovieLens 25M dataset [11] into binary engagements by treating all ratings of greater or equal to three as positive engagements, following prior work (e.g., [35]). We defined sessions to comprise a user's entire rating history but note that other definitions are possible as discussed in Section 2.1. We also removed movies with fewer than 30 positive engagements from the inventory because there were no similarity judgements available for those. This left us with 20,314,297 ratings for 13,987 movies from 162,542 users.

5.2 Task and Metrics

Given the bias that exists in observational datasets, we cannot follow the standard hold-out procedure [30] as it would not allow us to test the counterfactual performance of our models. For example, in hold-out testing, a model would still be rewarded for recommending popular movies as they frequently co-occur.

We instead use the following simple ranking task, based on the fact that we have a set of positively labeled similarity judgements $\text{rel}(i, j)$ for each target item j. Given a target movie j, we rank the entire inventory $\mathcal{I}$ of movies with the goal of retrieving *all* positively labeled candidate movies as far up in the ranking as possible.

We measure performance via two metrics. Recall@K, which is simply the fraction of relevant movies ranked in the top K results. We chose higher values of $K = 25, 50$ and 100 to mitigate the difficulties that missing positively labeled movies pose – a model can still rank other candidate movies at the top if it feels confident in them. We also report the average ranks of all relevant movies in each ranking to understand performance beyond the top K. We also note that other ranking measures such as nDCG or MAP are too heavily focused on the few top results and therefore not appropriate in our case.

5.3 Baselines

We considered a wide range of baselines, including both factorization-based as well as count-based methods. Despite having been in existence for at least multiple years, they remain state-of-the-art on many datasets [8, 29].

- **Random.** Randomly shuffles all movies in the inventory to produce a ranking.
- **Pop.** A static baseline that ranks items according to their popularity in the observational dataset, i.e., the number of sessions that the item occurred, $\text{count}(i)$. Though simple, popularity has been found to be a competitive baseline in many domains.
- **Supervised.** Here, we train a supervised gradient-boosted decision tree on the same features that Ours receives, plus we add the $\text{count}(i, j)$ to the features so that both methods have the same amount of information. The classification task is to correctly predict the label of $\text{rel}(i \mid j)$ as 0 or 1.
- **Cooccur.** Estimates the probability of engaging with a movie i given a seed movie j via the MLE estimator in Equation 3. This essentially corresponds to ranking items i by the number of times they occur together, $\text{count}(i, j)$.
- **ItemKNN.** We used the item-based k-nearest neighbors method with a generalized cosine similarity presented in Equation (11) as proposed in [13].
- **PureSVD.** Simply factorizes the entire user-item matrix without any reweighing.
- **WmRF [14].** Weighted regularized matrix factorization where zero entries are receive a lower weight.
- **Bpr [28].** Bayesian personalized ranking is a current and well-performing method for pairwise matrix factorization from implicit feedback. It departs from classic matrix factorization by treating entries in the user-item matrix as relative preferences instead of absolute targets.
- **Slim [25].** Treats recommendation as a multivariate regression problem, predicting each item column $S[i]$ given all other item columns while encouraging for the regression weights to be sparse. We use the weights given to each of the other items directly as ranking scores.

5.4 Implementation Details

For all methods, we chose the optimal hyperparameter settings with respect to Recall@100 performance on the validation set. We used the implementation of the `implicit` library[1] for Bpr and WmRF with offsets being disabled to make scoring during training and inference more consistent. For both methods, we tuned the number of factors as well as the amount of L_2 regularization. Slim was implemented via the `ElasticNet` procedure provided by `scikit-learn` and we considered both the amount of L_1 and L_2 regularization during hyperparameter tuning. Similarly, the gradient-boosted decision tree was also implemented via `scikit-learn` and we tuned the number of trees and the maximum depoth. For ItemKNN, we optimized its weighing term α.

For our method, we implemented the objective of Equation (7) in `pyTorch` and did batch optimization via the Adam algorithm [18]. Because the number of pairs in C_j can grow quadratically in general,

[1] https://github.com/benfred/implicit/

Table 2: Ranking performance on the test set under various metrics. The best numbers of each column are in bold. Ours achieves the highest recall under all cutoff points k, as well ranks relevant items closest to the top.

method	Recall@25	Recall@50	Recall@100	mean ranks
Random	0.000	0.000	0.000	6959.3
Pop	0.000	0.016	0.025	1850.4
Supervised	0.399	0.539	0.646	520.7
Cooccur	0.058	0.123	0.268	676.3
ItemKNN	0.436	0.529	0.594	450.9
PureSVD	0.356	0.450	0.532	673.8
WmRF	0.361	0.469	0.539	890.5
Bpr	0.365	0.455	0.515	785.8
Slim	0.487	0.639	0.657	2191.5
Ours	**0.532**	**0.652**	**0.761**	**213.5**

we constructed sampled versions of the training set $\tilde{D}_j$ where we kept all relevant items of D_j and sampled T irrelevant items from the remaining inventory $\mathcal{I} \setminus \{(i,j) \in D_j : \text{rel}(i,j) = 1\}$. For the features of a pair (i,j), we used the release year of i, the year when the first ratings were recorded for i in MovieLens, and the log-transformed popularity of i, the proportion of engagement with respect to i's popularity ($\text{count}(i,j)/\text{count}(i)$). We also used relative versions of the first three features we computed the absolute difference of the feature values between i and j. The hyperparameters we optimized were the number of negative samples T and the learning rate.

6 RESULTS AND DISCUSSION

Quantitative Results. Table 2 shows the performance of all method under various metrics with best results in bold. Ours improves upon all baselines and across all metrics. It improves recall scores by up to 16% and achieves a substantially lower mean rank of relevant movies.

Going through the individual methods, we can see that Random and Pop only produce recall values close to zero, demonstrating the high difficulty level of this task. Cooccur based on the MLE estimator in Section 3.1 performs somewhat better than Pop, but still has overall poor performance, reiterating the importance of accounting for confounding properly. ItemKNN, as a special case of Ours, is the third-best performing method in terms of recall, demonstrating the impact that even addressing popularity bias has. It also has the second-best mean rank score so that overall, ItemKNN provides robust performance while being conceptually simple which helps explain why it has been a popular choice among practitioners [21]. Supervised fairs slightly worse than ItemKNN on some dimensions and slightly better on others. We hypothesize that this is because it uses the co-occurrence counts as features which may generalize worse. Interestingly, all methods relying on low-rank matrix factorization, WmRF, PureSVD and Bpr, perform similarly. Albeit worse than ItemKNN, they still improve upon Cooccur's performance. We believe part of the improvement comes from the smoothing properties of low-rank matrix factorization which may dampen some of the biases present in the data. However, fundamentally, these factorization methods still are based on co-occurrences, and

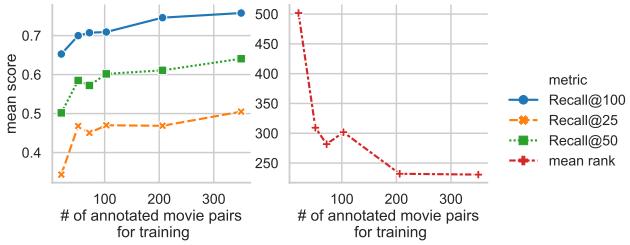

Figure 1: Test set performance with varying sizes of annotated data for training (averages from 5 runs each). Close to full performance is reached after learning from only about 200 annotated movie pairs.

so they all are also still subject to confounding of the latter. The best performing baseline with regard to recall is Slim. It particularly excels at higher ranks as is evidenced by the relative improvements over other methods at Recall@25 and Recall@50. Slim is fundamentally different from the other low-rank factorization methods since it uses regression coefficients directly as item similarities. This apparently helps again address some of the biases in the data; for example it likely de-emphasize the highly frequent but not very predictive co-occurrences. However, Slim fails to retrieve some items at all at reasonable cutoffs; its mean rank score of 2191 is the second worst overall. These results make sense given Slim's underlying L_1 sparsity assumption which strongly encourages the model set regression coefficients to zero. Finally, Ours shows superior performance across all metrics. In particular, it shows relevant movies higher up on average than any of the other models. Ours's strong performance is evidence for the importance of properly addressing confounding in observational data. Moreover, with Ours only using a few more features during scoring, it virtually possesses the same computational cost as ItemKNN and Cooccur during inference, making it a highly practical method.

Qualitative Results. To give more intuition for the recommendations that the individual methods produce, Table 3 shows the top ten recommendations for the animated movie "Toy Story (1995)". While Ours retrieves all animated movies in the top, the other three baselines struggle to produce consistently good rankings. All three baselines still contain popular movies but irrelevant movies, such as "Star Wars IV (1977)" or "Back to the Future (1985)", implying that they have only insufficiently adjusted for confounders such as candidate item popularity. Table 3 also illustrates why it is necessary to use larger cutoff values when evaluating rankings; there are typically many relevant movies for a target movie j, but only a few are actually annotated in the data. With so many potentially relevant items, we must give models the freedom to rank other items at the top as long as they sufficiently recall the annotated examples.

Performance vs. size of annotated data. One important question for the practicality of our method is how much annotated data would be needed to achieve an adequate performance level. For this, we sampled a varying number of target movies j from the annotated training set and report ranking performance on the test set. Note that each target movie j came with around 10.3 annotated

Table 3: Top ranked movies given the target movie "Toy Story (1995) for three well performing baselines.

rank	Ours	ItemKNN	Slim	WmRF
1.	Toy Story 2 (1999)	Toy Story 2 (1999)	Toy Story 2 (1999)	Toy Story 2 (1999)
2.	Toy Story 3 (2010)	Willy Wonka & t... (1971)	Toy Story 3 (2010)	Toy Story 3 (2010)
3.	Finding Nemo (2003)	Back to the Future (1985)	Willy Wonka & t... (1971)	Muppet Treasure... (1996)
4.	Incredibles, The (2004)	Monsters, Inc. (2001)	Aladdin (1992)	James and the Gi... (1996)
5.	Monsters, Inc. (2001)	Lion King, The (1994)	Star Wars IV (1977)	Willy Wonka & t... (1971)
6.	Shrek 2 (2004)	Bug's Life, A (1998)	Monsters, Inc. (2001)	Bug's Life, A (1998)
7.	Shrek (2001)	Independence Day (1996)	Independence Day (1996)	101 Dalmatians (1996)
8.	Bug's Life, A (1998)	Star Wars IV (1977)	Back to the Future (1985)	Space Jam (1996)
9.	Ratatouille (2007)	Aladdin (1992)	James and the Gi... (1996)	Star Wars IV (1977)
10.	Up (2009)	Star Wars VI (1983)	Finding Nemo (2003)	Aladdin (1992)

candidates, and we report performance with respect to the total number of annotated pairs of movies. The results are shown in Figure 1. Each marker represents the average of five repetitions with the same size. We can see that performance quickly increases in the beginning, reaching almost maximum performance at about 200 labeled movie pairs. Overall, this demonstrates the high efficiency and practicality of our approach as even small amounts of labeled data help in producing better recommendations.

7 DISCUSSION AND FUTURE WORK

Both our quantitative and qualitative results show that even having small amounts of annotated data to learn from can substantially reduce the amount of bias in observational data. Common methods that are based on low-rank matrix factorization showed only mediocre performance in item-to-item recommendation. This is not surprising given that they are essentially factorizing the (counfounded) co-occurrence matrix [20]. Among the baselines, we found that ItemKNN and Slim managed to compensate for some of the confounding in the data, albeit not in a principled way. Our approach instead is grounded in counterfactual learning. With that underpinning, we can leverage existing debiasing techniques as well as reason about how propensity models should be chosen based on the causal assumptions one has. We will discuss some of these issues in Section 7.1.

An exciting consequence of adopting a counterfactual perspective is that our framework produces *reusable* propensities that could be plugged into propensity-weighted risk minimizers for traditional matrix factorization models [30]. This means that one could use a small annotated dataset to estimate propensities once, and then not only use them for item-to-item recommendation but also to debias the empirical risk estimates of a factorization model [30]. We plan to pursue this direction in future work.

Another interesting avenue for extensions is to examine other modeling assumptions that link annotated data to an underlying propensity model. For example, in this paper, we only considered the pairwise ordering constraint that we introduce in Section 4.1. However, many other choices may be viable too, for example an assumption saying that the relevance score is proportional to the full-information conditional probability, $\text{rel}(i \mid j) \propto \text{P}(S[i] = 1 \mid S[j] = 1)$.

7.1 Considerations in choosing propensity models

So far, we have only stated our choice for the propensity model that we used in our experiments. Here, we want to discuss two key aspects of a propensity model to consider – namely the statistical efficiency of a propensity model and its causal validity. Statistical efficiency is the number of annotated samples that we need to estimate all parameters of a propensity model while causal validity is the degree to which a model is able to capture the confounding we suspect in the data. The two aspects do not necessarily have to be trade-offs, but often are, as more complex models tend to be less statistically efficient but have more capacity to model causal relationships.

In what follows, we will compare two different propensity models for a simplified version of the movie domain. Figure 2 shows the causal graph for this scenario, focusing on only one item i. We can see that item i's time in inventory, its popularity and release year not only determine the engagement $S^{\text{obs}}[i]$, but also influence whether we get to observe the engagement in $O[i]$ through a latent factor we call salience. The latter is what causes the confounding of the data we observe. Causal inference now tells us that in order to get an unbiased estimate, we need to have the propensity model that uses the set of variables $\mathcal{X}$ such that this set fulfills the so-called back-door criterion [27] with respect to $O[i]$. In our graph, a set of variables $\mathcal{X}$ meets the back-door criterion if (i) $\mathcal{X}$ blocks all paths from $O[i]$ into $S^{\text{obs}}[i]$; and (ii) $\mathcal{X}$ contains no descendant of $O[i]$. In our graph from Figure 2, there are two different, but causally equivalent ways of meeting the back-door criterion. The first option is to consider $\mathcal{X} = \{i$'s time in inventory, popularity of i, release year of $i\}$ and use it in an attribute-based model analogous to what we did in Section 4.2. For a linear model, this would result in three parameters to be estimated. The second option would be to choose $\mathcal{X} = \{\text{salience of } i\}$. One can verify that this too, blocks all paths from $O[i]$ into $S^{\text{obs}}[i]$. This then corresponds to learning a propensity model that has a separate salience value for each item $p_{k,i} = \theta_i$ resulting in $|\mathcal{I}|$ many parameters to be estimated. So, in terms of statistical efficiency the first propensity model is preferable over the second one since it has fewer parameters to estimate. Both models also have the same causal validity, thus making the first model more preferable overall in this case. In what situations would one choose the second model? One such situation is when we

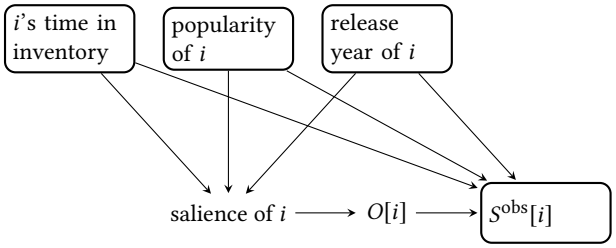

Figure 2: Causal graph for a simplified version of the partial information setting in the movie domain. A movies time in the inventory, its popularity and release year are all confounding factors in the engagement data we get to observe $S^{\text{obs}}[i]$.

cannot observe all of the factors contributing to the salience of an item, for example we may not know an item's release year. In this situation, the second propensity model is still causally valid while the first one is not. In summary, we recommend to consider the causal structure of a domain as well as the statistical properties of a propensity model when choosing a model in practice.

8 RELATED WORK

Our work touches on work from three different subareas of recommender systems, item-to-item recommendation as a task, work on estimating item similarities, and work on recommending from biased observational data. In this work, we use the term item-to-item recommendation exclusively to refer to the *task* of finding related items given a seed item. This is different from the *technique* of item-item collaborative filtering which is not the subject of this paper where ratings are predicted from items a user has consumed [23].

8.1 The task of item-to-item recommendation

The task of item-to-item recommendation, also referred to as "*more like this*" or *related item* recommendation has a long standing in real-world recommender systems. Even though the exact ways in which those recommendations are surfaced differ, for example as a side panel or at the end of a page, one common theme is that they get shown in response to user having engaged or clicked on an item. Online retailers use item-to-item recommendation as a natural way to help people explore their inventory [17, 21]. Other application domains are video streaming services [9] or news websites [1]. The task of item-to-item recommendation may be solved by using different techniques; for example by solving a classification problem [7]. Instead of focusing only on one specific technique, we formalize item-to-item recommendation as a general estimation problem where we need to estimate the probability of an item being engaged with given another seed item. This allows us to characterize the observational biases that occur in a partial information setting as well as adjust for the latter in a principled manner.

8.2 Estimating item similarities

Having a good understanding of similarities between items is an important part of real-world recommender systems [3, 39]. There are roughly two approaches to estimating item similarities – attributed-based (or content-based) methods leverage characteristics of item themselves and collaborative filtering-based approaches which look at how the interactions with the items from a crowd of users.

Attribute-based approaches for similarity estimation require that items can be represented by a list of attributes that are thought to be relevant for establishing relatedness. For example, Winecoff et al. represent dresses by sleeve length, dress length, pattern style and color family [36]. In the movie domain, Yao and Maxwell [38] found in a user survey on MovieLens that users strongly consider the genre, mood and plot of movies when assessing similarity. Trattner and Janner [33] found similar results when comparing various similarity functions via crowdsourcing. In both studies, however, the similarity function that was best correlated with human judgements was based on user-assigned tags for each movie [34]. Another broader class of attributes explored is text associated with items, for example user reviews [19]. The large limitation of attribute-based methods is that they pose a substantial cost for cataloging all items in the inventory, moreover, complex items such as art are hard to encode. For this reason, we do not consider attribute-based methods in this paper. Most propensity model require none of only *some* information about items needs to be known, but this is only the information that is involved in debiasing the data. In the movie domain, the information needed to debias the observational data is only coarse metadata, while for attribute-based models one would need fine-grained genre, mood, and plot information [38].

Collaborative-filtering (CF) based methods rely on information based on how a crowd of users interactions with the items. For example, for movies, similarities are typically derived based on all the ratings that users assigned to an item. CF-based approaches differ in whether raw interaction signals are used or lower dimensional embeddings. A prominent example of the former is ItemKNN [10] which most often uses the cosine similarity between items' interaction vectors to compute similarity. Cosine similarity is only one possible similarity measure and others have been examined, however it has been found to work best empirically [6, 31]. Other similarity measures can be constructed from using the cosine similarity on the resulting item embeddings of factorization methods [14, 28] or embedding items directlyItem2Vec [2]. We compare against current CF-based similarity measures such as similarities induced by weighted regularized matrix factorization [14] and Bayesian personalized ranking [28], and show that their performance is substantially impeded by biases in the data.

8.3 Recommendations from biased data

It has long been noted that virtually all data that collaborative filtering methods is trained with suffers from selection bias, meaning that the data we observe is missing not at random. For example, it is well known that users are less likely to provide ratings for movies they do not like [24]. Implicit data such as clicks is often confounded by the time that an item is in the inventory [7], or interface biases such as the position on the screen [16]. Older approaches to dealing with these biases include choosing more robust target metrics [32], or modeling the generative process of missing data explicitly [12, 24]. Newer approaches [30, 37] take a counterfactual perspective, resulting in a principled approach for adjusting for

selection bias via propensities. We follow these steps by providing a counterfactual view of item-to-item recommendation and complement previous work by showing how to estimate propensities from a small amount of annotated data alone.

9 CONCLUSIONS

Observational data abounds in modern recommender systems. However, it is often confounded by a variety of factors, such as popularity effects or presentation biases which can cause naive methods to fail. In this paper, we treat item-to-item recommendation from observational data as a fundamentally counterfactual problem, making it possible to adjust for confounding and biases in a principled manner using propensity models. Our core contribution in this paper is to provide a general framework leveraging small annotated datasets for debiasing larger, observational datasets for item-to-item recommendation. Our key innovation is to use the small annotated dataset to infer the parameters of a reusable propensity model. Our experiments on real-world movie recommendation data show that this substantially improves recommendation performance over existing methods.

ACKNOWLEDGMENTS

We would like to thank Adith Swaminathan and Longqi Yang for their in-depth feedback on earlier drafts of this work. We are also grateful for the helpful comments and suggested revisions we received from the anonymous reviewers.

REFERENCES

[1] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Raghu Ramakrishnan. 2013. Content recommendation on web portals. *Commun. ACM* 56, 6 (2013), 92–101.
[2] Oren Barkan and Noam Koenigstein. 2016. Item2Vec: neural item embedding for collaborative filtering. In *Workshop on Machine Learning for Signal Processing (MLSP)*. 1–6.
[3] Michael Bendersky, Lluis Garcia-Pueyo, Jeremiah Harmsen, Vanja Josifovski, and Dima Lepikhin. 2014. Up next: retrieval methods for large scale related video suggestion. In *KDD*. 1769–1778.
[4] Stephen Bonner and Flavian Vasile. 2018. Causal embeddings for recommendation. In *RecSys*. 104–112.
[5] Ben Carterette, Paul N Bennett, David Maxwell Chickering, and Susan T Dumais. 2008. Here or there. In *ECIR*. Springer, 16–27.
[6] Lucas Colucci, Prachi Doshi, Kun-Lin Lee, Jiajie Liang, Yin Lin, Ishan Vashishtha, Jia Zhang, and Alvin Jude. 2016. Evaluating item-item similarity algorithms for movies. In *CHI Extended Abstracts*. 2141–2147.
[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.
[8] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *RecSys*. 101–109.
[9] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. 2010. The YouTube video recommendation system. In *RecSys*. 293–296.
[10] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.
[11] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. 5, 4 (2015).
[12] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani. 2014. Probabilistic Matrix Factorization with Non-random Missing Data. In *ICML*. 1512–1520.
[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and D Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.
[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. 263–272.
[15] Guido W Imbens and Donald B Rubin. 2015. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press.
[16] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *WSDM*. 781–789.
[17] Jayasimha Katukuri, Tolga Könik, Rajyashree Mukherjee, and Santanu Kolay. 2014. Recommending similar items in large-scale online marketplaces. In *Big Data*. 868–876.
[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[19] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. 1188–1196.
[20] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. 2016. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *RecSys*. 59–66.
[21] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.
[22] R. J. A. Little and D. B. Rubin. 2002. *Statistical Analysis with Missing Data*. John Wiley.
[23] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*. Springer, 73–105.
[24] B. M. Marlin, R. S. Zemel, S. Roweis, and M. Slaney. 2007. Collaborative Filtering and the Missing at Random Assumption. In *UAI*. 267–275.
[25] Xia Ning and George Karypis. 2011. SLIM: Sparse linear methods for top-n recommender systems. In *ICDM*. 497–506.
[26] Peter Organisciak and J Stephen Downie. 2015. Improving consistency of crowdsourced multimedia similarity for evaluation. In *Joint Conference on Digital Libraries*. 115–118.
[27] Judea Pearl. 2009. *Causality*. Cambridge University Press.
[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[29] Steffen Rendle, Li Zhang, and Yehuda Koren. 2019. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395* (2019).
[30] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In *ICML*. 1670–1679.
[31] Ellen Spertus, Mehran Sahami, and Orkut Buyukkokten. 2005. Evaluating similarity measures: a large-scale study in the Orkut social network. In *KDD*. 678–684.
[32] H. Steck. 2010. Training and testing of recommender systems on data missing not at random. In *KDD*. 713–722.
[33] Christoph Trattner and Dietmar Jannach. 2020. Learning to recommend similar items from human judgments. *User Modeling and User-Adapted Interaction* 30, 1 (2020), 1–49.
[34] Jesse Vig, Shilad Sen, and John Riedl. 2012. The Tag Genome: Encoding community knowledge to support novel interaction. *Interactive Intelligent Systems (TiiS)* 2, 3 (2012), 1–44.
[35] Menghan Wang, Mingming Gong, Xiaolin Zheng, and Kun Zhang. 2018. Modeling dynamic missingness of implicit feedback for recommendation. In *NeurIPS*. 6669–6678.
[36] Amy A Winecoff, Florin Brasoveanu, Bryce Casavant, Pearce Washabaugh, and Matthew Graham. 2019. Users in the loop: a psychologically-informed approach to similar item retrieval. In *RecSys*. 52–59.
[37] Longqi Yang, Yin Cui, Yuan Xuan, Chenyang Wang, Serge Belongie, and Deborah Estrin. 2018. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In *RecSys*. 279–287.
[38] Yuan Yao and F Maxwell Harper. 2018. Judging similarity: a user-centric study of related item recommendations. In *RecSys*. 288–296.
[39] Yifan Zhong, Tahir Lazaro Sousa Menezes, Vikas Kumar, Qian Zhao, and F Maxwell Harper. 2018. A field study of related video recommendations: newest, most similar, or most relevant?. In *RecSys*. 274–278.