It Is Different When Items Are Older: Debiasing Recommendations When Selection Bias and User Preferences Are Dynamic

1 INTRODUCTION

User interactions with recommender systems (RSs) are subject to selection bias, as a consequence of the selective behavior of users and of the fact that RSs actively restrict the items from which a user can choose [32, 34, 37, 41, 43]. A typical form of selection bias in RSs is popularity bias: popular items are often overrepresented in interaction logs because users are more likely to rate them [7, 37, 43]. Without correction, bias can affect user preference prediction [22, 41, 56] and lead to problems of over-specialization [1], filter bubbles [33, 36], and unfairness [9]. To correct for selection bias in interaction data from RSs, the task of debiased recommendation has been proposed. A widely-adopted method for this task makes use of inverse propensity scoring (IPS), a causal inference technique [24], and integrates it in the learning process of rating-prediction for recommendation [10, 22, 27, 41]. It estimates the probability of a rating to be observed in the dataset, and inversely weights ratings according to these probabilities so that in expectation each user-item pair is equally represented.

While the existing IPS-based debiasing method improves recommendations over methods that ignore the effect of bias, we identify two significant limitations. The way that IPS-based debiasing is being applied for recommendations assumes that (1) the effect of selection bias is static over time, and (2) user preferences remain unchanged as items get older. As we will show in Section 4, current IPS-based methods are unable to debias recommendations when the selection bias and user preferences are dynamic, i.e., when they change over time.

In practice, selection bias is usually dynamic, not static [9, 26]. Typically, the popularity of an item changes with item-age [8, 26], i.e., the time since its publication. Figure 1 shows the number of ratings items received as they get older in the MovieLens dataset (red line). On average, items receive the most attention during a short initial period of time after being published. Hence, instead of static selection bias, real-world user behavior may be better captured with dynamic selection bias that assumes different probabilities of observing user ratings at different item-ages. Besides selection bias, user preferences may also change over time [2, 25, 50]. In this paper, we will focus on the effect of item-age on user preferences, and thus, on capturing the change in user preferences as items become older. From Figure 1, it is clear that the average observed user rating varies with the item-age (blue line), despite the increased variance observed due to a decreasing number of logged interactions. We use the term dynamic scenario to refer to the combination of dynamic selection bias and dynamic user preferences occurring in a recommendation setting.

In this paper we first analyze real-world logged data to verify that the dynamic scenario is real: selection bias and user preferences are dynamic. The dynamic scenario poses a two-fold problem for existing IPS-based debiasing methods for RSs. First, they are not valid.
unbiased in dynamic scenarios. Second, existing methods \cite{7, 41} for estimating static selection bias cannot be used to estimate dynamic selection bias. Hence, we propose and evaluate a debiasing method to account for dynamic selection bias and dynamic user preferences.

All in all, we make a three-fold contribution: (1) an analysis and estimation of dynamic selection bias and dynamic user preferences in the MovieLens dataset; (2) DANCER: a general debiasing method that is adaptable for DebiAsing in the dyNaMic scEnaRio; and (3) time-aware matrix factorization (TMF)-DANCER: to our knowledge it is the first recommendation method that corrects for dynamic selection bias and models dynamic user preferences.

2 RELATED WORK

General Recommendation. Early work on RSs typically uses collaborative filtering (CF) to predict user ratings on items or make recommendations to users based on the feedback of similar users with similar behavior. It is customary to divide recommendation tasks into the rating prediction task with explicit feedback (e.g., user ratings) and the top-K ranking task with implicit feedback (e.g., clicks). In this paper, we focus on rating prediction with explicit feedback. The traditional matrix factorization (MF) algorithm directly embeds users and items as vectors and models user-item interactions with an inner product \cite{15, 30}. Some recent work has used deep neural networks to improve CF, e.g., by using multi-layer perceptrons \cite{12, 19}, convolutional neural networks \cite{18}, or graph neural networks \cite{17, 48}. While they significantly improve recommendation accuracy \cite{28}, they ignore the effect of time.

Time-aware Recommendation. Recently, a wide range of algorithms have been proposed that consider temporal information to improve RSs. Such methods are often classified as time-aware or sequence-aware recommendation methods. Sequence-aware recommendation methods focus on the sequential order of interactions and aim to capture a user’s short-term preferences \cite{38, 59} such as recurrent neural networks \cite{21, 51, 57}, graph neural networks \cite{52, 54}, and networks with attention \cite{11, 23, 45}.

We focus on time-aware recommendation methods \cite{6} rather than sequence-aware recommendation methods, by considering changes in user preferences over exact time-periods. One of the best known examples is time-aware matrix factorization (TMP) \cite{29}, which takes the effect of time into consideration by adding time-dependent terms to the MF model, thus allowing predicted ratings to vary over time. Koren \cite{29} lists and compares various variants of TMF, in how well they can capture item-related or user-related temporal effects. Xiong et al. \cite{53} propose time-aware tensor factorization (TTF): a factorization based model that uses additional latent factors for each time period based on a probabilistic latent factor model. Lastly, the effect of time is sometimes modelled by utilizing contextual attributes related to time (e.g., day of the week or season of the year) as input features for context-aware RSs \cite{4, 6, 35, 47}.

Debiased Recommendation. User selection bias is prevalent in logged data, meaning that many logged user ratings are missing not at random (MNAR) \cite{20, 32, 41}. Two typical forms of bias in RSs are known as popularity bias and positivity bias. Positivity bias is characterized by a long tail distribution over the number of interactions per item logged data because users are more likely to interact with more popular items \cite{37, 43}. Positivity bias leads to an over-representation of positive feedback because users rate the items they like more often \cite{37}. The effect of these biases is generally dynamic: they can change drastically over time \cite{9, 26, 58}. For instance, items are rarely popular for very extended periods of time, and therefore, we may expect a dynamic effect between the age of items and popularity bias.

Existing debiasing methods for reducing the effect of selection bias address MNAR problems as follows: (1) the error-imputation-based model (EIB) fills in missing ratings with predicted values, which may introduce bias due to inaccurate predictions \cite{42}, (2) inverse propensity scoring (IPS) weights the loss associated with each observed rating inversely to their propensity, i.e., the probability of observing that rating \cite{10, 27, 41}, and (3) the doubly robust (DR) method integrates the EIB and IPS approaches to overcome the high variance of IPS and the potential bias of EIB \cite{49}.

While the impact of dynamic bias has previously been pointed out \cite{26, 58}, no prior debiasing method considers a scenario in which both selection bias and user preferences change over time. All existing debiasing recommendation methods assume a static effect of selection bias regardless of whether they model dynamic user preferences. Hence, there is currently no method that can effectively correct for bias in the dynamic scenario. This is the research gap that we address.
time periods in $\mathcal{T}$, it likely also generalizes well into the near future.

In our setting, logged interaction data is available to provide user ratings that can be used for optimization. However, it is unrealistic for all users to provide ratings for all items. In practice, user interaction data is very sparse. We will use an observation indicator matrix $O \in \{0, 1\}^{|U| \times |I| \times |T|}$ that indicates what ratings are recorded in the logged interaction data and during which time period. We use $o_{u,i,t} \in O$ to indicate this per rating: $o_{u,i,t} = 1$ indicates that the rating for user $u$ on item $i$ during time period $t$ has been recorded in the logged data, and $o_{u,i,t} = 0$ that it is missing. The matrix $O$ is strongly influenced by selection bias: certain ratings are much more likely to be observed than others. This can be due to self-selection bias: users choosing to rate certain items more often [37, 43]; or algorithmic bias: the RS used for logging choosing to show certain items.

4.1 Effect of Dynamic Selection Bias

Ignoring dynamic selection bias, the recommendation methods that use the naive or static IPS estimation are not unbiased in dynamic scenarios. To illustrate how this may happen, we use a simple example $X$ with one user $u$, one item $i$ and two time periods $t_1$ and $t_2$. Let $y_{u_1}$ and $y_{u_2}$ be the user ratings on the item at $t_1$ and $t_2$ respectively; $p_{i_1}$ and $p_{i_2}$ denote the probabilities of observing the ratings at $t_1$ and $t_2$, respectively. We omit the subscript of $u$ and $i$ if no confusion can arise. Due to dynamic user preferences and dynamic selection bias, the user ratings and observation probabilities are not constant over the different time periods: $y_{u_1} \neq y_{u_2}$, $p_{i_1} \neq p_{i_2}$. Remember that in this example the loss we wish to estimate is:

$$L^X = \frac{1}{2} (L(y_{u_1}, y_{t_1}) + L(y_{u_2}, y_{t_2})).$$

The expected naive loss over the observation variables becomes:

$$\mathbb{E}[L_{\text{Naive}}^X] = p_{i_1} L(\hat{y}_{u_1}, y_{t_1}) + p_{i_2} L(\hat{y}_{u_2}, y_{t_2}) - \frac{p_{i_1} p_{i_2}}{2} (L(\hat{y}_{u_1}, y_{t_1}) + L(\hat{y}_{u_2}, y_{t_2})).$$

Clearly, it is not proportional to the true loss $L^X$ when selection bias and user preferences are dynamic: if $y_{u_1} \neq y_{u_2}$ and $p_{i_1} \neq p_{i_2}$, then $\mathbb{E}[L_{\text{Naive}}^X] \neq L^X$. This happens because the rating with the higher probability of being observed is over-represented in the observations.

Then the static IPS-based debiasing method uses static propensity $p_{u,i} = p_{i_1} + (1 - p_{i_1}) p_{i_2}$, that is the probability of observing a rating at time $t_1$ or $t_2$. If we consider the expected value of this estimator:

$$\mathbb{E}[L_{\text{staticIPS}}^X] = \frac{1}{2} \left( \frac{p_{i_1}}{p_{u,i}} L(\hat{y}_{u_1}, y_{t_1}) + \frac{p_{i_2}}{p_{u,i}} L(\hat{y}_{u_2}, y_{t_2}) \right),$$

we see that it is not proportional to the true loss in the dynamic scenario: if $y_{u_1} \neq y_{u_2}$ and $p_{i_1} \neq p_{i_2}$, then $\mathbb{E}[L_{\text{staticIPS}}^X] \neq L^X$, because the static IPS estimation fails to address the problem that the user’s rating at a time with a higher probability of being observed is more likely to be represented in logged data than at any other time. We note that the above counterexample holds regardless of whether the prediction of user ratings allows for dynamic preferences, i.e., whether $y_{u_1} = \hat{y}_{u_2}$ or $\hat{y}_{u_1} \neq \hat{y}_{u_2}$.

Our example is overly simplistic as it only contains a single user and a single item and two time periods; however, it can trivially be extended to any number of items, users or time periods. Thus, it is a significant problem for RSs that optimization with the naive or static IPS is not unbiased if both the user preferences and the selection bias are dynamic; it will lead to biased optimization. Selection bias and user preferences are practically never static in the real-world; in support of this claim, Sections 7 and 8 provide evidence that the dynamic nature of bias and preferences can be observed in the MovieLens dataset.

5 DANCER: DEBIASING RECOMMENDATIONS IN THE DYNAMIC SCENARIO

We introduce DANCER, a method for DeBiasing in the dyNamiC scE- naRio. We apply DANCER to time-aware matrix factorization (TMF), resulting in a novel rating prediction method that corrects for dynamic bias and models dynamic preferences. We introduce a propensity estimation method to estimate the probabilities of ratings being observed per time period.

5.1 Debiasing Recommendations

As discussed in Section 4, existing debiasing methods that use the naive or static IPS estimation are unable to debias in the dynamic scenario where selection bias and user preferences are both dynamic. As
a solution, we propose DANCER. With accurate propensities \( p_{u,i,t} \), dynamic selection bias can be fully corrected by applying DANCER to inversely weight the evaluation of the predicted ratings:

\[
L_{\text{DANCER}} = \frac{1}{|U| \cdot |I| \cdot |T|} \sum_{u,i,t} L(\hat{y}_{u,i,t} | y_{u,i,t}) p_{u,i,t}.
\]

(8)

Unlike the naive approach \( L_{\text{Naive}} \) (Eq. 3) and the static IPS approach with a static estimator \( L_{\text{StaticIPS}} \) (Eq. 4), the proposed debiasing method \( L_{\text{DANCER}} \) is unbiased in the dynamic scenario:

\[
E[L_{\text{DANCER}}] = \frac{1}{|U| \cdot |I| \cdot |T|} \sum_{u \in U} \sum_{i \in I} \sum_{t \in T} \frac{\mathbb{E}[p_{u,i,t}]}{p_{u,i,t}} E[\hat{y}_{u,i,t} | y_{u,i,t}] = 1.
\]

(9)

Because DANCER utilizes propensities that vary per time period \( t \), it can correct for dynamic effects of bias that the existing static IPS estimators cannot. For instance, in our example \( X \) with a user, an item and two time periods (see Section 4), the expected DANCER loss becomes:

\[
E[L_{\text{DANCER}}^X] = \frac{1}{2} \left( \frac{p_{t_1} L(\hat{y}_{u_1, i_1} | y_{u_1, i_1})}{p_{t_1}} + \frac{p_{t_2} L(\hat{y}_{u_2, i_2} | y_{u_2, i_2})}{p_{t_2}} \right) = L_{\text{DANCER}}^X,
\]

(10)

where we can see that \( L_{\text{DANCER}}^X \) is an unbiased estimation of the true loss \( L^X \). Combined with a time-aware recommendation method, DANCER is able to predict that the user ratings change over time.

5.2 A Debiased Time-Aware Recommendation

Because we expect both selection bias and user preferences to change over time in a dynamic scenario, the rating prediction that is optimized by DANCER should also be able to account for changes in user preferences. While DANCER is not model specific, we will apply it to a time-aware matrix factorization (TMF) [29] model that accounts for temporal effects. We refer to this combination of TMF and debiasing method as TMF-DANCER. Given an observed rating \( y_{u,i,t} \) from user \( u \) on item \( i \) at time \( t \), TMF computes the predicted rating \( \hat{y}_{u,i,t} \) as:

\[
\hat{y}_{u,i,t} = p_u q_i + b_u + b_i + b_t,
\]

where \( p_u \in \mathbb{R}^d \) and \( q_i \in \mathbb{R}^d \) are embedding vectors of user \( u \) and item \( i \), and \( b_u \in \mathbb{R}, b_i \in \mathbb{R}, \) and \( b_t \in \mathbb{R} \) are user, item and global offsets, respectively. Crucially, \( b_t \) is a time-dependent offset and models the impact of time in rating prediction. Under this model, the proposed TMF-DANCER is optimized by minimizing the following loss:

\[
\min_{P,Q,B} \sum_{u,i,t} \delta(\hat{y}_{u,i,t}, y_{u,i,t}) + \lambda \left( \|P\|^2_F + \|Q\|^2_F + \|B\|^2_F \right),
\]

(11)

where \( P, Q, B \) denote the embeddings of all users, all items and all the offset terms, respectively, \( \delta \) is the MSE loss function.

5.3 Propensity Estimation

DANCER requires accurate propensities \( p_{u,i,t} \) to remove the effect of dynamic selection bias. Because it is the first method to consider dynamic selection bias in RSs, it thus also needs a novel method to estimate \( p_{u,i,t} = P(\text{user} = u, \text{item} = i, \text{time} = t) \), i.e., the probability that the rating for user \( u \) and item \( i \) is observed at time \( t \). We propose to apply a Negative Log-Likelihood (NLL) loss to the propensity estimates \( \hat{p}_{u,i,t} \) and the observations made in a dataset (indicated by \( o_{u,i,t} \)):

\[
L_{\text{PE}} = \frac{1}{|U| \cdot |I| \cdot |T|} \sum_{u \in U} \sum_{i \in I} \sum_{t \in T} L_o(\hat{p}_{u,i,t} | o_{u,i,t}),
\]

(12)

where the function \( L_o \) is the NLL for each individual propensity:

\[
L_o(\hat{p}_{u,i,t} | o_{u,i,t}) = o_{u,i,t} \log \hat{p}_{u,i,t} + (1-o_{u,i,t}) \log (1-\hat{p}_{u,i,t}).
\]

Due to the large number of estimated propensities \( \hat{p}_{u,i,t} \), we argue that it is best to predict them with a model. Similar to the rating prediction task, TMF and TTF [53] are potential choices to model how the propensities vary over users, items and time periods. Alternatively, one can also make simplifying assumptions in the estimations of dynamic popularity bias. For instance, \( \hat{p}_{u,i,t} = \text{Pop}(i):= \frac{\sum t \in T \text{ratings received by item } i \text{ at time } t}{|T|} \) uses the ratio of ratings received by item \( i \) at time \( t \). The \( \text{Pop}(i) \) estimate is easy to compute, but it does assume that there are no differences between users when it comes to providing ratings.

Finally, we note that our proposed propensity estimation method Eq. 12 builds on existing methods for propensity estimation for static selection bias. Saito et al. [40] use MF instead of TMF or TTF. Similarly, the \( \text{Pop}(i):= \frac{\sum t \in T \text{ratings received by item } i \text{ at time } t}{|T|} \) is a common way to measure (static) popularity bias [7, 13, 58]. Our propensity estimation method makes these methods applicable to the dynamic scenario, and enables them to provide propensities for the DANCER debiasing method.

6 EXPERIMENTS

In our experiments, we focus on the age of an item (item-age) and the dynamic effect it has on selection bias and user preferences. From this point onwards, our notation will use \( t \) to denote how long an item has been available in the system, we will refer to this as the age of the item.

Because the distribution of ratings is very skewed towards young items, we divide the item-ages into seven bins whose edges are \([0,1,3,5,8,11,15,\infty] \) in years. For instance, a rating on an item when it is two-and-a-half years old will be assigned to \( t = 2 \), and a rating when it is 15 years old will be assigned \( t = 7 \). This can be interpreted as a specific choice for the time periods \( T \) and thus does not change any of the previously stated theory.

We first wish to investigate whether real-world selection bias and user preferences are affected by item-age and – are thus dynamic – and whether TMF-DANCER is more effective in a dynamic scenario than existing rating prediction methods that do not consider dynamic bias. Our experimental analysis is organized around three research questions (RQ1) Does item-age affect selection bias present in logged data? (RQ2) Does item-age affect real-world user preferences? (RQ3) Does the proposed TMF-DANCER method better mitigate the effect of bias in the dynamic scenario than existing debiasing methods designed for static selection bias? To answer these questions, we make use of three different tasks based around the MovieLens-Latest-small dataset [16]. The following sections will each introduce one of these tasks and answer the corresponding research question.

All tasks use embeddings with 32 dimensions, hyperparameter tuning is applied per method and task in the following ranges: learning rate \( \eta \in [10^{-5}, \ldots, 0.1] \) and \( L_2 \) regularization weights \( \lambda \in (0, 10^{-2}, 10^{-4}, \ldots, 1.0) \). Our implementation and hyperparameter choices are available at https://github.com/BetsyHJ/DANCER.

7 RQ1: IS SELECTION BIAS DYNAMIC?

To answer RQ1: Does item-age affect selection bias present in real-world logged data?, we will evaluate whether methods that consider item-age can better predict which items will be rated than methods that do not. If item-age has a large effect on selection bias, it should be an essential feature for predicting whether users will rate an item.
7.1 Experimental Setup for RQ1

The goal of our first task is to predict which ratings will be observed in real-world data, in other words, across users \( u \), items \( i \) and item-ages \( t \) the aim is to predict the observation \( o_{u,i,t} \) variables. With \( \tilde{p}_{u,i,t} \) as the predicted probability of observation, the metrics for this task are the NLL (Eq. (13)) and Perplexity (PPL):

\[
2^{-\frac{1}{|U|\cdot|I|\cdot|T|}} \sum_{u \in U} \sum_{i \in I} \sum_{t \in T} \log \tilde{p}_{u,i,t} \cdot \sum_{\sigma = 0}^{1-o_{u,i,t}} \log (1-\tilde{p}_{u,i,t}).
\]  

(14)

To evaluate whether item-age has a significant effect on the observation probabilities – and thus the dynamic selection bias in the data – we compare the performance of observation prediction methods that assume static bias with others that take item-age into account. Our comparison contains three baselines, one static method and four time-aware methods; when specifying the methods, we use \( \sigma \) to denote the sigmoid function, \( p_i \) for a learned user embedding, \( q_i \) for an item embedding, \( a_i \) for an embedding representing an item-age, and \( b_1 \) is a learned parameter that varies per item-age \( t \).

(1) **Constant**: The fraction of all ratings, this assumes no selection bias is present: \( \tilde{p}_{u,i,t} = \frac{\sum_{u \in U} \sum_{i \in I} \sum_{t \in T} \sigma (p_i^T q_i)}{|U| \cdot |I| \cdot |T|} \).

(2) **Static Item Popularity (Pop)**: The fraction of all ratings that have been given to the item; this assumes that selection bias is static over users and time: \( \tilde{p}_{u,i,t} = \frac{\sum_{u \in U} \sum_{t \in T} \sigma (p_i^T q_i)}{|U| \cdot |T|} \).

(3) **Time-aware Item Popularity (T-Pop)**: The item popularity per item-age; defined as the fraction of all ratings that have been given to item \( i \) of age \( t \): \( \tilde{p}_{u,i,t} = \frac{\sum_{u \in U} \sum_{t \in T} \sigma (p_i^T a_i)}{|U| \cdot |T|} \).

(4) **Static matrix factorization (MF)**: A standard MF model that assumes selection bias is static: \( \tilde{p}_{u,i,t} = \sigma (p_i^T q_i) + b_1 \).

(5) **Time-aware matrix factorization (TMF)** [29]: TMF captures the drift in popularity as items get older by adding an age-dependent bias term: \( \tilde{p}_{u,i,t} = \sigma (p_i^T q_i) + a_i + b_1 \).

(6) **Time-aware tensor factorization (TTF) [53]**: TTF extends MF by modelling the effect of item-age via element-wise multiplication: \( \tilde{p}_{u,i,t} = \sigma (p_i^T q_i \cdot a_i) \).

(7) **TTF++**: We propose a variation on TTF that models the effect via summation instead: \( \tilde{p}_{u,i,t} = \sigma (p_i^T (q_i + a_i)) \).

(8) **Time-aware matrix & tensor factorization (TMTF)**: Lastly, we propose a novel integration of TMF with TTF++: \( \tilde{p}_{u,i,t} = \sigma (p_i^T (q_i + a_i^T + b_1)) \).

All models are optimized with the NLL loss as described in Section 5.3.

We split the dataset into training, validation and test partitions following a ratio of 7:1:2. The MovieLens-Latest-small dataset [16] consists of 100,836 ratings applied to 9,742 movies by 610 users between 1996 and 2018. We apply two splitting strategies to the data: (1) a time-based split that per user places the latest 20% of their ratings into the test set [6]; and (2) a random split that uniformly samples 20% of ratings per user. The time-based split is more realistic but may be more ratings on younger items in the training set. Since most users have an active lifecycle of less than one year, the time-based split results in a ratio that is four times higher than the ratio in the test set; to account for this large difference in distributions we scale the predicted \( \tilde{p}_{u,i,t} \) by 0.25 in this setting. This leads to considerable performance improvements for all methods. Lastly, we ignore ratings outside of the user’s presence in the dataset, i.e., before their first rating or after their last; this prevents the methods from having to predict when users became active so that they can focus on the effect of item-age.

7.2 Results for RQ1

The results for the first task are presented in Table 1. Clearly, under both splitting strategies, the time-aware methods TMF, TTF++ and TMTF are significantly more accurate than Pop and MF, which assumes that selection bias is static, while MF outperforms Constant, which assumes no bias. Interestingly, T-Pop performs worst among all the methods, probably due to the high variance caused by sparsity. Under the random splitting strategy, TMF and TTF++ outperform TF, while TMF outperforms all other methods. Thus it appears that modelling item-age via a learned embedding better captures its effect than a single learned parameter, but moreover, TMF shows us that combining both results in the most accurate method. Under the time-based splitting strategy, TMF performs slightly better than TTF++ and TMTF, while TF performs worse than them. Also, Pop performs worse than Constant. A plausible reason for this inconsistency is the difference in distribution between the training and test set caused by the time-based split. The number of ratings per year displayed in Figure 2 displays this difference. This suggests that TMF is more robust to differences in distribution and that the other methods are somewhat overfitted on the training set. Nevertheless, most time-aware methods still predict the selection bias significantly better than the static MF.

We thus conclude that time-aware methods can better predict selection bias in real-world data than static methods. While the skewed rating distribution in Figure 1 already suggests that item-age has a large influence, our experimental results strongly show that item-age is an essential factor for accurately capturing the selection bias in users’ rating behavior. Consequently, we answer RQ1 affirmatively: item-age significantly affects the selection bias present in real-world data. This result strongly implies that the assumption of static bias in previous work is incorrect, at least in recommendation settings similar to that of the MovieLens dataset.
8 RQ2: ARE USER PREFERENCES DYNAMIC?
To answer RQ2: Does item-age affect real-world user preferences?, we compare rating prediction methods that assume preferences are static with ones that allow for dynamic preferences. If item-age has a significant effect, the latter group should perform better.

8.1 Experimental Setup for RQ2
The average rating per item-age in Figure 1 does not reveal a clear influence from the item-age on rating behavior. However, the averages should not be taken at face value because they are subject to selection bias. Users are generally more likely to rate movies they like (i.e., positivity bias [37]), thus it is possible that while the true average rating drops, the observed remains stable due to selection bias.

To find out whether item-age has a substantial effect, we compare methods that assume static preferences with others that allow for dynamic preferences in terms of the Mean Squared Error (MSE), Mean Absolute Error (MAE) and Accuracy (ACC) metrics. We train and evaluate in two settings: (1) in the observed setting the dataset is used without any corrections to mitigate selection bias; and (2) in the debiased setting self-normalized inverse propensity scoring (SNIPS) [46, 55] is applied during training and metric calculation to mitigate the effect of selection bias. The advantage of the debiased setting is that – in expectation – it bases evaluation on the true rating distribution; however, it has drawbacks: it requires accurate propensities and can be subject to increased variance. The observed setting will provide biased estimates but does not have these drawbacks. Our evaluation considers both settings so that their advantages can complement each other.

The comparison includes two baselines:
(1) **Static Average Item Rating (Avg)**: The average observed rating across all item-ages: $\bar{y}_{u,i,t} = \frac{\sum_{d'|s'|t'|i',t'\in U} y_{u,i',t'}}{\sum_{d'|s'|t'|i',t'\in U} 1}$.
(2) **Time-aware Average Item Rating (T-Avg)**: the average observed rating per item-age: $\bar{y}_{u,i,t} = \frac{\sum_{d'|s'|t'|i',t'\in U} y_{u,i',t'}}{\sum_{d'|s'|t'|i',t'\in U} 1}$. 

In addition, we also compare with the static MF and the time-aware TMF, TTF, TTF++ and TMTF. These methods are analogues to those used in Section 7; the main difference is that for this task the σ sigmoid function is not applied. Additionally, we add a global offset $b$, a user offset $b_u$, and an item offset $b_i$ to MF, TMF and TMTF. All methods are optimized to minimize MSE; in the debiased setting optimization is performed with DANCER following Section 5. We use the propensity values estimated for the previous observation prediction task by TMTF under the random-split (see Section 7.1).

The dataset is again partitioned into a training, validation and test set according to a ratio of 7:1:2. Unlike for the previous task (Section 7.1), the data for this task only consists of observed ratings, and furthermore, the partitioning is only made via uniform random sampling. As displayed in Figure 2, we find that a time-based split leads to extremely different rating distributions. This makes it infeasible to obtain convincing conclusions from the results of this task. Nevertheless, because a random split is perfectly suitable for evaluating a possible relationship between user preferences and item-age, our results are completely appropriate to answer RQ2.

8.2 Results for RQ2
Table 2 displays the evaluation results for the second task; in both settings the time-aware methods outperform the static MF. There is a significant time window effect on rating behavior. However, the publicly available datasets that meet this criterion – YAHOO!R3 [32] and COAT SHOPPING [41] – lack any form of temporal information. As a result, we cannot apply DANCER or any other method designed for static selection bias.

9 RQ3: CAN TMF-DANCER BETTER MITIGATE DYNAMIC SELECTION BIAS?
Section 4 showed that the static IPS-based debiasing method is biased in a dynamic scenario. Subsequently, in Section 7 and 8 we discovered that selection bias and user preferences in the MovieLens dataset are indeed dynamic. Therefore, we can already conclude that theoretically TMF-DANCER is the first method that is potentially unbiased for the dynamic scenario. Our final research question considers whether this theoretical advantage translates into improved recommendation performance: RQ3: Does the proposed TMF-DANCER method better mitigate the effect of bias in the dynamic scenario than existing debiasing methods designed for static selection bias?

9.1 Experimental Setup for RQ3
The most common technique for evaluating debiasing methods for recommendation, without actual deployment to real-world users, makes use of unbiased test sets [41, 49]. This requires a dataset that has a training set consisting of biased logged ratings and a test set of user ratings on uniformly randomly selected items. Such a test set can be created by randomly sampling items and asking users to provide a rating for them, thus avoiding the selection bias that usually heavily affects what items are rated. However, the publicly available datasets that meet this criterion – YAHOO!R3 [32] and COAT SHOPPING [41] – lack any form of temporal information. As a result, we cannot apply DANCER or any other form of dynamic debiasing to them.
As an alternative to using real-world datasets, we utilize a semi-synthetic simulation based on a real-world dataset for our evaluation. This simulation first estimates a simulated Ground Truth (sim-GT) based on the actual dataset, and then generates a new biased training set from this sim-GT. Debiasing methods can be applied to the generated training set and evaluated on the sim-GT, since in this setting, the debiased estimates should match the sim-GT as close as possible. The creation of our semi-synthetic simulation has three steps:

1. First, we estimate the complete rating matrix using the TMF method, which simply uses an age-dependent bias term to model the dynamics of user preferences, thus making the simulation understandable and not prone to overfitting. This provides us with an estimated rating for each item, user and item-age combination which we will treat as the sim-GT. By optimizing TMF with the real user ratings in the debiased setting, we hope the sim-GT reflects the real-world scenario as closely as possible.

2. Second, dynamic selection bias is simulated using MF to model the interactions between items and item-ages. Following Section 7.1, we fit the following model: \( p_{u,i,t} = \sigma(q_i^T a_i) \), to predict if the ratings are observed in the MovieLens dataset. To mimic real-world dynamic popularity bias more closely, we follow the user presence of the original dataset: propensities are zero before a user’s first rating and after their last rating in the dataset, we also normalize the predicted probabilities so that their mean is 4%, the same value as the dataset has.

3. Third, to prevent overlap between the training and test set, we utilize both random and time-based splitting: per user, 50% of items are randomly selected for the test set, and a split timestep is chosen at 80% of the user presence. The test set consists of all sim-GT ratings on the randomly selected items at the last presence of each user; as a result, the test set reflects future preferences on previously unseen items. The training set uses the other 50% of items per user and samples from the ratings before the split timestamp following the estimated propensities \( p_{u,i,t} \) from the previous step. The result is a training set where due to dynamic selection bias only ~2% of the \( u, i, t \) ratings are observed.

Figure 3 compares the original MovieLens dataset with our semi-synthetic simulation. The popularity of items, in terms of how many ratings they receive, is closely approximated by the simulated training set and MovieLens: the simulated training set rates older items lower than MovieLens. It seems likely that this is the result of positivity bias, which is not part of our simulation.

As an alternative to using real-world datasets, we utilize a semi-synthetic simulation based on a real-world dataset for our evaluation. This simulation first estimates a simulated Ground Truth (sim-GT) based on the actual dataset, and then generates a new biased training set from this sim-GT. Debiasing methods can be applied to the generated training set and evaluated on the sim-GT, since in this setting, the debiased estimates should match the sim-GT as close as possible. The creation of our semi-synthetic simulation has three steps:

1. First, we estimate the complete rating matrix using the TMF method, which simply uses an age-dependent bias term to model the dynamics of user preferences, thus making the simulation understandable and not prone to overfitting. This provides us with an estimated rating for each item, user and item-age combination which we will treat as the sim-GT. By optimizing TMF with the real user ratings in the debiased setting, we hope the sim-GT reflects the real-world scenario as closely as possible.

2. Second, dynamic selection bias is simulated using MF to model the interactions between items and item-ages. Following Section 7.1, we fit the following model: \( p_{u,i,t} = \sigma(q_i^T a_i) \), to predict if the ratings are observed in the MovieLens dataset. To mimic real-world dynamic popularity bias more closely, we follow the user presence of the original dataset: propensities are zero before a user’s first rating and after their last rating in the dataset, we also normalize the predicted probabilities so that their mean is 4%, the same value as the dataset has.

3. Third, to prevent overlap between the training and test set, we utilize both random and time-based splitting: per user, 50% of items are randomly selected for the test set, and a split timestep is chosen at 80% of the user presence. The test set consists of all sim-GT ratings on the randomly selected items at the last presence of each user; as a result, the test set reflects future preferences on previously unseen items. The training set uses the other 50% of items per user and samples from the ratings before the split timestamp following the estimated propensities \( p_{u,i,t} \) from the previous step. The result is a training set where due to dynamic selection bias only ~2% of the \( u, i, t \) ratings are observed.

Figure 3 compares the original MovieLens dataset with our semi-synthetic simulation. The popularity of items, in terms of how many ratings they receive, is closely approximated by the simulated training set and MovieLens: the simulated training set rates older items lower than MovieLens. It seems likely that this is the result of positivity bias, which is not part of our simulation.
Table 3: RQ3 – Performance of TMF-DANCER compared with different methods. † indicates that the improvement of TMF-DANCER over all the baselines is significant at the level of 0.01.

<table>
<thead>
<tr>
<th>Method</th>
<th>MSE↓</th>
<th>MAE↓</th>
<th>ACC†</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>0.3155</td>
<td>0.4321</td>
<td>0.3623</td>
</tr>
<tr>
<td>T-Avg</td>
<td>0.3280</td>
<td>0.4326</td>
<td>0.3614</td>
</tr>
<tr>
<td>MF</td>
<td>0.1811 (0.0010)</td>
<td>0.3314 (0.0028)</td>
<td>0.4680 (0.0040)</td>
</tr>
<tr>
<td>TMF</td>
<td>0.1338 (0.0019)</td>
<td>0.2818 (0.0022)</td>
<td>0.5396 (0.0038)</td>
</tr>
<tr>
<td>MF-StaticIPS</td>
<td>0.1879 (0.0035)</td>
<td>0.3377 (0.0032)</td>
<td>0.4598 (0.0044)</td>
</tr>
<tr>
<td>TMF-StaticIPS</td>
<td>0.1086 (0.0021)</td>
<td>0.2491 (0.0027)</td>
<td>0.6065 (0.0057)</td>
</tr>
<tr>
<td>MF-DANCER</td>
<td>0.1533 (0.0016)</td>
<td>0.3032 (0.0017)</td>
<td>0.5074 (0.0023)</td>
</tr>
<tr>
<td>TMF-DANCER</td>
<td>0.1045† (0.0014)</td>
<td>0.2444† (0.0018)</td>
<td>0.6151† (0.0019)</td>
</tr>
</tbody>
</table>

Table 4: RQ3 – Performance of TMF-DANCER with estimated propensities and the (simulated) ground truth propensities.

<table>
<thead>
<tr>
<th>Method</th>
<th>MSE↓</th>
<th>MAE↓</th>
<th>ACC†</th>
</tr>
</thead>
<tbody>
<tr>
<td>TG-Pop</td>
<td>0.1182 (0.0012)</td>
<td>0.2644 (0.0016)</td>
<td>0.5677 (0.0032)</td>
</tr>
<tr>
<td>T-Pop</td>
<td>0.1041 (0.0015)</td>
<td>0.2448 (0.0022)</td>
<td>0.6115 (0.0055)</td>
</tr>
<tr>
<td>Ground Truth</td>
<td>0.1045 (0.0014)</td>
<td>0.2444 (0.0018)</td>
<td>0.6151 (0.0039)</td>
</tr>
</tbody>
</table>

performs well in the scenario that it assumes, the differences with other methods are considerable and statistically significant.

In addition, Table 4 displays the performance of TMF-DANCER using different propensities. We see that with estimated propensities the performance of TMF-DANCER is comparable to when it is using the actual sim-GT propensities. Moreover, TMF-DANCER outperforms the most baselines, except TMF-StaticIPS, even when using simple time-aware propensity estimation.

To better understand the improvements of TMF-DANCER, Figure 4 shows the average predicted rating from different methods across item-ages and the actual average rating. The MF methods are unable to model changes in ratings as items get older; the differences in the average ratings are purely caused by different item distributions: i.e., items that become available later in the dataset will never achieve the oldest item-ages. The TMF methods better capture the overall trend. TMF without debiasing consistently overestimates ratings; TMF-staticIPS reduces overestimation by correcting for static bias; the overestimation becomes worse for older items in both models. Instead, TMF-DANCER approximates the actual average rating at each item-age; its accuracy is quite consistent over time.

Lastly, to get more insights into the behavior of TMF-DANCER, Figure 5 shows the propensities and (predicted) ratings per item-age and averaged across users for two handpicked movies. We observe that TMF-DANCER outperforms TMF, especially when the popularity of items decreases as items get older.

Finally, we can answer RQ3 in the affirmative: the TMF-DANCER method better mitigates the effect of bias in a dynamic scenario than existing debiasing methods designed for static selection bias. This conclusion still holds when propensities are estimated, and the accuracy of TMF-DANCER is consistent across item-ages.

10 CONCLUSION

In this paper, we considered the dynamic scenario in recommendation where selection bias and user preferences change over time. Our experimental results revealed that in the real-world MovieLens dataset: (1) selection bias changes as items get older, and (2) user preferences are also affected by the age of items. Therefore, it appears that the dynamic scenario better captures the real-world situation, and thus, poses a serious problem that existing static IPS-based method cannot correct for dynamic bias in dynamic scenarios. As a solution, we proposed the DANCER debiasing method that takes into account the dynamic aspects of bias and user preferences, the first method that is unbiased in the dynamic scenario. The results on a semi-synthetic simulation based on the MovieLens dataset showed that TMF-DANCER provides significant gains in performance that are consistent across item-ages and robust to misspecified propensities. Our findings about the dynamic scenario have implications for state-of-the-art recommendation methods, as they are strongly affected by dynamic selection bias. With the DANCER debiasing method, RSs can now be expanded to deal with dynamic scenarios.

A limitation of our work is that we only considered the rating prediction task and the effect of item-age on bias and preferences; future work should consider the ranking task and look at other aspects of time, e.g., seasonal effects, weekday, time of day, etc.

ACKNOWLEDGMENTS

This research was partially supported by the Netherlands Organisation for Scientific Research (NWO) under project number 024.004.022 and the Google Research Scholar Program. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.
Debiasing Recommendations When Selection Bias and User Preferences Are Dynamic

REFERENCES


