UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

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

Huang, J.; Oosterhuis, H.; de Rijke, M.

DOI
10.1145/3488560.3498375

Publication date
2022

Document Version
Final published version

Published in
WSDM '22

License
CC BY

Link to publication

Citation for published version (APA):
Huang, J., Oosterhuis, H., & de Rijke, M. (2022). It Is Different When Items Are Older: Debiasing Recommendations When Selection Bias and User Preferences Are Dynamic. In *WSDM '22: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining : February 21-25, 2022 : virtual event, Tempe, AZ, USA* (pp. 381–389). Association for Computing Machinery. https://doi.org/10.1145/3488560.3498375

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:07 Oct 2026

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

Jin Huang
University of Amsterdam
Amsterdam, The Netherlands
j.huang2@uva.nl

Harrie Oosterhuis
Radboud University
Nijmegen, The Netherlands
harrie.oosterhuis@ru.nl

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

ABSTRACT

User interactions with recommender systems (RSs) are affected by user selection bias, *e.g.,* users are more likely to rate popular items (popularity bias) or items that they expect to enjoy beforehand (positivity bias). Methods exist for mitigating the effects of selection bias in user ratings on the evaluation and optimization of RSs. However, these methods treat selection bias as static, despite the fact that the popularity of an item may change drastically over time and the fact that user preferences may also change over time.

We focus on the age of an item and its effect on selection bias and user preferences. Our experimental analysis reveals that the rating behavior of users on the MovieLens dataset is better captured by methods that consider effects from the age of item on bias and preferences. We theoretically show that in a dynamic scenario in which both the selection bias and user preferences are dynamic, existing debiasing methods are no longer unbiased. To address this limitation, we introduce DebiAsing in the dyNamiC scEnaRio (DANCER), a novel debiasing method that extends the inverse propensity scoring debiasing method to account for dynamic selection bias and user preferences. Our experimental results indicate that DANCER improves rating prediction performance compared to debiasing methods that incorrectly assume that selection bias is static in a dynamic scenario. To the best of our knowledge, DANCER is the first debiasing method that accounts for dynamic selection bias and user preferences in RSs.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Evaluation of retrieval results.*

KEYWORDS

Selection bias; Dynamic user preferences

ACM Reference Format:
Jin Huang, Harrie Oosterhuis, and Maarten de Rijke. 2022. It Is Different When Items Are Older: Debiasing Recommendations When Selection Bias and User Preferences Are Dynamic. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498375

This work is licensed under a Creative Commons Attribution International 4.0 License.

WSDM '22, February 21–25, 2022, Tempe, AZ, USA.
© 2022 Copyright is held by the owner/author(s).
ACM ISBN 978-1-4503-9132-0/22/02.
https://doi.org/10.1145/3488560.3498375

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).[1] 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

[1] https://grouplens.org/datasets/movielens/latest/

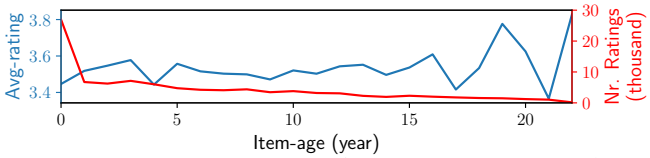

Figure 1: The number of ratings (indicative of popularity) and the average (observed) rating of items for different item-ages on the MovieLens-Latest-small dataset.

unbiased in dynamic scenarios. Second, existing methods [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 [15, 30]. Some recent work has used deep neural networks to improve CF, *e.g.,* by using multi-layer perceptrons [12, 19], convolutional neural networks [18], or graph neural networks [17, 48]. While they significantly improve recommendation accuracy [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 [38]. Various deep learning methods have been applied to this task [38, 59] such as recurrent neural networks [21, 51, 57], graph neural networks [52, 54], and networks with attention [11, 23, 45].

We focus on time-aware recommendation methods [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 (TMF) [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 [29] lists and compares various variants of TMF, in how well they can capture item-related or user-related temporal effects. Xiong et al. [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 [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) [20, 32, 41]. Two typical forms of bias in RSs are known as *popularity bias* and *positivity bias*. Popularity bias is characterized by a long tail distribution over the number of interactions per item in logged data because users are more likely to interact with more popular items [37, 43]. Positivity bias leads to an over-representation of positive feedback because users rate the items they like more often [37]. The effect of these biases is generally dynamic: they can change drastically over time [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 [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 [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 [49].

While the impact of dynamic bias has previously been pointed out [26, 58], no prior debiasing method considers a scenario in which both selection bias and user preferences change over time. All existing debiased 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.

3 PROBLEM DEFINITION

We follow the common RS setting where items from the set $\mathcal{I} = \{i_1,...,i_M\}$ are recommended to users from the set $\mathcal{U} = \{u_1,...,u_N\}$ [44]. Users have preferences towards items, generally modelled by a label $y_{u,i,t}$ (*e.g.,* a rating $y_{u,i,t} \in \{1,2,3,4,5\}$) per user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$. Similar to time-aware recommendations [6, 29, 53], we also consider the effect of time on user preferences: let $\mathcal{T} = \{t_1,...,t_T\}$ be a set of T time periods; we allow the user preference $y_{u,i,t}$ to vary over different periods $t \in \mathcal{T}$. Our goal is to optimize an RS that best captures the user preferences across all items i and time periods t. We formulate this goal as a loss function: let $\hat{y}_{u,i,t}$ be a predicted rating by the RS and $L(\hat{y},y)$ a comparison function between the predicted rating and actual rating. Then our loss is:

$$\mathcal{L} = \frac{1}{|\mathcal{U}|\cdot|\mathcal{I}|\cdot|\mathcal{T}|} \sum_{u\in\mathcal{U}} \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}} L(\hat{y}_{u,i,t}, y_{u,i,t}). \tag{1}$$

The function L can be chosen according to common RS metrics, for example, the prevalent Mean Squared Error (MSE) metric:

$$L(\hat{y}_{u,i,t}, y_{u,i,t}) = (\hat{y}_{u,i,t} - y_{u,i,t})^2. \tag{2}$$

The choice for RSs to perform well across all time periods t in $\mathcal{T}$ is partially made for practical reasons; arguably, at any particular time one only needs RSs to perform well for the present and future [25]. However, in practice, data is only available about past user preferences, thus making optimization w.r.t. future preferences infeasible. Moreover, we expect that if RSs' performance generalizes well across the

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 $\mathbf{O} \in \{0,1\}^{|\mathcal{U}|\cdot|\mathcal{I}|\cdot|\mathcal{T}|}$ that indicates what ratings are recorded in the logged interaction data and during which time period. We use $o_{u,i,t} \in \mathbf{O}$ 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 $\mathbf{O}$ 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 more often [3, 14]. Well-known prevalent biases in RS data include: (1) popularity bias [37, 43] – often a small group of popular items receive most interactions; and (2) positivity bias [37] – users are usually more likely to rate items they prefer. We model selection bias using the probability of a rating being recorded: $p_{u,i,t}=P(o_{u,i,t}=1)$, which we also refer to as the *observation probability* or *propensity*. Again, we deviate from the common existing method by explicitly allowing $p_{u,i,t}$ to vary over different time periods t. This enables our method to not only model a bias such as popularity bias but also how that bias changes as items get older and decline in popularity.

4 ESTIMATION IGNORING DYNAMIC BIAS

Before we introduce our recommendation method for dealing with the dynamic scenario in which both selection bias and user preferences are dynamic, we will show that, in a dynamic scenario, the existing recommendation methods that either assume no bias or static bias are not unbiased. The standard estimation of how well the predicted user preferences reflect the true user preferences shown in Eq. 1 is the full-information loss (*i.e.,* the loss based on all the ratings), which is impractical since user preferences are only partially known in the logged data. The naive loss ignores the effect of selection bias completely and thus assumes that the observed data represents the true user preferences unbiasedly. Under this assumption, the naive loss can be estimated by a simple average on the observed ratings:

$$\mathcal{L}_{\text{Naive}} = \frac{1}{|\{u,i,t : o_{u,i,t}=1\}|} \sum_{u,i,t:o_{u,i,t}=1} L(\hat{y}_{u,i,t}, y_{u,i,t}). \tag{3}$$

And the widely-used debiasing method uses IPS estimation [24, 31] to correct for the probability that a user rates an item [41]. It uses static propensities $p_{u,i}$ that are the probability of observing a rating for item i by user u in any of the time-periods [32, 39]. These propensities ignore the dynamic aspect of selection bias, *i.e.,* that these probabilities can vary per time period t, resulting in the *static* IPS estimator:

$$\mathcal{L}_{\text{staticIPS}} = \frac{1}{|\mathcal{U}|\cdot|\mathcal{I}|\cdot|\mathcal{T}|} \sum_{u,i,t:o_{u,i,t}=1} \frac{L(\hat{y}_{u,i,t}, y_{u,i,t})}{p_{u,i}}. \tag{4}$$

Now that we have described the naive and *static* IPS-based loss functions for recommendation (that assume no bias and only static bias, respectively), we can consider the effect of dynamic selection bias.

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 $\mathcal{X}$ with one user u, one item i and two time periods t_1 and t_2. Let y_{t_1} and y_{t_2} be the user ratings on the item at t_1 and t_2 respectively; p_{t_1} and p_{t_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: $y_{t_1} \neq y_{t_2}$, $p_{t_1} \neq p_{t_2}$. Remember that in this example the loss we wish to estimate is:

$$\mathcal{L}^{\mathcal{X}} = \frac{1}{2}\big(L(\hat{y}_{t_1}, y_{t_1}) + L(\hat{y}_{t_2}, y_{t_2})\big). \tag{5}$$

The expected naive loss over the observation variables becomes:

$$\begin{aligned}&\mathbb{E}\Big[\mathcal{L}^{\mathcal{X}}_{\text{Naive}}\Big] \\ &= p_{t_1} L(\hat{y}_{t_1}, y_{t_1}) + p_{t_2} L(\hat{y}_{t_2}, y_{t_2}) - \frac{p_{t_1} p_{t_2}}{2}\big(L(\hat{y}_{t_1}, y_{t_1}) + L(\hat{y}_{t_2}, y_{t_2})\big).\end{aligned} \tag{6}$$

Clearly, it is not proportional to the true loss $\mathcal{L}^{\mathcal{X}}$ when selection bias and user preferences are dynamic: if $y_{t_1} \neq y_{t_2}$ and $p_{t_1} \neq p_{t_2}$, then $\mathbb{E}\big[\mathcal{L}^{\mathcal{X}}_{\text{Naive}}\big] \propto\!\!\!\!\!\!\not\;\; \mathcal{L}^{\mathcal{X}}$. 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 $p_{u,i} = p_{t_1} + (1-p_{t_1})p_{t_2}$ that is the probability of observing a rating at time t_1 or t_2. If we consider the expected value of this estimator:

$$\mathbb{E}\Big[\mathcal{L}^{\mathcal{X}}_{\text{staticIPS}}\Big] = \frac{1}{2}\left(\frac{p_{t_1}}{p_{u,i}} L(\hat{y}_{t_1}, y_{t_1}) + \frac{p_{t_2}}{p_{u,i}} L(\hat{y}_{t_2}, y_{t_2})\right), \tag{7}$$

we see that it is not proportional to the true loss in the dynamic scenario: if $y_{t_1} \neq y_{t_2}$ and $p_{t_1} \neq p_{t_2}$, then $\mathbb{E}\big[\mathcal{L}^{\mathcal{X}}_{\text{staticIPS}}\big] \propto\!\!\!\!\!\!\not\;\; \mathcal{L}^{\mathcal{X}}$, 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 $\hat{y}_{t_1} = \hat{y}_{t_2}$ or $\hat{y}_{t_1} \neq \hat{y}_{t_2}$.

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 scEnaRio. 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:

$$\mathcal{L}_{\text{DANCER}} = \frac{1}{|\mathcal{U}|\cdot|\mathcal{I}|\cdot|\mathcal{T}|}\sum_{u,i,t:o_{u,i,t}=1}\frac{L(\hat{y}_{u,i,t},y_{u,i,t})}{p_{u,i,t}}. \tag{8}$$

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

$$\begin{aligned}\mathbb{E}[\mathcal{L}_{\text{DANCER}}] &= \frac{1}{|\mathcal{U}|\cdot|\mathcal{I}|\cdot|\mathcal{T}|}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\frac{\mathbb{E}\left[o_{u,i,t}\right]}{p_{u,i,t}}\cdot L(\hat{y}_{u,i,t},y_{u,i,t})\\ &= \frac{1}{|\mathcal{U}|\cdot|\mathcal{I}|\cdot|\mathcal{T}|}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}L(\hat{y}_{u,i,t},y_{u,i,t})\propto\mathcal{L}.\end{aligned} \tag{9}$$

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 $\mathcal{X}$ with a user, an item and two time periods (see Section 4), the expected DANCER loss becomes:

$$\mathbb{E}\left[\mathcal{L}^{\mathcal{X}}_{\text{DANCER}}\right] = \frac{1}{2}\left(p_{t_1}\frac{L(\hat{y}_{t_1},y_{t_1})}{p_{t_1}}+p_{t_2}\frac{L(\hat{y}_{t_2},y_{t_2})}{p_{t_2}}\right)=\mathcal{L}^{\mathcal{X}}, \tag{10}$$

where we can see that $\mathcal{L}^{\mathcal{X}}_{\text{DANCER}}$ is an unbiased estimation of the true loss $\mathcal{L}^{\mathcal{X}}$. 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: $\hat{y}_{u,i,t} = \boldsymbol{p}_u^T\boldsymbol{q}_i + b_u + b_i + b + b_t$, where the $\boldsymbol{p}_u \in \mathbb{R}^d$ and $\boldsymbol{q}_i \in \mathbb{R}^d$ are embedding vectors of user u and item i, and $b_u \in \mathbb{R}$, $b_i \in \mathbb{R}$, and $b \in \mathbb{R}$ 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:

$$\arg\min_{\boldsymbol{P},\boldsymbol{Q},\boldsymbol{B}}\left[\sum_{u,i,t:o_{u,i,t}=1}\frac{\delta(\hat{y}_{u,i,t},y_{u,i,t})}{p_{u,i,t}}+\lambda\left(\|\boldsymbol{P}\|_F^2+\|\boldsymbol{Q}\|_F^2+\|\boldsymbol{B}\|_F^2\right)\right], \tag{11}$$

where $\boldsymbol{P}$, $\boldsymbol{Q}$ and $\boldsymbol{B}$ denote the embeddings of all users, all items and all the offset terms, respectively; δ 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 $p_{u,i,t} = P(o_{u,i,t}=1)$, *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 $o_{u,i,t}$):

$$\mathcal{L}_{\text{PE}} = \frac{1}{|\mathcal{U}|\cdot|\mathcal{I}|\cdot|\mathcal{T}|}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}L_o(\hat{p}_{u,i,t},o_{u,i,t}), \tag{12}$$

where the function L_o is the NLL for each individual propensity:

$$L_o(\hat{p}_{u,i,t},o_{u,i,t}) = o_{u,i,t}\cdot\log\hat{p}_{u,i,t}+(1-o_{u,i,t})\cdot\log(1-\hat{p}_{u,i,t}). \tag{13}$$

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, $\hat{p}_{u,i,t} = \text{Pop}(i,t) := \frac{\sum_{u'\in\mathcal{U}}o_{u',i,t}}{|\mathcal{U}|}$ uses the ratio of ratings received by item i at time t. The $\text{Pop}(i,t)$ 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 $\text{Pop}(i) := \frac{\sum_{u'\in\mathcal{U}}\sum_{t'\in\mathcal{T}}o_{u',i,t'}}{|\mathcal{U}|\cdot|\mathcal{T}|}$ 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 $\mathcal{T}$ 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 $\lambda \in \{0, 10^{-7}, 10^{-6}, \ldots, 1.0\}$. 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 $\hat{p}_{u,i,t}$ as the predicted probability of observation, the metrics for this task are the NLL (Eq. 13) and Perplexity (PPL):

$$2^{-\frac{1}{|\mathcal{U}|\cdot|\mathcal{I}|\cdot|\mathcal{T}|}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}} o_{u,i,t}\cdot\log_2\hat{p}_{u,i,t}+(1-o_{u,i,t})\cdot\log_2(1-\hat{p}_{u,i,t})}. \quad (14)$$

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 σ to denote the sigmoid function, $\boldsymbol{p}_u$ for a learned user embedding, $\boldsymbol{q}_i$ for an item embedding, $\boldsymbol{a}_t$ for an embedding representing an item-age, and b_t is a learned parameter that varies per item-age t.

(1) **Constant**: The fraction of all ratings, this assumes no selection bias is present: $\hat{p}_{u,i,t} = \frac{\sum_{u'\in\mathcal{U}}\sum_{i'\in\mathcal{I}}\sum_{t'\in\mathcal{T}} o_{u',i',t'}}{|\mathcal{U}|\cdot|\mathcal{I}|\cdot|\mathcal{T}|}$.
(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: $\hat{p}_{u,i,t} = \frac{\sum_{u'\in\mathcal{U}}\sum_{t'\in\mathcal{T}} o_{u',i,t'}}{|\mathcal{U}|\cdot|\mathcal{T}|}$.
(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: $\hat{p}_{u,i,t} = \frac{\sum_{u'\in\mathcal{U}} o_{u',i,t}}{|\mathcal{U}|}$.
(4) **Static matrix factorization (MF)**: A standard MF model that assumes selection bias is static: $\hat{p}_{u,i,t} = \sigma(\boldsymbol{p}_u^T\boldsymbol{q}_i)$.
(5) **Time-aware matrix factorization (TMF)** [29]: TMF captures the drift in popularity as items get older by adding an age-dependent bias term: $\hat{p}_{u,i,t} = \sigma(\boldsymbol{p}_u^T\boldsymbol{q}_i + b_t)$.
(6) **Time-aware tensor factorization (TTF)** [53]: TTF extends MF by modelling the effect of item-age via element-wise multiplication: $\hat{p}_{u,i,t} = \sigma(\boldsymbol{p}_u^T(\boldsymbol{q}_i \times \boldsymbol{a}_t))$.
(7) **TTF++**: We propose a variation on TTF that models the effect via summation instead: $\hat{p}_{u,i,t} = \sigma(\boldsymbol{p}_u^T(\boldsymbol{q}_i + \boldsymbol{a}_t))$.
(8) **Time-aware matrix & tensor factorization (TMTF)**: Lastly, we propose a novel integration of TMF with TTF++: $\hat{p}_{u,i,t} = \sigma(\boldsymbol{p}_u^T(\boldsymbol{q}_i + \boldsymbol{a}_t) + b_t)$.

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 makes the training and test data follow different distributions: *i.e.,* there will be more ratings on younger items in the training set than in the test set. Alternatively, the random split ensures both partitions follow the same distribution but is less realistic: *i.e.,* ratings in the test set may have taken place before ratings in the training set. For both settings, the training and validation set are uniformly randomly sampled from the data outside the test set. Since most users have an active lifecycle of less than one year, the time-based split results in a ratio between observed and missing ratings that is four times higher than the ratio in the test set; to account for this large difference in distributions we scale the predicted $\hat{p}_{u,i,t}$ 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.

Table 1: RQ1 – Performance in observation prediction. Results are averages of 10 independent runs, the standard deviations are shown in brackets. † indicates a significant improvement over MF ($p < 0.01$) according to the paired-samples t-test.

Method	Random				Time-Based			
	NLL		PPL		NLL		PPL	
Constant	0.0973		1.1022		0.0337		1.0343	
Pop	0.0890		1.0931		0.0404		1.0412	
MF	0.0697	(0.0015)	1.0722	(0.0016)	0.0271	(0.0000)	1.0275	(0.0000)
T-Pop	0.1234		1.1314		0.0523		1.0537	
TMF	0.0658†	(0.0001)	1.0680†	(0.0001)	**0.0267**†	(0.0000)	**1.0271**†	(0.0000)
TTF	0.0637†	(0.0002)	1.0657†	(0.0003)	0.0273	(0.0004)	1.0277	(0.0004)
TTF++	0.0632†	(0.0002)	1.0653†	(0.0002)	0.0268†	(0.0001)	1.0271†	(0.0001)
TMTF	**0.0621**†	(0.0001)	**1.0641**†	(0.0001)	0.0268†	(0.0000)	1.0272†	(0.0000)

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 assume 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, TTF and TTF++ outperform TMF, while TMTF 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, TMTF 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 TTF 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: $\hat{y}_{u,i,t} = \frac{\sum_{u',i,t':o_{u',i,t'}=1} y_{u',i,t'}}{\sum_{u' \in \mathcal{U}} \sum_{t' \in \mathcal{T}} o_{u',i,t'}}$.

(2) **Time-aware Average Item Rating (T-Avg)**: the average observed rating per item-age: $\hat{y}_{u,i,t} = \frac{\sum_{u',i,t:o_{u',i,t}=1} y_{u',i,t}}{\sum_{u' \in \mathcal{U}} o_{u',i,t}}$.

In addition, we also compare with the static MF and the time-aware TMF, TTF, TTF++ and TMTF. These methods are analogous 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 single exception: TTF performs worst in both settings, probably due to over-fitting. The differences between the other time-aware methods and static MF are larger in the debiased setting than in the observed setting. This suggests that selection bias in the data reduces the dynamic effect of item-age on the observed ratings. We speculate that the effect of positivity bias could increase with item-age: users are less likely to try and rate movies that are older unless they already expect to enjoy them. Due to sparsity, T-Avg performs worse than Avg in both settings. Interestingly, Avg performs even better than MF in the debiased setting; this confirms prior observations that Avg is more robust in highly biased scenarios [7]. Regardless, in both settings most time-aware methods significantly outperform MF and the two baselines, and therefore, we answer **RQ2** in the affirmative: item-age has a significant effect on user preferences.

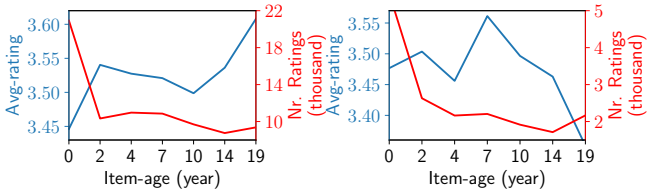

Figure 2: Average rating and number of ratings over item-age in the time-based partitioned training (left) and test set (right).

Our conclusions for RQ1 and RQ2 indicate that the dynamic scenario, where selection bias and user preferences change over time, better captures real-world logged data, than a static view. Moreover, Section 4 showed that the existing *static* IPS approach cannot debias in this scenario. Consequently, our answers to RQ1 and RQ2 reveal a real need for a method that can deal with the dynamic scenario.

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.[2] As a result, we cannot apply DANCER or any other form of dynamic debiasing to them.

[2] A recent music dataset [5] contains randomized observations and temporal information, but it only tracks user behavior during short sessions rather than for extended periods of time.

Table 2: RQ2 – Performance comparison of different methods in predicting ratings logged in MovieLens-Latest-small. † indicates that the improvement of the models over MF is significant ($p < 0.01$). ↑/↓ indicates whether larger or smaller values are better.

Method	Observed						Debiased					
	MSE ↓		MAE ↓		ACC↑		SNIPS-MSE↓		SNIPS-MAE↓		SNIPS-ACC↑	
Avg	0.9535		0.7540		0.2241		1.1436		0.8360		0.2048	
MF	0.7551	(0.0046)	0.6679	(0.0021)	0.2515	(0.0016)	1.2911	(0.0242)	0.8985	(0.0095)	0.1829	(0.0065)
T-Avg	1.0850		0.7974		0.2181		1.3105		0.8865		0.1955	
TMF	0.7505	(0.0058)	0.6656	(0.0026)	0.2525	(0.0014)	1.1210†	(0.0464)	0.8383†	(0.0173)	0.1944†	(0.0067)
TTF	1.1515	(0.0542)	0.8187	(0.0181)	0.2120	(0.0054)	1.8834	(0.1247)	1.0879	(0.0388)	0.1504	(0.0058)
TTF++	0.7526	(0.0011)	0.6645†	(0.0006)	**0.2552**†	(0.0007)	1.0839†	(0.0159)	0.8067†	(0.0067)	**0.2134**†	(0.0059)
TMTF	**0.7503**†	(0.0014)	**0.6637**†	(0.0008)	0.2533†	(0.0009)	**1.0727**†	(0.0173)	**0.8026**†	(0.0047)	0.2127†	(0.0060)

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, we fit the following model: $p_{u,i,t} = \sigma(\boldsymbol{q}_i^T \boldsymbol{a}_t)$, 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 value 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 $y_{u,i,t}$ 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. In terms of average rating, there is some deviation from 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.

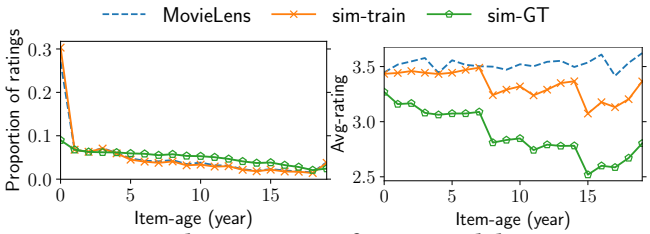

Figure 3: RQ3 – The proportion of ratings and the average rating of items over item-age on MovieLens, the simulated training set (sim-train) and the simulated Ground Truth (sim-GT).

Nonetheless, we clearly see that both dynamic selection bias and dynamic user preferences are represented in our simulation.

We compare the performance of TMF-DANCER with the following baselines: (1) Four methods that ignore bias altogether: Avg, T-Avg, MF and TMF (see Section 8). (2) Two methods optimized with the *static* IPS estimator: MF-staticIPS [41] and TMF-staticIPS, which use the Static Item Popularity propensities from Section 7. (3) A static preference method with dynamic debiasing: MF-DANCER, which optimizes a (static) MF while correcting for the effect of dynamic bias.

Finally, to evaluate whether TMF-DANCER is robust to misspecified propensities, we compare its performance with using Time-Aware General Popularity (TG-Pop): $\hat{p}_{u,i,t} = \frac{\sum_{u' \in \mathcal{U}} \sum_{i' \in \mathcal{I}} o_{u',i',t}}{|\mathcal{U}| \cdot |\mathcal{I}|}$, and Time-aware Item Popularity (T-Pop) (see Section 7.1).

9.2 Results for RQ3

The main results of our comparison are displayed in Table 3. Based on the displayed results we can make four observations: (1) The average methods (Avg and T-Avg) perform considerably worse than all other methods. Clearly, matrix factorization is preferable over averaging baselines. (2) The time-based methods outperform their static counterparts by substantial margins: TMF > MF, TMF-StaticIPS > MF-StaticIPS, and TMF-DANCER > MF-DANCER, except T-Avg < Avg due to sparsity. This shows that assuming static preferences can substantially hurt the performance of a method when user preferences are actually dynamic. (3) The debiased methods increase performance: MF-DANCER > MF and TMF-DANCER > TMF-StaticIPS > TMF. There is a single exception: MF > MF-staticIPS under the assumption of static bias. This surprising observation shows that DANCER is more robust to certain dynamic scenarios. (4) Finally, the best performing method is TMF-DANCER, which both models dynamic preferences and is debiased under the assumption of dynamic selection bias. While it is not a surprise that this method

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.

Method	MSE↓		MAE↓		ACC↑	
Avg	0.3155		0.4321		0.3623	
T-Avg	0.3280		0.4326		0.3614	
MF	0.1811	(0.0030)	0.3314	(0.0028)	0.4680	(0.0040)
TMF	0.1338	(0.0019)	0.2818	(0.0022)	0.5396	(0.0038)
MF-StaticIPS	0.1879	(0.0035)	0.3377	(0.0032)	0.4598	(0.0044)
TMF-StaticIPS	0.1086	(0.0021)	0.2491	(0.0027)	0.6065	(0.0057)
MF-DANCER	0.1533	(0.0016)	0.3032	(0.0017)	0.5074	(0.0023)
TMF-DANCER	**0.1045**†	(0.0014)	**0.2444**†	(0.0018)	**0.6151**†	(0.0039)

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

Method	MSE↓		MAE↓		ACC↑	
TG-Pop	0.1182	(0.0012)	0.2644	(0.0016)	0.5677	(0.0032)
T-Pop	**0.1041**	(0.0015)	0.2448	(0.0022)	0.6115	(0.0055)
Ground Truth	0.1045	(0.0014)	**0.2444**	(0.0018)	**0.6151**	(0.0039)

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-StatisIPS, 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

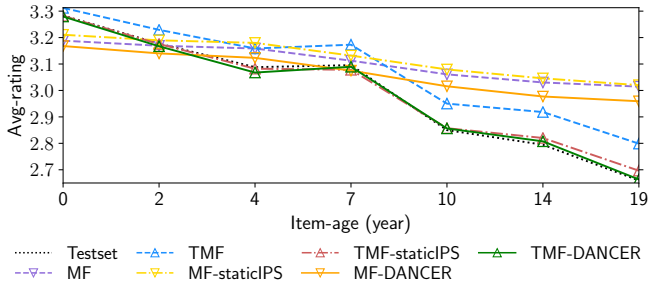

Figure 4: RQ3 – Average rating on items predicted by different models over the item-age.

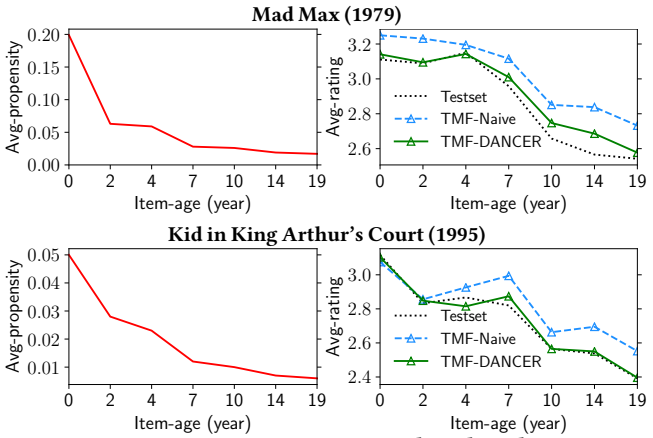

Figure 5: RQ3 – Average propensities and predicted average rating over item-age of the very popular movie "Mad Max (1979)" and the less popular "Kid in King Arthur's Court (1995)".

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.

REFERENCES

[1] Panagiotis Adamopoulos and Alexander Tuzhilin. 2014. On Over-specialization and Concentration Bias of Recommendations: Probabilistic Neighborhood Selection in Collaborative Filtering Systems. In *ACM Recsys*. 153–160.

[2] Ismail Ahmed Al-Qasem Al-Hadi, Nurfadhlina Mohd Sharef, Md Nasir Sulaiman, and Norwati Mustapha. 2017. Review of the Temporal Recommendation System with Matrix Factorization. *Int. J. Innov. Comput. Inf. Control* 13, 5 (2017), 1579–1594.

[3] Ricardo Baeza-Yates. 2018. Bias on the Web. *Commun. ACM* 61, 6 (2018), 54–61.

[4] Linas Baltrunas and Xavier Amatriain. 2009. Towards Time-dependant Recommendation based on Implicit Feedback. In *Workshop on context-aware recommender systems (CARS'09)*. Citeseer, 25–30.

[5] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. 2019. The Music Streaming Sessions Dataset. In *WWW*. 2594–2600.

[6] Pedro G Campos, Fernando Díez, and Iván Cantador. 2014. Time-aware Recommender Systems: A Comprehensive Survey and Analysis of Existing Evaluation Protocols. *User Modeling and User-Adapted Interaction* 24, 1 (2014), 67–119.

[7] Rocío Cañamares and Pablo Castells. 2018. Should I Follow the Crowd?: A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems. In *SIGIR*. ACM, 415–424.

[8] Abhijnan Chakraborty, Saptarshi Ghosh, Niloy Ganguly, and Krishna P Gummadi. 2017. Optimizing the Recency-relevancy Trade-off in Online News Recommendations. In *WWW*. 837–846.

[9] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and Debias in Recommender System: A Survey and Future Directions. *arXiv preprint arXiv:2010.03240* (2020).

[10] Ruey-Cheng Chen, Qingyao Ai, Gaya Jayasinghe, and W Bruce Croft. 2019. Correcting for Recency Bias in Job Recommendation. In *CIKM*. ACM, 2185–2188.

[11] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential Recommendation with User Memory Networks. In *WSDM*. 108–116.

[12] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[13] Giovanni Luca Ciampaglia, Azadeh Nematzadeh, Filippo Menczer, and Alessandro Flammini. 2018. How Algorithmic Popularity Bias Hinders or Promotes Quality. *Scientific reports* 8, 1 (2018), 1–7.

[14] Sara Hajian, Francesco Bonchi, and Carlos Castillo. 2016. Algorithmic Bias: From Discrimination Discovery to Fairness-aware Data Mining. In *KDD*. 2125–2126.

[15] Peng Han, Shuo Shang, Aixin Sun, Peilin Zhao, Kai Zheng, and Panos Kalnis. 2019. AUC-MF: point of interest recommendation with AUC maximization. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1558–1561.

[16] F Maxwell Harper and Joseph A Konstan. 2015. The Movielens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 1–19.

[17] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[18] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-based Neural Collaborative Filtering. *arXiv preprint arXiv:1808.03912* (2018).

[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[20] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. 2014. Probabilistic Matrix Factorization with Non-random Missing Data. In *ICML*. 1512–1520.

[21] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *arXiv preprint arXiv:1511.06939* (2015).

[22] Jin Huang, Harrie Oosterhuis, Maarten de Rijke, and Herke van Hoof. 2020. Keeping Dataset Biases out of the Simulation: A Debiased Simulator for Reinforcement Learning based Recommender Systems. In *ACM RecSys*. 190–199.

[23] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving Sequential Recommendation with Knowledge-enhanced Memory Networks. In *SIGIR*. 505–514.

[24] Guido W Imbens and Donald B Rubin. 2015. *Causal Inference in Statistics, Social, and Biomedical Sciences*. Cambridge University Press.

[25] Rolf Jagerman, Ilya Markov, and Maarten de Rijke. 2019. When People Change their Mind: Off-policy Evaluation in Non-stationary Recommendation Environments. In *WSDM*. 447–455.

[26] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A Re-visit of the Popularity Baseline in Recommender Systems. In *SIGIR*. 1749–1752.

[27] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *WSDM*. 781–789.

[28] Yehuda Koren. 2009. The Bellkor Solution to the Netflix Grand Prize. *Netflix prize documentation* 81, 2009 (2009), 1–10.

[29] Yehuda Koren. 2009. Collaborative Filtering with Temporal Dynamics. In *KDD*. 447–456.

[30] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[31] Roderick JA Little and Donald B Rubin. 2019. *Statistical Analysis with Missing Data*. Vol. 793. John Wiley & Sons.

[32] Benjamin M Marlin and Richard S Zemel. 2009. Collaborative Prediction and Ranking with Non-random Missing Data. In *ACM RecSys*. ACM, 5–12.

[33] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. 2014. Exploring the Filter Bubble: The Effect of Using Recommender Systems on Content Diversity. In *WWW*. 677–686.

[34] Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, Kathryn Vasilaky, and Elena Zheleva. 2020. Correcting for Selection Bias in Learning-to-Rank Systems. In *WWW*. 1863–1873.

[35] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. 2009. Experimental Comparison of Pre-vs. Post-filtering Approaches in Context-aware Recommender Systems. In *ACM RecSys*. 265–268.

[36] Eli Pariser. 2011. *The Filter Bubble: How the New Personalized Web is Changing What We Read and How We Think*. Penguin.

[37] Bruno Pradel, Nicolas Usunier, and Patrick Gallinari. 2012. Ranking with Non-random Missing Ratings: Influence of Popularity and Positivity on Evaluation Metrics. In *ACM RecSys*. 147–154.

[38] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware Recommender Systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[39] Paul R Rosenbaum. 2002. Overt Bias in Observational Studies. In *Observational Studies*. Springer, 71–104.

[40] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased recommender learning from missing-not-at-random implicit feedback. In *WSDM*. 501–509.

[41] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In *ICML*. 1670–1679.

[42] Harald Steck. 2010. Training and Testing of Recommender Systems on Data Missing Not At Random. In *KDD*. 713–722.

[43] Harald Steck. 2011. Item Popularity and Recommendation Accuracy. In *ACM RecSys*. 125–132.

[44] Harald Steck. 2013. Evaluation of Recommendations: Rating-prediction and Ranking. In *ACM RecSys*. 213–220.

[45] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*. 1441–1450.

[46] Adith Swaminathan and Thorsten Joachims. 2015. The Self-normalized Estimator for Counterfactual Learning. In *NIPS*. 3231–3239.

[47] Moshe Unger, Alexander Tuzhilin, and Amit Livne. 2020. Context-Aware Recommendations Based on Deep Learning Frameworks. *TMIS* 11, 2 (2020), 1–15.

[48] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *SIGIR*. 1001–1010.

[49] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2019. Doubly Robust Joint Learning for Recommendation on Data Missing Not At Random. In *ICML*. 6638–6647.

[50] Charinya Wangwatcharakul and Sartra Wongthanavasu. 2020. Dynamic Collaborative Filtering Based on User Preference Drift and Topic Evolution. *IEEE Access* 8 (2020), 86433–86447.

[51] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent Recommender Networks. In *WSDM*. 495–503.

[52] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Networks. In *AAAI*. 346–353.

[53] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. 2010. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In *SDM*. SIAM, 211–222.

[54] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *IJCAI*. 3940–3946.

[55] 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 *ACM RecSys*. 279–287.

[56] Sirui Yao, Yoni Halpern, Nithum Thain, Xuezhi Wang, Kang Lee, Flavien Prost, Ed H Chi, Jilin Chen, and Alex Beutel. 2021. Measuring Recommender System Effects with Simulated Users. *arXiv preprint arXiv:2101.04526* (2021).

[57] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *SIGIR*. 729–732.

[58] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. *arXiv preprint arXiv:2105.06067* (2021).

[59] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. 2021. RecBole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *CIKM*. 4653–4664.