Mitigating Sentiment Bias for Recommender Systems

Chen Lin
School of Informatics, Xiamen University
Xiamen, China
chenlin@xmu.edu.cn

Xinyi Liu
School of Informatics, Xiamen University
Xiamen, China
xinyiliu@stu.xmu.edu.cn

Guipeng Xv
School of Informatics, Xiamen University
Xiamen, China
xuguipeng@stu.xmu.edu.cn

Hui Li*
School of Informatics, Xiamen University
Xiamen, China
hui@xmu.edu.cn

ABSTRACT

Biases and de-biasing in recommender systems (RS) have become a research hotspot recently. This paper reveals an unexplored type of bias, i.e., sentiment bias. Through an empirical study, we find that many RS models provide more accurate recommendations on user/item groups having more positive feedback (i.e., positive users/items) than on user/item groups having more negative feedback (i.e., negative users/items). We show that sentiment bias is different from existing biases such as popularity bias: positive users/items do not have more user feedback (i.e., either more ratings or longer reviews). The existence of sentiment bias leads to low-quality recommendations to critical users and unfair recommendations for niche items. We discuss the factors that cause sentiment bias. Then, to fix the sources of sentiment bias, we propose a general de-biasing framework with three strategies manifesting in different regularizers that can be easily plugged into RS models without changing model architectures. Experiments on various RS models and benchmark datasets have verified the effectiveness of our de-biasing framework. To our best knowledge, sentiment bias and its de-biasing have not been studied before. We hope that this work can help strengthen the study of biases and de-biasing in RS.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

sentiment bias, de-biasing, recommender systems

ACM Reference Format:
Chen Lin, Xinyi Liu, Guipeng Xv, and Hui Li. 2021. Mitigating Sentiment Bias for Recommender Systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462943

*Corresponding author

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462943

1 INTRODUCTION

Recommender Systems (RS) are powering our everyday life. As people are producing more data than ever, RS have become an essential component in many online services to save us from information overload [6].

Unfortunately, the prevalence of RS is accompanied by the assertion that RS are severely biased [9]. User feedback data, which builds the foundation to train RS models, is observational rather than experimental [9]. Therefore, various biases (e.g., selection bias [30], popularity bias [1] and position bias [11]) exist in user feedback data. Simply fitting RS models to biased user feedback will increase discrepancy between offline evaluation and online test, and undermine user satisfaction.

From a broader perspective, biases in Machine Learning (ML) systems have become a bustling topic in recent years [32]. ML exhibits systematic error on certain groups of data samples (i.e., biases). In computer vision, demographic bias has been observed broadly, e.g., face recognition models perform poorly on certain demographic groups defined by sex, age, and race [42]. In natural language processing, word embeddings pre-trained on massive text have shown a strong gender bias, e.g., certain professions are wrongly correlated with a single gender [39]. Discovering the sources of biases and mitigating biases will lead to more fair and unbiased ML systems [32].

Users interact with RS, using reviews, ratings and other formats, to express their emotions and opinions concerning items. In this paper, we are interested in an unexplored question: *are RS models biased towards a certain user/item group with a specific sentiment polarity?* Through an empirical study, we discover a new type of bias, i.e., *sentiment bias*: *RS models make significantly more accurate recommendations on users/items having more positive feedback (i.e., positive users/items) than on users/items having more negative feedback (i.e., negative users/items).* Sentiment bias incurs low-quality recommendations to critical users (who are more likely to give negative feedback) and unfair recommendations for niche items (which are appreciated by only a small population). We observe the existence of sentiment bias in various review-based and non-review based RS models over several benchmark datasets. In our empirical study, we also show that sentiment bias is not the same as existing biases such as the popularity bias: the performance degradation on negative users/items is not caused by their lack of user feedback data. Due to the wide existence of sentiment bias and its unique

nature, we seek general de-biasing strategies that can be applied to different RS models in this paper.

We summarize our contributions as follows: (1) Firstly, we identify sentiment bias in RS and find that many RS models achieve lower accuracy on negative users/items. To our best knowledge, *sentiment bias and its de-biasing strategies have not been studied before*. (2) Moreover, we show that sentiment bias is not identical to existing biases in RS such as popularity bias. We discuss the possible underlying factors that cause the degraded performance. (3) Finally, we present a general de-biasing framework to mitigate sentiment bias in RS. Extensive experiments verify its effectiveness for de-biasing in various RS models. We hope that this work can help strengthen the study of biases and de-biasing in RS.

2 RELATED WORK

2.1 Biases in Recommender Systems

In the following, we illustrate several biases in the literature which exist in RS and may affect recommendation quality [9]:

Selection bias indicates that the observed ratings are not a representative sample of all ratings. Marlin et al. [30] conduct a user study and find that: (1) users tend to rate items that they love more often than other items; and (2) the sample of random ratings has markedly different properties than ratings of user-selected items, i.e., missing not at random [38].

Conformity bias appears as users rate similarly due to various factors (e.g., social influence [29]) but doing so does not conform with their own preferences. Krishnan et al. [22] observe the conformity bias by allowing users to rate twice: before and after seeing average ratings, and they find that the two rating distributions are different. Similar observations (i.e., discrepancy between individual and public opinions) are also mentioned in the literature [28, 44].

Position bias denotes that users tend to interact with items in higher position of the recommendation list even if they are not relevant. Collins et al. [11] observe the strong relationship between the rank of items shown to the user and whether the user decided to click them. Other works have also confirmed the existence of position bias in RS [14, 18], though Zheng et al. [49] found that position bias is negligible in their study.

Inductive bias, which introduces assumptions and may not be harmful, have been added to many RS models so that better generalization beyond training data can be achieved [9]. For instance, one common assumption for designing RS is that inner product [34] or learned similarity from neural networks [17] can be used to model user-item interactions.

Popularity bias, the most representative bias studied in the literature [9], indicates that RS prefer to recommend popular items more frequently than their original popularity in the data [3, 21]. Popularity bias is caused by training on the long-tail data of RS, i.e., most of the interactions are related to only a small number of popular items [1, 2, 4, 5]. Popularity bias will make popular items become even more popular while less popular items do not get the deserved attention (i.e., Matthew effect) [9].

Exposure bias happens as users are only exposed to part of the data (i.e., users are not randomly exposed to items [46]) and unobserved interactions do not always indicate "dislike". Exposure bias is the result of recommendations, i.e., what RS recommend are exposed to users [25]. Thus, one cause of exposure bias is popularity bias: unpopular items are not recommended frequently to users but it does not mean that they do not like them [3].

Note that *sentiment bias studied in this paper is not the same as any of the above biases*. More discussions are provided in Sec. 3.3.

2.2 De-biasing Recommendations

Causal inference is commonly used to discover the actual cause of an user-item interaction so that biases in RS models can be mitigated. To correct exposure bias, Wang et al. [45, 46] use causal-effect to explain recommendations. They first use the exposure data to estimate an exposure model offering items that the user is likely to consider. The exposure model is then used to generate substitute confounders for unobserved confounders. Finally, the predicted rating is conditioned on the substitute. Wei et al [47] propose to learn the contribution of each cause (user-item matching, item popularity and user activity) for an interaction via multi-task learning. When predicting, the effect of item popularity is removed by counterfactual inference. Bonner and Vasile [7] propose to use causal inference as domain adaption and design a causal embedding method that learns outcomes from biased RS to make better predictions.

In addition to causal inference, there are other works trying to de-bias via using new model architectures. Saito [36] introduces meta learning into de-biasing RS. He proposes to use one predictor to generate pseudo-ratings and another predictor to make the predictions. Liang et al. [24] develop a probabilistic model that separately captures whether a user has been exposed to an item and when a user has ultimately decided to click on it.

Unlike above works which de-bias in the training phase, another direction is to eliminate biases during the evaluation phase. Schnabel et al. [37] connect recommendation to causal inference and derive unbiased estimators for various RS performance measures. With these estimators, they propose an Empirical Risk Minimization framework for RS under selection bias. Similarly, Steck [38] also designs unbiased metrics for evaluating RS better.

The de-biasing framework introduced in this paper is different compared with existing works, as it mitigates a new bias, i.e., sentiment bias. Our framework can be easily plugged into existing RS models without changing model architectures, and it is orthogonal to de-biasing methods for the evaluation phase [37, 38].

3 EMPIRICAL STUDY

In this section, we conduct an empirical study on sentiment bias.

Preliminaries. Without loss of generality, the input to RS models is a rating matrix $\mathbf{X} \in \mathcal{R}^{|\mathcal{U}|\times|\mathcal{I}|}$, where $\mathcal{U}$ and $\mathcal{I}$ are user universe and item universe, respectively. We consider each rating $\mathbf{X}_{i,p}$, where $i \in \mathcal{U}, p \in \mathcal{I}$, is associated with a review $\mathbf{R}_{i,p}$ if $\mathbf{X}_{i,p} \neq 0$.[1] Each review $\mathbf{R}_{i,p}$ is a short piece of comment text posted by user i on item p. $\mathbf{R}_{i,p}$ contains a sequence of tokens, i.e., $\mathbf{R}_{i,p} = \left\{w_0, \cdots, w_{L(\mathbf{R}_{i,p})}\right\}$, where $L(\mathbf{R}_{i,p})$ is the length of review $\mathbf{R}_{i,p}$, and every token w belongs to the vocabulary $\mathcal{W}$. For a missing rating $\mathbf{X}_{i,p} = 0$, the corresponding review is also missing, i.e., $\mathbf{R}_{i,p} = \emptyset$. RS models can perform rating prediction task or top-K recommendation task. We

[1] In some E-commerce platforms such as Amazon, not all ratings are accompanied with reviews. For simplicity, we assume that every rating is associated with a review.

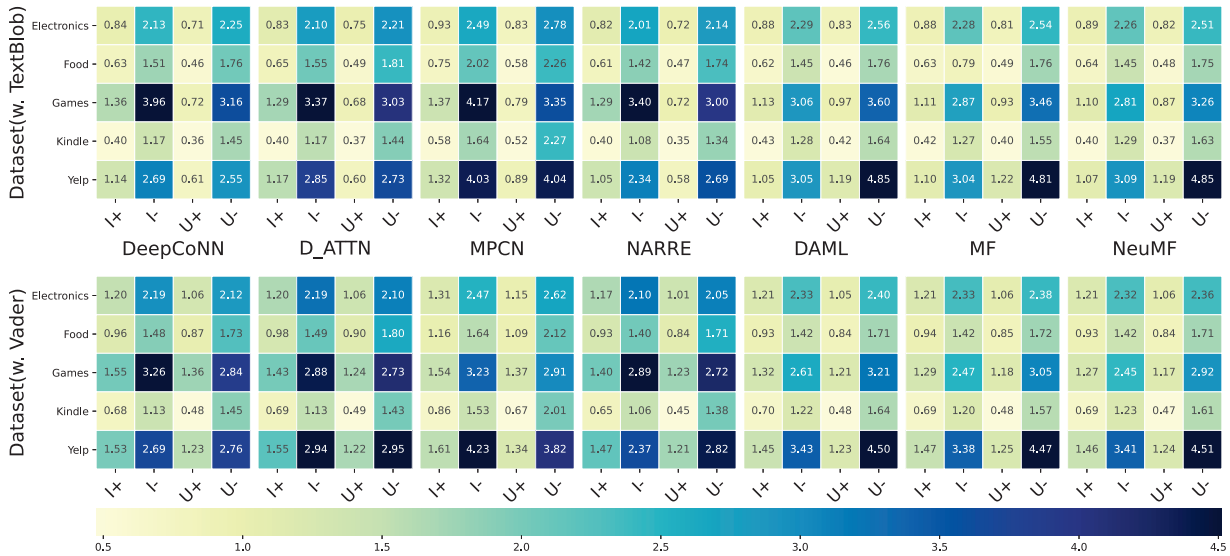

Figure 1: MSE on different user/item groups. $\mathcal{U}^+$ ($\mathcal{I}^+$) column is with lighter color than $\mathcal{U}^-$ ($\mathcal{I}^-$) column, indicating that RS performance on positive users (items) is better than on negative users (items).

start by assuming that the goal of an RS model is to predict the missing ratings based on the available ratings (and reviews) by optimizing a loss function $\mathcal{L}^{(RS)}$ computed over non-zero ratings $\mathbf{X}$ and its predictions $\hat{\mathbf{X}}$ from the RS model. If the RS model performs top-K recommendation task, $\hat{\mathbf{X}}$ is used in ranking candidate items for each user.

3.1 Empirical Study Protocol

User & Item Profile. Our empirical study aims at studying the difference of recommendation performance on two users/items groups: users/items generally with more positive feedback v.s. users/items generally with fewer positive feedback. We decide to use reviews to segment users/items, because reviews are more expressive and provide solid ground to analyze the sentiment of feedback than pure ratings. Our empirical study adopts the common paradigm to exploit reviews [27]: build user and item profiles by aggregating all relevant reviews to obtain user/item-level information.

Definition 1 (User Profile and Item Profile). *A user profile* $\mathbf{up}_i$ *is a piece of text that concatenates all reviews written by user* i*, i.e.,* $\mathbf{up}_i = concat(\mathbf{R}_{i,p}, \forall p \in \mathcal{I})$*. An item profile* $\mathbf{ip}_p$ *is a piece of text that concatenates all reviews written on item* p*, i.e.,* $\mathbf{ip}_p = concat(\mathbf{R}_{i,p}, \forall i \in \mathcal{U})$.

User/Item Grouping. We perform unsupervised sentiment analysis on user and item profiles. We experiment with two sentiment analysis tools: TextBlob (lexicon-based analysis)[2] and VADER (lexicon and rule-based analysis)[3]. These tools return a sentiment polarity value for each user/item profile, which is used to rank the user/item profiles so that more positive profiles are ranked higher than negative profiles. Then, we extract two groups of users: the top 10% users are denoted as *positive users* $\mathcal{U}^+$, and the bottom 10% users are denoted as *negative users* $\mathcal{U}^-$. Similarly, we have *positive items* $\mathcal{I}^+$ and *negative items* $\mathcal{I}^-$. Next, we train RS models on the training set, and make predictions on the test set. Finally, we report recommendation qualities on positive users, negative users, positive items, and negative items, respectively.

Table 1: Statistics of the data.

Dataset	#Users	#Items	#Reviews	Sparsity (%)
Amazon Gourmet Food	14,683	8,715	151,253	99.8818
Amazon Kindle Store	68,225	61,936	982,618	99.9767
Amazon Video Games	826,769	50,212	1,324,753	99.9968
Amazon Electronics	192,405	63,003	1,689,188	99.9861
Yelp	1,070,074	36,490	3,766,145	99.9904

Datasets. We use five public datasets with different degrees of sparsity, including four Amazon product rating 5-core datasets [31] (i.e., each training user/item has at least five ratings), and Yelp[4] dataset. We apply 5-core pre-processing on Yelp to make sure each user/item has sufficient feedback. We use the default training/test split. Tab. 1 lists the statistics of the five datasets.

RS Models. We investigate two representative non-review-based RS models MF [20] and NeuMF [17], and several review-based RS models including DeepCoNN [50], MPCN [40], NARRE [8], DAML [26] and D_ATTN [19]. We use public code[5] [35], with the default parameter settings, in the empirical study.

3.2 Existence of Sentiment Bias in RS

Fig. 1 depicts the recommendation performance evaluated by MSE (the definition of MSE will be given later in Eq. 1). We can see that RS models almost always perform better on positive users and items (with smaller MSE) than on negative users and items (with larger MSE). *The performance gap is significant and universal across different RS models and datasets, regardless of the nature of RS models (i.e., shallow or deep, review-based or non-review-based), the sparsity of*

[2]https://textblob.readthedocs.io
[3]https://github.com/cjhutto/vaderSentiment
[4]https://www.yelp.com/dataset
[5]https://github.com/noveens/reviews4rec

datasets and the sentiment analysis tools adopted. Though we only report the results for top and bottom 10% users/items ranked by their sentiment polarity values, we have observed similar, significant divergence of performance, between positive and negative groups with smaller and larger coverage percentage, e.g., from top/bottom 1% to 20%. Thus, the empirical results demonstrate the existence of *sentiment bias* in RS:

Definition 2 (sentiment bias). *Sentiment bias is defined as the divergence between recommendation performance on positive users/items and negative users/items. Specifically, the user sentiment bias and the item sentiment bias for an RS model can be defined as* $BU(RS) = E(RS, \mathcal{U}^-, \mathcal{I}) - E(RS, \mathcal{U}^+, \mathcal{I})$, *and* $BI(RS) = E(RS, \mathcal{U}, \mathcal{I}^-) - E(RS, \mathcal{U}, \mathcal{I}^+)$, *respectively.* E *is the evaluation metric, e.g., MSE.*

Sentiment bias is harmful. An RS model with severe sentiment bias makes low-quality recommendations to critical users (i.e., users tend to give fewer positive comments) and unfair recommendations for niche items (i.e., items that receive positive comments from a small population). Critical users are valuable resources to RS. When critical users provide informative reviews explaining their dissatisfaction, they contribute to the community and help RS attract more users. However, sentiment bias hurts their user experiences as they will frequently receive unsatisfying recommendations, which will drive them away. Furthermore, it is unfair to niche items, because they might receive fewer recommendations, which lowers their exposure to users.

To study the impacts of sentiment bias on recommendation qualities, we first investigate the performance of RS models in terms of two common evaluation metrics, i.e., *Mean Square Error (MSE)* for the rating prediction task and *NDCG@K* for the top-K recommendation task [6]:

$$MSE = \frac{\sum_{\mathrm{X}_{i,p} \neq 0} \left(\hat{\mathrm{X}}_{i,p} - \mathrm{X}_{i,p}\right)^2}{N}, \quad NDCG@K = \frac{1}{Z_K} \sum_{j=1}^{K} \frac{2^{rel_j} - 1}{\log_2(1+j)} \quad (1)$$

where $i \in \mathcal{U}, p \in \mathcal{I}$, N is the total number of data instance, and Z_K is a normalizer which ensures that perfect ranking has a value of 1. rel_j is the relevance of item at position j, i.e., the ground-truth rating value. Lower MSE or higher NDCG@K suggests better recommendations. MSE and NDCG@K evaluate an RS model from the perspective of accuracy.

When the recommendation lists for different users are very similar (i.e., they are highly overlapping), the system actually fails to provide *personalized* recommendations, leading to unsatisfying user experiences [6]. Therefore, in addition to accuracy metrics MSE and NDCG@K, we also include *Diversity@K* in our empirical study. Diversity@K is defined as the number of unique items in all top-K recommendation lists that the RS model offers to all users. Higher Diversity@K indicates better recommendations.

In Fig. 2, we plot the correlations between item bias (i.e., BI in Def. 2) and user bias (i.e., BU in Def. 2), item bias and MSE, item bias and NDCG@5, and item bias and Diversity@5. Note that each point in Fig. 2 indicates an RS model on one dataset. As shown in Fig. 2, BU, MSE and NDCG@5 can fit a linear model with BI well. Diversity@5 can fit a logarithmic function with BI. We can see that BU and BI are highly positively correlated. On the other hand, less biased models (i.e., with lower BI) generally deliver recommendations with higher qualities, i.e., with lower MSE, higher NDCG@5 and higher

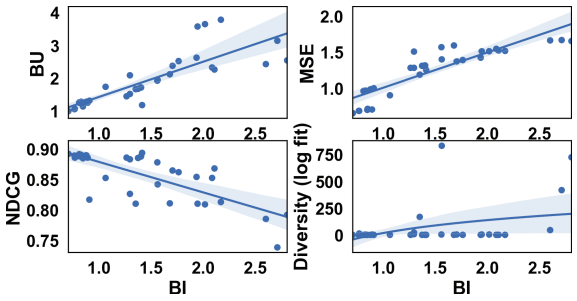

Figure 2: Correlations of BI v.s. BU/MSE/NDCG@5/Diversity@5 for all RS models and datasets.

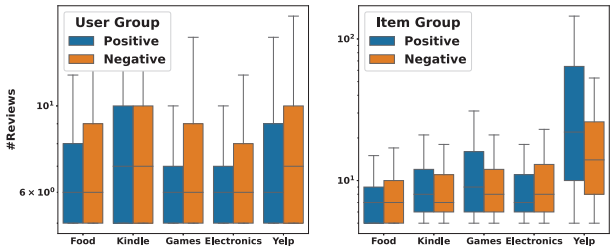

Figure 3: Distributions of #reviews for positive and negative users/items.

Diversity@5. Based on these observations, we can conclude that *eliminating BI may lead to reduced BU and higher recommendation qualities*, which motivates us to design a de-biasing framework in Sec. 4 to lower item sentiment bias, improve recommendation qualities, and raise user satisfaction.

3.3 Comparisons with Other Biases

We have verified the existence of sentiment bias in Sec. 3.2: numerous review-based RS models and representative non-review-based RS models are biased towards positive users and items, i.e., recommendations are more accurate on positive users and items. In this subsection, we show that *sentiment bias is a new type of bias in RS.* We choose popularity bias to illustrate the uniqueness of sentiment bias, because popularity bias is one of the most important bias, and it has been shown to cause other biases in RS such as exposure bias and so on [9].

We count the number of reviews (i.e., popularity) for each positive and negative user/item. As shown in Fig. 3, in two out of five datasets, positive items receive fewer reviews than negative items. Note that popularity bias means RS models are in favor of popular items. Therefore, item sentiment bias, where RS models show degraded performance on negative items (which are sometimes more popular items), is not the same as popularity bias.

Fig. 3 also excludes data imbalance as a reason for performance degradation. It is well regarded that, imbalanced training data, either by popularity bias, selection bias or exposure bias, causes poor performance on long-tail or cold-start users/items [10]. Fig. 3 shows that positive and negative items have balanced user feedback data. In addition, positive users, in general, have written fewer reviews than negative users. Consequently, it is clear that poor performance on negative users, is not due to the lack of sufficient training data.

Next, we discuss two other possible factors that may incur sentiment bias so that we can focus on the exact cause (i.e., the inadequacy of RS models themselves) when designing de-biasing methods in the next section.

Length of Reviews. In many natural language processing tasks such as text classification, it is more challenging for models to analyze shorter documents. However, according to Fig. 4 which reports distributions of review lengths of different groups, reviews for negative users/items are generally longer. Thus, sentiment bias is not the result of shorter text.

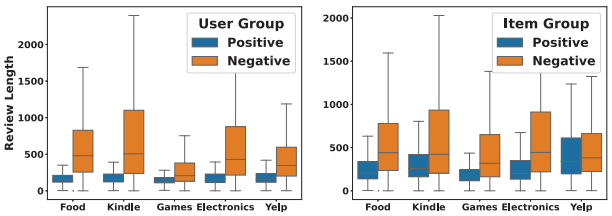

Figure 4: Distributions of review lengths for positive and negative users/items.

Published Time of Reviews. We also demonstrate when the reviews of positive and negative users/items are posted. As illustrated in Fig. 5, reviews for negative users/items are in general published earlier (i.e., with smaller UNIX timestamps). This is contrary to existing findings [13], which claims that, later feedback is more negative because consumers have difficulties to diagnose previous reviews and thus make more purchase mistakes. The observation also suggests that sentiment bias can not be reduced by mitigating social influence bias [22]. RS models are incompetent on negative users, who purchase items early and are less likely to be influenced by late consumers.

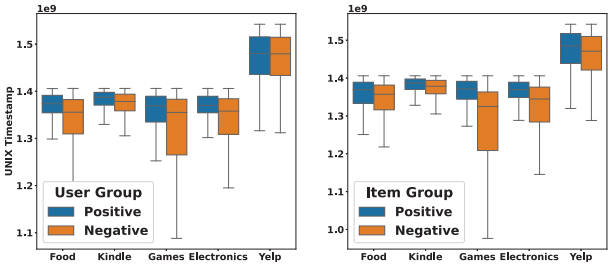

Figure 5: Distributions of review timestamps for positive and negative users/items.

4 ELIMINATING SENTIMENT BIAS

In Sec. 3, we have demonstrated the existence of sentiment bias which hurts the performance of RS and is most likely to be caused by the inadequacy of RS models. In this section, we introduce our de-biasing framework with three strategies manifesting in different regularizers to help existing RS models mitigate sentiment bias without modifications of model architectures.

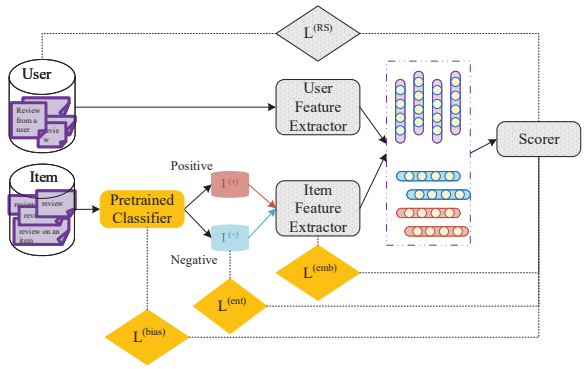

Figure 6: Overview of the proposed de-biasing framework.

4.1 Base RS Model

We first formulate the architecture of existing RS models. Most, if not all, RS models consist of two core modules: *feature extractors* and a *scorer*. The feature extractors and scorer are trained via minimizing a recommendation loss function $\mathcal{L}^{(RS)}$.

Feature extractors transform the input (e.g. the rating matrix and/or reviews) to low dimensional numerical vectors (i.e., embeddings), which represent features of users, items and/or reviews. For simplicity, we assume that all embeddings are with the same size d. We use $\mathbf{u}_i \in \mathcal{R}^d$ to denote the embedding for user i, $\mathbf{v}_p \in \mathcal{R}^d$ to denote embedding for item p, $\mathbf{r}_{i,p} \in \mathcal{R}^d$ to denote the embedding for the review written by user i on item p.

The *scorer* converts the output of feature extractors, i.e., embeddings, to a numerical score. The score is provided to users as the predicted rating in the rating prediction task, or it is used in ranking the items recommended to users in the top-K recommendation task.

4.2 Overview of De-biasing Framework

Our goal is to design a general framework that can be applied to most RS models to eliminate sentiment bias without changing their structures. The overall framework is depicted in Fig. 6.

Motivated by our observations in Sec. 3.2 (i.e., reduced item sentiment bias possibly leads to reduced user sentiment bias and improved recommendation qualities), we first classify the items. We employ a pre-trained classifier (e.g., VADER) on item profiles illustrated in Sec. 3.1 to extract positive items $\mathcal{I}^{(+)}$ and negative items $\mathcal{I}^{(-)}$. Note that $\mathcal{I}^{(+)}$ and $\mathcal{I}^{(-)}$ are not necessarily identical to $\mathcal{I}^{+}$ and $\mathcal{I}^{-}$, which are used in measuring item sentiment bias, as defined in Def. 2. Then, we adopt a new overall loss function in the optimization of RS models as shown in Eq. 2, which includes the original recommendation loss function $\mathcal{L}^{(RS)}$ and three additional regularization terms:

$$\mathcal{L} = \mathcal{L}^{(RS)} + \lambda_1 \mathcal{L}^{(bias)} + \lambda_2 \mathcal{L}^{(ent)} + \lambda_3 \mathcal{L}^{(emb)}, \tag{2}$$

where regularizers are combined with coefficients λ_1, λ_2, and λ_3.

Our framework is general. It can be easily injected into any RS model while keeping the original loss $\mathcal{L}^{(RS)}$. It does not require any change of the original RS models. The item sentiment bias is reduced via the three additional regularizations $\mathcal{L}^{(bias)}$, $\mathcal{L}^{(ent)}$ and $\mathcal{L}^{(emb)}$. The regularizations are computed over embeddings that can be directly extracted from the original RS models.

4.3 $\mathcal{L}^{(bias)}$: Regularization with Item Bias

A straightforward way for de-biasing sentiment bias, without modifying the RS model, is to directly regularize sentiment bias in the loss function. However, doing so will not conform to our expectations. Assuming that we directly integrate the item sentiment bias as a regularization term, as shown in the first line of Eq. 3, where we use $\mathcal{I}^{(+)}$ and $\mathcal{I}^{(-)}$ in computing item sentiment bias, and $\mathcal{I}^{(rest)}$ represents the rest items (if any) that are labeled as neither positive nor negative items. Since both the original recommendation loss $\mathcal{L}^{(RS)}$ and the item sentiment bias term $BI(RS)$ are summing up the differences between predictions and ground-truth values, the error terms of positive items in $\mathcal{L}^{(RS)}$ and $BI(RS)$ will be canceled off, as shown in the second and third lines of Eq. 3:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}^{(RS)} + BI(RS) \\ &\approx E(RS, \mathcal{U}, \mathcal{I}^{(+)}) + E(RS, \mathcal{U}, \mathcal{I}^{(-)}) + E(RS, \mathcal{U}, \mathcal{I}^{(rest)}) \\ &\quad + E(RS, \mathcal{U}, \mathcal{I}^{(-)}) - E(RS, \mathcal{U}, \mathcal{I}^{(+)}) \\ &= 2 \times E(RS, \mathcal{U}, \mathcal{I}^{(-)}) + E(RS, \mathcal{U}, \mathcal{I}^{(rest)}).\end{aligned} \tag{3}$$

Note that the first and second lines in Eq. 3 are not completely equivalent (thus we use "≈"). For example, most RS models adopt the RMSE loss function or the cross entropy loss function. For such cases, $\mathcal{L}^{(RS)}$ cannot be simply divided into three independent terms $E(RS, \mathcal{U}, \mathcal{I}^{(+)})$, $E(RS, \mathcal{U}, \mathcal{I}^{(-)})$ and $E(RS, \mathcal{U}, \mathcal{I}^{(rest)})$. The impact of positive users on the overall loss function is not completely ignored. However, their importance is still significantly downgraded.

To address the above issue, our strategy is to regularize with partial item sentiment bias, i.e., divergence between representative items. Intuitively, users' intrinsic preferences are not affected by how the reviews are expressed. Thus, for each user, in predicting his/her favorite items, the RS model should not be biased towards the positive item set or the negative item set. For each user, we compute the difference between ratings on two representative items, each from the positive and negative item sets, and use it in the first regularizer $\mathcal{L}^{(bias)}$:

$$\mathcal{L}^{(bias)} = \sum_{i \in \mathcal{U}} max_{p \in \mathcal{I}^{(+)}} \left(\hat{\mathbf{X}}_{i,p} - \bar{\mathbf{X}}_p\right) - max_{q \in \mathcal{I}^{(-)}} \left(\hat{\mathbf{X}}_{i,q} - \bar{\mathbf{X}}_q\right), \tag{4}$$

where $\hat{\mathbf{X}}_{i,p}$ is the predicted rating of user i on item p. $\bar{\mathbf{X}}_p$ is the average rating of item p, which acts as a baseline to adjust comparisons among different items, so that $\hat{\mathbf{X}}_{i,p} - \bar{\mathbf{X}}_p$ correctly reflects the user preference. Incorporating $\mathcal{L}^{(bias)}$ allows fair treatments on both positive and negative items. $\mathcal{L}^{(bias)}$ is minimized, when each user's favorite items in the positive and negative item sets have similar ratings. In other words, $\mathcal{L}^{(bias)}$ helps keep the RS models from being in favor of (i.e., systematically giving higher ratings on) a certain item group.

4.4 $\mathcal{L}^{(ent)}$: Regularization with Entropy

In our preliminary study, we have found that negative items and positive items have different rating distributions. As an illustrative example, we plot the distributions of actual ratings (integers) and predicted ratings (real numbers) by DeepCoNN model on Amazon Gourmet Food dataset in Fig. 7. We can see that, the actual ratings of positive items follow a single mode distribution which can be captured well by the RS model. On the contrary, actual ratings of negative items follow a bimodal distribution and the RS model is incompetent to predict them.

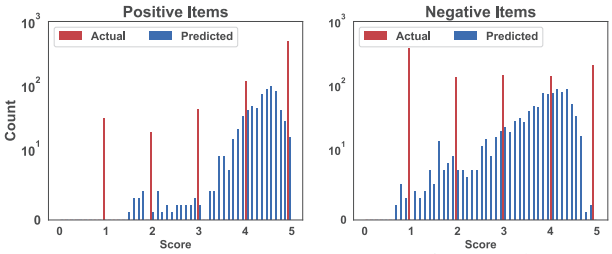

Figure 7: Distributions of actual ratings (integers) and predicted ratings (real numbers) on positive and negative items by DeepCoNN on Amazon Gourmet Food.

The above observation inspires us to adopt a strategy to "force" the predicted ratings of negative items to spread evenly in the space so that they can fit the distribution of the actual ratings better. Many RS models estimate the probabilities of different rating values for the rating prediction. Thus, we introduce an entropy-based regularizer $\mathcal{L}^{(ent)}$, which increases the uncertainty of rating prediction and thus spreads the ratings over the prediction space. There remains one issue to define the entropy-based regularization: the rating variable. Most RS models produce the output $\hat{\mathbf{X}}_{i,p} \in [0, 1]$, which corresponds to the probability of implicit feedback $\mathbf{X}_{i,p} \in \{0, 1\}$, i.e., $\hat{\mathbf{X}}_{i,p} = Pr(\mathbf{X}_{i,p} = 1)$, where Pr denotes the probability. Then, the entropy-based regularizer can be defined in the format of $-\hat{\mathbf{X}}_{i,p} \log \hat{\mathbf{X}}_{i,p} - (1 - \hat{\mathbf{X}}_{i,p}) \log(1 - \hat{\mathbf{X}}_{i,p})$. However, we empirically find that such a binarized entropy regularizer does not work well.

Therefore, we opt to present a regularization term for *Likert-scale ratings*. Most productive RS, including Amazon and Yelp, require users to provide ratings using one to five stars. Assume that $\mathbf{S}_{i,p,t} = Pr(\mathbf{X}_{i,p} = t)$ denotes the probability that user i gives item p a rating of t, where $0 \le \mathbf{S}_{i,p,t} \le 1$, $\sum_t \mathbf{S}_{i,p,t} = 1$, and $t \in \{1, 2, 3, 4, 5\}$. The entropy-based regularizer $\mathcal{L}^{(ent)}$ is given as follows:

$$\mathcal{L}^{(ent)} = \sum_{i \in \mathcal{U}} \sum_{p \in \mathcal{I}^{(-)}} \sum_{t=1}^{5} \mathbf{S}_{i,p,t} \log \mathbf{S}_{i,p,t}. \tag{5}$$

$\mathcal{L}^{(ent)}$ is aggregated over $\mathcal{I}^{(-)}$, because we want to improve fitting on negative items. With a minimized $\mathcal{L}^{(ent)}$, the RS model is encouraged to predict more evenly distributed ratings than before. To obtain $\mathbf{S}$ for a user and an item, we can simply connect a feedforward neural network layer with softmax to the final layer of the scorer in RS.

4.5 $\mathcal{L}^{(emb)}$: Regularization with Embeddings

The third strategy in our framework for eliminating sentiment bias only applies to review-based RS models.

To increase the variability of rating predictions for negative items, in addition to $\mathcal{L}^{(ent)}$ in Sec. 4.4, a possible remedy (i.e., the third strategy) is to *decrease similarities among the learned negative item embeddings by RS models* via an additional regularizer. As the core idea of most RS models encourages models to make similar predictions on similar items [6], decreasing similarities among negative item embeddings may keep RS models from providing

Figure 8: Typical reviews for a highly controversial SSD product in Amazon: positive reviews (five or four stars) are different, and negative reviews (one star) are also dissimilar.

predictions that are densely packed. The similarity between two item embeddings $\mathbf{v}_p$ and $\mathbf{v}_q$ for items p and q can be represented by their inner product $\mathbf{v}_p^T\mathbf{v}_q$ [34], i.e., lower $\mathbf{v}_p^T\mathbf{v}_q$ for more dissimilar items. Hence, the simplest way to implement the third strategy is directly adding a regularizer in the format of $\mathbf{v}_p^T\mathbf{v}_q$ to the overall loss function. Nevertheless, we find this method does not work well for de-biasing.

Recall that we have discovered the difference of review lengths between positive and negative items in Sec. 3.3. It is reasonable to suspect that current review-based RS models have difficulties learning review embeddings well, which indirectly incurs the low-quality item embeddings and poor recommendation performance on negative items. After we investigate the five datasets, we find that reviews for a highly appreciated item are all alike, while reviews for a highly controversial item, which usually belongs to the negative items $\mathcal{I}^{(-)}$, are positive or negative in their own ways. We illustrate this phenomenon in Fig. 8 with typical reviews for a real SSD product in Amazon[6] that is highly controversial.

A possible direction to address the above problem is to design new feature extractors to enhance the representation learning for reviews of negative items so that negative item embeddings can also get improved (i.e., more dissimilar). Since our goal is to de-bias without changing the RS model, an alternative is to *design a regularizer targeting at review embeddings of negative items and manifesting the relation between review embeddings and item embeddings.*

Inspired by the success of Translation-based Knowledge Graph Completion [43], we can interpret the review embedding $\mathbf{r}$ as the "translation" from user embedding $\mathbf{u}$ to item embedding $\mathbf{v}$ [12]:

$$\mathbf{v}_p \approx \mathbf{r}_{i,p} + \mathbf{u}_i, \tag{6}$$

where $\mathbf{r}_{i,p}$ is the embedding of review written by user i on item p. Note that "$\approx$" describes the low-error "translation" among different feature spaces. Similar ideas of "translation" have also been adopted in previous works [12, 15, 16, 23, 33, 48] to provide better recommendations. In our framework, we adopt the following "translation" adapted from Eq. 6 to interpret the similarity between negative items p and q as well as the relation between their item embeddings and review embeddings:

$$\mathbf{v}_p^T\mathbf{v}_q \approx \mathbf{r}_{i,p}^T\mathbf{r}_{j,q} + \mathbf{u}_i^T\mathbf{u}_j. \tag{7}$$

We expect a small value of $\mathbf{v}_p^T\mathbf{v}_q$ for two negative items p and q (i.e., the intuition for the third strategy: make negative items more dissimilar). Based on the adapted "translation", we propose the third regularizer in our framework:

$$\mathcal{L}^{(emb)} = \sum_{p,q\in\mathcal{I}^-} \sum_{\substack{i,j\in\mathcal{U} \\ g(\mathbf{X}_{i,p})=g(\mathbf{X}_{j,q})}} \left(\mathbf{r}_{i,p}^T\mathbf{r}_{j,q} + \mathbf{u}_i^T\mathbf{u}_j\right)^2, \tag{8}$$

where $g(\cdot)$ is a grouping function that returns 1 if the input variable is smaller than 3, and returns 0 otherwise. $\mathcal{L}^{(emb)}$ affects negative items only. Without $\mathcal{L}^{(emb)}$, if two reviews from different users on the negative items are associated with similar ratings (i.e., $g(\mathbf{X}_{i,p}) = g(\mathbf{X}_{j,q})$), the RS model will attempt to infer similar review embeddings. The reason is that $\mathcal{L}^{(RS)}$ fits ratings by review embeddings. Hence, similar review embeddings may cause similar negative item embeddings according to Eq. 7 and result in the "crowded" predictions. As a comparison, using $\mathcal{L}^{(emb)}$ will lower $\mathbf{r}_{i,p}^T\mathbf{r}_{j,q} + \mathbf{u}_i^T\mathbf{u}_j$, leading to smaller $\mathbf{v}_p^T\mathbf{v}_q$ (i.e., dissimilar negative item embeddings) and eventually more scattered predictions.

Note that embeddings used in Eq. 8 can be easily extracted. All RS models produce item embeddings and user embeddings. For review embeddings, we notice that there are two main paradigms [27]: review-level modeling and document-level modeling. Some models (e.g., MPCN) perform review-level modeling and directly produce the review embedding for each review. Document-level review-based models (e.g., DeepCoNN, NARRE, DAML and D_ATTN), do not extract individual review embeddings. However, review-based RS models generally consider word tokens as numerical word embeddings, which are updated in the optimization. Thus, we use the average of word embeddings for a review as its review embedding for document-level review-based models.

5 EXPERIMENTS ON DE-BIASING

In this section, we conduct experiments on de-biasing to answer the following research questions:

RQ1: Does the proposed framework eliminate sentiment bias in RS models?

RQ2: What are the impacts of the proposed framework on recommendations qualities?

RQ3: How does each regularizer in our framework contribute to de-biasing?

We use the same datasets and RS models as described in Sec. 3.1. In the experiments, we first separately train each RS model without de-biasing on the training set of each dataset. Then, as described in Sec. 4.2, we use VADER to extract positive and negative items. VADER assigns a compound score within the range of $[-1, 1]$ to each of the sentence in the item profile. If the compound score is larger than zero, then the sentence is positive. If the ratio of positive sentences in an item profile is larger than 50%, we put the corresponding item in the positive item set. Otherwise, we add it to the negative item set. Finally, we incorporate the three de-biasing regularizations proposed in Sec. 4, re-train the RS models from scratch, and report the results.

5.1 Effects on Sentiment Bias (RQ1)

When measuring sentiment bias, we use the top and bottom 10% items as positive and negative items, and the details are described in Sec. 3.1. Tabs. 2, 3, 4, 5 and 6 report the user bias (BU) and the

[6] https://www.amazon.com/dp/B084RCMLXL

Table 2: User bias, item bias, and overall recommendation performance of different RS models with and without de-biasing on Amazon Gourmet Food.

Methods	BU	BI	MSE	NDCG@5	Diversity@5
DeepCoNN	1.2958	0.8749	0.9942	0.8905	5
w/.De-bias	**1.2699**	**0.7856**	**0.9829**	0.8904	**10**
D_ATTN	1.3244	0.9047	1.0060	0.8173	5
w/.De-bias	**1.2622**	**0.8224**	**0.9982**	**0.8892**	**7**
MPCN	1.6758	1.3502	1.1966	0.8109	173
w/.De-bias	**1.4547**	**1.0461**	**1.1297**	**0.8343**	**948**
NARRE	1.2759	0.8067	0.9669	0.8921	5
w/.De-bias	**1.2344**	**0.7655**	**0.9652**	**0.8943**	**16**
DAML	1.2999	0.8279	0.9672	0.8911	5
w/.De-bias	**1.2403**	**0.7927**	**0.9671**	**0.8945**	**13**
MF	1.2681	0.8282	0.9728	0.8882	8
w/.De-bias	**1.2472**	**0.7885**	**0.9698**	**0.8889**	7
NeuMF	1.2668	0.8079	0.9693	0.8887	5
w/.De-bias	**1.2259**	**0.7675**	**0.9687**	**0.8898**	**7**

Table 3: User bias, item bias, and overall recommendation performance of different RS models with and without de-biasing strategies on Amazon Kindle Store.

Methods	BU	BI	MSE	NDCG@5	Diversity@5
DeepCoNN	1.0811	0.7665	0.6962	0.8863	17
w/.De-bias	**0.9870**	**0.6901**	**0.6532**	**0.8923**	**30**
D_ATTN	1.0723	0.7633	0.6960	0.8900	5
w/.De-bias	**0.9482**	**0.6785**	**0.6546**	**0.8927**	**10**
MPCN	1.7521	1.0606	0.9077	0.8531	5
w/.De-bias	**1.3143**	**0.9370**	**0.7941**	0.8531	**939**
NARRE	1.0024	0.7044	0.6612	0.8929	5
w/.De-bias	**0.9247**	**0.6469**	**0.6244**	**0.8979**	**13**
DAML	1.2250	0.8485	0.7213	0.8872	5
w/.De-bias	**1.0237**	**0.7468**	**0.6461**	**0.8949**	**7**
MF	1.1546	0.8446	0.7074	0.8852	5
w/.De-bias	**1.1164**	**0.8098**	**0.6769**	**0.8919**	**13**
NeuMF	1.2605	0.8891	0.7130	0.8865	7
w/.De-bias	**1.1842**	**0.8479**	**0.6859**	**0.8902**	**21**

Table 4: User bias, item bias, and overall recommendation performance of different RS models with and without de-biasing strategies on Amazon Video Games.

Methods	BU	BI	MSE	NDCG@5	Diversity@5
DeepCoNN	2.4418	2.5994	1.6701	0.7857	50
w/.De-bias	**2.1032**	**1.6850**	**1.5140**	**0.8621**	**118**
D_ATTN	2.3413	2.0832	1.5500	0.8534	6
w/.De-bias	**1.1893**	**1.8587**	**1.5027**	**0.8590**	**10**
MPCN	2.5574	2.8042	1.6608	0.7924	728
w/.De-bias	**2.3943**	**2.6592**	**1.6466**	**0.7932**	**910**
NARRE	2.2774	2.1054	1.5260	0.8686	5
w/.De-bias	**1.9752**	**1.4594**	**1.4388**	**0.8705**	**13**
DAML	2.6363	1.9346	1.4302	0.8546	5
w/.De-bias	**2.0396**	**1.6465**	**1.3977**	**0.8605**	**28**
MF	2.5329	1.7625	1.3963	0.8629	5
w/.De-bias	**2.2652**	**1.5989**	**1.3844**	**0.8631**	**11**
NeuMF	2.3867	1.7048	1.3781	0.8655	8
w/.De-bias	**2.2633**	**1.6505**	**1.3713**	**0.8685**	7

item bias (BI) of the seven RS models with/without de-biasing on the five datasets. Performance after de-biasing is shown in bold if it is better than results without de-biasing. For review-based RS, all three regularizers are used. For non-review-based RS, only $\mathcal{L}^{(bias)}$ and $\mathcal{L}^{(ent)}$ are used. The coefficients are tuned to achieve the best results using grid search. After tuning, $\lambda_1 = 0.25$, $\lambda_2 = 0.05$ and $\lambda_3 = 0.05$. We also report the impacts of coefficients in Sec. 5.3.

Table 5: User bias, item bias, and overall recommendation performance of different RS models with and without de-biasing strategies on Amazon Electronics.

Methods	BU	BI	MSE	NDCG@5	Diversity@5
DeepCoNN	1.5338	1.2952	1.2912	0.8836	26
w/.De-bias	**1.4029**	**1.1806**	**1.2579**	**0.8939**	25
D_ATTN	1.4579	1.2635	1.2906	0.8867	5
w/.De-bias	**1.4210**	**1.2281**	**1.2834**	**0.8917**	**6**
MPCN	1.9555	1.5603	1.4075	0.8426	835
w/.De-bias	**1.7611**	**1.4899**	**1.3750**	**0.8439**	**965**
NARRE	1.4132	1.1890	1.2588	0.8944	6
w/.De-bias	**1.3522**	**1.1352**	**1.2394**	**0.8980**	**23**
DAML	1.7306	1.4007	1.3240	0.8890	5
w/.De-bias	**1.4927**	**1.2227**	**1.2527**	**0.8950**	**7**
MF	1.7254	1.3997	1.3215	0.8881	6
w/.De-bias	**1.5508**	**1.2443**	**1.2893**	**0.8886**	6
NeuMF	1.6880	1.3744	1.3187	0.8859	6
w/.De-bias	**1.1842**	**1.2812**	**1.2871**	**0.8898**	**8**

Table 6: User bias, item bias, and overall recommendation performance of different RS models with and without de-biasing strategies on Yelp.

Methods	BU	BI	MSE	NDCG@5	Diversity@5
DeepCoNN	1.9377	1.5571	1.5740	0.8790	7
w/.De-bias	**1.8886**	**1.3805**	**1.5290**	0.8262	**8**
D_ATTN	2.1309	1.6794	1.5990	0.8112	5
w/.De-bias	**2.1053**	**1.4936**	**1.5764**	**0.8170**	**7**
MPCN	3.1545	2.7100	1.6718	0.7388	422
w/.De-bias	**3.0293**	**2.5072**	1.6844	**0.7415**	**823**
NARRE	2.1043	1.2952	1.5119	0.8272	11
w/.De-bias	**1.7749**	**1.0028**	**1.4535**	**0.8336**	8
DAML	3.7942	2.1677	1.5250	0.8138	5
w/.De-bias	**3.4675**	**1.7165**	**1.4660**	**0.8186**	**41**
MF	3.5910	1.9455	1.5207	0.8107	7
w/.De-bias	**3.5607**	**1.7825**	**1.4928**	**0.8150**	6
NeuMF	3.6603	2.0143	1.5167	0.8097	8
w/.De-bias	**3.6113**	**1.8427**	**1.4920**	**0.8147**	5

From the results, we can clearly observe that our de-biasing framework is able to consistently eliminate sentiment bias for all RS models on different datasets. Both BU and BI are reduced after de-biasing, showing the effectiveness of our framework in de-biasing RS models. On the other hand, the consistent de-biasing results have proved that our framework is general and can be easily injected into a wide range of RS models regardless of their detailed designs (e.g., shallow or deep, review-based or non-review-based).

5.2 Impacts on Recommendation Quality (RQ2)

In the empirical study (Sec. 3.2), we have observed that sentiment bias is positively correlated with several common recommendation metrics, including MSE, NDCG@K and Diversity@K. An interesting question is whether recommendation qualities can be enhanced after using our de-biasing framework.

Tabs. 2, 3, 4, 5 and 6 also report the results of MSE, NDCG@5 and Diversity@5 of each RS model with/without de-biasing on different datasets. Performance after de-biasing is shown in bold if it is better than the results without de-biasing. We can see that, the proposed framework can universally improve recommendation accuracy, i.e., smaller MSE and higher NDCG@5 are obtained. Furthermore, Diversity@5 is improved most of the time. The results have demonstrated that eliminating sentiment bias using the proposed framework will eventually lead to the improvements of

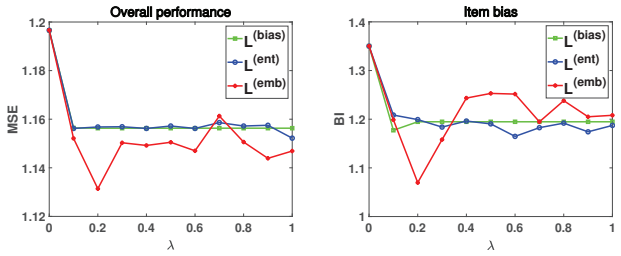

Figure 9: Changes of MSE and item bias with respect to the coefficients on MPCN and Amazon Gourmet Food.

both the recommendation accuracy (MSE and NDCG@K) and the personalization of RS (Diversity@K).

5.3 Contribution of Each Regularizer (RQ3)

We further analyze the role of each regularizer in our framework, by showing the impacts of coefficients λ_1, λ_2 and λ_3. Fig. 9 provides the changes of MSE and item bias (i.e., BI) on MPCN over Amazon Gourmet Food dataset using different coefficients. For better visualizations, we do not change the coefficients simultaneously. Instead, we change one coefficient at a time and fix the other two coefficients as 0. From Fig. 9, we can observe that changes of each coefficient noticeably affect MSE and BI, showing that all regularizers play pivotal roles in our de-biasing framework. Specifically, when only using $\mathcal{L}^{(bias)}$, lowest MSE and BI are both obtained at $\lambda_1 = 0.1$; when using only $\mathcal{L}^{(ent)}$, lowest MSE and BI can be achieved when setting λ_2 to 1.0 and 0.6, respectively; when using only $\mathcal{L}^{(emb)}$, $\lambda_3 = 0.2$ is the best value. It is worth pointing out that the result of grid search when using all three regularizers is $\lambda_1 = 0.25$, $\lambda_2 = 0.05$, $\lambda_3 = 0.05$, as mentioned in Sec. 5.1.

5.4 Visualization

In this subsection, we visualize the impacts of de-biasing via a case study: de-bias DeepCoNN on Amazon Gourmet Food dataset.

We first provide the improvement of recommendation accuracy on different user and item groups. We rank users and items in the test set, by the sentiment polarity values of their profiles, as described in Sec. 3.1. We divide users/items into 10 groups by their sentiment polarity values, and each group contains 10% of the users/items denoted as P_n and N_n where $n = \{1, \cdots, 5\}$. Note that P_1 is the positive user/item group (i.e., $\mathcal{U}^+$ and $\mathcal{I}^+$), and N_1 is the negative user/item group (i.e., $\mathcal{U}^-$ and $\mathcal{I}^-$), which are used in computing sentiment bias of Def. 2. Fig. 10 depicts MSE for different user/item groups with/without de-biasing, We can observe that MSE has been improved for all user/item groups after de-biasing.

We then demonstrate the distributions of ratings on negative items. As shown in Fig. 11, after de-biasing, predicted ratings on negative items spread more evenly across the rating space. It confirms to our expectations in Sec. 4.4: predicted ratings on negative items will become evenly distributed after de-biasing.

Finally, we analyze how negative item embeddings change after de-biasing. We use t-SNE [41] to project negative item embeddings to a two-dimensional space. As shown in Fig. 12, negative item embeddings in the original DeepCoNN are crowded together and do not show enough diversity along the projected y-axis. After applying de-biasing, negative item embeddings are scattered over

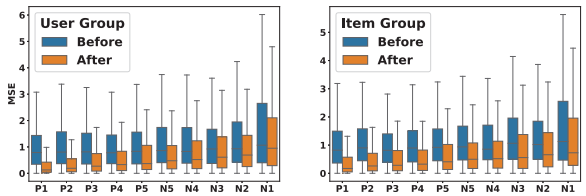

Figure 10: MSE for different user/item groups for DeepCoNN on Amazon Gourmet Food before/after de-biasing.

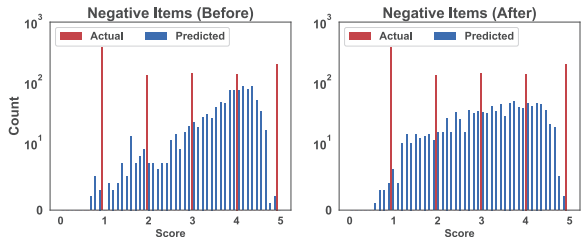

Figure 11: Distributions of actual ratings (integers) and predicted ratings (real numbers) on negative items by DeepCoNN on Amazon Gourmet Food before/after de-biasing.

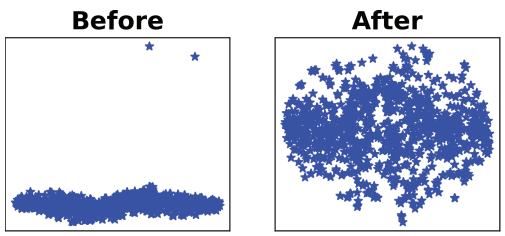

Figure 12: Projection of negative item embeddings in DeepCoNN on Amazon Gourmet Food before/after de-biasing.

the entire space. Considering the changes of both negative item embeddings (Fig. 12) and predicted ratings on negative items (Fig. 11), we can conclude that the de-biasing framework fulfills our design goal in Sec. 4.5, i.e., enforce smaller similarities among negative items so that predicted ratings of negative items will spread over different values instead of crowding around 3 or 4 stars in Fig. 7.

6 CONCLUSION

In this paper, we have revealed the unexplored sentiment bias in RS models and proposed an effective de-biasing framework. Our empirical study has demonstrated the existence and impacts of sentiment bias in RS, and our experiments have verified the effectiveness of our de-biasing framework. For future study, we intend to further investigate sentiment bias based on other types of user-item interactions (e.g., images posted by users to demonstrate the quality of an item), and study the possibility of multimodal de-biasing for RS.

ACKNOWLEDGMENTS

Chen Lin is supported by the Natural Science Foundation of China (No. 61972328) and Joint Innovation Research Program of Fujian Province China (No. 2020R0130). Hui Li is supported by the Natural Science Foundation of China (No. 62002303) and Natural Science Foundation of Fujian Province China (No. 2020J05001).

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling Popularity Bias in Learning-to-Rank Recommendation. In *RecSys*. 42–46.

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing Popularity Bias in Recommender Systems with Personalized Re-Ranking. In *FLAIRS Conference*. 413–418.

[3] Himan Abdollahpouri and Masoud Mansoury. 2020. Multi-sided Exposure Bias in Recommendation. In *IRS2020@KDD*.

[4] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2019. The Unfairness of Popularity Bias in Recommendation. In *RMSE@RecSys*, Vol. 2440.

[5] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2020. The Connection Between Popularity Bias, Calibration, and Fairness in Recommendation. In *RecSys*. 726–731.

[6] Charu C. Aggarwal. 2016. *Recommender Systems - The Textbook*. Springer.

[7] Stephen Bonner and Flavian Vasile. 2018. Causal embeddings for recommendation. In *RecSys*. 104–112.

[8] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *WWW*. 1583–1592.

[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* (2020). https://arxiv.org/abs/2010.03240

[10] Zhihong Chen, Rong Xiao, Chenliang Li, Gangfeng Ye, Haochuan Sun, and Hongbo Deng. 2020. ESAM: Discriminative Domain Adaptation with Non-Displayed Items to Improve Long-Tail Performance. In *SIGIR*. 579–588.

[11] Andrew Collins, Dominika Tkaczyk, Akiko Aizawa, and Jöran Beel. 2018. A Study of Position Bias in Digital Library Recommender Systems. *arXiv Preprint* (2018). https://arxiv.org/abs/1802.06565

[12] Alberto García-Durán, Roberto Gonzalez, Daniel Oñoro-Rubio, Mathias Niepert, and Hui Li. 2020. TransRev: Modeling Reviews as Translations from Users to Items. In *ECIR*, Vol. 12035. 234–248.

[13] David Godes and Jose C. Silva. 2012. Sequential and Temporal Dynamics of Online Opinion. *Marketing Science* 31, 3 (2012), 448–473.

[14] Huifeng Guo, Jinkai Yu, Qing Liu, Ruiming Tang, and Yuzhou Zhang. 2019. PAL: a position-bias aware learning framework for CTR prediction in live recommender systems. In *RecSys*. 452–456.

[15] Ruining He, Wang-Cheng Kang, and Julian J. McAuley. 2017. Translation-based Recommendation. In *RecSys*. 161–169.

[16] Ruining He, Wang-Cheng Kang, and Julian J. McAuley. 2018. Translation-based Recommendation: A Scalable Method for Modeling Sequential Behavior. In *IJCAI*. 5264–5268.

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

[18] Katja Hofmann, Anne Schuth, Alejandro Bellogín, and Maarten de Rijke. 2014. Effects of Position Bias on Click-Based Recommender Evaluation. In *ECIR*, Vol. 8416. 624–630.

[19] Dongmin Hyun, Chanyoung Park, Min-Chul Yang, Ilhyeon Song, Jung-Tae Lee, and Hwanjo Yu. 2018. Review Sentiment-Guided Scalable Deep Recommender System. In *SIGIR*. 965–968.

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

[21] Dominik Kowald, Markus Schedl, and Elisabeth Lex. 2020. The Unfairness of Popularity Bias in Music Recommendation: A Reproducibility Study. In *ECIR*, Vol. 12036. 35–42.

[22] Sanjay Krishnan, Jay Patel, Michael J. Franklin, and Ken Goldberg. 2014. A methodology for learning, analyzing, and mitigating social influence bias in recommender systems. In *RecSys*. 137–144.

[23] Hui Li, Ye Liu, Nikos Mamoulis, and David S. Rosenblum. 2020. Translation-Based Sequential Recommendation for Complex Users on Sparse Data. *IEEE Trans. Knowl. Data Eng.* 32, 8 (2020), 1639–1651.

[24] Dawen Liang, Laurent Charlin, James McInerney, and David M. Blei. 2016. Modeling User Exposure in Recommendation. In *WWW*. 951–961.

[25] Dugang Liu, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2020. A General Knowledge Distillation Framework for Counterfactual Recommendation via Uniform Data. In *SIGIR*. 831–840.

[26] Donghua Liu, Jing Li, Bo Du, Jun Chang, and Rong Gao. 2019. DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation. In *KDD*. 344–352.

[27] Hongtao Liu, Wenjun Wang, Hongyan Xu, Qiyao Peng, and Pengfei Jiao. 2020. Neural Unified Review Recommendation with Cross Attention. In *SIGIR*. 1789–1792.

[28] Yiming Liu, Xuezhi Cao, and Yong Yu. 2016. Are You Influenced by Others When Rating?: Improve Rating Prediction by Conformity Modeling. In *RecSys*. 269–272.

[29] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *WSDM*. 287–296.

[30] Benjamin M. Marlin, Richard S. Zemel, Sam T. Roweis, and Malcolm Slaney. 2007. Collaborative Filtering and the Missing at Random Assumption. In *UAI*. 267–275.

[31] Julian J. McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring Networks of Substitutable and Complementary Products. In *KDD*. 785–794.

[32] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A Survey on Bias and Fairness in Machine Learning. *arXiv Preprint* (2019). https://arxiv.org/abs/1908.09635

[33] Rajiv Pasricha and Julian J. McAuley. 2018. Translation-based factorization machines for sequential recommendation. In *RecSys*. 63–71.

[34] Steffen Rendle, Walid Krichene, Li Zhang, and John R. Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *RecSys*. 240–248.

[35] Noveen Sachdeva and Julian J. McAuley. 2020. How Useful are Reviews for Recommendation? A Critical Review and Potential Improvements. In *SIGIR*. 1845–1848.

[36] Yuta Saito. 2020. Asymmetric Tri-training for Debiasing Missing-Not-At-Random Explicit Feedback. In *SIGIR*. 309–318.

[37] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In *ICML*, Vol. 48. 1670–1679.

[38] Harald Steck. 2010. Training and testing of recommender systems on data missing not at random. In *KDD*. 713–722.

[39] Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth M. Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating Gender Bias in Natural Language Processing: Literature Review. In *ACL*. 1630–1640.

[40] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. In *KDD*. 2309–2318.

[41] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* 9 (2008), 2579–2605.

[42] Mei Wang and Weihong Deng. 2020. Mitigating Bias in Face Recognition Using Skewness-Aware Reinforcement Learning. In *CVPR*. 9319–9328.

[43] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Trans. Knowl. Data Eng.* 29, 12 (2017), 2724–2743.

[44] Ting Wang and Dashun Wang. 2014. Why Amazon's Ratings Might Mislead You: The Story of Herding Effects. *Big Data* 2, 4 (2014), 196–204.

[45] Yixin Wang, Dawen Liang, Laurent Charlin, and David M. Blei. 2018. The Deconfounded Recommender: A Causal Inference Approach to Recommendation. *arXiv Preprint* (2018). https://arxiv.org/abs/1808.06581

[46] Yixin Wang, Dawen Liang, Laurent Charlin, and David M. Blei. 2020. Causal Inference for Recommender Systems. In *RecSys*. 426–431.

[47] Tianxin Wei, Fuli Feng, Jiawei Chen, Chufeng Shi, Ziwei Wu, Jinfeng Yi, and Xiangnan He. 2020. Model-Agnostic Counterfactual Reasoning for Eliminating Popularity Bias in Recommender System. *arXiv Preprint* (2020). https://arxiv.org/abs/2010.15363

[48] Yin Zhang, Yun He, Jianling Wang, and James Caverlee. 2020. Adaptive Hierarchical Translation-based Sequential Recommendation. In *WWW*. 2984–2990.

[49] Hua Zheng, Dong Wang, Qi Zhang, Hang Li, and Tinghao Yang. 2010. Do clicks measure recommendation relevancy?: an empirical user study. In *RecSys*. 249–252.

[50] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *WSDM*. 425–434.