Denoising Implicit Feedback for Recommendation

Wenjie Wang
wenjiewang96@gmail.com
National University of Singapore

Fuli Feng
fulifeng93@gmail.com
National University of Singapore

Xiangnan He
hexn@ustc.edu.cn
University of Science and Technology of China

Liqiang Nie
nieliqiang@gmail.com
Shandong University

Tat-Seng Chua
dcscts@nus.edu.sg
National University of Singapore

ABSTRACT

The ubiquity of implicit feedback makes them the default choice to build online recommender systems. While the large volume of implicit feedback alleviates the data sparsity issue, the downside is that they are not as clean in reflecting the actual satisfaction of users. For example, in E-commerce, a large portion of clicks do not translate to purchases, and many purchases end up with negative reviews. As such, it is of critical importance to account for the inevitable noises in implicit feedback for recommender training. However, little work on recommendation has taken the noisy nature of implicit feedback into consideration.

In this work, we explore the central theme of denoising implicit feedback for recommender training. We find serious negative impacts of noisy implicit feedback, *i.e.,* fitting the noisy data hinders the recommender from learning the actual user preference. Our target is to identify and prune the noisy interactions, so as to improve the efficacy of recommender training. By observing the process of normal recommender training, we find that noisy feedback typically has large loss values in the early stages. Inspired by this observation, we propose a new training strategy named *Adaptive Denoising Training* (ADT), which adaptively prunes noisy interactions during training. Specifically, we devise two paradigms for adaptive loss formulation: *Truncated Loss* that discards the large-loss samples with a dynamic threshold in each iteration; and *Reweighted Loss* that adaptively lowers the weights of large-loss samples. We instantiate the two paradigms on the widely used binary cross-entropy loss and test the proposed ADT strategies on three representative recommenders. Extensive experiments on three benchmarks demonstrate that ADT significantly improves the quality of recommendation over normal training.

CCS CONCEPTS

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

* Corresponding author: Fuli Feng (fulifeng93@gmail.com).

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

Recommender System, False-positive Feedback, Adaptive Denoising Training

1 INTRODUCTION

Recommender systems have been a promising solution for mining user preference over items in various online services such as E-commerce [29], news portals [32] and social media [3]. As the clue to user choices, implicit feedback (*e.g.,* click and purchase) are typically the default choice to train a recommender due to their large volume. However, prior work [18, 32, 39] points out the gap between implicit feedback and the actual user satisfaction due to the prevailing presence of *noisy interactions* (*a.k.a. false-positive interactions*) where the users dislike the interacted item. For instance, in E-commerce, a large portion of purchases end up with negative reviews or being returned. This is because implicit interactions are easily affected by the first impression of users and other factors such as caption bias [4, 17, 33] and position bias [19]. Moreover, existing studies [33, 39] have demonstrated the detrimental effect of such false-positive interactions on user experience of online services. Nevertheless, little work on recommendation has taken the noisy nature of implicit feedback into consideration.

In this work, we argue that such false-positive interactions would hinder a recommender from learning the actual user preference, leading to low-quality recommendations. Table 1 provides empirical evidence on the negative effects of false-positive interactions when we train a competitive recommender, Neural Matrix Factorization (NeuMF) [16], on two real-world datasets. In particular, we construct a "clean" testing set by removing the false-positive interactions for recommender evaluation[1]. As can be seen, training NeuMF with false-positive interactions (*i.e., normal training*) results in an average performance drop of 16.65% and 10.29% over the two datasets *w.r.t.* Recall@20 and NDCG@20, as compared to the NeuMF trained without false-positive interactions (*i.e., clean training*). As such, it is of critical importance to account for the inevitable noises in implicit feedback and eliminate the impact of false-positive interactions for recommender training.

Indeed, some efforts [7, 21, 41] have been dedicated to eliminating the effects of false-positive interactions by: 1) negative experience identification [21, 33] (illustrated in Figure 1(b)); and 2) the incorporation of various feedback [41, 43] (shown in Figure 1(c)). The former processes the implicit feedback in advance by predicting

[1]Each false-positive interaction is identified by auxiliary information of post-interaction behaviors, *e.g.*, rating score ($[1, 5]$) < 3, indicating that the interacted item dissatisfies the user. Refer to Section 2 for more details.

the false-positive ones with additional user behaviors (*e.g.,* dwell time and gaze pattern) and auxiliary item features (*e.g.,* length of the item description) [33]. The latter incorporates extra feedback (*e.g.,* favorite and skip) into recommender training to prune the effects of false-positive interactions [43]. A key limitation with these methods is that they require additional data to perform denoising, which may not be easy to collect. Moreover, extra feedback (*e.g.,* rating and favorite) is often of a smaller scale, which may suffer from the sparsity issue. For instance, many users do not give any feedback after watching a movie or purchasing a product [18].

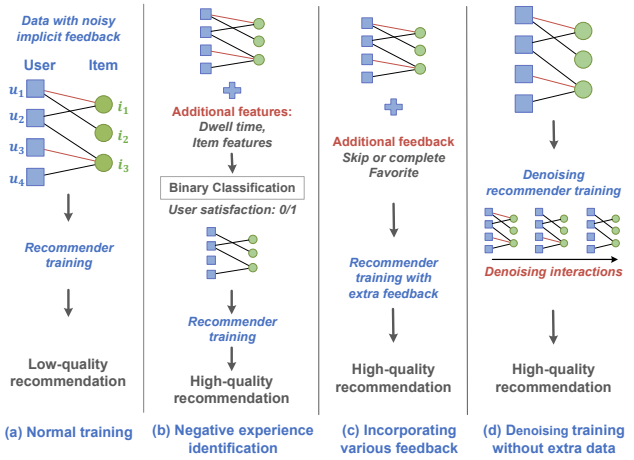


Figure 1: The comparison between normal training (a); two prior solutions to eliminate false-positive interactions through extra data (b) and (c); and denoising training without extra data (d). Note that the red lines in the user-item graph denote false-positive interactions.

This work explores denoising implicit feedback for recommender training, which automatically reduces the influence of false-positive interactions without using any additional data (Figure 1(d)). That is, we only count on the implicit interactions and distill signals of false-positive interactions across different users and items. Prior study on robust learning [13, 20] and curriculum learning [2] demonstrate that noisy samples are relatively harder to fit into models, indicating distinct patterns of noisy samples' loss values in the training procedure. Primary experiments across different recommenders and datasets (*e.g.,* Figure 3) reveals similar phenomenon: the loss values of false-positive interactions are larger than those of the true-positive ones in the early stages of training, while their loss values decrease to the same range at the end. Consequently, due to the larger loss, false-positive interactions can largely mislead the recommender training in the early stages. Worse still, the recommender ultimately fits the false-positive interactions due to its high representation capacity, which could be overfitting and hurt the generalization. As such, a potential idea of denoising is to reduce the impact of false-positive interactions, *e.g.,* pruning the interactions with large loss values, where the key challenge is to simultaneously decrease the sacrifice of true-positive interactions.

To this end, we propose Adaptive Denoising Training (ADT) strategies for recommenders, which dynamically prunes the large-loss interactions along the training process. To avoid the lost of generality, we focus only on formulating the training loss, which can be applied to any differentiable models. In detail, we devise two paradigms to formulate the training loss: 1) *Truncated Loss*, which discards the large-loss interactions dynamically, and 2) *Reweighted Loss*, which adaptively reweighs the interactions. For each training iteration, the Truncated Loss removes the large-loss samples (*i.e.,* hard samples) with a dynamic threshold which is automatically updated during training. Besides, the Reweighted Loss dynamically assigns "harder" interactions with smaller weights to weaken their effects on the optimization. We instantiate the two loss functions on the basis of the widely used binary cross-entropy loss. On three benchmarks, we test ADT trained with the Truncated Loss or Reweighted Loss over three representative recommenders: Generalized Matrix Factorization (GMF) [16], NeuMF [16], and Collaborative Denoising Auto-Encoder (CDAE) [40]. The results show significant performance improvements of ADT over normal training.

Table 1: Performance comparison between the clean training and normal training of NeuMF on Adressa and Amazon-book. #Drop denotes the relative performance drop of normal training as compared to clean training.

Dataset	Adressa		Amazon-book	
Metric	Recall@20	NDCG@20	Recall@20	NDCG@20
Clean training	0.4040	0.1963	0.0293	0.0159
Normal training	0.3081	0.1732	0.0265	0.0145
#Drop	23.74%	11.77%	9.56%	8.81%

Our main contributions are summarized as:

- We formulate the task of denoising implicit feedback for recommender training. We find the negative effect of false-positive interactions and identify their characteristics (*i.e.,* hard samples) during training.
- We propose Adaptive Denoising Training to prune the large-loss interactions dynamically, which introduces two paradigms to formulate the training loss: Truncated Loss and Reweighted Loss.
- We instantiate two paradigms on the binary cross-entropy loss and apply ADT to three representative recommenders. Extensive experiments on three benchmarks validate the effectiveness of ADT in improving the recommendation quality.

2 STUDY ON FALSE-POSITIVE FEEDBACK

The effect of noisy training samples has been studied in conventional machine learning tasks such as image classification [13, 20]. However, little attention has been paid to such effect on recommendation, which is inherently different from conventional tasks with training samples highly related to each other, *e.g.,* interactions on the same item. We investigate the effects of false-positive interactions on recommender training by comparing the performance of recommenders trained with and without false-positive interactions. An interaction is identified as false-positive or true-positive one according to the explicit feedback. For instance, a purchase is false-positive if the following rating score ($[1, 5]$) < 3. Although the size of such explicit feedback is typically insufficient for building robust recommenders in real-world scenarios, the scale is sufficient for a pilot experiment. In detail, we train a competitive recommender model NeuMF under two different settings: 1) "clean training" which trains NeuMF on the true-positive interactions only;

and 2) "normal training" which trains NeuMF on all observed user-item interactions. We evaluate the recommendation performance on the holdout clean testing set with only true-positive interactions kept, *i.e.,* the evaluation focuses on recommending more satisfying items to users. More details can be seen in Section 5.

Results. Table 1 summarizes the performance of NeuMF under normal training and clean training *w.r.t.* Recall@20 and NDCG@20 on the two representative datasets, Adressa and Amazon-book. From Table 1, we can observe that, as compared to the ideal setting, *i.e.,* clean training, the performance of normal training drops by 11.77% and 8.8% *w.r.t.* NDCG@20 on Adressa and Amazon-book, respectively. This result shows the *negative effects* of false-positive interactions on recommending satisfying items to users. Worse still, recommendations from normal training have higher risk on leading to further false-positive interactions, which would hurt the user experience [33]. Despite the success of clean training in the pilot study, it is not a reasonable choice in practical applications because of the sparsity issues of reliable feedback such as rating scores. As such, it is worth exploring denoising implicit feedback such as click, view, or buy for recommender training.

3 METHOD

In this section, we detail the proposed Adaptive Denoising Training strategy for recommenders. Prior to that, task formulation and observations that inspire the strategy design are introduced.

3.1 Task Formulation

Generally, the target of recommender training is to learn user preference from user feedback, *i.e.,* learning a scoring function $\hat{y}_{ui} = f(u, i|\Theta)$ to assess the preference of user u over item i with parameters Θ. Ideally, the setting of recommender training is to learn Θ from a set of reliable feedback between N users ($\mathcal{U}$) and M items ($\mathcal{I}$). That is, given $\mathcal{D}^* = \{(u, i, y^*_{ui})|u \in \mathcal{U}, i \in \mathcal{I}\}$, we learn the parameters Θ^* by minimizing a recommendation loss over $\mathcal{D}^*$, *e.g.,* the binary Cross-Entropy (CE) loss:

$$\mathcal{L}_{CE}\left(\mathcal{D}^*\right) = -\sum_{(u,i,y^*_{ui})\in\mathcal{D}^*} y^*_{ui}\log\left(\hat{y}_{ui}\right) + \left(1 - y^*_{ui}\right)\log\left(1 - \hat{y}_{ui}\right),$$

where $y^*_{ui} \in \{0, 1\}$ represents whether the user u really prefers the item i. The recommender with Θ^* would be reliable to generate high-quality recommendations. In practice, due to the lack of large-scale reliable feedback, recommender training is typically formalized as: $\bar{\Theta} = \min \mathcal{L}_{CE}(\bar{\mathcal{D}})$, where $\bar{\mathcal{D}} = \{(u, i, \bar{y}_{ui})|u \in \mathcal{U}, i \in \mathcal{I}\}$ is a set of implicit interactions. $\bar{y}_{ui}$ denotes whether the user u has interacted with the item i implicitly, such as click and purchase.

However, due to the existence of noisy interactions which would mislead the learning of user preference, the typical recommender training might result in a poor model (*i.e.,* $\bar{\Theta}$) that lacks generalization ability on the clean testing set. As such, we formulate a *denoising recommender training* task as:

$$\Theta^* = \min \mathcal{L}_{CE}\big(denoise(\bar{\mathcal{D}})\big), \tag{1}$$

aiming to learn a reliable recommender with parameters Θ^* by denoising implicit feedback, *i.e.,* pruning the impact of noisy interactions. Formally, by assuming the existence of inconsistency between y^*_{ui} and $\bar{y}_{ui}$, we define noisy interactions as $\left\{(u, i)|y^*_{ui} = 0 \wedge \bar{y}_{ui} = 1\right\}$. According to the value of y^*_{ui} and $\bar{y}_{ui}$, we can separate implicit feedback into four categories similar to a confusion matrix as shown in Figure 2.

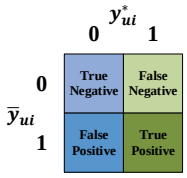
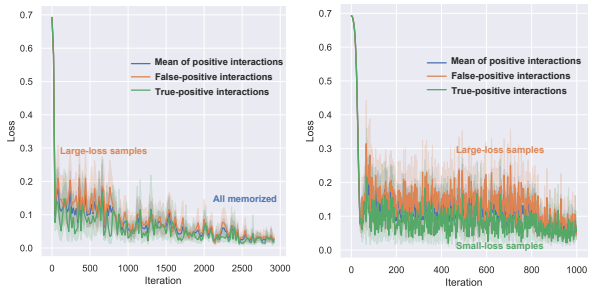


Figure 2: Four types of implicit interactions.

In this work, we focus on denoising false-positive interactions and omit the false-negative ones since positive interactions are much fewer in recommendation and thus false-positive interactions would induce worse effects on recommender training. Note that we do not incorporate any additional data such as explicit feedback or reliable implicit feedback into the task of denoising, despite their success in a few applications [33, 39]. This is because such feedback is of a smaller scale in most cases, suffering more severely from the sparsity issue.



(a) Whole training process (b) Early training stages

Figure 3: The training loss of true- and false-positive interactions on Adressa in the normal training of NeuMF.

3.2 Observations

False-positive interactions are harder to fit in the early stages. In robust learning [13, 20] and curriculum learning [2], one theory is that easy samples are more likely to be the clean ones and fitting the hard samples may hurt the generalization. To explore whether it also exists in recommendation, we conduct experiments by training NeuMF with all observed implicit interactions (*i.e.,* normal training) on Adressa and Amazon-book. The loss of true- and false-positive interactions in Adressa is visualized in Figure 3. Note that similar trends are also found over other recommenders and datasets (see more details in Section 5.2.1). From Figure 3, we observe that:

- Ultimately, the loss of both of true- and false-positive interactions converges to a stable state with close values, which implies that NeuMF fits both of them well. It reflects that deep models with substantial capacity would "memorize" all the training data, including the noisy samples. As such, if the data is noisy, the memorization will lead to poor generalization performance.
- In the early stages of training, the loss values of true- and false-positive interactions decrease differently. Furthermore, we zoom in to visualize the changes of the loss *w.r.t.* iterations ranging from 0 to 1,000 in Figure 3(b). From the figure, we can see that the loss of false-positive interactions is clearly larger than that of the true-positive ones, which indicates that false-positive interactions are

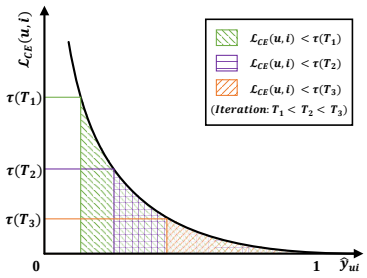


Figure 4: Illustration of T-CE loss for the observed interactions (*i.e.,* $\bar{y}_{ui} = 1$). T_i is the iteration number and $\tau(T_i)$ refers to the threshold. Dash area indicates the effective loss and the loss values larger than $\tau(T_i)$ are truncated.

harder to memorize than the true-positive ones in the early stages. The reason might be that false-positive ones represent the items that the user dislikes, and they are more similar to the items that the user didn't interact with (*i.e.,* the negative samples). The findings also support the prior theory in robust learning and curriculum learning [2, 13].

Overall, the results are consistent with the memorization effect [1]: deep models will first learn the easy and clean patterns in the early stage, and eventually memorize all training samples [13].

3.3 Adaptive Denoising Training

Based on the observations, we propose ADT strategies for recommenders, which estimate $P(y^*_{ui} = 0|\bar{y}_{ui} = 1, u, i)$ according to the training loss. To reduce the impact of false-positive interactions, ADT dynamically prunes the large-loss interactions during training. In particular, ADT either *discards* or *reweighs* the interactions with large loss values to reduce their influences on the training objective. Towards this end, we devise two paradigms to formulate loss functions for denoising training:

- *Truncated Loss.* This is to truncate the loss values of large-loss interactions to 0 with a dynamic threshold function.
- *Reweighted Loss.* It adaptively assigns hard samples (*i.e.,* the large-loss ones) with smaller weights during training.

Note that the two paradigms formulate various recommendation loss functions, *e.g.,* CE loss, square loss [35], and BPR loss [34]. In the work, we take CE loss as an example to elaborate them.

3.3.1 ***Truncated Cross-Entropy Loss.*** Functionally speaking, the Truncated Cross-Entropy (shorted as T-CE) loss discards positive interactions with large values of CE loss. Formally, we can define it as:

$$\mathcal{L}_{T\text{-}CE}(u,i) = \begin{cases} 0, & \mathcal{L}_{CE}(u,i) > \tau \wedge \bar{y}_{ui} = 1 \\ \mathcal{L}_{CE}(u,i), & \text{otherwise,} \end{cases} \tag{2}$$

where τ is a pre-defined threshold. The T-CE loss removes any positive interactions with CE loss larger than τ from the training. While this simple T-CE loss is easy to interpret and implement, the fixed threshold may not work properly. This is because the loss value is decreasing with the increase of training iterations. Inspired by the dynamic gradient descent methods [22], we replace the fixed threshold with a dynamic threshold function $\tau(T)$ *w.r.t.* the training iteration T, which changes the threshold value along the training process (Figure 4). Besides, since the loss values vary across different datasets, it would be more flexible to devise $\tau(T)$ as a function of the drop rate $\epsilon(T)$. Note that there is a bijection between the drop rate and the threshold, *i.e.,* for any training iteration, if the drop rate is given, we can calculate the threshold to filter out samples.

Algorithm 1 Adaptive Denoising Training with T-CE loss

Input: the set of all trainable parameters Θ, the training set of observed implicit interactions $\bar{\mathcal{D}}$, the maximum number of iterations T_{max}, learning rate η, ϵ_{max}, α, $\mathcal{L}_{CE}$

1: **for** $T = 1 \rightarrow T_{max}$ **do** ▷ shuffle samples every epoch
2: **Fetch** mini-batch data $\bar{\mathcal{D}}_{pos}$ from $\bar{\mathcal{D}}$
3: **Sample** unobserved interactions $\bar{\mathcal{D}}_{neg}$ randomly for users in $\bar{\mathcal{D}}_{pos}$ with the proportion of 1:1
4: **Define** $\bar{\mathcal{D}}_T = \bar{\mathcal{D}}_{pos} \cup \bar{\mathcal{D}}_{neg}$
5: **Obtain** $\hat{\mathcal{D}} = \underset{\hat{\mathcal{D}} \in \bar{\mathcal{D}}_{pos}, |\hat{\mathcal{D}}| = \epsilon(T)|\bar{\mathcal{D}}_T|}{\arg\max} \sum_{(u,i)\in\hat{\mathcal{D}}} \mathcal{L}_{CE}(u,i|\Theta_{T\text{-}1})$
6: **Update** $\Theta_T = \Theta_{T\text{-}1} - \eta \nabla \frac{1}{|\hat{\mathcal{D}}|} \sum_{u,i\in\hat{\mathcal{D}}} \mathcal{L}_{CE}(u,i|\Theta_{T\text{-}1})$
7: **Update** $\epsilon(T) = min(\alpha T, \epsilon_{max})$
8: **end for**

Output: the optimized parameters $\Theta_{T_{max}}$ of the recommender

Based on prior observations, a proper drop rate function should have the following properties: 1) $\epsilon(\cdot)$ should have an upper bound to limit the proportion of discarded samples so as to prevent data missing; 2) $\epsilon(0) = 0$, *i.e.,* it should allow all the samples to be fed into the models in the beginning; and 3) $\epsilon(\cdot)$ should increase smoothly from zero to its upper bound, so that the model can learn and distinguish the true- and false-positive interactions gradually.

Towards this end, we formulate the drop rate function as:

$$\epsilon(T) = min(\alpha T, \epsilon_{max}), \tag{3}$$

where ϵ_{max} is an upper bound and α is a hyper-parameter to adjust the pace to reach the maximum drop rate. Note that we increase the drop rate in a linear fashion rather than a more complex function such as a polynomial function or a logarithm function. Despite the expressiveness of these functions, they will inevitably increase the number of hyper-parameters, resulting in the increasing cost of tuning a recommender. The whole algorithm is explained in Algorithm 1. Note that T-CE loss discards the hard samples which are more likely to be the noisy ones. It is symmetrically contrary to the Hinge loss, and T-CE loss limits the model to be overfitting.

3.3.2 ***Reweighted Cross-Entropy Loss.*** Functionally speaking, the Reweighted Cross-Entropy (shorted as R-CE) loss down-weights the positive interactions with large loss values, which is defined as:

$$\mathcal{L}_{R\text{-}CE}(u,i) = \omega(u,i)\,\mathcal{L}_{CE}(u,i), \tag{4}$$

where $\omega(u,i)$ is a weight function that adjusts the contribution of an observed interaction to the training objective. To achieve the target of properly down-weighting the large-loss samples, the weight function $\omega(u,i)$ is expected to have the following properties: 1) it dynamically adjusts the weights of samples during training; 2) the function will reduce the influence of a hard sample to be weaker than an easy sample; and 3) the degree of weight reduction can be easily adjusted so that it can fit different models and datasets.

Inspired by the success of Focal Loss [30], we estimate $\omega(u,i)$ with a function of $f(\hat{y}_{ui})$ that takes the prediction score as the

input. Note that the prediction score and CE loss are equivalent to identify hard samples (*i.e.,* the large-loss ones). We use the prediction score as the input of the weight function since its value is within $[0, 1]$ rather than $[0, +\infty]$, which is more accountable to further computation. Towards this end, we formulate it as:

$$f(\hat{y}_{ui}) = \hat{y}_{ui}^{\beta}, \tag{5}$$

where $\beta \in [0, +\infty]$ is a hyper-parameter to control the range of weights. From Figure 5(a), we can see that R-CE loss equipped with the proposed weight function can significantly reduce the loss of hard samples (*i.e.,* $\hat{y}_{ui} \ll 0.5$) as compared to the original CE loss. Furthermore, the proposed weight function satisfies the aforementioned requirements:

- $f(\hat{y}_{ui}) = \hat{y}_{ui}^{\beta}$ is sensitive to $\hat{y}_{ui}$ which is closely related to the loss value. As such, it generates dynamic weights during training.
- The interactions with extremely large CE loss (*e.g.,* the "outlier" in Figure 5(b)) will be assigned with very small weights because $\hat{y}_{ui}$ is close to 0. Therefore, the influence of such large-loss samples is largely reduced. In addition, as shown in Figure 5(b), harder samples always have smaller weights because the function $f(\hat{y}_{ui})$ monotonically increases when $\hat{y}_{ui} \in [0, 1]$ and $\beta \in [0, +\infty]$. As such, it can avoid that false-positive interactions with large loss values dominate the optimization during training [42].
- The hyper-parameter β dynamically controls the gap between the weights of hard and easy samples. By observing the examples in Figure 5(b), we can find that: 1) if β increases, for the same pair of easy and hard samples, the gap between their weights becomes larger (*e.g.,* $d_{0.4} < d_{1.0}$ in Figure 5(b)); and 2) if we set β as 0, the R-CE loss will degrade to the standard CE loss.

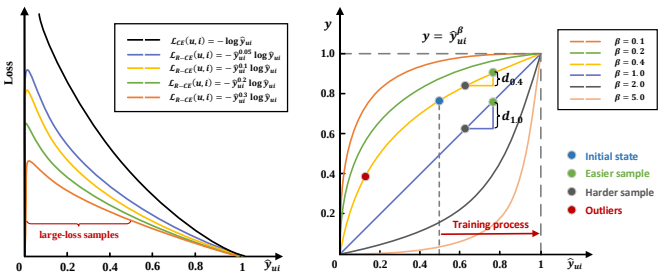
In practice, to ensure the loss values of all samples are within the same range, preventing negative samples with large loss values from dominating the optimization, negative samples are also weighted in this paradigm. Formally, we revise the weight function as:

$$\omega(u, i) = \begin{cases} \hat{y}_{ui}^{\beta}, & \bar{y}_{ui} = 1 \\ (1 - \hat{y}_{ui})^{\beta}, & \text{otherwise}, \end{cases} \tag{6}$$

Indeed, it may provide a possible solution to alleviate the impact of false-negative interactions, which is left for future exploration.

3.3.3 ***In-depth Analysis.*** Since ADT depends totally on recommenders to identify false-positive interactions, one question might be whether it is reliable. Actually, many existing work [13, 20] has pointed out the connection between the large loss and noisy samples, and explained the underlying causality: the "memorization" effect of deep models. That is, deep models will first learn easy and clean patterns in the initial training phase, and then gradually memorize all samples, including noisy ones. As such, the loss of deep models in the early stage can help to filter out noisy interactions. We discuss the memorization effect of recommenders by experiments in Section 3.2 and 5.2.1. And the performance of T-CE loss also shows that it can is explored in Section 5.2.2.

Another concern is that some hard samples may be more informative than easy samples and discarding hard samples would limit the model's learning ability. Indeed, as indicated in the prior studies [2], hard samples in the noisy data probably confuse the model rather than help it to establish the right decision surface. As such, they may induce poor generalization. It's actually a trade-off between denoising and learning. In ADT, the $\epsilon(\cdot)$ of T-CE loss and β of R-CE loss are to control the balance. And the sensitivity to the hyper-parameters is studied in Section 5.2.3.



(a) R-CE loss for the observed positive interactions. The contributions of large-loss samples are greatly reduced.

(b) The weight function with different parameters β, where β controls the weight difference between hard and easy samples.

Figure 5: Illustration and analysis of R-CE loss.

4 RELATED WORK

This work aims to denoise implicit feedback for recommenders, which is highly related to the negative experience identification, incorporating various feedback, and the robustness of recommenders.

Negative Experience Identification. To reduce the gap between implicit feedback and the actual user preference, many researchers have paid attention to identify negative experiences in implicit signals [7, 21]. Prior work usually collects the various users' feedback (*e.g.,* dwell time [21], gaze patterns [46], and skip [7]) and the item characteristics [32, 33] to predict the user's satisfaction. Lu *et al.* [32] predicted users' actual preference in news recommendation based on various user behaviors, news quality, and the interaction context. However, these methods need additional feedback and extensive manual label work, *e.g.,* users have to tell if they are satisfied for each interaction. Besides, the quantification of item quality and characteristics is non-trivial [32], which largely relies on the manually feature design and the labeling of domain experts [32, 33]. The unaffordable labor cost hinders the practical usage of these methods, especially in the scenarios with constantly changing items.

Incorporating Various Feedback. To alleviate the impact of false-positive interactions, previous approaches [8, 26, 31, 41, 44] also consider incorporating more feedback (*e.g.,* dwell time [43], skip [27, 45], and adding to favorites) into training directly. For instance, Wen *et al.* [39] proposed to train the recommender using three kinds of items: "click-complete", "click-skip", and "non-click" ones. The last two kinds of items are both treated as negative samples but with different weights. However, additional feedback might be unavailable in complex scenarios. For example, we cannot acquire dwell time and skip patterns after users buy products or watch movies in a cinema. Most users even don't give any informative feedback after clicks. In an orthogonal direction, this work explores denoising implicit feedback without additional information during training.

Robustness of Recommender Systems. Gunawardana *et al.* [12] defined the robustness of recommender systems as "the stability of the recommendation in the presence of fake information". Prior work [25, 36] has tried to evaluate the

Table 2: Statistics of the datasets. In particular, #FP interactions refer to the number of false-positive interactions.

Dataset	#User	#Item	#Interaction	#FP Interaction
Adressa	212,231	6,596	419,491	247,628
Amazon-book	80,464	98,663	2,714,021	199,475
Yelp	45,548	57,396	1,672,520	260,581

robustness of recommender systems under various attack methods, such as shilling attacks [25] and fuzzing attacks [36]. To build more robust recommender systems, some auto-encoder based models [28, 37, 40] introduce the denoising techniques. These approaches (*e.g.,* CDAE [40]) first corrupt the interactions of user by random noises, and then try to reconstruct the original one with auto-encoders. However, these methods focus on heuristic attacks or random noises, and ignore the natural false-positive interactions in data. This work highlights the negative impact of natural noisy interactions, and improve the robustness against them.

5 EXPERIMENT

Dataset. To evaluate the effectiveness of the proposed ADT on recommender training, we conducted experiments on three publicly accessible datasets: Adressa, Amazon-book, and Yelp.

- **Adressa:** This is a real-world news reading dataset from Adressavisen[2] [11]. It includes user clicks over news and the dwell time for each click, where the clicks with dwell time < 10s are thought of as false-positive ones [21, 43].
- **Amazon-book:** It is from the Amazon-review datasets[3] [14]. It covers users' purchases over books with rating scores. A rating score below 3 is regarded as a false-positive interaction.
- **Yelp:** It's an open recommendation dataset[4], in which businesses in the catering industry (*e.g.,* restaurants and bars) are reviewed by users. Similar to Amazon-book, the rating scores below 3 are regarded as false-positive feedback.

These datasets comprise the common implicit feedback: click, purchase, and consumption, which are suitable to explore the effectiveness of denoising implicit feedback although explicit feedback also exists in each interaction. We followed former work [15, 28, 38] to remove users and items with extremely sparse interactions and split the dataset into training, validation, and testing (see Table 2 for statistics). To evaluate the effectiveness of denoising implicit feedback, we kept all interactions, including the false-positive ones, in training and validation, and tested recommenders only on true-positive interactions. That is, the models are expected to recommend more satisfying items to users.

Evaluation Protocols. For each user in the testing set, we predicted the preference score over all the items except the positive ones used during training. Following existing studies [16, 38], we reported the recommendation performance *w.r.t.* two widely used metrics: Recall@K and NDCG@K, where higher scores indicate better performance. For both metrics, we set K as 50 and 100 for Amazon-book and Yelp, while 3 and 20 for Adressa due to its much smaller item space.

Testing Recommenders. To demonstrate the effectiveness of our proposed ADT strategy on denoising implicit feedback, we compared the performance of recommenders trained with T-CE or R-CE and normal training with standard CE. We selected two representative user-based neural CF models, GMF and NeuMF [16], and one item-based model, CDAE [40]. Note that CDAE is also a representative model of robust recommender which can defend random noises within implicit feedback.

- **GMF [16]:** This is a generalized version of matrix factorization by replacing the inner product with the element-wise product and a linear neural layer as the interaction function.
- **NeuMF [16]:** NeuMF is a representative CF neural model, which models the relationship between users and items by combining GMF and a Multi-Layer Perceptron (MLP).
- **CDAE [40]:** CDAE corrupts the interactions with random noises, and then employs a MLP model to reconstruct the original input.

We only tested neural recommenders and omit conventional ones such as MF [24] and SVD++ [23] due to their inferior performance [16, 40].

Parameter Settings. For the three testing recommenders, we followed their default settings, and verified the effectiveness of our methods under the same conditions. For GMF and NeuMF, the factor numbers of users and items are both 32. As to CDAE, the hidden size of MLP is set as 200. In addition, the batch size is always 1,024 and Adam [22] is applied to optimize all the parameters with the learning rate initialized as 0.001. As to the ADT strategies, they have three hyper-parameters in total: α and ϵ_{max} in T-CE loss, and β in R-CE loss. In detail, ϵ_{max} is searched in $\{0.05, 0.1, ..., 0.5\}$ and β is tuned in $\{0.05, 0.1, ..., 0.25, 0.5, 1.0\}$. As for α, we controlled its range by adjusting the iteration number ϵ_N to the maximum drop rate ϵ_{max}, and ϵ_N is adjusted in $\{1k, 5k, 10k, 20k, 30k\}$.

5.1 Performance Comparison

Table 3 summarizes the recommendation performance comparison of the three testing models trained with standard CE, T-CE, or R-CE over three datasets. From Table 3, we can observe:

- In all cases, both the T-CE loss and R-CE loss effectively improve the performance, *e.g.,* NeuMF+T-CE outperforms vanilla NeuMF by 12.98% on average over three datasets. The significant performance gain indicates the better generalization ability of neural recommenders trained by T-CE loss and R-CE loss. It validates the effectiveness of adaptive denoising training, *i.e.,* discarding or down-weighting hard samples during training.
- By comparing the T-CE Loss and R-CE Loss, we found that the T-CE loss performs better in most cases. We postulate that the recommender still suffers from the false-positive interactions when it is trained with the Reweighted Loss, even though they have smaller weights and contribute little to the overall training loss. In addition, we suspect that the superior performance of the Truncated Loss could be attributed to the additional hyper-parameters in the dynamic threshold function which can be tuned more granularly. Further improvement might be achieved by a finer-grained user-specific or item-specific tuning of these parameters, which can be done automatically [5].
- Across the recommenders, NeuMF performs worse than GMF and CDAE, especially on Amazon-book and Yelp, which is criticized for the vulnerability to noisy interactions. Because our testing is only on the true-positive interactions, the inferior performance

[2] https://www.adressa.no/
[3] http://jmcauley.ucsd.edu/data/amazon/
[4] https://www.yelp.com/dataset/challenge

Table 3: Overall performance of three testing recommenders trained with ADT strategies and normal training over three datasets. Note that Recall@K and NDCG@K are shorted as R@K and N@K to save space, respectively, and "RI" in the last column denotes the relative improvement of ADT over normal training on average. The best results are highlighted in bold.

Dataset	Adressa				Amazon-book				Yelp				
Metric	R@3	R@20	N@3	N@20	R@50	R@100	N@50	N@100	R@50	R@100	N@50	N@100	RI
GMF	0.0880	0.2141	0.0780	0.1237	0.0610	0.0953	0.0252	0.0328	0.0830	0.1344	0.0348	0.0463	-
GMF+T-CE	**0.0904**	**0.2210**	**0.0805**	**0.1275**	**0.0707**	**0.1113**	**0.0292**	**0.0382**	**0.0871**	**0.1437**	0.0359	**0.0486**	8.10%
GMF+R-CE	0.0890	0.2152	0.0788	0.1248	0.0682	0.1075	0.0275	0.0362	0.0860	0.1363	**0.0366**	0.0480	5.13%
NeuMF	0.1094	0.3081	0.0947	0.1732	0.0509	0.0813	0.0210	0.0279	0.0771	0.1259	0.0317	0.0427	-
NeuMF+T-CE	0.1416	0.3158	0.1267	0.1885	0.0600	0.0972	0.0240	0.0323	**0.0800**	**0.1314**	**0.0325**	**0.0440**	12.98%
NeuMF+R-CE	**0.1416**	**0.3172**	**0.1267**	**0.1900**	**0.0628**	**0.1028**	**0.0248**	**0.0334**	0.0788	0.1304	0.0320	0.0436	14.36%
CDAE	0.1394	0.3208	0.1168	0.1808	0.0989	0.1507	0.0414	0.0527	0.1112	0.1732	0.0471	0.0611	-
CDAE+T-CE	**0.1406**	**0.3220**	0.1176	**0.1839**	**0.1088**	**0.1645**	**0.0454**	**0.0575**	**0.1165**	**0.1806**	**0.0504**	**0.0652**	5.36%
CDAE+R-CE	0.1388	0.3164	**0.1200**	0.1827	0.1022	0.1560	0.0424	0.0542	0.1161	0.1801	0.0488	0.0632	2.46%

of NeuMF is reasonable since NeuMF with more parameters can fit more false-positive interactions during training.

- Both T-CE and R-CE achieve the biggest performance increase on NeuMF, which validates the effectiveness of ADT to prevent vulnerable models from the disturbance of noisy data. On the contrary, the improvement over CDAE is relatively small, showing that the design of defending random noise can also improve the robustness against false-positive interactions to some extend. Nevertheless, applying T-CE or R-CE still leads to performance gain, which further validates the rationality of denoising implicit feedback.

In the following, GMF is taken as an example to conduct thorough investigation for the consideration of computation cost.

Further Comparison against Using Additional Feedback. To avoid the detrimental effect of false-positive interactions, a popular idea is to incorporate the additional user feedback for training although they are usually sparse. Existing work either adopts the additional feedback by multi-task learning [9, 10], or leverages it to identify the true-positive interactions [32, 39]. In this work, we introduce two classical models for comparison: Neural Multi-Task Recommendation (NMTR) [10] and Negative feedback Re-weighting (NR) [39]. In particular, NMTR with multi-task learning is to capture multiple user behaviors (*i.e.,* click and satisfaction) while NR uses the addition feedback (*i.e.,* dwell time and rating) to identify true-positive interactions with user satisfaction and re-weight the false-positive and non-interacted ones as negative samples. We applied NMTR and NR on the testing recommenders and reported the results of GMF in Table 4. The results of other recommenders with similar trends are omitted to save space.

From Table 4, we can find that: 1) NMTR and NR achieve better performance than GMF, which validates the effectiveness of additional feedback; and 2) the results of NMTR and NR are inferior to that of ADT, both T-CE and R-CE. This is attributed to the sparsity of additional feedback. Indeed, the clicks with satisfaction is much fewer than the total number of clicks, and thus NR will lose extensive positive training samples. Besides, not all clicks without labeled user satisfaction indicate users' dislikes because many users seldom give explicit feedback even if they are satisfied. Therefore, treating them as negative samples will hurt the performance, which is also found by the experiments in [6].

Table 4: Performance *w.r.t.* GMF on Amazon-book.

Metric	R@50	R@100	N@50	N@100
GMF	0.0600	0.0945	0.0247	0.0324
GMF+T-CE	**0.0707**	**0.1113**	**0.0292**	**0.0382**
GMF+R-CE	0.0682	0.1075	0.0275	0.0362
GMF+NMTR	0.0616	0.0967	0.0254	0.0332
GMF+NR	0.0615	0.0958	0.0254	0.0331

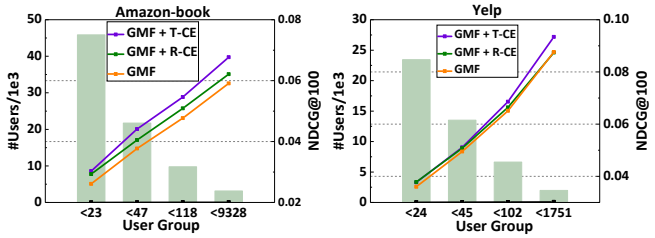
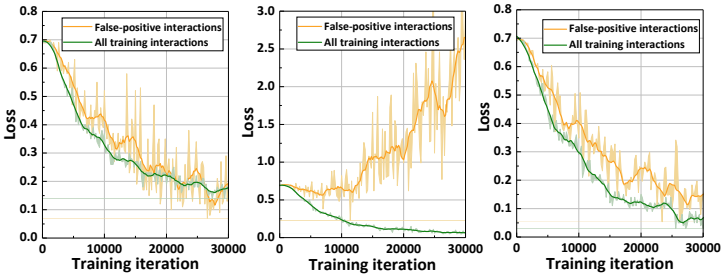


Figure 6: Performance comparison of GMF over user groups with different sparsity levels on Amazon-book and Yelp. The histograms represent the user number in each group and the lines denote the performance *w.r.t.* NDCG@100.



(a) Normal Training (b) Truncated Loss (c) Reweighted Loss

Figure 7: Loss of GMF (a), GMF+T-CE (b) and GMF+R-CE (c).

Performance Comparison w.r.t. Interaction Sparsity. Since ADT prunes many interactions during training, we explored whether ADT hurts the preference learning of inactive users because their interacted items are sparse. Following the former studies [38], we split testing users into four groups according to the interaction number of each user where each group has the same number of interactions. Figure 6 shows the group-wise performance comparison where we can observe that the proposed ADT strategies achieve stable performance gain over normal training in all cases. It validates that ADT is also effective for the inactive users.

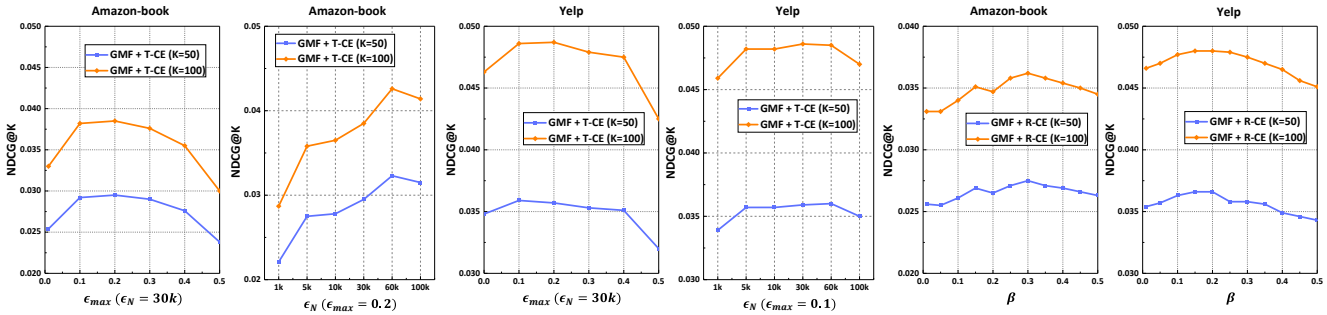


Figure 8: Performance comparison of GMF trained with ADT on Yelp and Amazon-book *w.r.t.* different hyper-parameters.

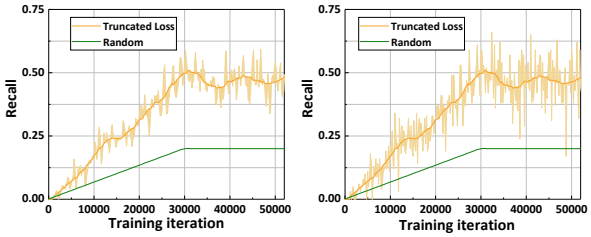


Figure 9: Recall and precision of false-positive interactions over GMF trained with the Truncated Loss on Amazon-book.

5.2 In-depth Analysis

5.2.1 Memorization of False-positive Interactions. Recall that false-positive interactions are memorized by recommenders eventually under normal training, leading to poor generalization (cf. Section 3.2). We then investigated whether false-positive interactions are also fitted well by the recommenders trained with ADT strategies. Considering that the original CE loss values indicates the model's fitting ability on the samples, we depicted the CE loss of false-positive interactions during the training procedure with the real training loss as reference.

From Figure 7(a), we can find that the observations in section 3.2 also exist in the training of GMF on Amazon-book. The loss values of false-positive interactions eventually become similar to other samples, indicating that GMF fits false-positive samples well at last. On the contrary, as shown in Figure 7(b), by applying T-CE, the loss values of false-positive interactions keep increasing while the overall training loss stably decreases step by step. The increased loss indicates that the recommender parameters are not optimized over the false-positive interactions, validating the capability of T-CE to identify and discard such interactions. As to R-CE (Figure 7(c)), the loss of false-positive interactions also shows a decreasing trend, showing that the recommender still fits such interactions. However, their loss values are still larger than the real training loss, indicating that the false-positive interactions are assigned with smaller weights by R-CE, which prevents the model from fitting them. Therefore, we can conclude that both paradigms reduce the effect of false-positive interactions on recommender training, which can explain their improvement over normal training.

5.2.2 Study of Truncated Loss. Since the Truncated Loss achieves promising performance in the experiments, we studied how well it performs to identify and discard false-positive interactions. We first defined Recall to represent what percentage of false-positive interactions in the training data are discarded, and precision as the ratio of discarded false-positive interactions to all discarded samples. Figure 9 visualizes the changes of the recall and precision along the training process. The green line in Figure 9 indicates the recall and precision under the settings of random discarding. In particular, the recall of random discarding equals the drop rate during training while its precision is the proportion of noisy interactions in all training samples at each iteration.

From Figure 9, we observed that: 1) the Truncated Loss discards nearly half of false-positive interactions after the drop rate keeps stable, greatly reducing the impact of noisy interactions; and 2) the precision of Truncated Loss is about twice as large as that of random discarding. It demonstrates that the Truncated Loss effectively utilizes the distill signals of false-positive interactions and weakens their contributions to the model training. In spite of this, we can find that a key limitation of the Truncated Loss is the low precision, *e.g.,* only 10% precision in Figure 9, which implies that it inevitably discards many clean interactions. This also partly proves that it's worth pruning noisy interactions at the cost of losing many clean samples. And the hyper-parameters in ADT control the trade-off between denoising and losing clean samples. Besides, how to further improve the precision so as to decrease the loss of clean samples is a promising research direction in the future.

5.2.3 Hyper-parameter Sensitivity. Our proposed ADT strategies incorporate three hyper-parameters to adjust the dynamic threshold function and the weight function in two paradigms. In particular, ϵ_{max} and ϵ_N are used to control the drop rate in the Truncated Loss, and β adjusts the weight function in the Reweighted Loss. In this section, we studied how the hyper-parameters affect the performance. Only the results of GMF trained with ADT strategies on Amazon-book and Yelp are reported in Figure 8 due to space limitation. Other methods over three datasets have similar patterns. From Figure 8, we can find that: 1) the recommender trained with the T-CE loss performs better when $\epsilon_{max} \in [0.1, 0.3]$. If ϵ_{max} exceeds 0.4, the performance drops significantly because a large proportion of samples are discarded. Therefore, the upper bound ϵ_{max} should be restricted. 2) The recommender is relatively sensitive to ϵ_N, especially on Amazon-book, and the performance still increases when $\epsilon_N >$ 30k. Nevertheless, a limitation of T-CE loss is the big search space of hyper-parameters. 3) The adjustment of β in the Reweighted Loss is consistent over different datasets, and the best results happen when β ranges from 0.15 to 0.3. These observations provide insights on how to tune the hyper-parameters of ADT if it's applied to other recommenders and datasets.

6 CONCLUSION AND FUTURE WORK

In this work, we aim to denoise implicit feedback for recommender training. We explore the negative effects of noisy implicit feedback, and propose Adaptive Denoising Training strategies to reduce their impact. In particular, this work contributes two paradigms to formulate the loss functions: Truncated Loss and Reweighted Loss. Both paradigms are general and can be applied to different loss functions, neural recommenders, and optimizers. In this work, we applied the two paradigms on the widely used binary cross-entropy loss and conduct extensive experiments over three recommenders on three datasets, showing that the paradigms effectively reduce the disturbance of noisy implicit feedback.

This work takes the first step to denoise implicit feedback for recommendation without using additional feedback for training, and points to some new research directions. Specifically, it is interesting to explore how the proposed two paradigms perform on other loss functions, such as Square Loss [35], Hinge Loss [35] and BPR Loss [34]. Besides, how to further improve the precision of the paradigms is worth studying. Lastly, our Adaptive Denoising Training is not specific to the recommendation task, and it can be widely used to denoise implicit interactions in other domains, such as Web search and question answering.

ACKNOWLEDGMENTS

This research is supported by the National Research Foundation, Singapore under its International Research Centres in Singapore Funding Initiative, and the National Natural Science Foundation of China (61972372, U19A2079). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. 2017. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 233–242.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 41–48.

[3] Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. 2019. An Efficient Adaptive Transfer Neural Network for Social-Aware Recommendation. In *Proceedings of the 42nd International SIGIR Conference on Research and Development in Information Retrieval*. ACM, 225–234.

[4] 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).

[5] Yihong Chen, Bei Chen, Xiangnan He, Chen Gao, Yong Li, Jian-Guang Lou, and Yue Wang. 2019. lambdaOpt: Learn to Regularize Recommender Models in Finer Levels. In *Proceedings of the 25th SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 978–986.

[6] Jingtao Ding, Guanghui Yu, Xiangnan He, Fuli Feng, Yong Li, and Depeng Jin. 2019. Sampler design for bayesian personalized ranking by leveraging view data. *Transactions on Knowledge and Data Engineering* (2019).

[7] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. 2005. Evaluating Implicit Measures to Improve Web Search. *Transactions on Information Systems* 23, 2 (2005), 147–168.

[8] Evgeny Frolov and Ivan Oseledets. 2016. Fifty Shades of Ratings: How to Benefit from a Negative Feedback in Top-N Recommendations Tasks. In *Proceedings of the 10th Conference on Recommender Systems*. ACM, 91–98.

[9] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019. Neural multi-task recommendation from multi-behavior data. In *Proceedings of the 35th International Conference on Data Engineering*. IEEE, 1554–1557.

[10] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yueming Li, Tat-Seng Chua, Lina Yao, Yang Song, and Depeng Jin. 2019. Learning to Recommend with Multiple Cascading Behaviors. *Transactions on Knowledge and Data Engineering* (2019).

[11] Jon Atle Gulla, Lemei Zhang, Peng Liu, Özlem Özgöbek, and Xiaomeng Su. 2017. The Adressa Dataset for News Recommendation. In *Proceedings of the International Conference on Web Intelligence*. ACM, 1042–1048.

[12] Asela Gunawardana and Guy Shani. 2015. Evaluating Recommender Systems. In *Recommender Systems handbook*. Springer, 265–308.

[13] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. MIT Press, Curran Associates Inc., 8527–8537.

[14] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web*. IW3C2, 507–517.

[15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 639–648.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web*. IW3C2, 173–182.

[17] Katja Hofmann, Fritz Behr, and Filip Radlinski. 2012. On Caption Bias in Interleaving Experiments. In *Proceedings of the 21st International Conference on Information and Knowledge Management*. ACM, 115–124.

[18] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th International Conference on Data Mining*. TEEE, 263–272.

[19] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions. In *Proceedings of the 42nd International SIGIR Conference on Research and Development in Information Retrieval*. ACM, 15–24.

[20] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2304–2313.

[21] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. 2014. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th International Conference on Web Search and Data Mining*. ACM, 193–202.

[22] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[23] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 426–434.

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

[25] Shyong K. Lam and John Riedl. 2004. Shilling Recommender Systems for Fun and Profit. In *Proceedings of the 13th International Conference on World Wide Web*. IW3C2, 393–402.

[26] Gal Lavee, Noam Koenigstein, and Oren Barkan. 2019. When Actions Speak Louder Than Clicks: A Combined Model of Purchase Probability and Long-term Customer Satisfaction. In *Proceedings of the 13th Conference on Recommender Systems*. ACM, 287–295.

[27] Ruirui Li, Liangda Li, Xian Wu, Yunhong Zhou, and Wei Wang. 2019. Click Feedback-Aware Query Recommendation Using Adversarial Examples. In *Proceedings of the 28th International Conference on World Wide Web*. IW3C2, 2978–2984.

[28] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 27th International Conference on World Wide Web*. IW3C2, 689–698.

[29] Tzu-Heng Lin, Chen Gao, and Yong Li. 2019. CROSS: Cross-Platform Recommendation for Social E-Commerce. In *Proceedings of the 42nd International SIGIR Conference on Research and Development in Information Retrieval*. ACM, 515–524.

[30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the International Conference on Computer Vision*. IEEE, 2980–2988.

[31] Chao Liu, Ryen W White, and Susan Dumais. 2010. Understanding web browsing behaviors through Weibull analysis of dwell time. In *Proceedings of the 33rd international SIGIR Conference on Research and development in information retrieval*. ACM, 379–386.

[32] Hongyu Lu, Min Zhang, and Shaoping Ma. 2018. Between Clicks and Satisfaction: Study on Multi-Phase User Preferences and Satisfaction for Online News Reading. In *Proceedings of the 41st International SIGIR Conference on Research and Development in Information Retrieval*. ACM, 435–444.

[33] Hongyu Lu, Min Zhang, Weizhi Ma, Ce Wang, Feng xia, Yiqun Liu, Leyu Lin, and Shaoping Ma. 2019. Effects of User Negative Experience in Mobile News Streaming. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 705–714.

[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.

[35] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. 2004. Are Loss Functions All the Same? *Neural Computation*. 16, 5 (2004), 1063–1076.

[36] David Shriver, Sebastian G. Elbaum, Matthew B. Dwyer, and David S. Rosenblum. 2019. Evaluating Recommender System Stability with Influence-Guided Fuzzing. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. AAAI Press, 4934–4942.

[37] Florian Strub and Jeremie Mary. 2015. Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs. In *Workshop of Neural Information Processing Systems*. MIT Press.

[38] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International SIGIR Conference on Research and Development in Information Retrieval*. ACM, 165–174.

[39] Hongyi Wen, Longqi Yang, and Deborah Estrin. 2019. Leveraging Post-click Feedback for Content Recommendations. In *Proceedings of the 13th Conference on Recommender Systems*. ACM, 278–286.

[40] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n Recommender Systems. In *Proceedings of the 9th International Conference on Web Search and Data Mining*. ACM, 153–162.

[41] Byoungju Yang, Sangkeun Lee, Sungchan Park, and Sang goo Lee. 2012. Exploiting Various Implicit Feedback for Collaborative Filtering. In *Proceedings of the 21st International Conference on World Wide Web*. IW3C2, 639–640.

[42] Peng Yang, Peilin Zhao, Vincent W. Zheng, Lizhong Ding, and Xin Gao. 2018. Robust Asymmetric Recommendation via Min-Max Optimization. In *The 41st International SIGIR Conference on Research & Development in Information Retrieval*. ACM, 1077–1080.

[43] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: dwell time for personalization. In *Proceedings of the 8th Conference on Recommender Systems*. ACM, 113–120.

[44] Peifeng Yin, Ping Luo, Wang-Chien Lee, and Min Wang. 2013. Silence is Also Evidence: Interpreting Dwell Time for Recommendation from Psychological Perspective. In *Proceedings of the 19th SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 989–997.

[45] Aston Zhang, Amit Goyal, Weize Kong, Hongbo Deng, Anlei Dong, Yi Chang, Carl A Gunter, and Jiawei Han. 2015. adaqac: Adaptive query auto-completion via implicit negative feedback. In *Proceedings of the 38th International SIGIR Conference on Research and Development in Information Retrieval*. ACM, 143–152.

[46] Qian Zhao, Shuo Chang, F. Maxwell Harper, and Joseph A. Konstan. 2016. Gaze Prediction for Recommender Systems. In *Proceedings of the 10th Conference on Recommender Systems*. ACM, 131–138.