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
xiangnanhe@gmail.com
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 prevents the recommender from learning the actual user preference. Our target is to identify and prune the noisy interactions, so as to improve the quality 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 weight 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*.

KEYWORDS

Recommender System, False-positive Feedback, Denoising Recommendation, Adaptive Training

1 INTRODUCTION

Recommender systems have been a promising solution for mining user preference over items in various online services such as E-commerce [21], news portals [24] and social media [2]. As clues to user choices, implicit feedback (*e.g.,* click and purchase) are typically the default choice to train a recommender due to their large volume. Unfortunately, prior work [10, 24, 32] has pointed out the gap between implicit feedback and the actual satisfaction of users due to the common existence 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 even returns. This is because implicit interactions are easily affected by the first impression of users and other factors such as caption bias [24, 25] and position bias [11]. Moreover, existing studies [25, 32] have demonstrated the detrimental effect of such false-positive interactions on user experience of online services. However, 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 mislead 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 where we train a competitive recommender, Neural Matrix Factorization (NeuMF) [9], 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 research efforts [3, 13, 34] have been dedicated to eliminating the effects of false-positive interactions by 1) negative experience identification [13, 25] (illustrated in Figure 1(b)) and 2) the incorporation of various feedback [34, 36] (shown in Figure 1(c)). The former could process the implicit feedback in advance by predicting 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) [25]. The latter incorporates extra feedback (*e.g.,* favorite and skip) into recommender training to prune the effects of false-positive interactions [36]. 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 of a smaller scale, which may suffer more severely from the sparsity issue. For instance, many

[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.

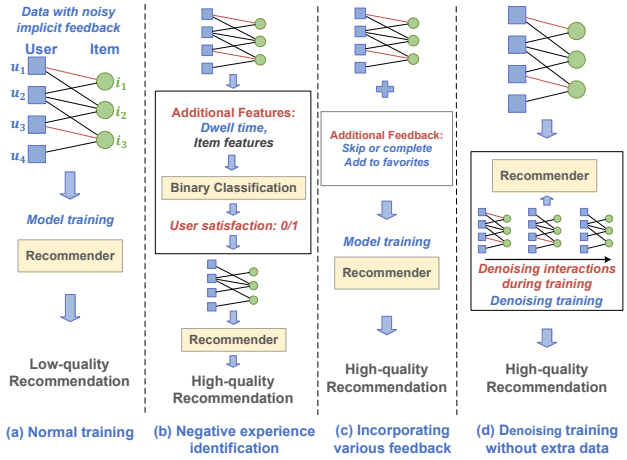


Figure 1: The comparison between normal training (a); two prior solutions to eliminate false-positive interactions through extra data (b) and (c); and denoising training without extra data (d). Note that the red lines in the user-item graph denote false-positive interactions.

users do not give any feedback after watching a movie or purchasing a product [10].

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[2]. By observing the process of normal training over various recommenders trained on different datasets (*e.g.*, Figure 3), we investigate the properties of false-positive interactions and their effects on the recommender. An important observation is that 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. These results indicate that: 1) false-positive interactions are "harder" to fit for the recommender, largely misleading the training objective in the early stages. One reason is that false-positive interactions represent the items which users actually dislike so that they are similar to the negative samples; and 2) the recommender ultimately fits the false-positive interactions due to its high capacity, which could be overfitting. To reduce the impact of false-positive interactions, a potential idea of denoising is to adaptively prune the interactions with large loss values along the training process. Considering that the training process is affected by many factors, including model, dataset, and initialization, we aim to devise a general training strategy to perform the denoising (*i.e.,* pruning) appropriately.

Towards this end, we propose Adaptive Denoising Training (ADT) strategies for recommenders, which dynamically prunes the large-loss interactions along the training process. To avoid losing generality, we revise only the way of formulating the loss. In particular, 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 ignores the large-loss samples (*i.e.,* hard samples) with a dynamic threshold which is automatically updated during training. In addition, we devise the Reweighted Loss which dynamically assigns "harder" interactions with smaller weights to weaken their effects on model optimization. We implement the two loss functions on the basis of the widely used binary cross-entropy loss. On three benchmarks, we test ADT equipped with the Truncated Loss or Reweighted Loss over three representative recommenders: Generalized Matrix Factorization (GMF) [9], NeuMF [9], and Collaborative Denoising Auto-Encoder (CDAE) [33]. The results show significant performance improvements of ADT over normal training. Our codes and data will be publicly available upon acceptance.

[2]Note that we assume false-positive interactions are the minority of the data. Otherwise, the data is not suitable for learning a recommender.

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:

- To our knowledge, this is the first work of denoising implicit feedback for recommender training without auxiliary data. We find the negative effect of false-positive interactions and identify their characteristics (*i.e.,* hard samples) during training.
- We propose Adaptive Denoising Training, which dynamically prunes the large-loss interactions. Specifically, we devise 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 recommendation quality.

2 STUDY ON FALSE-POSITIVE FEEDBACK

In this section, we investigate the effects of false-positive interactions on recommender training. Suppose that we are able to identify false-positive interactions by reliable explicit feedback from users, we will be able to construct a "clean" testing set and compare the performance of recommenders trained with and without false-positive interactions. Fortunately, such reliable user feedback (*e.g.,* dwell time or rating scores) are available in a few applications, providing an opportunity for us to study the influence of false-positive interactions.

Dataset. We conduct experiments on two datasets (see more details in Section 5):

- **Adressa**: This is a news reading dataset [5] that contains the dwell time of user click on news articles. Based on the experience of former researchers [13, 36], we identify the clicks with dwell time shorter than 10 seconds as false-positive interactions.
- **Amazon-book**: This is a product recommendation dataset [8] which contains purchase history with rating score ranges from 1 to 5 (5 indicates the best). We intuitively treat the user-item pairs with rating scores below 3 as false-positive interactions.

Settings. 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 follow the all-ranking protocol [31] to evaluate the recommendation performance on the holdout clean testing set and report Recall@20 and NDCG@20 (see the details of protocols in Section 5).

Results. Table 1 summarizes the recommendation performance of NeuMF under normal training and clean training. From Table 1, we can observe that, as compared to the ideal setting, *i.e.,* clean training, the performance of full 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 training recommenders. Despite the success of clean training on the experimental datasets, it is not a reasonable choice in practical applications since 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 the 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 recommender's parameters Θ^* by minimizing a recommendation loss over $\mathcal{D}^*$ such as the binary Cross-Entropy (CE) loss:

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

Wherein, $y^*_{ui} \in \{0, 1\}$ 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 reliable feedback in a large volume, 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 implicit interactions are observed for the user u and item i.

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

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

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

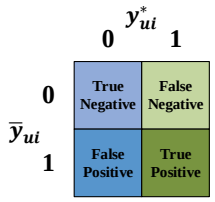
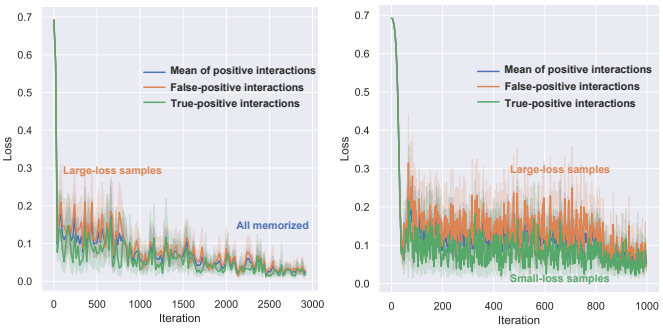


Figure 2: Illustration of four different types of implicit interactions according to the value of user satisfaction (y^*_{ui}) and implicit feedback ($\bar{y}_{ui}$).

we can separate implicit feedback into four categories similar to a confusion matrix as shown in Figure 2. In this work, we focus on denoising false-positive interactions and omit the false-negative ones since positive interactions are more sparse in the recommendation and thus false-positive interactions would induce worse effects on recommender training. Note that we don't incorporate any additional data such as explicit feedback or reliable implicit feedback into the task of denoising recommender training, despite their success in estimating $P(y^*_{ui} = 0|\bar{y}_{ui} = 1, u, i)$ (*i.e.,* denoise) for a few applications [25, 32]. 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 trend of loss over true- and false-positive interactions in Adressa during the normal training of NeuMF.

3.2 Observations

False-positive interactions are harder to fit in the early stages. We conduct experiments by training NeuMF with all observed implicit interactions (*i.e.,* normal training) on Adressa and Amazon-book. The loss changes of true- and false-positive interactions in Adressa are visualized in Figure 3. Note that similar trends are also found over other recommenders trained on Amazon-book (see details in Section 5.2.1). From Figure 3, we have the following observations:

- Ultimately, the loss of both of true-positive and false-positive interactions converges to a stable state with close values, which implies that NeuMF fits both of them well. This result reflects the memorization effect of the cutting-edge recommender based on deep neural networks. That is, deep models with substantial capacity would "memorize" all the training data, including the noisy samples with wrong labels [1, 40]. As such, if the training data is noisy, the memorization effect will lead to poor generalization performance.

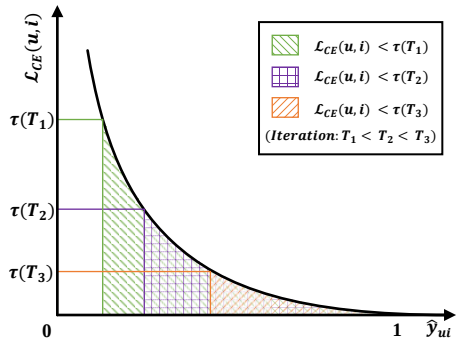


Figure 4: Illustration of T-CE loss for the observed user-item interactions (*i.e.,* samples labeled with $\bar{y}_{ui} = 1$). T_i denotes the iteration number and $\tau(T_i)$ refers to the threshold function. Note that the dash area indicates the effective loss and the loss values larger than $\tau(T_i)$ are truncated.

- In the early stages of the training process, 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 find that the loss of false-positive interactions is clearly larger than that of the true-positive ones. The smaller loss of true-positive interactions implies that they have relatively easier patterns and can be well memorized earlier. Intuitively, true-positive interactions reflect the actual user preference and users are more certain to interact with these items. On the contrary, false-positive interactions with larger loss values are harder to memorize than the true-positive ones in the early stages of training. The reason might be that false-positive ones represent the items that the user dislikes, practically. And they are probably more similar to the items that the user didn't interact with (*i.e.,* the negative samples). Because the false-positive interactions are assigned with wrong positive labels, their loss would be larger than the clean samples. Similar conclusions about noisy samples are also found in the computer vision domain [7, 12, 38], further demonstrating that the phenomenon that noisy samples are harder to fit in the early training stages isn't model-specific, data-specific, or even domain-specific.

3.3 Adaptive Denoising Training

Based on the key observation that *false-positive interactions have larger loss values in the early training stages*, we propose ADT strategies for recommenders, which estimates $P(y^*_{ui} = 0|\bar{y}_{ui} = 1, u, i)$ 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 so as to lower 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 loss functions based on a conventional recommendation loss (*e.g.,* CE loss, square loss [28], and BPR loss [27]). In the work, we take CE loss as an example to elaborate the two paradigms.

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 Θ^* of the recommender

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. To satisfy the basic requirement, we can formally define it as:

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

where τ is a pre-defined threshold. The T-CE loss removes any positive interactions with CE loss larger than τ from the optimization of recommender parameters. While this simple T-CE loss is easy to interpret and implement, the fixed threshold may not work properly in the whole training process. This is because the loss value is decreasing with the increase of training iterations. To be adaptive to the overall trend of training loss, we can 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. In addition, since loss values vary across different datasets, it would be more flexible to devise it as a function of the drop rate $\epsilon(T)$. Note that there is a bijection between the drop rate and the truncation threshold, *i.e.,* for any training iteration, once the drop rate is given, we can figure out the threshold to filter out samples.

Based on the prior observations, a proper drop rate function should have the following properties:
- $\epsilon(\cdot)$ should have an upper bound to limit the proportion of discarded samples so as to prevent data missing.
- $\epsilon(0) = 0$, *i.e.,* it should allow all the samples to be fed into the models in the beginning.
- $\epsilon(\cdot)$ 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:

$$\epsilon(T) = min(\alpha T, \epsilon_{max}), \tag{2}$$

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

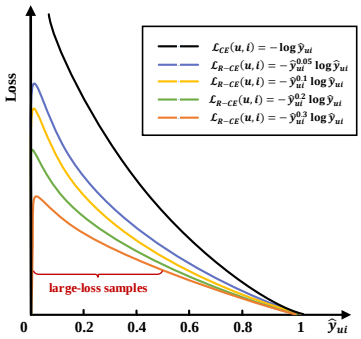


Figure 5: Illustration of R-CE loss for the observed positive interactions. Note that the contributions of large-loss samples are greatly reduced.

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, increasing the cost of tuning a recommender. The whole algorithm is explained in Algorithm 1.

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:

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

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:
- The function should dynamically adjust weights of samples during training.
- The function should down-weight the influences of a hard sample (*i.e.,* large-loss interactions) to be weaker than a easy sample.
- 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 [22], we estimate $\omega(u,i)$ with a function of $f(\hat{y}_{ui})$ that takes the prediction score as input. Note that the prediction score and CE loss are equivalent as used to identify hard samples (*i.e.,* the large-loss ones). We use the prediction score as input of the weight function since its value is within $[0,1]$ rather than $[0,+\infty]$, which is friendly to further computation. Towards this end, we formulate it as:

$$f(\hat{y}_{ui}) = \hat{y}_{ui}^{\beta}, \tag{3}$$

where $\beta \in [0,+\infty]$ is a hyper-parameter to control the range of weights. From Figure 5, 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 could generate dynamic weights along the training process.
- The interactions with extremely large CE loss (*e.g.,* the "outlier" in Figure 6) will be assigned with very small weights because $\hat{y}_{ui}$ is close to 0. Therefore, the influences of such large-loss samples are largely reduced. In addition, as shown in Figure 6, harder samples always have smaller weights, while both hard and easy samples are down-weighted. This is because the function $f(\hat{y}_{ui}) = \hat{y}_{ui}^{\beta}$ monotonically increases when $\hat{y}_{ui} \in [0,1]$ and $\beta \in [0,+\infty]$. Note that the prediction scores of harder samples are smaller than those of the easy ones. Therefore, it can avoid that false-positive interactions with large loss values dominate the optimization during training [35].

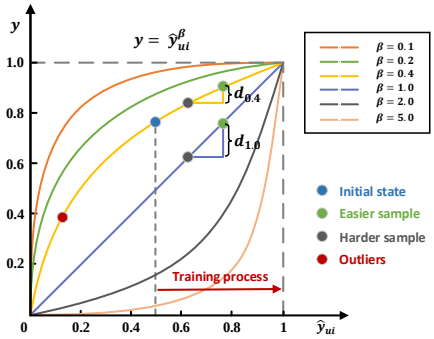


Figure 6: The weight function with different parameters β. $\hat{y}_{ui}$ changes from 0.5 to 1 gradually along training process if $\bar{y}_{ui} = 1$ and β controls the weight difference between hard and easy samples.

- The hyper-parameter β dynamically controls the gap between the weights of hard and easy interactions during the training process. Figure 6 plots the weight function $f(\hat{y}_{ui}) = \hat{y}_{ui}^{\beta}$ under various settings of β. Note that during the training, the prediction score $\hat{y}_{ui}$ fluctuates around 0.5 in the initial iterations, and then gradually moves to 1. According to the value of $\hat{y}_{ui}$, we intuitively plot four cases for initial states, easier sample, harder sample, and outlier. From Figure 6, we can find that: 1) Along this process, the corresponding weight $\hat{y}_{ui}^{\beta}$ increases. 2) If the value of β is increased, 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 6). Besides, 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:

$$\omega(u,i) = \begin{cases} \hat{y}_{ui}^{\beta}, & \bar{y}_{ui} = 1 \\ (1-\hat{y}_{ui})^{\beta}, & \text{otherwise}, \end{cases} \tag{4}$$

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

Moreover, it is worth noting that $\hat{y}_{ui}^{\beta}$ in the weight function is regarded as a constant when computing the gradients to update model parameters (Θ). Otherwise, it will mislead the optimization direction. Formally, we compute the gradient of R-CE loss *w.r.t.* Θ as follows,

$$\frac{\partial \mathcal{L}_{R\text{-}CE}}{\partial \Theta} = \frac{\partial \mathcal{L}_{R\text{-}CE}}{\partial \hat{y}_{ui}} \frac{\partial \hat{y}_{ui}}{\partial \Theta} = -\hat{y}_{ui}^{\beta} \frac{1}{\hat{y}_{ui}} \frac{\partial \hat{y}_{ui}}{\partial \Theta} = \hat{y}_{ui}^{\beta} \left(\frac{\partial \mathcal{L}_{CE}}{\partial \hat{y}_{ui}} \frac{\partial \hat{y}_{ui}}{\partial \Theta}\right).$$

Note that R-CE only introduces one extra hyper-parameters (β).

4 RELATED WORK

In this work, we aim to improve the anti-noise capability of the recommendation methods. Therefore, this work is highly related to the study of false-positive interactions, negative experience identification, the incorporation of various feedback, and the robustness of recommender systems.

4.1 False-positive Interaction

In recent years, implicit feedback (*e.g.*, click and watch) has been widely used as indications of user preference in recommendation systems [9] since the collection of explicit feedback is time-consuming and possibly expensive [3, 10, 18, 42]. Even though implicit feedback is somehow correlated with user preference, there exist a large proportion of noisy interactions in the implicit signals [10, 13]. Many research studies have pointed out that implicit signals, especially the clicks, are easily affected by different factors, such as the position bias [11] and the caption bias [24]. Therefore, there is actually a gap between the implicit interaction signals and the actual user preference in various scenarios, such as news recommendation [24, 25], music recommendation [32, 34], and micro-video recommendation [32]. For example, Lu *et al.* [24] found that more than half of the click signals don't capture the actual user preference in the news recommendation. Users might be attracted by the title of news while he doesn't like its content after the click. More importantly, negative experiences are detrimental to users' following behaviors and overall satisfaction [25].

4.2 Negative Experience Identification

To reduce the gap between implicit feedback and the actual user preference, many researchers have paid attention to the identification of negative experiences in the implicit signals [3, 13, 24, 25]. Prior work usually collects the various users' feedback (*e.g.*, dwell time [13], gaze patterns [41], skip [3], and scroll intervals [24]) and the characteristics of the recommended items [24, 25] to predict the user's satisfaction. Lu *et al.* [24] focused on predicting users' actual preference in the news recommendation with the help of various user behaviors, news quality, and interaction context. More studies on identifying negative experiences from implicit feedback are under the web search scenario. For instance, Fox *et al.* [3] utilized multiple implicit users' behaviors (*e.g.*, clicks, dwell time, skip or completion) to predict user satisfaction in web search. In addition to the dwell time, Kim *et al.* [13] incorporated the features of web pages to train a click-level satisfaction model, such as topics, readability levels, and content lengths. However, a big problem in the negative experience identification is that it needs various feedback and extensive manual label work. Users have to tell whether they are satisfied or not for each interaction. In addition, the quality and characteristics of the recommended items are hard to measure. Existing work usually designs features manually and labels the item quality with the help of domain experts [24, 25].

4.3 Incorporating Various Feedback

To alleviate the impact of false-positive interactions, previous approaches [23, 32, 34, 36, 37] also consider incorporating more feedback (*e.g.*, dwell time, adding to favorites) to mitigate the gap between clicks and the user preference. For example, Yang *et al.* [34] explored various positive and negative implicit feedback, such as play completion and skip, and then incorporated a new rating function with weighted feedback into the recommendation. Besides, dwell time has been considered to be well correlated with the post-click user preference [18, 36]. Yi *et al.* [36] explored how to normalize dwell time, and then used them to train collaborative filtering methods. Moreover, some approaches try to learn from negative feedback [4, 19, 32, 39]. Many researchers [19, 39] utilized the non-clicked query candidates as the negative samples explicitly. For instance, Wen *et al.* [32] 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, implicit feedback is easily affected by various factors [11, 24], such as the characteristics of items and the interaction context. In addition, additional feedback might be unavailable in complex scenarios. For example, we cannot acquire dwell time and skip patterns when movies or products are recommended to users. Most users even don't give any other informative feedback in these scenarios. Therefore, in this work, we train the models with noisy implicit feedback, and focus on denoising implicit feedback without additional information.

4.4 Robustness of Recommender systems

Gunawardana *et al.* [6] defined the robustness of recommender systems as âĂIJthe stability of the recommendation in the presence of fake informationâĂİ. Prior work [17, 29] has tried to evaluate the robustness of recommender systems under various attack methods, such as shilling attacks [17] and fuzzing attacks [29]. To build more robust recommender systems, some auto-encoder based models [20, 30, 33] introduce the denoising techniques. These approaches [33] first corrupt the input, *i.e.,* the preference vector of users by different noises, and then try to reconstruct the original input with auto-encoders. However, existing work usually considers the heuristic attacks or random noises, totally ignoring the natural existence of false-positive interactions in implicit feedback. The objective of this work is to verify their negative effect, and improve the robustness of neural recommenders to noisy implicit feedback.

5 EXPERIMENT

Dataset. To evaluate the effectiveness of our proposed ADT strategy on existing recommenders, we conducted experiments on three publicly accessible datasets: Adressa, Amazon-book, Yelp. The users and items with extremely sparse interactions are removed to ensure the data quality based on the experience of former work [20, 31]. In addition, we first filtered out the false-positive interactions in the testing set based on reliable users' feedback, and then trained the recommendation models with the noisy implicit feedback. Ultimately we evaluated their performance on the clean testing set. Note that these three datasets comprise the common implicit feedback: click, purchase, and consumption so that they are suitable for us to explore the effectiveness of denoising implicit feedback. More statistics about the three datasets are summarized in Table 2.

- **Adressa:** This is a real-world news reading dataset from Adressavisen[3] [5]. It includes anonymous users and their

[3] https://www.adressa.no/

Table 2: Statistics of the datasets. In particular, FP interactions refer to false-positive ones in implicit feedback.

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

clicked news. Besides, dwell time is recorded for each user-item interaction. Based on the existing work [13, 36], the interactions whose dwell time is less than 10 seconds are thought of as false-positive ones.

- **Amazon-book:** Amazon-book is selected from the repository of Amazon-review datasets[4] [8]. This dataset also covers users' ratings for their purchased books.
- **Yelp:** It's an open recommendation dataset[5], 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.

For Adressa, we split the user-item interactions into the training set, validation set, and testing set according to the ratio of 8:1:1 in chronological order due to the timeliness of news. As to the other two datasets, we randomly split the observed interactions of each user according to the same ratio. Ultimately, we removed all the false-positive interactions in the testing set to ensure the reliability of the evaluation.

Evaluation Protocols. For each user in the testing set, we predicted the preference score over all the items except the positive samples used during training. All the items that the user never interacts with are regarded as negative ones. Following the existing studies [9, 31], we adopted two widely used objective metrics to evaluate the performance of the recommendation models: Recall@K and NDCG@K. We reported the averaged value of all users in the testing set *w.r.t.* Recall and NDCG. For both of them, higher scores indicate better performance. Considering that the item number of Adressa is much smaller than those of Amazon-book and Yelp, we set K as 50 and 100 by default while K is 3 or 20 to evaluate the performance on Adressa.

Testing Recommenders. To demonstrate the effectiveness of our proposed ADT strategy on denoising implicit feedback, we equipped several neural models with two paradigms, and then explore if they achieve better generalization performance on the clean testing set. We chose GMF, NeuMF, and CDAE as our testing recommenders because GMF and NeuMF are representative collaborative filtering methods, and CDAE incorporates random noises to reduce the disturbance of noisy feedback.

- **GMF:** 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:** 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:** CDAE corrupts the observed interactions with random noises, and then employs a MLP model to reconstruct the original interactions, partly increasing its anti-noise capability.

Note that we only test neural recommenders and omit conventional recommenders such as MF [16], PMF [26], and SVD++ [15] since recent work has validated the advantages of neural recommenders over conventional recommenders.

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 [14] is applied to optimize all the parameters with the learning rate initialized as 0.001. As to our proposed ADT strategies, they have three hyper-parameters in total: α and ϵ_{max} in T-CE loss, and β in R-CE loss. In particular, 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 Overall Performance

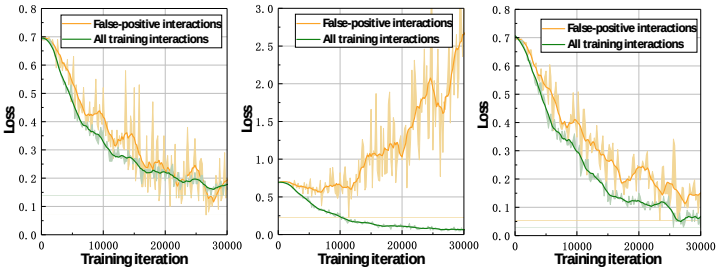
Three testing recommenders are trained with our proposed two paradigms to formulate the CE loss function, respectively, and Table 3 summarizes the performance comparison over three datasets. From Table 3, we can observe the following points:

- Our proposed ADT strategy effectively improves the performance of three testing recommenders over the clean testing set. The relative improvements under two paradigms are both significant, indicating that the proposed ADT strategies successfully reduce the impact of noisy implicit feedback and enhance the generalization ability.
- By comparing the Truncated Loss and Reweighted Loss, we found that the Truncated Loss performs better in most cases. This is because that the Reweighted Loss is still affected by the false-positive interactions even if they have smaller weights. In addition, the dynamic threshold function in the Truncated Loss can be tuned more granularly with two hyper-parameters.
- ADT achieves the biggest performance increase on NeuMF and the improvement over GMF and CDAE is relatively smaller. By comparing their performance under normal training, we can find that NeuMF performs worse than GMF and CDAE, especially on Amazon-book and Yelp, indicating that NeuMF is more vulnerable to noisy interactions. The reason might be that the MLP model in NeuMF increases its complexity and makes NeuMF fit more false-positive interactions. The bigger performance improvement of ADT over NeuMF demonstrates ADT could effectively prevent vulnerable models from the disturbance of noisy data.
- As to the difference over three datasets, we can observe that the biggest performance improvement is over Amazon-book while the smallest one is on Adressa. It's probably because that the training samples of Adressa are less, and even worse, false-positive interactions of Adressa occupies more than 59% in all observed positive ones, which greatly affects the denoising training of neural recommenders. It implies that a great proportion of noisy interactions in the training data partly restrict the improvements of ADT.

[4] http://jmcauley.ucsd.edu/data/amazon/
[5] 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%



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

Figure 7: Loss comparison of false-positive interactions between Normal Training (a), Truncated Loss (b) and Reweighted Loss (c).

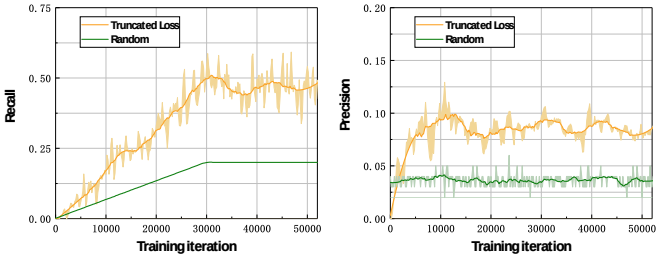


Figure 8: Recall and precision of false-positive interactions over GMF trained the Truncated Loss on Amazon-book.

5.2 Discussion

5.2.1 Memorization of False-positive Interactions. As discussed in Section 3.2, false-positive interactions are memorized by recommenders eventually during normal training, which leads to poor generalization performance. Here we wanted to explore whether false-positive interactions are also fitted well by the recommenders trained with ADT strategies. Figure 7 shows the trend of the CE loss values of false-positive interactions and all training samples along the process of training GMF on Amazon-book. The results of other testing recommenders with similar trends are omitted to save space.

From Figure 7, we can have the following findings: 1) 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 memorizes false-positive samples well at last. 2) When GMF is trained with T-CE loss, the loss values of false-positive interactions are becoming extremely large while the loss values of all training samples are stable and small. This is because the recommender always selects the small-loss samples to optimize parameters, and more and more false-positive interactions are not fitted during training. 3) The loss of false-positive interactions is also decreasing along the training process when GMF is trained with R-CE loss. However, their loss values are always larger than those of other interactions because false-positive interactions are assigned with smaller weights, which delays the models' memorizing them. From the aforementioned observations, we can conclude that both paradigms reduce the effect of false-positive interactions on recommender training, which can explain why they achieve performance 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 defined Recall to represent what percentage of false-positive interactions in the training data are discarded, and treated precision as the ratio of discarded false-positive interactions to all discarded samples. Figure 8 visualizes the changes of the recall and precision along the training process. The green line in Figure 8 indicates the recall and precision under the settings of random discard. In particular, the recall of random discard equals the drop rate during training while its precision is the proportion of noisy interactions in all training samples at each iteration.

From Figure 8, we observed the following points: 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 discard. 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 limitation of the Truncated Loss is low precision, *e.g.*, only 10% precision in Figure 8, which implies it inevitably discards many clean interactions. This also partly proves that it's worth pruning noisy interactions with the cost of losing many 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.

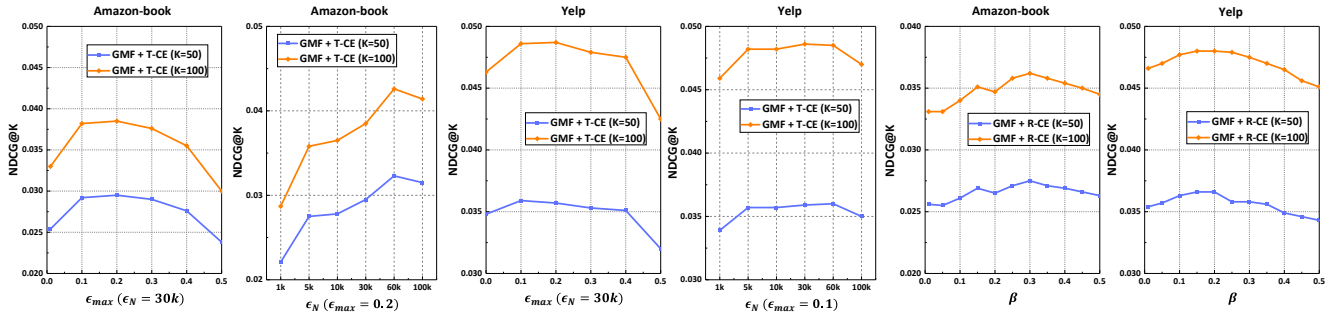


Figure 9: Performance comparison of GMF trained with ADT strategies on Yelp and Amazon-book *w.r.t.* different hyper-parameter settings.

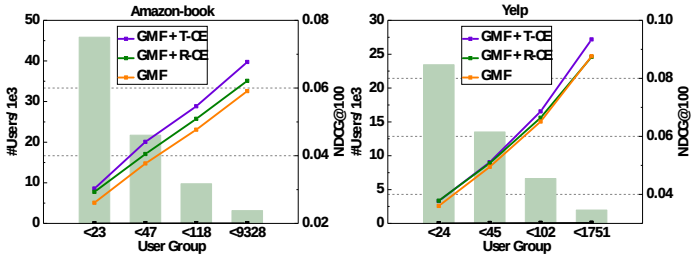


Figure 10: Performance comparison of GMF over user groups with different sparsity levels on Amazon-book and Yelp. The histograms represent the user number in each group and the lines denote the performance *w.r.t.* NDCG@100.

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 recommendation performance. Only the results of GMF trained with ADT strategies on Amazon-book and Yelp are reported in Figure 9 due to the space limitation. And other methods over three datasets have similar patterns. From Figure 9, we can find that: 1) The recommender trained with the T-CE loss achieves the maximum performance improvement 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} in the Truncated Loss should be restricted. 2) The recommender is relatively sensitive to ϵ_N, especially on Amazon-book, and the performance still increases when ϵ_N is greater than 30k. Nevertheless, a limitation is that the big search space restricts the hyper-parameter tuning. 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 will indicate how to tune the hyper-parameters of ADT strategies when they are applied to train other recommenders on different datasets.

5.2.4 Performance Comparison w.r.t. Interaction Sparsity. Since the sparsity issue prevents the recommenders from learning the preference of inactive users, we explored whether our ADT strategies affect the learning of inactive users' preference because they prune many interactions during training. Following the former studies [31], we split testing users into four groups according to the interaction number of each user. And each group has the same number of interactions in total. Only the results of GMF over different groups are shown in Figure 10 due to the space limitation. From Figure 10, we can observe that the proposed ADT strategies improve the performance over all user groups on different datasets, indicating that ADT strategies are stable and also effective for the inactive users.

6 CONCLUSION AND FUTURE WORK

In this work, we aim to denoise implicit feedback for recommender training. We first explored the negative effects of noisy implicit feedback on the training of recommenders, and then proposed Adaptive Denoising Training strategies to reduce their impacts during training. In particular, this work contributes two paradigms to formulate the loss functions: Truncated Loss and Reweighted Loss. The former truncates the loss values of noisy samples to 0 with a dynamic threshold function; the latter reweighs all interactions adaptively during training. These two paradigms are general adaptive denoising training strategies, which can be applied to various settings with different loss functions, recommender model, and optimizers. In this work, we implemented the two paradigms on the widely used binary cross-entropy loss and conducted extensive experiments over three testing recommenders on three datasets, demonstrating that the proposed Adaptive Denoising Training strategies effectively reduce the disturbance of noisy implicit feedback.

This work takes the first step to denoise implicit feedback for recommendation without using any additional data, and points out some new research directions. Specifically, it is interesting to explore how the proposed two paradigms perform on other loss functions, such as Square Loss [28], Hinge Loss [28] and BPR Loss [27]. In addition, how to further improve the precision of the Truncated Loss is worth studying since low precision means it loses a proportion of training samples. Lastly, it is worth noting that our Adaptive Denoising Training strategies are not specific to the recommendation, and it can be widely used to denoise implicit interactions in many other domains, such as Web search, question answering, and query auto-completion.

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] 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 ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 225âĂŞ234.

[3] 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.

[4] 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 ACM Conference on Recommender systems*. ACM, 91–98.

[5] 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, 1042âĂŞ1048.

[6] Asela Gunawardana and Guy Shani. 2015. Evaluating recommender systems. In *Recommender systems handbook*. Springer, 265–308.

[7] 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.

[8] 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, 507âĂŞ517.

[9] 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.

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

[11] 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 ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 15–24.

[12] 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.

[13] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. 2014. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 193–202.

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

[15] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 426âĂŞ434.

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

[17] 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.

[18] 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 ACM Conference on Recommender Systems*. ACM, 287–295.

[19] 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.

[20] 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.

[21] Tzu-Heng Lin, Chen Gao, and Yong Li. 2019. CROSS: Cross-Platform Recommendation for Social E-Commerce. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 515âĂŞ524.

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. IEEE, 2980–2988.

[23] 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 ACM SIGIR conference on Research and development in information retrieval*. ACM, 379–386.

[24] 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 ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 435–444.

[25] 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.

[26] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*. MIT Press, 1257–1264.

[27] 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.

[28] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. 2004. Are Loss Functions All the Same? *Neural Computation*. 16, 5 (2004), 1063âĂŞ1076.

[29] 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.

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

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

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

[33] 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 ACM International Conference on Web Search and Data Mining*. ACM, 153–162.

[34] 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, 639âĂŞ640.

[35] Peng Yang, Peilin Zhao, Vincent W. Zheng, Lizhong Ding, and Xin Gao. 2018. Robust Asymmetric Recommendation via Min-Max Optimization. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 1077âĂŞ1080.

[36] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 113âĂŞ120.

[37] 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 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 989âĂŞ997.

[38] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. 2019. How does Disagreement Help Generalization against Label Corruption?. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 7164–7173.

[39] 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 ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 143âĂŞ152.

[40] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

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

[42] Qian Zhao, F. Maxwell Harper, Gediminas Adomavicius, and Joseph A. Konstan. 2018. Explicit or Implicit Feedback? Engagement or Satisfaction?. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. ACM, 1331–1340.