Journal of Computer Science & Technology

Indexed in: SCIE, EI, INSPEC, CBST, DBLP, etc.

Sponsored by: ICT, CAS & CCF

JCST Papers

Only for academic and non-commercial use

Thanks for reading!

[Survey]

[Computer Architecture and Systems]

[Artificial Intelligence and Pattern Recognition]

[Computer Graphics and Multimedia]

[Data Management and Data Mining]

[Software Systems]

[Computer Networks and Distributed Computing]

[Theory and Algorithms]

[Emerging Areas]

JCST WeChat Subscription Account

JCST URL: https://jcst.ict.ac.cn

SPRINGER URL: https://www.springer.com/journal/11390

E-mail: jcst@ict.ac.cn

Online Submission: https://mc03.manuscriptcentral.com/jcst

Twitter: JCST_Journal

LinkedIn: Journal of Computer Science and Technology

Liu PS, Zheng LN, Chen JL *et al.* Enhancing recommendation with denoising auxiliary task. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 39(5): 1123-1137 Sept. 2024. DOI: 10.1007/s11390-024-4069-5

Enhancing Recommendation with Denoising Auxiliary Task

Peng-Sheng Liu[1] (刘鹏圣), Li-Nan Zheng[2, 3] (郑力南), Jia-Le Chen[2, 3] (陈加乐), Guang-Fa Zhang[2, 3] (张广发) Yang Xu[1] (徐　杨), and Jin-Yun Fang[2, *] (方金云)

[1] *College of Big Data and Information Engineering, Guizhou University, Guiyang 550025, China*

[2] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China*

[3] *University of Chinese Academy of Sciences, Beijing 100049, China*

E-mail: ie.psliu21@gzu.edu.cn; zhenglinan21@mails.ucas.ac.cn; chenjiale21s@ict.ac.cn; zhangguangfa14@mails.ucas.ac.cn xuy@gzu.edu.cn; fangjy@ict.ac.cn

Received December 25, 2023; accepted June 25, 2024.

Abstract The historical interaction sequences of users play a crucial role in training recommender systems that can accurately predict user preferences. However, due to the arbitrariness of user behaviors, the presence of noise in these sequences poses a challenge to predicting their next actions in recommender systems. To address this issue, our motivation is based on the observation that training noisy sequences and clean sequences (sequences without noise) with equal weights can impact the performance of the model. We propose the novel self-supervised Auxiliary Task Joint Training (ATJT) method aimed at more accurately reweighting noisy sequences in recommender systems. Specifically, we strategically select subsets from users' original sequences and perform random replacements to generate artificially replaced noisy sequences. Subsequently, we perform joint training on these artificially replaced noisy sequences and the original sequences. Through effective reweighting, we incorporate the training results of the noise recognition model into the recommender model. We evaluate our method on three datasets using a consistent base model. Experimental results demonstrate the effectiveness of introducing the self-supervised auxiliary task to enhance the base model's performance.

Keywords auxiliary task learning, recommender system, sequence denoising

1 Introduction

Recommender systems play a crucial role in today's Internet and e-commerce domains, offering users improved information retrieval and shopping experiences, while also yielding substantial economic benefits for businesses[1–3]. Click-through rate (CTR) prediction holds a significant role within personalized recommender systems[4–7]. By analyzing users' historical interaction sequences, these systems recommend products aligned with user interests and preferences, facilitating the discovery of potentially engaging content[8]. This method enhances user experience, fosters sales and propagates content[9, 10]. In the context of sequence-based recommendation, the issue of noise present in sequences significantly impacts the establishment of accurate and reliable recommender models, presenting a complex and pivotal challenge within the field. Sequence noise can arise from various sources, including user curiosity, data collection inaccuracies and environmental shifts, consequently leading to misjudgments of user interests and inaccurate recommender model outcomes[7, 11–17]. Models trained on clean sequences significantly outperform those trained on original, noise-containing sequences. This underscores the imperative of exploring denoising strategies in recommender systems[18].

To address the challenges mentioned above, denoising of sequences has garnered increasing attention from researchers. Recent studies demonstrate that using denoising methods in recommender systems can lead to more efficient model training and

Regular Paper

The work was supported by the Program for Student Innovation Through Research and Training of Guizhou University under Grant No. 2023SRT071.

*Corresponding Author

better performance at a reasonable computational cost[19–21]. The existing denoising process involves two steps: recognizing noise and handling noisy sequences.

In practice, recognizing for noise typically judges sequences with high loss values as noisy sequences. Based on the handling of noisy sequences, existing methods can be categorized into two types: truncated denoising and reweighted denoising. For truncated denoising methods[22, 23], the objective is to train a network capable of recognizing noise and discarding noisy sequences, allowing the models to only learn from clean sequences. Regarding reweighted denoising methods[18, 24], once noisy sequences are recognized, these methods tend to assign smaller weights to these sequences throughout the entire model training process, thereby reducing the contribution of these sequences to the recommender models.

Although these denoising methods improve the performance of the recommender model, user behaviors encompass diverse interests and motivations. Some interactions may be temporary, random or influenced by other factors, which increases the difficulty of recognizing between noisy and clean sequences. Moreover, due to complex data distributions and inherent learning difficulties, high loss values do not necessarily indicate noisy sequences. Additionally, thresholds in the truncated denoising method heavily relies on the sampling distribution during the decision-making process, inevitably discarding many clean sequences and potentially exhibiting biased selections[18]. Reweighted denoising methods require specific configurations for a given model or recommendation task, which can be time-consuming and challenging to transfer to other settings[25].

To address the aforementioned issues, from an intuitive perspective, we posit that using a noise recognition model to identify noise sequences and then assigning smaller weights to these sequences to mitigate their influence can enhance the performance of the recommender model. Unlike traditional noise recognition methods, we propose a direct method by constructing a noise recognition model as an auxiliary task to specifically identify noisy sequences. Moreover, to mitigate the impact of reduced training data on the recommender model, we use a novel adaptive reweighting method: training the noise recognition model and the recommender model jointly. This method allows for assigning the most suitable weights for different sequences, optimizing the performance of the recommender model.

Initially, we construct a noise recognition model to differentiate between clean and noisy sequences in the original dateset. Given the difficulty of identifying noisy sequences within the original dataset[25], we artificially create noisy sequences by replacing historical click items of the original sequences with random data. Due to the inherent limitations of human intervention, the artificially replaced noisy sequences may not fully replicate the authentic noisy sequences present in the original sequences. However, since certain authentic noisy sequences also result from users' sporadic, unintentional clicks, there are some similarities between them. Based on the assumption of the existence of certain similarities, we believe that the artificially replaced noisy sequences can represent a portion of the original noisy sequences, and thus we regard the artificially replaced noisy sequences as noise data. Given the scarcity of true noise data within the original sequences, we regard the original sequences as clean data. At this point, we can conduct labeled training for the noise recognition model.

Furthermore, we cannot simply discard the noisy sequences from the original sequences, as these noisy sequences may contain factors that are beneficial for the training of the recommender model, and different noisy sequences have varying impacts on the training of the recommender model. Therefore, we use a novel adaptive reweighting method. Taking into account that a fixed weighting strategy does not adapt to model variations and that the contributions of noisy data to model training are not uniform, we opt to design the sequence weights as learnable parameters associated with the denoising method, which is beneficial for the performance of the recommender model. Specifically, we train the noise recognition model using original sequences and randomly replaced noisy sequences. The noise recognition model then weights non-overlapping original sequences not used in its training. These weighted sequences are subsequently used to train the recommender model. This joint training is accomplished through the auxiliary task, ensuring that the noise recognition model accurately identifies noisy sequences while optimizing the results of sequence reweighting.

After training the noise recognition model, the noise recognition model becomes adept at accurately distinguishing between these two types of sequences. In other words, the noise recognition model tends to

classify the original sequences it is trained on as clean sequences, which results in the inability to recognize the noisy sequences in the original sequences. Taking this issue into consideration, we choose to use the non-overlapping original sequences that are not involved in the training of the noise recognition model as inputs for the recommender model allow us to determine which of the input sequences used during the training of the recommender model contain noise.

The main contributions of this work are as follows.

- We introduce the novel self-supervised Auxiliary Task Joint Training (ATJT) method, where the weights obtained from the joint training of the noise recognition model and the recommender model are reweighted onto the sequences used for training the recommender model. This method enhances the performance of the recommender model.
- The ATJT method is versatile and can be applied to various underlying recommender models.
- We evaluate the ATJT method on three datasets using a consistent base model. Experimental results show that our method improves the performance of the recommender model.

The paper is structured as follows. Section 2 introduces related work, focusing on CTR models and denoising methods. Section 3 presents the preliminary work, describes the training processes for both the noise recognition and recommender models, and explains the ATJT method. Section 4 presents the experimental setup, results and model analysis. Section 5 concludes the paper with a summary of our work and discusses future research directions.

2 Related Work

In this section, we introduce the CTR models and provide a comprehensive overview of the methods related to sequence denoising in CTR models.

2.1 CTR Models

In recent years, deep learning based models have gained significant traction in CTR prediction[15]. These models exhibit strong representation learning capabilities, enabling them to capture more intricate and challenging patterns and features. Existing deep learning based recommender models can be broadly categorized into two types: sequence-based[7, 26–34] and graph-based[35–37] models. We propose a sequence-based denoising method in this paper. This subsection focuses on sequence-based recommender models. Wide & Deep[11] and DCN[14] leverage the memory and generalization capabilities of feature interactions by combining traditional generalized linear models with deep neural networks. DIN[7] uses self-attention mechanisms to enhance the representation of user interests. SASRec[38] and S3Rec[39] utilize a multi-head self-attention mechanism to model relationships within sequences. PS-SA[40] employs a learnable progressive sampling strategy to identify the most valuable items. FEARec[41] enhances recommendation by converting user historical behavior sequences into frequency domain representations and combining them with a self-attention mechanism.

CTR models leverage self-supervised learning[42] methods to improve data utilization and learn feature representations. For instance, DuoRec[43] and MPT[44] enhance item embedding distributions through contrastive learning. ICL[45] and simple CL method[46] address data sparsity and popularity bias by learning user intent representations. Pre-training GNN[47], multi-channel hypergraph convolutional network[48], DHCN[49], and self-supervised tri-training[50] integrate self-supervised learning with other relevant techniques to enhance the performance of recommender systems.

2.2 Denoising Methods

Identifying noisy sequences is an essential step in sequence denoising. DROP[51] and three instance selection methods[52] discuss how to reduce the number of sequences in the training set without affecting classification accuracy. AutoDenoise[25] deletes sequences that have a counteractive effect on the model through rewards. Hierarchical reinforcement learning for course recommendation in MOOCs[53] removes noisy courses by jointly training a hierarchical reinforcement learning based modifier and a basic recommender model. DeCA[24] determines noisy sequences by analyzing the discrepancies in user preferences predicted by two recommender models. MMInfoRec[54] and ContrastVAE[55] address issues such as sparsity and uncertainty in recommender systems by leveraging contrastive learning techniques. DT4SR[56] effectively resolves the problem of neglecting user dynamic preferences and item relationships in traditional methods by introducing uncertainty into sequential

modeling. The SDK framework[57] deals with the challenges of knowledge graphs (KGs) in knowledge-aware recommendation by modeling hyper-relational facts and using self-supervised learning mechanisms. SGL[58] improves the recommendation performance of long-tail items and the robustness against interaction noises by using an auxiliary self-supervised learning task. We propose a denoising auxiliary task that neither requires considering the impact on the model nor adds excessive additional training steps. We define a model capable of recognizing noise, thereby enhancing the model's performance.

After recognizing noisy sequences, we need to handle them to improve the performance of the recommender model. Existing methods for handling noisy sequences can be classified into two categories: truncated denoising[18, 19, 21] and reweighted denoising[18]. WBPR[19] and T-CE[18] define thresholds for samples, truncating sequences with loss values higher than the threshold at each iteration. IR[21] modifies labels to train downstream modules for recommendation tasks. In R-CE[18], smaller weights are assigned to high-loss sequences to prevent the model from fitting them too quickly. However, truncated denoising methods risk filtering out many clean sequences, while reweighted denoising methods suffer from limited transferability. We propose ATJT similar to reweighted denoising methods, but it addresses limitations by adaptively adjusting the weighting degree.

3 Methodology

We introduce the preliminary work and discuss the training processes for the noise recognition model and the recommender model, and provide a detailed explanation of how to implement the ATJT method.

3.1 Preliminary

In this paper, we use batch b composed of training sequences as the input for both the noise recognition model and the recommender model. Each batch has a size M, and the sequences have a length N.

All batches are divided into two groups, $\mathcal{B}^R$ and $\mathcal{B}^D$. The batch in the first group, denoted as $b_i^R = \{s_{i,1}, \ldots, s_{i,m}, \ldots, s_{i,M}\} \in \mathcal{B}^R$, undergoes obtaining the weights of historical interaction sequences through the noise recognition model. We then use the reweighted sequences to train the recommender model. We use s_i to represent the sequences of the i-th batch that are used for training the recommender model. And $\mathcal{B}^R$ consists of a total of I batches. The batch in the second group, denoted as $b_j^D = \{s_{j,1}, \ldots, s_{j,m}, \ldots, s_{j,M}\} \in \mathcal{B}^D$, is used to train the noise recognition model capable of accurately recognizing noisy sequences. We use s_j to represent the sequences of the j-th batch that are used for training the noise recognition model. And $\mathcal{B}^D$ consists of a total of J batches. In summary, $\mathcal{B}^R \cup \mathcal{B}^D = \mathcal{B}$ and $\mathcal{B}^R \cap \mathcal{B}^D = \emptyset$.

We further divide the batch b_j^D into two batches, $b_j^{D(+)}$ and $b_j^{D(-)}$. $b_j^{D(+)}$ represents the clean batch within b_j^D consisting of original sequences. $b_j^{D(-)} = \{s'_{j,1}, \ldots, s'_{j,m}, \ldots, s'_{j,M}\}$ represents noisy batch consisting of randomly replaced sequences from b_j^D, where $s'_{j,m} = \{v_1, \ldots, v'_n, \ldots, v_N\}$ represents the m-th noisy sequence that has undergone random replacement in the j-th batch of the noise recognition model. Within the sequence $s'_{j,m}$, v'_n represents the n-th interaction item that has been randomly replaced. At this point, the second batch transforms into $b_j^D = \{s_{j,1}, \ldots, s'_{j,m}, \ldots, s_{j,M}\} \in \mathcal{B}^D$. In summary, $b_j^{D(+)} \cup b_j^{D(-)} = b_j^D$ and $b_j^{D(+)} \cap b_j^{D(-)} = \phi$.

We use $f(\cdot;\Theta_R)$ to represent the recommender model and $g(\cdot;\Theta_D)$ to represent the noise recognition model. Given the users' historical interaction sequences $(u, s_i) \in b_i^R$, the recommender model can predict the probabilities $f(u, s_i;\Theta_R)$ of clicks. Similarly, given $(u, s_j) \in b_j^D$, the noise recognition model can predict the probabilities $g(u, s_j;\Theta_D)$ of noise contamination.

Summarily, we enhance the recommender model's performance by obtaining accurate weights w_i for the sequences s_i from the joint training of $f(\cdot;\Theta_R)$ and $g(\cdot;\Theta_D)$.

3.2 Recommender Model

In the training process of the recommender model, as shown in Fig.1, given a batch $b_i^R \in \mathcal{B}^R$, the sequences s_i in the batch initially pass through the noise recognition model to obtain weights w_i. Subsequently, the reweighted sequences are used to train the parameters of the recommender model. It is worth noting that the recommender model can be chosen based on specific requirements, such as DIN or DCN. Its training process aligns with these base models.

The CTR prediction of the recommender model

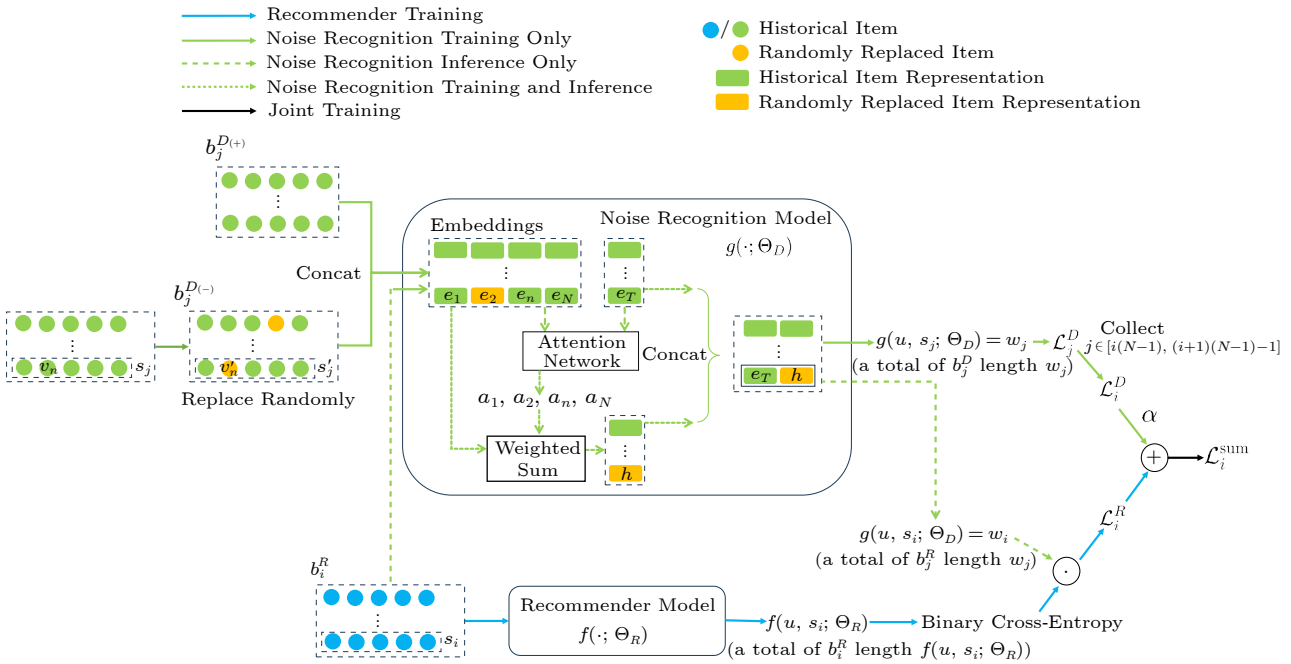


Fig.1. The ATJT method consists of two main components: 1) training the noise recognition model (composed of Noise Recognition Training Only and Noise Recognition Training and Inference), 2) training the recommender model using reweighted sequences (composed of Recommender Training, Noise Recognition Inference Only, and Noise Recognition Training and Inference).

can be viewed as a supervised binary classification task. Therefore, we optimize the recommender model using a binary cross-entropy loss function. Additionally, considering the impact of noisy sequences on the training of the recommender model, it is essential to recognize and assign smaller weights to mitigate the influence of noisy sequences. Consequently, we define the loss function for the recommender model as:

$$\mathcal{L}_i^R = -\frac{1}{|b_i^R|}\sum_{s_i \in b_i^R}^{|b_i^R|}(w_i(y_i \log f(u, s_i; \Theta_R) + (1 - y_i)\log(1 - f(u, s_i; \Theta_R)))), \quad (1)$$

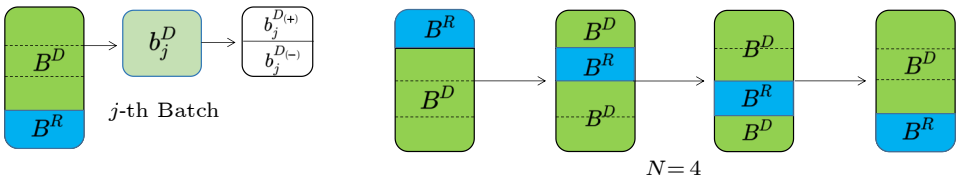
where y_i and $f(u, s_i; \Theta_R)$ represent the labels for clicks and the predicted probabilities of clicks for the sequences s_i in b_i^R, respectively. w_i represents the weights of s_i. Typically, noisy sequences have smaller weights w_i compared with clean sequences in model training (as shown in our experiments in Subsection 4.2.4). This approach reduces the impact of noisy sequences on model performance[18, 24]. We will elaborate on how to determine the sequence weights w_i that improve the performance of the recommender model in Subsection 3.3.2 and Subsection 3.4.2.

3.3 Noise Recognition Model

To build a noise recognition model capable of accurately distinguishing noisy sequences from clean sequences and weighting the sequences for the recommender model, we opt for a self-supervised training method. In this subsection, we focus on two essential components: data replacement and weight generation.

3.3.1 *Data Replacement*

As shown in Fig.2(a), we use the batch $b_j^D = b_j^{D(+)} \cup b_j^{D(-)}$ as the input for the noise recognition model, where $b_j^{D(+)}$ represents the clean batch consisting of original sequences, labeled as 1. $b_j^{D(-)}$ represents the noisy batch composed of randomly replaced noisy sequences, labeled as 0. While the selection of $b_j^{D(-)}$ from b_j^D is not fixed, it should not be too scant. Specifically, we assume that there are very few noisy sequences in $b_j^{D(+)}$. If we select too few sequences in $b_j^{D(-)}$, it may lead to a situation where the extremely few noisy sequences in $b_j^{D(+)}$ outnumber the sequences in $b_j^{D(-)}$, meaning that the number of sequences in $b_j^{D(+)}$ labeled as 1 while actually being 0 is greater than the number of sequences in $b_j^{D(-)}$ labeled as 0. This situation could make the noise recognition model incorrectly learn noisy sequences as positive (labeled as 1). Hence, it is essential to ensure an adequate number of sequences in $b_j^{D(-)}$ to avoid an unsta-



(a) (b)

Fig.2. (a) Division of noisy and clean sequences within a batch in the noise recognition model, following a 1:1 ratio. (b) Partitioning of training data for the recommender model and the noise recognition model. (Blue represents training data $\mathcal{B}^R$ for the recommender model, and green represents training data $\mathcal{B}^D$ for the noise recognition model).

ble situation that could make the noise recognition model learn in the wrong direction. Up to this point, we have discussed the training method for the recommender model and how input sequences for the noise recognition model are generated.

3.3.2 *Weight Generation*

In this subsection, we will describe the method for generating weights w_i. As depicted in Fig.1, the noise recognition model is a sequence-to-value model. The model takes $b_j^D \in \mathcal{B}^D$ as input. For $s_j \in b_j^D$, where s_j consists of items with length N and a target item to be predicted, s_j first passes through the neural network's embedding layer. S_j is then transformed into the sequences of embeddings $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n, \ldots, \boldsymbol{e}_N, \boldsymbol{e}_{\mathrm{T}})$, in which $\boldsymbol{e}_n$ represents the embedding of the n-th item in the sequences s_j after passes through the neural network's embedding layer. Then, we pass embeddings through the attention network to obtain user hidden representation of the sequences s_j:

$$\boldsymbol{h} = \sum_{n=1}^{N} a_n \boldsymbol{e}_n,$$

where

$$a_n = \frac{MLP(\boldsymbol{e}_n || \boldsymbol{e}_{\mathrm{T}})}{\sum_{n'=1}^{N} MLP(\boldsymbol{e}_{n'} || \boldsymbol{e}_{\mathrm{T}})},$$

$MLP()$ represents the fully connected layers (also known as multilayer perceptron, MLP), and $||$ represents the concatenation of embeddings. Subsequently, we concat $\boldsymbol{h}$ with the embedding $\boldsymbol{e}_{\mathrm{T}}$ of the target item, and then pass the results through an MLP to produce the weights w_j. Given that our noise recognition method can be viewed as a self-supervised binary classification task, we use the binary cross-entropy loss function for optimization:

$$\mathcal{L}_j^D = -\frac{1}{|b_j^D|} \sum_{s_j \in b_j^D}^{|b_j^D|} (y_j \log g(u, s_j; \Theta_D) + (1 - y_j) \log(1 - g(u, s_j; \Theta_D))), \tag{2}$$

where y_j and $g(u, s_j; \Theta_D) = w_j$ represent the labels for noise and the predicted probabilities of noise for the sequences s_j in b_j^D, respectively.

The noise recognition model similarly uses training set $b_i^R \in \mathcal{B}^R$, which is used in training the recommender model, as input. This set of sequences is non-overlapping with the training sequences used for the noise recognition model, which will be explained in detail in Subsection 3.4.1. At this point, we can use the results $g(u, s_i; \Theta_D)$ output by the noise recognition model as the weights for the training sequences of the recommender model, namely the weights w_i in (1).

Furthermore, the noise recognition model is used only during the training phase to help the recommender model learn better parameters. It is not used during the evaluation phase. Therefore, the ATJT method does not increase the number of parameters in the recommender model.

3.4 ATJT Method

3.4.1 *Data Partition*

To fully use the data, after fitting the parameters of both the recommender model and the noise recognition model, we can reverse the training data for $\mathcal{B}^R$ and $\mathcal{B}^D$. This means that B^D optimizes the recommender model while $\mathcal{B}^R$ optimizes the noise recognition model. It is worth noting that the recommender model continues to use the original model in the subsequent training, while the noise recognition model is trained using a duplicate model. The purpose of this is to ensure that all data can be used to train the recommender model, while the noise recognition model does not fit all the data. Throughout the training process, the reweighted recommender model and the

noise recognition model are trained together. This ensures that the noise recognition model can accurately recognize noisy sequences while optimizing the results of sequence reweighting.

Two important points are noted. First, we only use the original sequences to train the recommender model, and the noise recognition model is trained with original sequences and randomly replaced noisy sequences. Second, if there is a need to make more extensive use of the data, the training set sequences can be divided into N groups instead of two groups. As shown in Fig.2(b), we demonstrate a training method where the training set is divided into four groups. In extreme cases, only one sequence receives the best reweighting output by the noise recognition model and trains the recommender model, while the rest of the sequences are input into the noise recognition model to achieve the best recognition performance in training. However, this method increases the number of duplicate models for training the noise recognition model. Therefore, the minimum grouping is two groups, and the maximum grouping is the size of the training set sequences $|\mathcal{B}^R \cup \mathcal{B}^D|$. The specific grouping can be chosen based on available resources and performance considerations.

3.4.2 Loss Function

We focus on how to jointly train the recommender model with the noise recognition model by computing the loss value. When we partition the training set sequences into N groups, due to $|\mathcal{B}^R| : |\mathcal{B}^D| = 1 : N-1$, the batches used for training the recommender model should be in a ratio of $1 : N-1$ compared with those for training the noise recognition model, as illustrated in Fig.3. The loss function for the noise recognition model should be:

$$\mathcal{L}_i^D = \frac{1}{N-1} \sum_{i\times(N-1)}^{(i+1)(N-1)-1} \mathcal{L}_j^D,$$

where i represents the index of the batch used by the recommender model. $N-1$ represents the number of batches used by the noise recognition model corresponding to one batch of data used for training the recommender model, and $\mathcal{L}_j^D$ represents the binary cross-entropy loss function when training the noise recognition model with the j-th batch of data. During the training process, we combine the loss of the recommender model with the loss of the noise recognition model using a scaling factor α to obtain the total loss for model training:

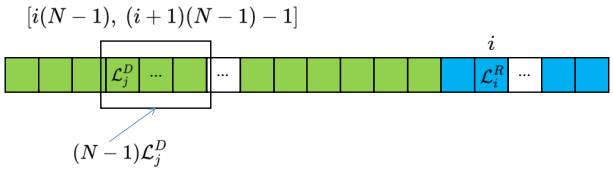


Fig.3. When training the recommender model with the i-th set of data from $\mathcal{B}^R$, the noise recognition model with data from $\mathcal{B}^D$ in the range $[i(N-1), (i+1)(N-1)-1]$ is concurrently trained. Green represents training data for the noise recognition model, and blue represents training data $\mathcal{B}^R$ for the recommender model.

$$\mathcal{L}_i^{\text{sum}} = \mathcal{L}_i^R + \alpha \mathcal{L}_i^D,$$

where α represents a tunable parameter that allows us to control the learning rates of the recommender model and the noise recognition model, thereby achieving the goal of joint training. Joint training enables the noise recognition model to learn the weights w_i for sequences s_i in b_i^R (as defined in (1)), which is more suitable for training the recommender model while accurately recognizing noise. This enables the recommender model to achieve better performance with the weighted sequences.

3.4.3 Overall Optimization Algorithm of Model Training

The joint training process is illustrated in Algorithm 1. The joint training consists of two parts: the calculation of the $\mathcal{L}_i^R$ for the recommender model (lines 6 and 7) and the calculation of the $\mathcal{L}_i^D$ for the noise recognition model (lines 8–14). We achieve joint training by summing the loss values from these two parts (line 15). Specifically, we start by initializing the recommender model $f(\cdot;\Theta_R)$ (line 2). Next, we iterate over all groups in the training set as described in Subsection 3.4.1 (line 3). We then initialize the noise recognition model (line 4) and retrieve the i-th batch from the training set of the recommender model (line 5). The sequences from the i-th batch are passed through the noise recognition model $g(\cdot;\Theta_D)$ to determine their weights w_i (line 6). These sequences are then input into the recommender model, and the weighted loss is calculated. After averaging the loss for all sequences in the batch, we obtain $\mathcal{L}_i^R$ (line 7). Subsequently, we iterate over the batches from the $(i(N-1))$-th to the $((i+1)(N-1)-1)$-th in the training set of the noise recognition model (line 9). From the batch of the noise recognition model, we select half of the sequences. For these sequences, we

perform random replacements of items, considering them as noisy sequences (lines 10 and 11). The original sequences and the noisy sequences from set b_j^D are input into the noise recognition model, and the loss is calculated. After averaging the loss for all the sequences in the batch, we obtain $\mathcal{L}_j^D$ (line 12). Next, we accumulate all $\mathcal{L}_j^D$ values during the iteration onto $\mathcal{L}_i^D$ to obtain the final noise recognition model loss (line 13). Finally, we add $\mathcal{L}_i^R$ and $\mathcal{L}_i^D$ to obtain $\mathcal{L}_i^{\text{sum}}$ (line 15). At this point, we can jointly optimize the recommender model and the noise recognition model based on $\mathcal{L}_i^{\text{sum}}$ (lines 16 and 17). At this point, we complete the training of the noise recognition model and explain how to implement the ATJT method.

Algorithm 1. Overall Optimization of Model Training

Require: $b_i^R \in \mathcal{B}^R$, $b_j^D \in \mathcal{B}^D$, recommender model $j(\cdot;\Theta_R)$, noise recognition model $g(\cdot;\Theta_D)$, dataset groups N, reweighted recommender model loss $\mathcal{L}_i^R$, noise recognition model loss $\mathcal{L}_j^D$, (N-1) batches of noise recognition model loss $\mathcal{L}_i^D$, sum loss $\mathcal{L}_i^{\text{sum}}$

Ensure: trained recommender model $f^*(\cdot;\Theta_R)$

1: Create $\mathcal{L}_i^R$, $\mathcal{L}_j^D$, $\mathcal{L}_i^{\text{sum}}$
2: Initialize $f(\cdot;\Theta_R)$
3: **for** $n \in [1, N]$ **do**
4: Initialize $g(\cdot;\Theta_D)$
5: **for** $i \in [1, |\mathcal{B}^R|]$ **do**
6: Estimate w_i using noise recognition model $g(\cdot;\Theta_D)$
7: Get $\mathcal{L}_i^R$ by (1)
8: $\mathcal{L}_i^D \leftarrow 0$
9: **for** $j \in [i(N-1), (i+1)(N-1)-1]$ **do**
10: $b_j^D = b_j^{D(+)} \cup b_j^{D(-)}$, $|b_j^{D(+)}| = |b_j^{D(-)}|$
11: $s_j^{'} = (v_1, \ldots, v_n^{'}, \ldots, v_N) \in b_j^{D(-)}$ represents the replaced sequence as described in Subsection 3.1
12: Get $\mathcal{L}_j^D$ by (2)
13: $\mathcal{L}_i^D \mathrel{+}= \mathcal{L}_j^D$
14: **end for**
15: $\mathcal{L}_i^{\text{sum}} \leftarrow \mathcal{L}_i^R + \alpha \frac{1}{N-1} \mathcal{L}_i^D$
16: Update Θ_R
17: Update Θ_D
18: **end for**
19: **end for**

4 Experiments

We conduct extensive experiments to address the following three questions.

- RQ1: how does the performance of the ATJT method compared with the base model?
- RQ2: what is the impact of different types of noisy sequences generation on performance?
- RQ3: how does different sequence weighting training methods affect performance?

4.1 Experimental Setting

4.1.1 Datasets and Baselines

We evaluate our method using datasets MovieLens20M[①], Amazon (Electro)[②], and Yelp[③]. We select these datasets for two reasons. 1) They represent diverse scenarios, namely an online movie platform and an e-commerce platform, with varying levels of product diversity. 2) They differ in size and characteristics. The statistical data for MovieLens20M, Amazon (Electro), and Yelp are shown in Table 1. Datasets MovieLens20M and Amazon (Electro) both consist of features such as user ID, historicalin-teraction item IDs, and their correspondingcate-gories. In the Yelp dataset, each item has featuresin-cluding its business_id, city, postal_code, star rating,and categories. Each user has features including theuser_id, useful, funny, cool and average star rating.

Table 1. Statistical Information for Datasets

Dataset	#Users	#Items	#Samples
MovieLens20M	138 493	27 278	20 000 263
Amazon (Electro)	192 403	63 001	1 689 188
Yelp	1 987 929	150 346	6 990 280

Note: # represents number of.

We employ several advanced recommender models as base models, including Wide & Deep[11], DCN[14], DIN[7], SASRec[38], S3Rec[39] and FEARec[41]. We use the results of these six base models on three datasets as the baseline and compare them with the ATJT method.

We conduct a total of 6 (the number of recommender models) $\times 2$ (the number of contrastive models) experiments to assess the performance improvement of ATJT on three specified datasets for the recommender model.

4.1.2 Evaluation Protocol

To accurately assess the performance of the rec-

[①] https://grouplens.org/datasets/movielens/20m, Jun. 2024.
[②] http://jmcauley.ucsd.edu/data/amazon/, Jun. 2024.
[③] https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset/data, Jun. 2024.

ommender model, we first divide users' historical interaction sequences into the training and testing sets in a 4:1 ratio. In this setup, we use the training set to train both the noise recognition model and the recommender model, while the testing set is used to evaluate the performance of the recommender model. Notably, we need to ensure that users' historical interaction sequences in the training and testing sets are non-overlapping. Additionally, to avoid the issue described in Subsection 3.4.1, where the noise recognition model fits the training data, we also need to ensure that the historical interaction sequences used for training the recommender model and the noise recognition model are non-overlapping.

We evaluate the testing set using standard AUC (relative improvement area under the ROC curve) scores, HR@5 and NDCG@5, which are are widely used in click prediction tasks[7, 8]. Higher values for all three metrics indicate superior model performance.

4.1.3 Implementation Details

The construction of the ATJT method is based on the PyTorch framework. We encapsulate the noise recognition model into a class, allowing it to be integrated as a plugin with most recommender models. The implementation of the noise recognition model follows a unified structure when integrated with different underlying recommender models. The implementation of the noise recognition model relies on an attention mechanism. Specifically, we implement it as an attention model with embedding and output layers. The attention part consists of two layers of the MLP. For the first layer, we set the dimensions as (64, 32), and for the second layer, we set the dimensions as (32, 1). Each layer consists of a linear layer, a PReLU activation function, and a dropout operation (rate=0.5). The output part after SUM pooling consists of three layers of the MLP. For the first layer, we set the dimensions as (40, 256); for the second layer, we set the dimensions as (256, 64); for the third layer, we set the dimensions as (64, 1). The structure of each layer is identical to the attention part. The structure of MLPs is identical to the attention part. The output layer is implemented with a sigmoid function, with a dimension of 1, in order to obtain different weights for training the recommender model with noisy and clean sequences. The noise recognition model is uniformly optimized using the Adagrad optimizer, and a learning rate search is conducted from {0.1, 0.01, 0.001, 0.000 1}.

When training recommender models, each method follows the following steps. 1) When training the DCN and Wide & Deep models, we treat historical interactions as item features. The DNN architectures for DCN and Wide & Deep are set as (128, 128) and (256, 128), respectively. 2) For the DIN model, we use user ID, historical interaction item IDs, and their corresponding categories as input features, following [7]. 3) For the SASRec and S3Rec models, we use historical interaction item IDs as input features, following [38] and [39]. 4) When training the FEARec model, our input features are the same as those used in the DIN model. Additionally, we use the default hyperparameter configurations provided by the original authors of the model on GitHub④.

4.2 Experimental Results

4.2.1 Overall Performance (RQ1)

We propose the ATJT method based on six fundamental recommender models and compare the performance of the models using the ATJT method with that of the base recommender models on three different datasets. Experimental results show that the ATJT method outperforms base models in terms of AUC, HR@5, and NDCG@5, as shown in Table 2.

We find that the ATJT method yields better improvements in the DCN and Wide & Deep models compared with the DIN, SASRec, S3Rec, and FEARec models. This phenomenon can be attributed to the attention mechanism possessed by DIN, SASRec, S3Rec, and FEARec, which adaptively learns users' interest representations from the historical interaction sequences, and thus mitigates the impact of behaviors unrelated to users' interest representations[7]. Furthermore, the enhancement of the ATJT method is more pronounced in the Wide & Deep model than in the DCN model. The reason may lie in that when we process the input features of the DCN model, we regard historical interactions as item features. Therefore, the cross network captures feature interactions and mitigates the impact of irrelevant features on model performance during training[14]. The Wide & Deep model lacks attention mechanisms like DIN, the multi-head attention mechanism like SASRec, S3Rec, and FEARec or the cross

④https://github.com/sudaada/FEARec, Jun. 2024

Table 2. Experimental Results on Three Datasets Based on Different Recommender Models

Model	MovieLens20M			Amazon (Electro)			Yelp		
	AUC	HR@5	NDCG@5	AUC	HR@5	NDCG@5	AUC	HR@5	NDCG@5
Wide & Deep	0.821 7	0.505 2	0.111 7	0.853 5	0.568 5	0.122 4	0.750 3	0.386 6	0.085 9
Wide & Deep+ATJT	0.831 0	0.512 4	0.114 3	0.857 2	0.574 3	0.124 8	0.759 9	0.397 4	0.089 7
+RI	1.13%	1.43%	2.33%	0.43%	1.02%	1.96%	1.28%	2.79%	4.42%
DCN	0.843 8	0.509 1	0.114 5	0.872 9	0.586 5	0.137 6	0.788 1	0.382 3	0.091 2
DCN+ATJT	0.845 0	0.513 7	0.115 7	0.873 5	0.587 9	0.138 2	0.788 5	0.391 7	0.092 7
+RI	0.14%	0.90%	1.05%	0.06%	0.24%	0.44%	0.05%	2.46%	1.64%
DIN	0.851 6	0.522 2	0.116 6	0.874 8	0.601 1	0.140 6	0.803 2	0.450 6	0.101 2
DIN+ATJT	0.851 9	0.522 6	0.116 7	0.874 9	0.602 4	0.141 1	0.803 6	0.451 4	0.101 5
+RI	0.04%	0.08%	0.09%	0.01%	0.22%	0.36%	0.05%	0.18%	0.30%
SASRec	0.847 5	0.522 2	0.118 0	0.876 7	0.588 1	0.136 8	0.774 7	0.410 9	0.091 2
SASRec+ATJT	0.849 0	0.522 4	0.118 3	0.877 2	0.588 9	0.137 0	0.776 3	0.412 5	0.091 7
+RI	0.18%	0.04%	0.25%	0.06%	0.14%	0.15%	0.21%	0.39%	0.55%
S3Rec	0.849 0	0.531 1	0.117 0	0.878 4	0.603 5	0.142 0	0.803 3	0.443 8	0.099 9
S3Rec+ATJT	0.849 9	0.531 2	0.117 2	0.878 7	0.605 8	0.142 4	0.804 8	0.446 0	0.100 3
+RI	0.11%	0.02%	0.17%	0.03%	0.38%	0.28%	0.19%	0.50%	0.40%
FEARec	0.853 7	0.531 1	0.120 8	0.880 4	0.606 3	0.143 1	0.804 9	0.451 2	0.102 2
FEARec+ATJT	0.853 9	0.533 1	0.121 3	0.881 2	0.609 4	0.143 8	0.806 0	0.456 6	0.103 3
+RI	0.02%	0.38%	0.41%	0.09%	0.51%	0.49%	0.14%	1.20%	1.08%

Note: "+RI" represents the relative improvement of the ATJT method over the base models.

network like the DCN model, which can filter out irrelevant or negative behaviors on model optimization. The ATJT method compensates for the Wide & Deep model's inability to filter out irrelevant or negative behaviors within user actions, resulting in a more noticeable performance improvement.

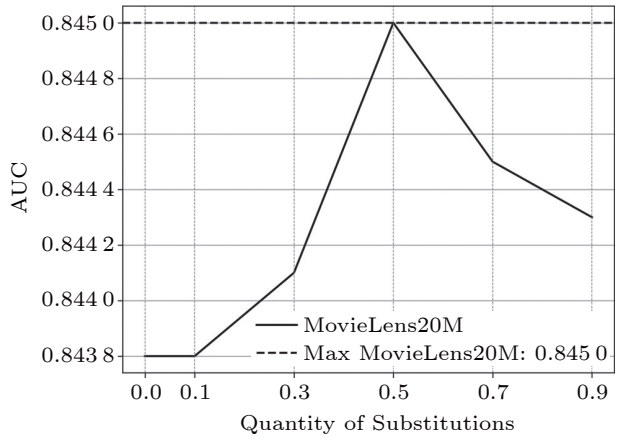
ATJT demonstrates superior performance on the Yelp dataset compared with the MovieLens20M and Amazon (Electro) datasets. This observation may be attributed to the relatively larger size of the Yelp dataset, which provides more data for training more complex models after denoising. The MovieLens20M dataset is also substantial in size, whereas the Amazon (Electro) dataset is relatively smaller, potentially impacting the model's performance after denoising. However, the ATJT method exhibits significant improvements across various base models and data sizes. Furthermore, experiments conducted with six different recommender models indicate the adaptability and efficacy of the ATJT method.

4.2.2 *Noise Generation Analysis (RQ2)*

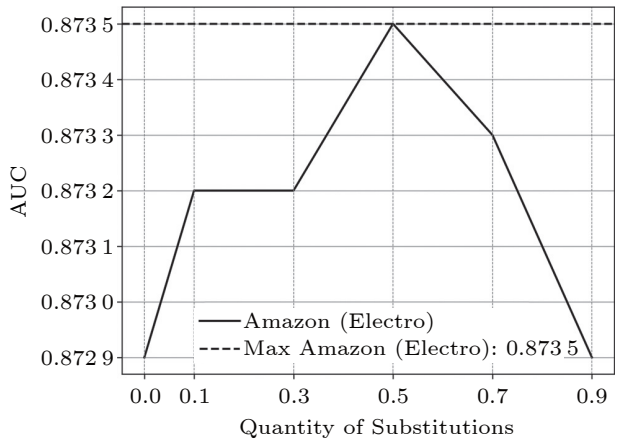
In this subsection, we analyze the effect of selecting how many sequences in the training data of the noise recognition model to replace, and the effect of replacing a different number of historical click items in the artificially replaced noisy sequences. Fig.4 shows the impact of selecting 0.1, 0.3, 0.5, 0.7, and 0.9 of the data in the noise recognition model training as noisy sequences on the performance of the recommender model. Table 3 shows the impact of replacing 1, 2, 3, 5, and 10 historical click items in the artificially replaced noisy sequences on the performance of the recommender model.

Based on Fig.4, it is evident that training the noise recognition model with varying number of sequences, corresponding to different quantities of $b_j^{D(-)}$ as described in Subsection 3.3.1, has a different impact on the fitting speed and the effectiveness of the noise recognition model. When selecting 0.3 or 0.7 of the data, there is a noticeable decrease in fitting speed and performance. This is due to the use of too few noisy sequences during the training of the noise recognition model, which leads to its inability to accurately recognize noisy sequences. Conversely, if artificially replaced noisy sequences are overly abundant, it can also hinder the noise recognition model's ability to accurately distinguish clean sequences. Furthermore, when opting for a smaller number of sequences, such as using 0.1 of the data as noisy sequences, a situation similar to what was described in Subsection 3.3.1 may occur, namely the noise recognition model struggling to differentiate between noisy and clean sequences. Selecting fewer data points also results in poorer fitting performance.

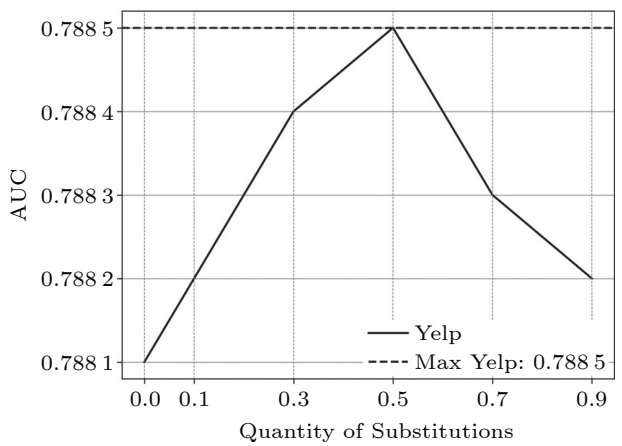
From Table 3, we observe that replacing fewer historical click items in the sequences during the noisy recognition model training leads to more accurate output weights for the sequences in the recommender model, resulting in improved performance. However, as the number of replaced items increases, the perfor-



(a)



(b)



(c)

Fig.4. Analyzing the impact of using different numbers of sequences as noisy sequences in the training of noise recognition models (base model: DCN). (a) MovieLens20M. (b) Amazon (Electro). (c) Yelp.

mance of the recommender model starts to deteriorate. This observation suggests that the original noisy sequences within the historical interaction sequences are mostly sparse. Sequences replacing fewer items exhibit greater similarity to the original noisy sequences, while sequences replacing 3 or more items show significant divergence from the original noisy sequences.

Table 3. Impact Analysis of Replacing Historical Click Items in Artificially Replaced Noisy Sequences on Recommender Model Performance

Replacement Quantity	AUC	HR@5	NDCG@5
Replaced one	0.873 5	0.587 9	0.138 2
Replaced two	0.873 2	0.586 6	0.137 9
Replaced three	0.873 1	0.586 6	0.137 7
Replaced five	0.873 1	0.586 6	0.137 6
Replaced ten	0.873 0	0.586 4	0.137 6

Note: We use the DCN model as the base model and use the Amazon (Electro) dataset.

4.2.3 Sequence Weighting Ablation Experiments (RQ3)

To investigate the impact of different noisy sequence weighting methods on the performance of the recommender model, we compare three weighting training methods: Without Auxiliary Task (WAT), Direct Noise Recognition (DNR), and ATJT. The sequence weights for the three methods are obtained through a consistent model structure as described in Subsection 4.1.3. In the DNR method, the noise recognition model is first separately trained to accurately distinguish artificially replaced noisy sequences from the original clean sequences as the training objective. Then, the training sequences of the recommender model are passed through this noise recognition model to obtain sequence weights, followed by training the recommender model. In the WAT method, the training sequences of the recommender model are directly passed through the targetless model structure to obtain the sequence weights, followed by training the recommender model.

As shown in Table 4, the ATJT method outperforms the other two methods, showing significant improvements in AUC, HR@5, and NDCG@5 metrics. The reason for this is, in comparison with the WAT weighted training method, the ATJT method takes recognition of noise as the auxiliary goal, which helps the training sequences of the recommender model find suitable training weights. Specifically, it assigns larger weights to clean sequences and smaller weights to noisy sequences, and thus mitigates the impact of noisy sequences on the performance of the recommender model. In contrast to the DNR method, the ATJT method not only identifies noisy sequences but also assigns appropriate weights to them. In other words, when training the recommender model with noisy sequences, the goal is not to minimize the

Table 4. Evaluation Metrics for Different Training Sequence Weighting Methods

Method	AUC	+RI (AUC) (%)	HR@5	+RI (HR@5) (%)	NDCG@5	+RI (NDCG@5) (%)
Base (w/o auxiliary task or joint training)	0.872 9	–	0.586 5	–	0.137 6	–
WAT	0.873 1	0.02	0.586 2	–0.05	0.137 7	0.07
DNR	0.873 2	0.04	0.587 8	0.22	0.138 1	0.36
ATJT	0.873 5	0.06	0.587 9	0.24	0.138 2	0.44

Note: We use the DCN model as the base model and use the Amazon (Electro) dataset.

weights assigned to them. Instead, the objective is to find the weights that optimize the performance of the recommender model. It is worth noting that using the ATJT method and the DNR method results in better performance compared with the base model and model trained using the WAT method. This underscores the meaningfulness of weighting sequences through the noise recognition auxiliary task. Additionally, the reason for the inferior performance of the recommender model trained using the WAT method compared with the base model is that the WAT method fails to capture the degree of noise in the sequences. It exhibits confusion in the early stages of training, potentially leading to incorrect weights. This also indicates that augmenting the complexity of the model does not lead to a significant improvement in performance.

4.2.4 *Impact of Noise on Sequence Weights*

To demonstrate that noisy sequences have smaller weights compared with clean sequences, we use the DCN model as the base model and conduct analysis on the three datasets. As shown in Table 5, the weights of the noisy sequences are approximately 19% smaller on average compared with the weights of the clean sequences. This aligns with our expectation that assigning smaller weights to noisy sequences can enhance the base model's performance.

5 Conclusions

In this study, we proposed a novel self-supervised ATJT method. This method leverages the training outcomes of a noise recognition model to reweight sequences for training the recommender model. Additionally, we conducted joint training of the recommender model and the noise recognition model to acquire more appropriate weights, further enhancing the performance of the recommender model. Our method was evaluated on three datasets and six base models, demonstrating its effectiveness. Finally, we validated the impact of different noisy sequences and training methods on the performance of the recommender model through noise generation analysis and sequence weighting ablation experiments.

Table 5. Analysis of Weights w_i in (1) Generated by the Noise Recognition Model

Dataset	Sequence	w_i
MovieLens20M	Noisy	$0.443\,7 \pm 0.028\,3$
	Clean	$0.557\,5 \pm 0.018\,8$
Amazon (Electro)	Noisy	$0.478\,6 \pm 0.009\,4$
	Clean	$0.520\,9 \pm 0.010\,4$
Yelp	Noisy	$0.447\,1 \pm 0.036\,9$
	Clean	$0.549\,4 \pm 0.006\,8$

Note: We use the DCN model as the base model. The average and variance of w_i for clean and noisy sequences across different datasets are analyzed.

In the context of future prospects, through adversarial networks, the well-trained noise recognition model can discriminate between artificially replaced noisy sequences, and is used as a discriminator to learn a generator that makes it unable to recognize whether the sequences has been artificially replaced with noise. At this point, the generator can create noisier sequences than those replaced by humans, which may be more similar to the noisy sequences in the original sequences. Therefore, using these generated sequences as noisy sequences might yield better results.

Conflict of Interest The authors declare that they have no conflict of interest.

References

[1] Zhou R, Khemmarat S, Gao L. The impact of YouTube recommendation system on video views. In *Proc. the 10th ACM SIGCOMM Conference on Internet Measurement*, Nov. 2010, pp.404–410. DOI: 10.1145/1879141.1879193.

[2] Lee D, Hosanagar K. Impact of recommender systems on sales volume and diversity. In *Proc. the 35th International Conference on Information Systems—Building a Better World through Information Systems*, Dec. 2014.

[3] Lin T H, Gao C, Li Y. CROSS: Cross-platform recommendation for social e-commerce. In *Proc. the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 2019, pp.515–524. DOI: 10.1145/3331184.3331191.

[4] Hidasi B, Karatzoglou A, Baltrunas L, Tikk D. Session-based recommendations with recurrent neural networks.

In *Proc. the 4th International Conference on Learning Representations*, May 2016.

[5] Ni Y, Ou D, Liu S, Li X, Ou W, Zeng A, Si L. Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks. In *Proc. the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2018, pp.596–605. DOI: 10.1145/3219819.3219828.

[6] Zhou G, Mou N, Fan Y, Pi Q, Bian W, Zhou C, Zhu X, Gai K. Deep interest evolution network for click-through rate prediction. In *Proc. the 33rd AAAI Conference on Artificial Intelligence*, Jul. 2019, pp.5941–5948. DOI: 10.1609/aaai.v33i01.33015941.

[7] Zhou G, Zhu X, Song C, Fan Y, Zhu H, Ma X, Yan Y, Jin J, Li H, Gai K. Deep interest network for click-through rate prediction. In *Proc. the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2018, pp.1059–1068. DOI: 10.1145/3219819.3219823.

[8] Chen H, Lin Y, Pan M, Wang L, Yeh C C M, Li X, Zheng Y, Wang F, Yang H. Denoising self-attentive sequential recommendation. In *Proc. the 16th ACM Conference on Recommender Systems*, Sept. 2022, pp.92–101. DOI: 10.1145/3523227.3546788.

[9] Davidson J, Liebald B, Liu J, Nandy P, Van Vleet T, Gargi U, Gupta S, He Y, Lambert M, Livingston B, Sampath D. The YouTube video recommendation system. In *Proc. the 4th ACM Conference on Recommender Systems*, Sept. 2010, pp.293–296. DOI: 10.1145/1864708.1864770.

[10] Wang J, Zhang Y. Opportunity model for e-commerce recommendation: Right product; right time. In *Proc. the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 28–Aug. 1, 2013, pp.303–312. DOI: 10.1145/2484028.2484067.

[11] Cheng H T, Koc L, Harmsen J, Shaked T, Chandra T, Aradhye H, Anderson G, Corrado G, Chai W, Ispir M, Anil R, Haque Z, Hong L, Jain V, Liu X, Shah H. Wide & deep learning for recommender systems. In *Proc. the 1st Workshop on Deep Learning for Recommender Systems*, Sept. 2016, pp.7–10. DOI: 10.1145/2988450.2988454.

[12] Guo H, Tang R, Ye Y, Li Z, He X. DeepFM: A factorization-machine based neural network for CTR prediction. In *Proc. the 26th International Joint Conference on Artificial Intelligence*, Aug. 2017, pp.1725–1731.

[13] Lin W, Zhao X, Wang Y, Xu T, Wu X. AdaFS: Adaptive feature selection in deep recommender system. In *Proc. the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Aug. 2022, pp.3309–3317. DOI: 10.1145/3534678.3539204.

[14] Wang R, Fu B, Fu G, Wang M. Deep & cross network for ad click predictions. In *Proc. the ADKDD2017*, Aug. 2017, Article No. 12. DOI: 10.1145/3124749.3124754.

[15] Fang H, Zhang D, Shu Y, Guo G. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Trans. Information Systems*, 2020, 39(1): Article No. 10. DOI: 10.1145/3426723.

[16] Wang T, Xia L, Huang C. Denoised self-augmented learning for social recommendation. In *Proc. the 32nd International Joint Conference on Artificial Intelligence*, Aug. 2023, pp.2324–2331.

[17] Fan Z, Xu K, Dong Z, Peng H, Zhang J, Yu P S. Graph collaborative signals denoising and augmentation for recommendation. In *Proc. the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 2023, pp.2037–2041. DOI: 10.1145/3539618.3591994.

[18] Wang W, Feng F, He X, Nie L, Chua T S. Denoising implicit feedback for recommendation. In *Proc. the 14th ACM International Conference on Web Search and Data Mining*, Mar. 2021, pp.373–381. DOI: 10.1145/3437963.3441800.

[19] Gantner Z, Drumond L, Freudenthaler C, Schmidt-Thieme L. Personalized ranking for non-uniformly sampled items. In *Proc. the 2011 International Conference on KDD Cup 2011*, Jan. 2011, pp.231–247.

[20] Hu K, Li L, Xie Q, Liu J, Tao X. What is next when sequential prediction meets implicitly hard interaction? In *Proc. the 30th ACM International Conference on Information & Knowledge Management*, Nov. 2021, pp.710–719. DOI: 10.1145/3459637.3482492.

[21] Wang Z, Xu Q, Yang Z, Cao X, Huang Q. Implicit feedbacks are not always favorable: Iterative relabeled one-class collaborative filtering against noisy interactions. In *Proc. the 29th ACM International Conference on Multimedia*, Oct. 2021, pp.3070–3078. DOI: 10.1145/3474085.3475446.

[22] Qin Y, Wang P, Li C. The world is binary: Contrastive learning for denoising next basket recommendation. In *Proc. the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 2021, pp.859–868. DOI: 10.1145/3404835.3462836.

[23] Yu W, Qin Z. Sampler design for implicit feedback data by noisy-label robust learning. In *Proc. the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 2020, pp.861–870. DOI: 10.1145/3397271.3401155.

[24] Wang Y, Xin X, Meng Z, Jose J M, Feng F, He X. Learning robust recommenders through cross-model agreement. In *Proc. the 2022 ACM Web Conference*, Apr. 2022, pp.2015–2025. DOI: 10.1145/3485447.3512202.

[25] Lin W, Zhao X, Wang Y, Zhu Y, Wang W. AutoDenoise: Automatic data instance denoising for recommendations. In *Proc. the 2023 ACM Web Conference*, Apr. 30–May 4, 2023, pp.1003–1011. DOI: 10.1145/3543507.3583339.

[26] He R, Kang W C, McAuley J. Translation-based recommendation. In *Proc. the 11th ACM Conference on Recommender Systems*, Aug. 2017, pp.161–169. DOI: 10.1145/3109859.3109882.

[27] He R, McAuley J. Fusing similarity models with Markov chains for sparse sequential recommendation. In *Proc. the 16th IEEE International Conference on Data Mining* (*ICDM*), Dec. 2016, pp.191–200. DOI: 10.1109/ICDM.2016.0030.

[28] Medsker L R, Jain L. Recurrent neural networks. *Design and Applications*, 2001, 5: 64–67.

[29] Graves A. Long short-term memory. In *Supervised Sequence Labelling with Recurrent Neural Networks*, Graves A (ed.), Springer, 2012, pp.37–45. DOI: 10.1007/978-3-642-24797-2_4.

[30] Han K, Xiao A, Wu E, Guo J, Xu C, Wang Y. Transformer in transformer. In *Proc. the 34th Advances in Neural Information Processing Systems*, Dec. 2021, pp.15908–15919.

[31] Fan X, Liu Z, Lian J, Zhao W X, Xie X, Wen J R. Lighter and better: Low-rank decomposed self-attention networks for next-item recommendation. In *Proc. the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 2021, pp.1733–1737. DOI: 10.1145/3404835.3462978.

[32] Tan Q, Zhang J, Yao J, Liu N, Zhou J, Yang H, Hu X. Sparse-interest network for sequential recommendation. In *Proc. the 14th ACM International Conference on Web Search and Data Mining*, Mar. 2021, pp.598–606. DOI: 10.1145/3437963.3441811.

[33] Xie X, Sun F, Liu Z, Wu S, Gao J, Zhang J, Ding B, Cui B. Contrastive learning for sequential recommendation. In *Proc. the 38th IEEE International Conference on Data Engineering* (*ICDE*), May 2022, pp.1259–1273. DOI: 10.1109/ICDE53745.2022.00099.

[34] Huang L, Ma Y, Liu Y, Du B D, Wang S, Li D. Position-enhanced and time-aware graph convolutional network for sequential recommendations. *ACM Trans. Information Systems*, 2023, 41(1): Article No. 6. DOI: 10.1145/3511700.

[35] He X, Deng K, Wang X, Li Y, Zhang Y, Wang M. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proc. the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 2020, pp.639–648. DOI: 10.1145/3397271.3401063.

[36] Mao K, Zhu J, Xiao X, Lu B, Wang Z, He X. UltraGCN: Ultra simplification of graph convolutional networks for recommendation. In *Proc. the 30th ACM International Conference on Information & Knowledge Management*, Nov. 2021, pp.1253–1262. DOI: 10.1145/3459637.3482291.

[37] Fan Z, Liu Z, Zhang J, Xiong Y, Zheng L, Yu P S. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *Proc. the 30th ACM International Conference on Information & Knowledge Management*, Nov. 2021, pp.433–442. DOI: 10.1145/3459637.3482242.

[38] Kang W C, McAuley J. Self-attentive sequential recommendation. In *Proc. the 2018 IEEE International Conference on Data Mining* (*ICDM*), Nov. 2018, pp.197–206. DOI: 10.1109/ICDM.2018.00035.

[39] Zhou K, Wang H, Zhao W X, Zhu Y, Wang S, Zhang F, Wang Z, Wen J R. S3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proc. the 29th ACM International Conference on Information & Knowledge Management*, Oct. 2020, pp.1893–1902. DOI: 10.1145/3340531.3411954.

[40] Hu J, Chan Z, Zhang Y, Han S, Lou S, Liu B, Zhu H, Jiang Y, Xu J, Zheng B. PS-SA: An efficient self-attention via progressive sampling for user behavior sequence modeling. In *Proc. the 32nd ACM International Conference on Information and Knowledge Management*, Oct. 2023, pp.4639–4645. DOI: 10.1145/3583780.3615495.

[41] Du X, Yuan H, Zhao P, Qu J, Zhuang F, Liu G, Liu Y, Sheng V S. Frequency enhanced hybrid attention network for sequential recommendation. In *Proc. the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 2023, pp.78–88. DOI: 10.1145/3539618.3591689.

[42] Hjelm R D, Fedorov A, Lavoie-Marchildon S, Grewal K, Bachman P, Trischler A, Bengio Y. Learning deep representations by mutual information estimation and maximization. In *Proc. the 7th International Conference on Learning Representations*, May 2019.

[43] Qiu R, Huang Z, Yin H, Wang Z. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proc. the 15th ACM International Conference on Web Search and Data Mining*, Feb. 2022, pp.813–823. DOI: 10.1145/3488560.3498433.

[44] Hao B, Yin H, Zhang J, Li C, Chen H. A multi-strategy-based pre-training method for cold-start recommendation. *ACM Trans. Information Systems*, 2023, 41(2): 31. DOI: 10.1145/3544107.

[45] Chen Y, Liu Z, Li J, McAuley J, Xiong C. Intent contrastive learning for sequential recommendation. In *Proc. the 2022 ACM Web Conference*, Apr. 2022, pp.2172–2182. DOI: 10.1145/3485447.3512090.

[46] Yu J, Yin H, Xia X, Chen T, Cui L, Nguyen Q V H. Are graph augmentations necessary?: Simple graph contrastive learning for recommendation. In *Proc. the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 2022, pp.1294–1303. DOI: 10.1145/3477495.3531937.

[47] Hao B, Zhang J, Yin H, Li C, Chen H. Pre-training graph neural networks for cold-start users and items representation. In *Proc. the 14th ACM International Conference on Web Search and Data Mining*, Mar. 2021, pp.265–273. DOI: 10.1145/3437963.3441738.

[48] Yu J, Yin H, Li J, Wang Q, Hung N Q V, Zhang X. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proc. the 2021 Web Conference*, Apr. 2021, pp.413–424. DOI: 10.1145/3442381.3449844.

[49] Xia X, Yin H, Yu J, Wang Q, Cui L, Zhang X. Self-supervised hypergraph convolutional networks for session-based recommendation. In *Proc. the 35th AAAI Conference on Artificial Intelligence*, May 2021, pp.4503–4511. DOI: 10.1609/aaai.v35i5.16578.

[50] Yu J, Yin H, Gao M, Xia X, Zhang X, Hung N Q V. Socially-aware self-supervised tri-training for recommendation. In *Proc. the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, Aug. 2021, pp.2084–2092. DOI: 10.1145/3447548.3467340.

[51] Wilson D R, Martinez T R. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 2000, 38(3): 257–286. DOI: 10.1023/A:1007626913721.

[52] Leyva E, Gonzalez A, Pérez R. Three new instance selec-

tion methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognition*, 2015, 48(4): 1523–1537. DOI: 10.1016/j.patcog.2014.10.001.

[53] Zhang J, Hao B, Chen B, Li C, Chen H, Sun J. Hierarchical reinforcement learning for course recommendation in MOOCs. In *Proc. the 33rd AAAI Conference on Artificial Intelligence*, Jul. 2019, pp.435–442. DOI: 10.1609/aaai.v33i01.3301435.

[54] Qiu R, Huang Z, Yin H. Memory augmented multi-instance contrastive predictive coding for sequential recommendation. In *Proc. the 2021 IEEE International Conference on Data Mining* (*ICDM*), Dec. 2021, pp.519–528. DOI: 10.1109/ICDM51629.2021.00063.

[55] Wang Y, Zhang H, Liu Z, Yang L, Yu P S. ContrastVAE: Contrastive variational autoencoder for sequential recommendation. In *Proc. the 31st ACM International Conference on Information & Knowledge Management*, Oct. 2022, pp.2056–2066. DOI: 10.1145/3511808.3557268.

[56] Fan Z, Liu Z, Wang S, Zheng L, Yu P S. Modeling sequences as distributions with uncertainty for sequential recommendation. In *Proc. the 30th ACM International Conference on Information & Knowledge Management*, Nov. 2021, pp.3019–3023. DOI: 10.1145/3459637.3482145.

[57] Liu Y, Xuan H, Li B, Wang M, Chen T, Yin H. Self-supervised dynamic hypergraph recommendation based on hyper-relational knowledge graph. In *Proc. the 32nd ACM International Conference on Information and Knowledge Management*, Oct. 2023, pp.1617–1626. DOI: 10.1145/3583780.3615054.

[58] Wu J, Wang X, Feng F, He X, Chen L, Lian J, Xie X. Self-supervised graph learning for recommendation. In *Proc. the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 2021, pp.726–735. DOI: 10.1145/3404835.3462862.

Peng-Sheng Liu is an undergraduate student at Guizhou University, Guiyang. His research interests include big data, recommender systems, and graph mining.

Li-Nan Zheng is a Master's student at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His research interests include recommender systems and information retrieval.

Jia-Le Chen is a Ph.D. student at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His research interests include recommender systems and search engines.

Guang-Fa Zhang received his B.S. degree from Anhui Jianzhu University, Hefei, in 2009, and his M.S. and Ph.D. degrees in computer application from the University of Chinese Academy of Sciences, Beijing, in 2017 and 2024, respectively. He is currently an engineer at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His research interests include e-commerce big data analysis, disciplinary inspection big data analysis, intelligent search engines, recommender systems, and spatial big data analysis.

Yang Xu received his B.S. degree from Guizhou University, Guiyang, in 2003, and his Ph.D. degree in engineering from the Institute of Modern Materials and Mechanics, Chinese Academy of Sciences, Beijing, in 2008. He is currently a master tutor and an associate professor with Guizhou University, Guiyang. His current research interests include big data acquisition, artificial intelligence, and the Internet of Things.

Jin-Yun Fang received his B.S. degree from China University of Geosciences, Wuhan, in 1990, and his Ph.D. degree in engineering from China University of Geosciences, Wuhan, in 1999. He is currently a professor at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His current research interests include big data, geographic information systems, recommender systems, and search engines.