Hierarchical Item Inconsistency Signal Learning for Sequence Denoising in Sequential Recommendation

Chi Zhang
Harbin Engineering University
Harbin, China
zhangchi20@hrbeu.edu.cn

Yantong Du
Harbin Engineering University
Harbin, China
duyantong94@hrbeu.edu.cn

Xiangyu Zhao*
City University of Hong Kong
Hong Kong
xianzhao@cityu.edu.hk

Qilong Han*
Harbin Engineering University
Harbin, China
hanqilong@hrbeu.edu.cn

Rui Chen*
Harbin Engineering University
Harbin, China
ruichen@hrbeu.edu.cn

Li Li
University of Delaware
Newark, United States
lilee@udel.edu

ABSTRACT

Sequential recommender systems aim to recommend the next items in which target users are most interested based on their historical interaction sequences. In practice, historical sequences typically contain some inherent noise (e.g., accidental interactions), which is harmful to learn accurate sequence representations and thus misleads the next-item recommendation. However, the absence of supervised signals (i.e., labels indicating noisy items) makes the problem of sequence denoising rather challenging. To this end, we propose a novel sequence denoising paradigm for sequential recommendation by learning hierarchical item inconsistency signals. More specifically, we design a hierarchical sequence denoising (HSD) model, which first learns two levels of inconsistency signals in input sequences, and then generates noiseless subsequences (i.e., dropping inherent noisy items) for subsequent sequential recommenders. It is noteworthy that HSD is flexible to accommodate supervised item signals, if any, and can be seamlessly integrated with most existing sequential recommendation models to boost their performance. Extensive experiments on five public benchmark datasets demonstrate the superiority of HSD over state-of-the-art denoising methods and its applicability over a wide variety of mainstream sequential recommendation models. The implementation code is available at https://github.com/zc-97/HSD.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Sequential recommendation, sequence denoising, contrastive learning, curriculum learning

*Corresponding authors.

ACM Reference Format:
Chi Zhang, Yantong Du, Xiangyu Zhao*, Qilong Han*, Rui Chen*, and Li Li. 2022. Hierarchical Item Inconsistency Signal Learning for Sequence Denoising in Sequential Recommendation. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3511808.3557348

1 INTRODUCTION

Sequential recommender systems aim to recommend the next items in which target users are most interested based on their historical interaction sequences. The majority of existing studies feed sequences into various deep models to learn sequence representations, including recurrent neural networks (RNNs) [7, 11, 35], convolutional neural networks (CNNs) [35], Transformer [15, 20, 32], and graph neural networks (GNNs) [46]. In practice, historical sequences typically contain some inherent noisy items (e.g., accidental interactions [36]), which cannot accurately reflect users' preferences and thus mislead the next-item recommendation.

Consequently, some pioneering studies have explored denoising approaches to reduce the influence of noise and thus learn better sequence representations. However, the absence of supervised signals (i.e., labels indicating noisy items) makes the problem of sequence denoising rather challenging. One line of research proposes to tackle this challenge in a "soft" way, which tries to implicitly reduce noise's influence on learning sequence representations. For example, attention-based methods [20, 21, 23, 24, 48, 53] assign lower attention weights to some less important items with respect to the final sequence representation. In contrast, FMLP-Rec [59] treats sequence representations as signals and further incorporates Fourier Transform to filter out noise. Although these ideas can learn better sequence representations, noisy items still exist in sequences and may jeopardize recommendation performance.

Another latest line of research relies on comparing item-level relevance to a target item to remove noisy items [29, 33, 37, 53]. Different from the previous line, this idea explicitly drops irrelevant items in sequences, which can more thoroughly purge noisy information. Nevertheless, it still has a significant drawback: historical sequences typically contain many items that are irrelevant to the next item, but may not be *inherent* noise. Eliminating them without careful discrimination may lose useful information. Motivated by the above observation, in this paper, we propose a novel sequence denoising paradigm for sequential recommendation by learning hierarchical item inconsistency signals. The key idea is to resort to more strict detectable signals to remove only reliable noisy items without dropping advantageous information.

More specifically, we consider two types of noisy item signals in sequences, including *user-level intent signals* and *sequence-level context signals*. Intuitively, informative items in a sequence should well reflect the user's personalized preference (i.e., user-level intent signals) and exhibit smooth sequentiality over surrounding items (i.e., sequence-level context signals). Sequential recommendation models should leverage only such items to learn high-quality sequence representations. For simplicity, we use "user-level signal" and "sequence-level signal" to mean the above signals. Naturally, we can make use of these two types of signals to identify inherent noisy items: items exhibiting inconsistency with respect to these signals are potentially noisy. However, in the absence of supervised signals, depending on user- or sequence-level signals alone may not be fully reliable. Therefore, we combine these signals to decide whether to consider an item as noisy while preserving as much useful information as possible. This general idea is illustrated in Figure 1. Given an input sequence, we use two levels of inconsistency signals to pinpoint noisy items and generate a clean subsequence for downstream sequential recommenders.

To instantiate the above idea, we propose a novel hierarchical sequence denoising (HSD) model, which consists of different signal generation layers for user- and sequence-level signals. The user-level signal generation layer is equipped with two discriminators to detect inconsistent items over the corresponding user's short- and long-term intents, while the sequence-level signal generation layer is powered by a context-aware autoencoder and another discriminator to learn contextual semantics and detect inconsistencies by comparing the difficulty levels of information reconstruction. Furthermore, two types of curriculum learning strategies are adopted in HSD to enhance the above layers' learning capabilities. We provide a case study in Section 3.4 to help understand how the learned signals affect the sequence denoising process.

We summarize our main technical contributions as follows.

- To the best of our knowledge, this is the first paper that proposes to learn hierarchical inconsistency signals for sequence denoising. The strict design choice enables to identify true inherent noisy items without requiring any additional prior knowledge (e.g., supervised item signals).
- We propose two types of useful signals to identify inherent noisy items in sequences. We show that sequence-level signals, which have not been studied before, are critical to sequence denoising and that the proposed two types of signals can complement each other to improve the reliability of identified noisy items.
- We propose a novel hierarchical sequence denoising (HSD) model to effectively learn different signals using different generation layers. Our solution is flexible to accommodate supervised item signals, if any, and can be seamlessly integrated into most existing sequential recommenders to boost their performance.
- Extensive experiments on five public benchmark datasets demonstrate the superiority of HSD over state-of-the-art denoising methods and its applicability over a wide variety of mainstream sequential recommendation models.

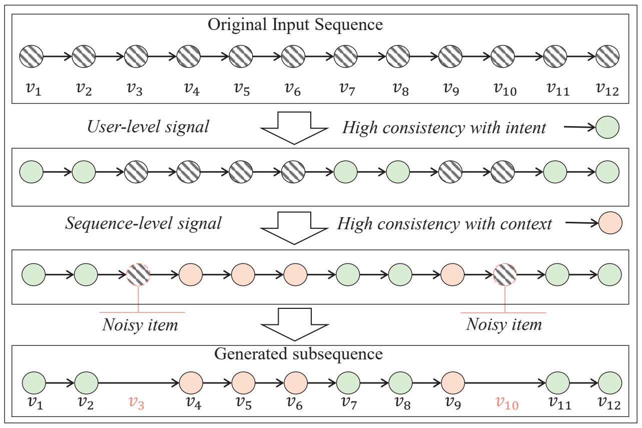
Original Input Sequence
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12}
User-level signal
High consistency with intent
Sequence-level signal
High consistency with context
Noisy item
Noisy item
Generated subsequence
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12}

Figure 1: An illustration of hierarchical item inconsistency signals for sequence denoising.

2 METHODOLOGY

Our denoising model takes as input a set of sequences (i.e., users' historical interactions) $\mathcal{S} = \{S^{u_1}, S^{u_2}, \cdots, S^{u_{|\mathcal{U}|}}\}$ over the user set $\mathcal{U} = \{u_1, u_2, \cdots, u_{|\mathcal{U}|}\}$ and item universe $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$. Each sequence is generated by a user u and defined as $S^u = [s^u_1, s^u_2, \cdots, s^u_n]$, where s^u_n is the last item interacted by user u and $s^u_i \in \mathcal{V}$. Our model aims to generate a noiseless subsequence $S^u_+ = [s^u_{1+}, s^u_{2+}, \cdots, s^u_{n+}]$ for each input sequence S^u, where $|S^u_+| \le |S^u|$. Then we take S^u_+ as the input to a downstream sequential recommender to learn the sequence representation and make recommendations.

As illustrated in Figure 2, HSD consists of four major components, including the user- and sequence-level signal generation layers to generate inconsistency signals for sequence denoising, the recommendation layer (i.e., an arbitrary sequential recommender) to learn the representation of the clean subsequence and the final prediction layer. In the following, we detail each component.

2.1 Embedding Layer

To train HSD, we generate item v_i's embedding by mapping its ID into a dense embedding vector $h_i \in \mathbb{R}^d$, where d is the dimension of the embedding vector. Specifically, we build a parameter matrix as an embedding look-up table for embedding initialization as

$$\boldsymbol{h}_{v_i} = \boldsymbol{x}_{v_i} \boldsymbol{W}_h, \tag{1}$$

where $\boldsymbol{W}_h$ is a trainable matrix, and $\boldsymbol{x}_{v_i}$ is the one-hot encoding of v_i's ID. Note that for Transformer-based recommenders, the position embeddings can be similarly learned from one-hot vectors and a trainable parameter matrix.

2.2 User-Level Signal Generation Layer

Users' historical interactions are typically long-term sequences, and the evolving user interests may cause inconsistencies between long-term and short-term interactions [25, 31, 58]. Our insight is that this phenomenon provides critical information for sequence denoising because noisy items typically differ from the user intent within the evolving process. Therefore, our key idea is to simulate the evolving process and detect inconsistencies as potential noisy items. To do so, we first disentangle user intents into long-term and

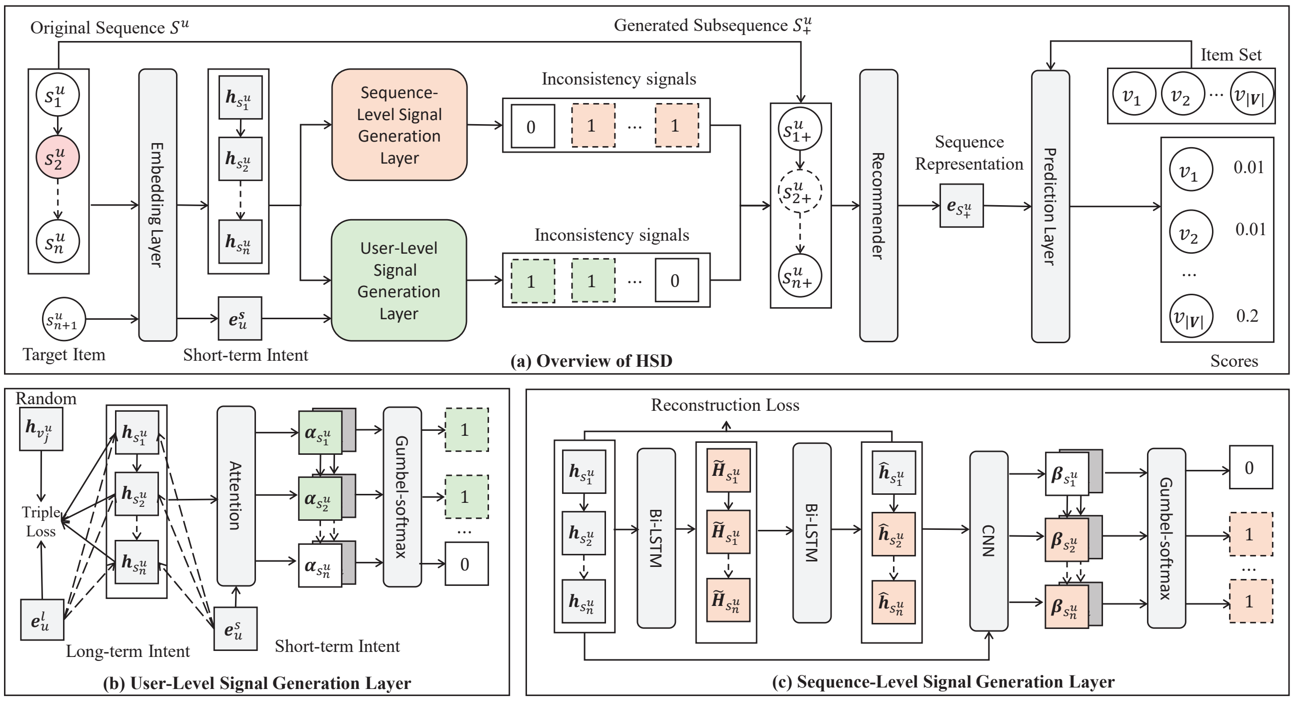

Figure 2: The overall architecture of the proposed HSD model.

short-term representations. Then, we simulate the evolving process by comparing each item with the above intents through different discriminators to detect inconsistencies.

To represent the short-term user intent, previous methods suppose that recent interactions are more critical to user intent representations. It is natural to use the last m items to well reflect short-term user intent [23, 24, 48], or as a query vector to assign weights [13, 20, 26, 46]. However, this is sub-optimal because the next item itself may raise a short-term intent shift. Therefore, we propose to leverage the actual next item as the short-term intent during the training process, which can satisfactorily avoid the problem. Furthermore, instead of utilizing the items straightly, we incorporate user embeddings to match all items in the sequence to learn the long-term user intent. This design has a notable advantage in that we can utilize sequentiality and personality simultaneously to reveal user intents in sequences. Consequently, we could formulate the short-term user intent $\boldsymbol{e}^s_u$ and long-term user intent $\boldsymbol{e}^l_u$ representations as follows:

$$\boldsymbol{e}^s_u = \boldsymbol{h}_{s^u_{n+1}}, \boldsymbol{e}^l_u = \boldsymbol{h}_u, \tag{2}$$

where $\boldsymbol{h}_{s^u_{n+1}} \in \mathbb{R}^d$ is the embedding of the input sequence S^u's next item s^u_{n+1}, and $\boldsymbol{h}_u \in \mathbb{R}^d$ is u's embedding, which can be similarly learned from one-hot vectors (i.e., u's ID) and a trainable parameter matrix. After that, we can leverage the above intent representations to query each item and obtain short-term and long-term inconsistency scores for short term and long term. An intuitive way is to calculate paired-wise similarity between the above intents and each item in a heuristic manner, e.g., cosine similarity. However, the straightforward similarity measure may mislead the subsequent detection process in the absence of supervised signals and additional prior knowledge. For this reason, we propose two discriminators to judge whether inconsistencies exist.

2.2.1 Short-Term User Intent Discriminator. First, we leverage a soft-attention mechanism as a discriminator to detect inconsistencies between items and the short-term intent. The key idea is to utilize the short-term intent representation as a query vector and assign different attention weights $\boldsymbol{\alpha}_{s^u_i}$ to each item $s^u_i \in S^u$ as

$$\boldsymbol{\alpha}_{s^u_i} = \sigma(\tanh(\boldsymbol{h}_{s^u_i}\boldsymbol{W}_1 + \boldsymbol{e}^s_u\boldsymbol{W}_2)\boldsymbol{W}_3), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function, $\boldsymbol{h}_{s^u_i} \in \mathbb{R}^d$ is item s^u_i's embedding, and $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d\times d}$, $\boldsymbol{W}_3 \in \mathbb{R}^{d\times 2}$ are trainable parameter matrices. Note that $\boldsymbol{\alpha}_{s^u_i} \in \mathbb{R}^2$, where the first dimension indicates the consistency between $\boldsymbol{h}_{s^u_i}$ and the short-term user intent representation $\boldsymbol{e}^s_u$, while the second dimension means inconsistency. Therefore, the scores can be treated as a binary distribution (i.e., consistency vs. inconsistency). To generate binary values (i.e., 0 vs. 1) and facilitate gradient back-propagation, we utilize a Gumbel-softmax function [39, 45, 52] suggested by CLEA [29] to support model learning via

$$\begin{aligned}\boldsymbol{a}_{s^u_i} &= \text{Gumbel-softmax}(\boldsymbol{\alpha}_{s^u_i}, \tau),\\ \boldsymbol{a}_{s^u_i} &= \frac{\exp(\log(\boldsymbol{\alpha}_{s^u_i,j}) + g_j)/\tau}{\sum_{j=0}^{1}\exp(\log(\boldsymbol{\alpha}_{s^u_i,j}) + g_j)/\tau},\end{aligned} \tag{4}$$

where $\boldsymbol{a}_{s^u_i} \in \mathbb{R}^2$ indicates whether an item s^u_i is consistent with the short-term user intent, $\tau > 0$ is the temperature parameter to control the selection distribution, and g_j is i.i.d sampled from a Gumbel distribution as noisy disturber. When $\tau \rightarrow 0$, $\boldsymbol{a}_{s_i^u}$ approximates a one-hot vector (i.e., hard selection). When $\tau \rightarrow \infty$, $\boldsymbol{a}_{s_i^u}$ approximates a uniform distribution. When $\tau \rightarrow 1$, the Gumbel-softmax function is identical to the general Softmax function.

2.2.2 Long-Term User Intent Discriminator. Unlike the short-term user intent that can be explicitly represented based on the next item in the training process, detecting inconsistent items with respect to the long-term user intent renders non-trivial technical challenges and needs a careful design. Intuitively, we can leverage a similar discriminator to detect items inconsistent with the long-term user intent. However, this attempt may be sub-optimal according to the following reasons. (1) Due to the complementary principle in multi-view learning [9, 47, 49], two similar discriminators may not be able to distinguish between long-term and short-term user intents and only provide less information. (2) Without another optimization objective function, the long-term user intent representation may be misled. Consequently, we propose to train another long-term user intent "discriminator" and gradually enhance its confidence. Specifically, we resort to the contrastive learning paradigm [5, 57] to detect inconsistencies through the training loss. Accordingly, we can enforce the long-term user intent to pull items in the sequence and push a randomly sampled item during training. Therefore, we deem that inconsistencies occur when the long-term user intent is difficult to fit a few items. Mathematically, we could formulate this process as a triple loss function for each user u to pull his/her interactions and push a randomly sampled one via

$$\begin{aligned} \mathcal{L}_u &= \frac{1}{n}\sum_{s_i^u \in S^u} \mathcal{L}(u, s_i^u, v_j^u), \\ \mathcal{L}(u, s_i^u, v_j^u) &= -\log(\sigma(\mathrm{sim}(\boldsymbol{e}_u^l, \boldsymbol{h}_{s_i^u}) - \mathrm{sim}(\boldsymbol{e}_u^l, \boldsymbol{h}_{v_j^u}))), \end{aligned} \tag{5}$$

where v_j^u is the randomly sampled item not in user u's sequence, and $\mathrm{sim}(\cdot)$ is a similarity calculation function, e.g., inner product.

However, the long-term user intent representation may be unreliable (i.e., pull noisy items) during the training process. To tackle this problem, we leverage the idea of curriculum learning [1, 4, 41] to derive a hard-to-easy strategy, which, unlike the general easy-to-hard strategy, learns a robust long-term user intent representation. Inspired by Wang *et al.* [40], we rank all items' loss values and set the loss values of the highest μ percent of items to 0, where μ is a dynamic ratio parameter. It is because these items are harder to fit by the user's long-term intent than others in the sequence and are considered inconsistencies. Specifically, during the training progress, we smoothly increase μ from 0 to an upper bound of M until epoch T with increasing training iterations, where M and T are pre-defined parameters. Therefore, at the early stage of the training process, we enforce the long-term intent representation to pull all interactions to learn more information and enhance its robustness by gradually discarding "difficult" items (i.e., with higher loss values) to perform the hard-to-easy strategy. Accordingly, we can treat the well-learned long-term user intent representation as a more robust user intent query vector and feed it to the short-term inconsistency discriminator (i.e., Eq. (4)) to identify potential noisy items during the inference process.

2.3 Sequence-Level Signal Generation Layer

With the above generation layer, we can learn which items are inconsistent with a user's intents. However, since the generated signals are pseudo-labels without supervised signals, the detected inconsistencies may not be the inherent noisy items in sequences. To this end, we further introduce sequence-level signals to improve the reliability of detected inherent noise.

2.3.1 Context-Aware Autoencoder. Recall that informative items in a sequence normally exhibit smooth sequentiality over surrounding items while noisy items do not. Therefore, once we fuse each item with its context to learn contextual semantics, noisy items should be difficult to be reconstructed from the mixture, and we can compare the difficulty levels of information reconstruction to detect inconsistencies. Consequently, we propose a context-aware autoencoder based on a bi-directional long-short term memory neural network (Bi-LSTM) to learn contextual semantics and perform information reconstruction. More specifically, we input a sequence to the Bi-LSTM and combine (e.g., sum pooling) the bi-directional hidden states of each item as an encoded vector. After that, we again input this state sequence to the same Bi-LSTM to reconstruct each item's original information. We can formulate this process via

$$\begin{aligned} \tilde{\boldsymbol{H}}_{S^u}^L, \tilde{\boldsymbol{H}}_{S^u}^R &= \text{Bi-LSTM}(\boldsymbol{H}_{S^u}, \Theta_1), \\ \tilde{\boldsymbol{H}}_{S^u} &= \tilde{\boldsymbol{H}}_{S^u}^L + \tilde{\boldsymbol{H}}_{S^u}^R, \\ \hat{\boldsymbol{H}}_{S^u}^L, \hat{\boldsymbol{H}}_{S^u}^R &= \text{Bi-LSTM}(\tilde{\boldsymbol{H}}_{S^u}, \Theta_1), \\ \hat{\boldsymbol{H}}_{S^u} &= \hat{\boldsymbol{H}}_{S^u}^L + \hat{\boldsymbol{H}}_{S^u}^R, \end{aligned} \tag{6}$$

where $\boldsymbol{H}_{S^u} = [\boldsymbol{h}_{s_1^u}, \boldsymbol{h}_{s_2^u}, \cdots, \boldsymbol{h}_{s_n^u}] \in \mathbb{R}^{n \times d}$ consists of the embeddings of all items in S^u, $\tilde{\boldsymbol{H}}_{S^u}^L, \tilde{\boldsymbol{H}}_{S^u}^R \in \mathbb{R}^{n \times d}$ are encoded bi-directional hidden states, $\hat{\boldsymbol{H}}_{S^u} \in \mathbb{R}^{n \times d}$ is the decoded embedding sequence, and Θ_1 is the set of trainable parameters of the Bi-LSTM. After obtaining the decoded embedding sequence, we can calculate the reconstruction loss for the original input sequence S^u via

$$\mathcal{L}_{S^u} = \frac{1}{n}\sum_{j=1}^{n}(\hat{\boldsymbol{h}}_{s_i^u} - \boldsymbol{h}_{s_i^u})^2. \tag{7}$$

We could minimize Eq. (7) to improve the reconstruction ability. However, at the beginning of the training process, the noisy items may mislead the autoencoder reconstruction (i.e., higher loss), making the decoded results unreliable for inconsistency detection. An intuitive way is to pre-train the autoencoder and leverage it to detect noisy items. However, it may not work because there is no absolute clean training dataset to train the autoencoder. In practice, it is reasonable to assume that there are only a few noisy items in a dataset, and thus we can leverage the idea of curriculum learning again to tackle the above problem. Since $\mathcal{L}_{S^u}$ accumulates the reconstruction loss values of all items in the sequence S^u, relatively clean sequences (i.e., containing less noise) should have lower loss values. Thus, at the early stage of training, we only use "easy" sequences (i.e., with lower loss values) in each mini-batch to train the autoencoder, and gradually increase the difficulty level to perform a general easy-to-hard strategy, which is different from the hard-to-easy strategy used in Section 2.2. Specifically, during the training process, we rank all sequences' loss values in each mini-batch according to Eq. (7), and set the loss values of the highest $M - \mu$ percent of sequences to 0 until epoch T.

2.3.2 Sequence-Level Context Discriminator. According to Eq. (6), we can obtain each item's original and decoded vectors. We could feed them into a fully connected layer to calculate inconsistency scores. However, minimizing the reconstruction loss makes the difference between the above representations difficult to detect. Inspired by ConvKB [27], we leverage a convolutional neural network to enhance the detection ability for capturing dimension-wise differences. Specifically, we concatenate each item's original embedding with the decoded one and utilize a convolution operator to preserve dimension-wise information and detect the subtle differences via

$$\begin{aligned} \boldsymbol{c}_{s_i^u} &= \mathrm{Conv}^i([\hat{\boldsymbol{h}}_{s_i^u} \| \boldsymbol{h}_{s_i^u}], \Theta_2^i), \\ \boldsymbol{\beta}_{s_i^u} &= \sigma(\boldsymbol{c}_{s_i^u} \boldsymbol{W}_4), \end{aligned} \tag{8}$$

where Conv$(\cdot)$ is a 2-d convolution operation with filter size 2×1 and stride 1, Θ_2^i is the trainable parameters for each channel (i.e., item), $\boldsymbol{W}_4 \in \mathbb{R}^{2d \times 2}$ is a trainable parameter matrix, and $\|$ is the concatenation operation. Similarly, $\boldsymbol{\beta}_{s_i^u} \in \mathbb{R}^2$, where the first dimension indicates the consistency between $\boldsymbol{h}_{s_i^u}$ and $\hat{\boldsymbol{h}}_{s_i^u}$, while the second dimension indicates the inconsistency. Since $\boldsymbol{\beta}_{s_i^u}$ is also a binary distribution, we use a similar process to generate binary values for $\boldsymbol{\beta}_{s_i^u}$ via

$$\begin{aligned} \boldsymbol{b}_{s_i^u} &= \text{Gumbel-softmax}(\boldsymbol{\beta}_{s_i^u}, \tau), \\ \boldsymbol{b}_{s_i^u} &= \frac{\exp(\log(\boldsymbol{\beta}_{s_i^u,j}) + g_j)/\tau}{\sum_{j=0}^{1} \exp(\log(\boldsymbol{\beta}_{s_i^u,j}) + g_j)/\tau}, \end{aligned} \tag{9}$$

where τ is the same temperature parameter used in Eq. (4) to tune the learned distribution from the Gumbel-softmax function.

2.4 Sequential Recommender

With the above generation layers, we are ready to identify inherent noisy items in sequences from both user- and sequence-level signals. A first attempt is to identify an item as noise if it is inconsistent with either the corresponding user intent or the sequence context. However, in practice, it will introduce many false positives and lose advantageous information for sequential recommendation. Thus, we propose to design more strict signals to remove only reliable noise without losing useful information. We can treat the above inconsistency generation processes as two different views of noisy items and consider items as inherent noise only if these two views detect some inconsistencies simultaneously. This is generally backed up by the consensus principle.

Formally, we generate the noiseless subsequence S_+^u from the input sequence S^u via

$$\begin{aligned} r_{s_i^u} &= a_{s_i^u} b_{s_i^u}, \\ \boldsymbol{H}_+^u &= [r_{s_1^u}\boldsymbol{h}_{s_1^u}, r_{s_2^u}\boldsymbol{h}_{s_2^u}, \cdots, r_{s_n^u}\boldsymbol{h}_{s_n^u}], \end{aligned} \tag{10}$$

where $r_{s_i^u} \in \{1, 0\}$ indicates whether an item s_i^u is noisy (i.e., $r_{s_i^u} = 1$) or not ($a_{s_i^u}$ and $b_{s_i^u}$ are the second dimension of $\boldsymbol{a}_{s_i^u}$ and $\boldsymbol{b}_{s_i^u}$, respectively). Note that we apply the signals to the original item embedding sequence $\boldsymbol{H}_S^u$, instead of the original input sequence S^u, to support gradient back-propagation. After that, we can feed $\boldsymbol{H}_+^u$ into various mainstream sequential recommendation models to learn better sequence representations as follows:

$$\boldsymbol{e}_{S_+^u} = \mathrm{F}(\boldsymbol{H}_+^u), \tag{11}$$

where F$(\cdot)$ is a sequential recommendation model based on representation learning, which takes as input the noiseless item embedding sequence $\boldsymbol{H}_+^u$, and outputs sequence representations.

2.5 Prediction and Model Optimization

After obtaining the noiseless representations, we make the next-item recommendation by computing a probability distribution of the next item over the entire item universe. For each candidate item $v_i \in \mathcal{V}$, we can calculate its relevance to the sequence via

$$z_i = \boldsymbol{e}_{S_+^u} \boldsymbol{h}_{v_i}^{\top}. \tag{12}$$

Then the predicted probability of the next item being v_i (i.e., $\hat{y}_i$) can be computed by

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_{v_j \in \mathcal{V}} \exp(z_j)}. \tag{13}$$

Therefore, we can formulate the recommendation task as minimizing the cross-entropy of the prediction results $\hat{y}_i$:

$$\mathcal{L}(\mathrm{y}, \hat{\mathrm{y}}) = -\sum_{i=1}^{|\mathcal{V}|} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)), \tag{14}$$

where y denotes the one-hot encoding of the ground truth item.

Finally, we train our model with the sequential recommender by minimizing the total loss function according to Eq. (5), Eq. (7) and Eq. (14) as follows:

$$Loss = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} (\mathcal{L}(\mathrm{y}, \hat{\mathrm{y}}) + \mathcal{L}_u + \mathcal{L}_{S^u}) + \lambda \|\Theta\|_2^2, \tag{15}$$

where λ is a hyper-parameter controlling the strength of L_2 regularization, and Θ is the set of model parameters.

3 EXPERIMENTS

In this section, we perform comprehensive experiments on our proposed HSD model and a large number of state-of-the-art sequential recommendation models over five public real-world datasets. Specifically, our experiments are devised to answer the key research questions as follows:

- **RQ1**: How does integrating HSD into different mainstream sequential models perform compared with the original ones and other state-of-the-art denoising methods?
- **RQ2**: How do different inconsistency signal generation layers in HSD contribute to the performance?
- **RQ3**: How do the learned inconsistency signals in HSD affect the sequence denoising process?

3.1 Experimental Settings

3.1.1 Datasets. We conduct experiments on five datasets widely used in the literature to evaluate the effectiveness of HSD: (1) **MovieLens**[1]: It contains users' reviews and ratings on movies and is a popular dataset for sequential recommendation models. We use both the 100k and 1M versions (ML-100k and ML-1M for

[1]https://movielens.org/

Table 1: Statistics of the datasets

Dataset	# Sequences	# users	# Items	# Sparsity
ML-100k	99,287	944	1,350	92.21%
ML-1M	999,611	6,041	3,417	95.16%
Beauty	198,502	22,364	12,102	99.93%
Sports	296,337	35,599	18,358	99.95%
Yelp	317,078	30,495	20,062	99.95%

short) in the experiments. (2) **Amazon-Beauty and Sports**[2]: the Amazon datasets record users' historical purchases over abundant products and are also widely used in previous works[37, 40, 59]. We conduct experiments on two representative subcategories: beauty and sports. (3) **Yelp**[3]: It is another popular dataset for business recommendations, which contains user reviews of restaurants and bars. As its large size, we only use the transaction records after January 1st, 2019. Following the previous studies [12, 30, 32, 59], we filter out infrequent items whose frequency is less than 5 and short sequences with less than 5 items. We set the maximum sequence length to 200 for ML-1M and 50 for all other datasets.

Identical to the previous methods[37, 53, 59], we group user interactions in chronological order on all datasets, and split data by the leave-one-out strategy. For example, for an input sequence $S^u = [s_1^u, \cdots, s_n^u, s_{n+1}^u, s_{n+2}^u, s_{n+3}^u]$, we use $([s_1^u, \cdots, s_n^u], s_{n+1}^u)$ for training, $([s_1^u, \cdots, s_{n+1}^u], s_{n+2}^u)$ for validation, and $([s_1^u, \cdots, s_{n+2}^u], s_{n+3}^u)$ for testing. We employ a variety of common evaluation metrics adopted in literature [55, 59] to evaluate the performance of all methods, including *Hit Ratio* (HR@K), *Normalized Discounted Cumulative Gain* (NDCG@K, N@K for short), and *Mean Reciprocal Rank* (MRR@K). In this paper, we report results on HR@{5,10,20}, NDCG@{5,10,20} and MRR@20 (MRR for short) over the entire item universe (i.e., full ranking), instead of sampling, to avoid the bias introduced by the sampling process [2, 19]. The properties of the preprocessed datasets are summarized in Table 1.

3.1.2 *Competing Models.*

To demonstrate the effectiveness of our proposed sequence denoising paradigm, we integrate HSD with a wide range of representative methods, and compare the boosted performance with the state-of-the-art denoising methods. We consider the following sequential recommendation models: (1) **GRU4Rec** [11] uses gated recurrent units (GRUs) to learn session representations. (2) **NARM** [20] leverages a hybrid encoder with an attention mechanism to model user sequential behavior. (3) **STAMP** [23] equips with a short-term attention/memory priority model to capture user intent. (4) **CASER** [35] leverages horizontal and vertical convolutions for sequential recommendation. (5) **SASRec** [15] uses the multi-head attention mechanism to learn sequence representations. (6) **BERT4Rec** [32] leverages deep bidirectional Transformer and a mask & fill task to enhance sequence representations.

We consider the following denoising methods: (1) **DSAN** [53] explores a trainable virtual target item embedding to model the user's current preference and applies an adaptive sparse transformation function to eliminate the effect of unrelated items. (2) **FMLP-Rec** [59] incorporates a Fast Fourier Transform (FFT) and an inverse FFT procedure to reduce the potential influence of noise to learn better sequence representations.

[2]http://jmcauley.ucsd.edu/data/amazon/
[3]https://www.yelp.com/dataset

3.1.3 *Implementation Details.*

Identical to the settings of previous methods [20, 23, 44, 46, 53], we fix the default embedding size to 100 and mini-batch size to 256 for all methods, and the embedding parameters are initialized with a Gaussian distribution. We use the Adam optimizer [18] with a default learning rate of 0.001, and adopt the early-stopping training strategy if the HR@20 performance on the validation set decreases for 10 continuous epochs. The L_2 regularization coefficient is searched in $\{0, 10^{-3}, 10^{-4}\}$. In particular, we set the default dynamic ratio parameter μ's upper bound as $M = 20$ and smoothly increase μ from 0 to M until epoch $T = 10$ to control the curriculum learning process, which is determined by the 20/80 principle [34]. Following previous studies [14, 29], we set the initial temperature parameter as $\tau = 0.5$ and anneal it after every 40 batches. The hyper-parameters of all competing models either follow the suggestions from the original papers or are carefully tuned on the validation data, and the best performances are reported. We implement our model in PyTorch 1.7.1, Python 3.7.0, and RecBole v1.0.1 [56]. We conduct experiments on a workstation with an Intel Xeon Platinum 2.40GHz CPU, an NVIDIA Quadro RTX 8000 GPU, and 754GB RAM.

3.2 Overall Performance Comparison (RQ1)

We report the main experimental results in Table 2 and Table 3, where the best results are boldfaced and the second-best results are underlined. *Imprv* stands for the average improvements, and all improvements are significant by performing two-sided t-test with $p < 0.05$ over the baseline.

- From Table 2, it can be observed that the base models with HSD show significant improvements in most cases on all datasets. In particular, compared to BERT4Rec, HSD+BERT4Rec achieves relative average improvements of all evaluation metrics as 52.68%, 95.82%, 262.99%, 102.39% and 246.14% on ML-100k, ML-1M, Beauty, Sports and Yelp, respectively. Such results generally demonstrate the superiority and applicability of our solution over a wide variety of mainstream sequential recommendation models. Compared with the original methods, integrating with HSD can learn better sequence representations from the generated noiseless subsequences, and thus improve their performance in most cases. For the only exception (i.e., N@10 of NARM with or without HSD on ML-100k), the performance drop is marginal (0.0202 vs. 0.0201). We suspect that it is caused by the learning ability of NARM itself on this specific dataset.
- From Table 3, it can be seen that HSD+BERT4Rec consistently yields the best performance on all cases and that even the worst combinations (i.e., HSD+NARM and HSD+Caser) can achieve significant improvements in most cases. Such results generally demonstrate the superiority and applicability of our solution again. In particular, when comparing HSD with FMLP-Rec, we can observe the benefits of explicitly removing noisy items; when comparing HSD with DSAN, we can observe the benefits of removing only inherently noisy items, instead of the items irrelevant to a target item.

Table 2: Experimental results of different sequential recommendation methods with (w) or without (w/o) HSD on the five datasets. The best results are boldfaced, and Imprv indicates the average improvements over all metrics. All improvements are statistically significant (i.e., two-sided t-tests with $p < 0.05$).

Datasets	Metric	GRU4Rec		NARM		STAMP		Caser		SASRec		BERT4Rec	
		w/o	w	w/o	w	w/o	w	w/o	w	w/o	w	w/o	w
ML-100k	HR@5	0.0191	**0.0276**	0.0180	**0.0223**	0.0201	**0.0255**	0.0212	**0.0255**	0.0191	**0.0371**	0.0191	**0.0339**
	HR@10	0.0286	**0.0488**	0.0403	**0.0403**	0.0392	**0.0467**	0.0339	**0.0477**	0.0371	**0.0583**	0.0414	**0.0732**
	HR@20	0.0594	**0.0742**	0.0657	**0.0785**	0.0700	**0.0870**	0.0679	**0.0742**	0.0764	**0.1018**	0.0912	**0.1294**
	N@5	0.0104	**0.0173**	0.0132	**0.0144**	0.0115	**0.0151**	0.0113	**0.0154**	0.0114	**0.0225**	0.0117	**0.0178**
	N@10	0.0134	**0.0241**	**0.0202**	0.0201	0.0176	**0.0217**	0.0153	**0.0224**	0.0172	**0.0292**	0.0189	**0.0305**
	N@20	0.0212	**0.0306**	0.0267	**0.0299**	0.0253	**0.0320**	0.0238	**0.0291**	0.0270	**0.0401**	0.0315	**0.0447**
	MRR	0.0109	**0.0185**	0.0162	**0.0169**	0.0132	**0.0171**	0.0119	**0.0168**	0.0139	**0.0234**	0.0157	**0.0218**
	Imprv.		**63.91%**		**7.82%**		**26.78%**		**35.74%**		**71.88%**		**52.68%**
ML-1M	HR@5	0.0194	**0.0260**	0.0151	**0.0260**	0.0232	**0.0301**	0.0104	**0.0252**	0.0397	**0.0500**	0.0224	**0.0477**
	HR@10	0.0373	**0.0427**	0.0349	**0.0417**	0.0440	**0.0530**	0.0215	**0.0424**	0.0666	**0.0858**	0.0495	**0.0886**
	HR@20	0.0690	**0.0745**	0.0591	**0.0677**	0.0677	**0.0899**	0.0589	**0.0725**	0.1007	**0.1326**	0.0980	**0.1399**
	N@5	0.0135	**0.0152**	0.0080	**0.0162**	0.0150	**0.0190**	0.0063	**0.0161**	0.0207	**0.0282**	0.0132	**0.0297**
	N@10	0.0190	**0.0206**	0.0144	**0.0212**	0.0218	**0.0264**	0.0099	**0.0216**	0.0294	**0.0397**	0.0218	**0.0429**
	N@20	0.0270	**0.0286**	0.0205	**0.0277**	0.0278	**0.0357**	0.0194	**0.0291**	0.0379	**0.0515**	0.0339	**0.0558**
	MRR	0.0159	**0.0161**	0.0100	**0.0168**	0.0168	**0.0209**	0.0091	**0.0173**	0.0203	**0.0290**	0.0169	**0.0328**
	Imprv.		**9.68%**		**62.42%**		**25.63%**		**108.61%**		**36.26%**		**95.82%**
Beauty	HR@5	0.0077	**0.0131**	0.0120	**0.0187**	0.0080	**0.0201**	0.0072	**0.0106**	0.0242	**0.0267**	0.0060	**0.0261**
	HR@10	0.0135	**0.0229**	0.0209	**0.0311**	0.0135	**0.0325**	0.0133	**0.0183**	0.0386	**0.0449**	0.0127	**0.0447**
	HR@20	0.0256	**0.0378**	0.0367	**0.0495**	0.0231	**0.0505**	0.0235	**0.0290**	0.0561	**0.0674**	0.0204	**0.0683**
	N@5	0.0045	**0.0082**	0.0071	**0.0114**	0.0046	**0.0128**	0.0044	**0.0064**	0.0129	**0.0147**	0.0037	**0.0147**
	N@10	0.0064	**0.0113**	0.0099	**0.0153**	0.0064	**0.0168**	0.0064	**0.0089**	0.0175	**0.0205**	0.0059	**0.0207**
	N@20	0.0094	**0.0150**	0.0139	**0.0199**	0.0088	**0.0213**	0.0090	**0.0116**	0.0219	**0.0261**	0.0078	**0.0266**
	MRR	0.0051	**0.0088**	0.0077	**0.0118**	0.0049	**0.0133**	0.0051	**0.0068**	0.0122	**0.0146**	0.0044	**0.0151**
	Imprv.		**71.71%**		**52.54%**		**161.44%**		**38.06%**		**17.05%**		**262.99%**
Sports	HR@5	0.0064	**0.0097**	0.0090	**0.0096**	0.0071	**0.0097**	0.0069	**0.0077**	0.0113	**0.0131**	0.0055	**0.0135**
	HR@10	0.0114	**0.0152**	0.0138	**0.0169**	0.0123	**0.0155**	0.0115	**0.0129**	0.0175	**0.0224**	0.0104	**0.0224**
	HR@20	0.0183	**0.0230**	0.0223	**0.0282**	0.0182	**0.0252**	0.0178	**0.0214**	0.0268	**0.0348**	0.0167	**0.0365**
	N@5	0.0035	**0.0067**	0.0058	**0.0063**	0.0046	**0.0065**	0.0046	**0.0049**	0.0059	**0.0071**	0.0036	**0.0072**
	N@10	0.0051	**0.0085**	0.0073	**0.0087**	0.0062	**0.0084**	0.0061	**0.0065**	0.0079	**0.0100**	0.0051	**0.0101**
	N@20	0.0068	**0.0105**	0.0094	**0.0115**	0.0077	**0.0108**	0.0077	**0.0086**	0.0102	**0.0132**	0.0067	**0.0137**
	MRR	0.0036	**0.0070**	0.0059	**0.0070**	0.0048	**0.0069**	0.0049	**0.0052**	0.0055	**0.0071**	0.0040	**0.0073**
	Imprv.		**71.53%**		**17.12%**		**39.05%**		**8.85%**		**25.80%**		**102.39%**
yelp	HR@5	0.0057	**0.0104**	0.0113	**0.0180**	0.0060	**0.0147**	0.0045	**0.0169**	0.0293	**0.0334**	0.0087	**0.0292**
	HR@10	0.0102	**0.0180**	0.0187	**0.0248**	0.0099	**0.0216**	0.0084	**0.0198**	0.0352	**0.0446**	0.0159	**0.0408**
	HR@20	0.0184	**0.0317**	0.0315	**0.0349**	0.0161	**0.0345**	0.0146	**0.0251**	0.0439	**0.0613**	0.0273	**0.0593**
	N@5	0.0034	**0.0066**	0.0075	**0.0142**	0.0038	**0.0107**	0.0028	**0.0151**	0.0251	**0.0255**	0.0054	**0.0223**
	N@10	0.0048	**0.0090**	0.0099	**0.0164**	0.0051	**0.0129**	0.0040	**0.0160**	0.0270	**0.0291**	0.0077	**0.0260**
	N@20	0.0068	**0.0124**	0.0131	**0.0189**	0.0066	**0.0162**	0.0055	**0.0173**	0.0292	**0.0333**	0.0105	**0.0307**
	MRR	0.0037	**0.0072**	0.0081	**0.0145**	0.0040	**0.0112**	0.0031	**0.0152**	0.0250	**0.0255**	0.0060	**0.0228**
	Imprv.		**87.83%**		**64.52%**		**158.53%**		**315.36%**		**11.33%**		**246.14%**

3.3 Ablation Study (RQ2)

To verify the contributions of different components in HSD, we conduct an ablation study with several variants on ML-100k. We consider three variants of HSD+BERT4Rec: (1) w/o long-term; (2) w/o short-term; (3) w/o sequence-level. As shown in Figure 3, each signal positively contributes to performance. The best results are consistently achieved with all signal generation layers, which well validates our motivation that leveraging multiple inconsistency

Table 3: The experimental comparison between HSD with the best/worst base model and the state-of-the-art denoising methods on the five datasets. The best results are boldfaced, the second-best results are underlined. Imprv indicates the average improvements between the best and worst ones over the baselines. All improvements are statistically significant (i.e., two-sided t-test with $p < 0.05$) over the best baselines.

Dataset	Model	HR@5	HR@10	HR@20	N@5	N@10	N@20	MRR
ML-100k	**DSAN (*AAAI'21*)**	0.0201	0.0435	0.0700	0.0115	0.0188	0.0254	0.0133
	FMLP-Rec (*WWW'22*)	0.0170	0.0477	0.0764	0.0117	0.0216	0.0288	0.0160
	HSD+NARM	0.0223	0.0403	0.0785	0.0144	0.0201	0.0299	0.0169
	HSD+BERT4Rec	**0.0339**	**0.0732**	**0.1294**	**0.0178**	**0.0305**	**0.0447**	**0.0218**
	Imprv.	40%±29%	19%±34%	36%±33%	38%±15%	17%±24%	30%±26%	21%±15%
ML-1M	**DSAN**	0.0098	0.0336	0.0651	0.0048	0.0122	0.0200	0.0081
	FMLP-Rec	0.0210	0.0449	0.0707	0.0120	0.0199	0.0263	0.0142
	HSD+NARM	0.0260	0.0417	0.0677	0.0162	0.0212	0.0277	0.0168
	HSD+BERT4Rec	**0.0477**	**0.0886**	**0.1399**	**0.0297**	**0.0429**	**0.0558**	**0.0328**
	Imprv.	75%±52%	45%±52%	47%±51%	91%±56%	61%±55%	59%±53%	75%±56%
Beauty	**DSAN**	0.0092	0.0152	0.0264	0.0058	0.0077	0.0105	0.0062
	FMLP-Rec	0.0095	0.0166	0.0284	0.0056	0.0078	0.0107	0.0060
	HSD+Caser	0.0106	0.0183	0.0290	0.0064	0.0089	0.0116	0.0068
	HSD+BERT4Rec	**0.0261**	**0.0447**	**0.0683**	**0.0147**	**0.0207**	**0.0266**	**0.0151**
	Imprv.	93%±82%	90%±80%	71%±69%	82%±72%	90%±76%	79%±70%	77%±67%
Sports	**DSAN**	0.0061	0.0105	0.0215	0.0042	0.0056	0.0084	0.0049
	FMLP-Rec	0.0068	0.0117	0.0180	0.0044	0.0059	0.0075	0.0046
	HSD+Caser	0.0077	0.0129	0.0214	0.0049	0.0065	0.0086	0.0052
	HSD+BERT4Rec	**0.0120**	**0.0190**	**0.0303**	**0.0078**	**0.0100**	**0.0129**	**0.0081**
	Imprv.	45%±32%	36%±26%	20%±21%	44%±33%	40%±30%	28%±26%	36%±30%
Yelp	**DSAN**	0.0269	0.0369	0.0541	0.0211	0.0242	0.0285	0.0216
	FMLP-Rec	0.0203	0.0294	0.0436	0.0142	0.0171	0.0207	0.0144
	HSD+GRU4Rec	0.0104	0.0180	0.0317	0.0066	0.0090	0.0124	0.0072
	HSD+BERT4Rec	**0.0292**	**0.0408**	**0.0593**	**0.0223**	**0.0260**	**0.0307**	**0.0228**
	Imprv.	-26%±35%	-20%±31%	-16%±26%	-32%±37%	-28%±35%	-24%±32%	-31%±36%

signals in a strict way can more reliably eliminate noisy items while preserving useful information. Similar results can be observed on other model combinations and datasets.

3.4 Case Study (RQ3)

Finally, we conduct a case study to illustrate how the hierarchical inconsistency signals learned in HSD+BERT4Rec can affect the sequence denoising process. In Figure 4, we show a user whose ID is 70 and whose historical interactions starts with item ID 4025 and ends with item ID 10379 from the ML-100k dataset. From the user-level signal, we can learn that items with ID 4025, 7089, 10984, and 10379 have high consistency with the user's intent. Therefore, the potential noisy item set is reduced to item ID {9087, 10759}. From the sequence-level signal, we can learn that items with ID 4025 and 9087 have high consistency with the sequence context. After that, the potential noisy item set is further reduced to item ID {10759}. In the above process, item ID {7089, 9087, 10759, 10984, 10379} are either inconsistent with the user intent or the sequence context. If we identify all the above items as noise and remove them from the sequence, there will be an inevitable loss of information. Combining the two types of signals, we consider only item ID 10759 as noise and drop it to generate a noiseless subsequence. This case aligns with our assumption that inherent noisy items in sequences should be of a small amount.

4 RELATED WORK

Sequential Recommendation Methods. The key idea of sequential recommendation is to explicitly exploit the temporal order of users' historical interactions for next-item prediction. Some earlier research utilizes Markov chains (MC) to mine sequential patterns in historical data [30]. With the rapid development of deep learning, most recent sequential recommendation methods feed sequences into various deep models to learn sequence representations and unveil user intent. Some examples include recurrent neural networks (RNNs) [7,

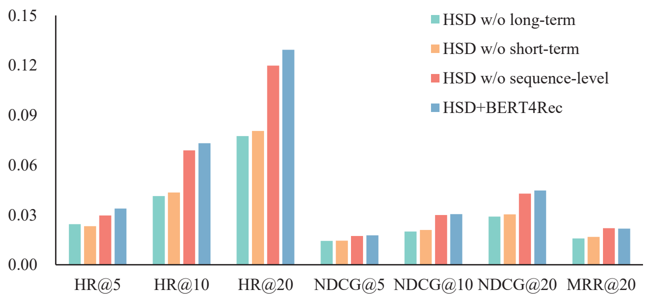

Figure 3: Impact of different inconsistency signal generation layers.

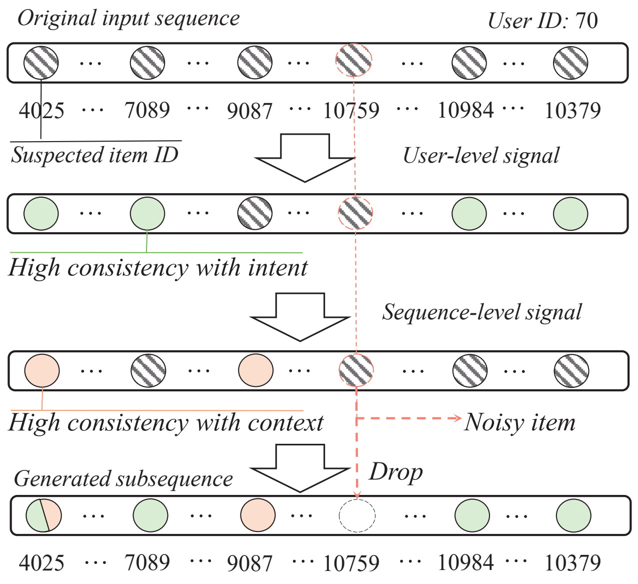

Figure 4: A case study to show how different inconsistency signals affect the denoising process.

11, 35], convolutional neural networks (CNNs) [35], attention mechanism [15, 20, 32], and graph neural networks (GNNs) [13, 22, 26, 28, 43, 46]. There are also studies [16, 38, 42] that exploit knowledge graphs to improve sequential recommendation. Some recent studies further utilize useful temporal information beyond temporal order to learn user behavior [3, 6, 10, 50]. All these works assume that the input sequences are clean. However, in practice, this is normally not the case. Noisy items in sequences may mislead the recommenders to learn sub-optimal sequence representations. Our method is orthogonal to these works in that the clean subsequences provided by HSD can be used by them to boost performance.

Denoising Approaches. Earlier works utilize user explicit feedback to reduce the gap between implicit feedback and user preference [4, 8, 17, 36, 51, 54]. Recently, Wang *et al.* [40] observe that noisy feedback typically has higher loss in the early stages of training and thus propose an adaptive denoising training strategy to reduce noise. In contrast, Sun *et al.* [33] argue that high-loss instances are not necessarily noise, and further integrate high-loss instances with an *uncertainty* measure to distinguish unreliable instances. However, the absence of supervised signals (i.e., labels indicating noisy items) makes the problem of sequence denoising rather challenging. One line of research proposes to implicitly reduce noise's influence on learning sequential representations, e.g., assigning lower attention weights to some less important items with respect to the final sequence representation [20, 21, 23, 24, 48, 53]. Recently, FMLP-Rec [59] treats sequence representations as signals and further incorporates learnable filters, which perform Fast Fourier Transform (FFT) to convert the input sequence representation into the frequency domain and filter out noise through an inverse FFT procedure. While these ideas can learn better sequence representations, there are still noisy items in sequences.

Another latest line of research explicitly removes some items by comparing their relevance with a target item (i.e., the next item). For example, DSAN [53] proposes to utilize the *entmax* function to automatically eliminate irrelevant items' attention weights with a virtual target item. CLEA [29] divides a basket into positive (i.e., items relevant to the next item) and negative (i.e., items irrelevant to the next item) sub-baskets by incorporating a discriminator, which could judge whether an item is noisy. In contrast, RAP [37] models the denoising process as a Markov Decision Processing (MDP). It proposes to learn a policy network and force the agent to select an appropriate action (i.e., whether to remove an item or not) to reach the maximum long-term award. Compared with the previous line, this idea explicitly drops some irrelevant items in sequences. However, a large number of interactions in sequences may be irrelevant to the next items, while the real inherent noisy items should be of a small amount. Motivated by the above observation, we propose to learn the hierarchical inconsistency signals and combine these signals to identify inherent noisy items in sequences without dropping advantageous information.

5 CONCLUSION

In this paper, we studied the problem of sequence denoising in sequential recommendation from a new perspective – how to learn hierarchical inconsistency signals to identify inherent noisy items in sequences. We identified two types of inconsistency signals, including sequence-level signals that have not been touched upon in the literature. We consequently devised a novel HSD model with different types of signal generation layers to effectively learn these signals that can be used to effectively identify truly reliable noisy items. We conducted comprehensive experiments on multiple benchmark datasets to show that our solution can be seamlessly integrated with a wide variety of mainstream sequential recommenders to boost their performance. We also showed that when combined with different base sequential models, HSD can outperform state-of-the-art denoising competitors.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China under Grant No. 2020YFB1710200. Xiangyu Zhao was supported by the APRC - CityU New Research Initiatives (No.9610565, Start-up Grant for the New Faculty of the City University of Hong Kong), the SIRG - CityU Strategic Interdisciplinary Research Grant (No.7020046, No.7020074), the HKIDS Early Career Research Grant (No.G0106), and the CCF-Tencent Open Fund.

REFERENCES

[1] Shuqing Bian, Wayne Xin Zhao, Kun Zhou, Jing Cai, Yancheng He, Cunxiang Yin, and Ji-Rong Wen. 2021. Contrastive Curriculum Learning for Sequential User Behavior Modeling via Data Augmentation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3737–3746.

[2] Renqin Cai, Jibang Wu, Aidan San, Chong Wang, and Hongning Wang. 2021. Category-Aware Collaborative Sequential Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 388–397.

[3] Gaode Chen, Xinghua Zhang, Yanyan Zhao, Cong Xue, and Ji Xiang. 2021. Exploring Periodicity and Interactivity in Multi-Interest Framework for Sequential Recommendation. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*. 1426–1433.

[4] Hong Chen, Yudong Chen, Xin Wang, Ruobing Xie, Rui Wang, Feng Xia, and Wenwu Zhu. 2021. Curriculum Disentangled Recommendation with Noisy Multi-feedback. *Advances in Neural Information Processing Systems* 34 (2021), 26924–26936.

[5] Yongjun Chen, Zhiwei Liu, Jia Li, Julian McAuley, and Caiming Xiong. 2022. Intent Contrastive Learning for Sequential Recommendation. In *Proceedings of the ACM Web Conference 2022*. 2172–2182.

[6] Junsu Cho, Dongmin Hyun, SeongKu Kang, and Hwanjo Yu. 2021. Learning Heterogeneous Temporal Patterns of User Preference for Timely Recommendation. In *Proceedings of the Web Conference 2021*. 1274–1283.

[7] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential User-Based Recurrent Neural Network Recommendations. In *Proceedings of the 11th ACM Conference on Recommender Systems*. 152–160.

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

[9] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. 2019. Explainable Recommendation Through Attentive Multi-View Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3622–3629.

[10] Xueliang Guo, Chongyang Shi, and Chuanming Liu. 2020. Intention Modeling from Ordered and Unordered Facets for Sequential Recommendation. In *Proceedings of The Web Conference 2020*. 1127–1137.

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

[12] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-Rich Session-Based Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 241–248.

[13] Chao Huang, Jiahui Chen, Lianghao Xia, Yong Xu, Peng Dai, Yanqing Chen, Liefeng Bo, Jiashu Zhao, and Jimmy Xiangji Huang. 2021. Graph-Enhanced Multi-task Learning of Multi-Level Transition Dynamics for Session-Based Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4123–4130.

[14] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144* (2016).

[15] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[16] Wang-Cheng Kang, Mengting Wan, and Julian McAuley. 2018. Recommendation Through Mixtures of Heterogeneous Item Relationships. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1143–1152.

[17] 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*. 193–202.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Walid Krichene and Steffen Rendle. 2020. On sampled Metrics for Item Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1748–1757.

[20] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-Based Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[21] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *Proceedings of the 13rd International Conference on Web Search and Data Mining*. 322–330.

[22] Yicong Li, Hongxu Chen, Xiangguo Sun, Zhenchao Sun, Lin Li, Lizhen Cui, Philip S Yu, and Guandong Xu. 2021. Hyperbolic Hypergraphs for Sequential Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 988–997.

[23] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-Based Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.

[24] Anjing Luo, Pengpeng Zhao, Yanchi Liu, Fuzhen Zhuang, Deqing Wang, Jiajie Xu, Junhua Fang, and Victor S. Sheng. 2020. Collaborative Self-Attention Network for Session-based Recommendation. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. 2591–2597.

[25] Yang Lv, Liansheng Zhuang, Pengyu Luo, Houqiang Li, and Zhengjun Zha. 2020. Time-Sensitive Collaborative Interest Aware Model for Session-Based Recommendation. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[26] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1253–1262.

[27] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A Novel Embedding Model for Knowledge Base completion Based on Convolutional Neural Network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 327–333.

[28] Yitong Pang, Lingfei Wu, Qi Shen, Yiming Zhang, Zhihua Wei, Fangli Xu, Ethan Chang, Bo Long, and Jian Pei. 2022. Heterogeneous Global Graph Neural Networks for Personalized Session-Based Recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 775–783.

[29] Yuqi Qin, Pengfei Wang, and Chenliang Li. 2021. The world is Binary: Contrastive Learning for Denoising Next Basket Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 859–868.

[30] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-Basket Recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. 811–820.

[31] Jing Song, Hong Shen, Zijing Ou, Junyi Zhang, Teng Xiao, and Shangsong Liang. 2019. ISLF: Interest Shift and Latent Factors Combination Model for Session-Based Recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 5765–5771.

[32] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International conference on Information and Knowledge Management*. 1441–1450.

[33] Yatong Sun, Bin Wang, Zhu Sun, and Xiaochun Yang. 2021. Does Every Data Instance Matter? Enhancing Sequential Recommendation by Eliminating Unreliable Data. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*. 1579–1585.

[34] Yang Sun, Fajie Yuan, Min Yang, Guoao Wei, Zhou Zhao, and Duo Liu. 2020. A Generic Network Compression Framework for Sequential Recommender Systems. In *Proceedings of the 43rd International Conference on Research and Development in Information Retrieval*. 1299–1308.

[35] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 565–573.

[36] Gabriele Tolomei, Mounia Lalmas, Ayman Farahat, and Andrew Haines. 2019. You Must Have Clicked on This ad by Mistake! Data-Driven Identification of Accidental Clicks on Mobile ads with Applications to Advertiser Cost Discounting and Click-Through Rate Prediction. *International Journal of Data Science and Analytics* 7, 1 (2019), 53–66.

[37] Xiaohai Tong, Pengfei Wang, Chenliang Li, Long Xia, and Shaozhang Niu. 2021. Pattern-Enhanced Contrastive Policy Learning Network for Sequential Recommendation. In *Proceedings of the 30th International Joint Conference on Artificial*. 1593–1599.

[38] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Make It a Chorus: Knowledge- and Time-Aware Item Modeling for Sequential Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 109–118.

[39] Qinyong Wang, Hongzhi Yin, Hao Wang, Quoc Viet Hung Nguyen, Zi Huang, and Lizhen Cui. 2019. Enhancing Collaborative Filtering with Generative Augmentation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 548–556.

[40] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising Implicit Feedback for Recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 373–381.

[41] Xin Wang, Yudong Chen, and Wenwu Zhu. 2022. A Survey on Curriculum Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 9 (2022), 4555–4576.

[42] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 950–958.

[43] Yifan Wang, Suyao Tang, Yuntong Lei, Weiping Song, Sheng Wang, and Ming Zhang. 2020. Disenhan: Disentangled Heterogeneous Graph Attention Network for Recommendation. In *Proceedings of the 29th ACM International Conference on*

Information & Knowledge Management. 1605–1614.

[44] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global Context Enhanced Graph Neural Networks for Session-Based Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 169–178.

[45] Sixing Wu, Ying Li, Dawei Zhang, Yang Zhou, and Zhonghai Wu. 2021. Topicka: Generating Commonsense Knowledge-Aware Dialogue Responses Towards the Recommended Topic Fact. In *Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence*. 3766–3772.

[46] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

[47] Chang Xu, Dacheng Tao, and Chao Xu. 2013. A Survey on Multi-View Learning. *arXiv preprint arXiv:1304.5634* (2013).

[48] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-Based Recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 3940–3946.

[49] Jie Xu, Yazhou Ren, Guofeng Li, Lili Pan, Ce Zhu, and Zenglin Xu. 2021. Deep Embedded Multi-View Clustering with Collaborative Training. *Information Sciences* 573 (2021), 279–290.

[50] Wenwen Ye, Shuaiqiang Wang, Xu Chen, Xuepeng Wang, Zheng Qin, and Dawei Yin. 2020. Time Matters: Sequential Recommendation with Complex Temporal Information. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1459–1468.

[51] 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*. 113–120.

[52] Junliang Yu, Min Gao, Hongzhi Yin, Jundong Li, Chongming Gao, and Qinyong Wang. 2019. Generating Reliable Friends via Adversarial Training to Improve Social Recommendation. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 768–777.

[53] Jiahao Yuan, Zihan Song, Mingyou Sun, Xiaoling Wang, and Wayne Xin Zhao. 2021. Dual Sparse Attention Network For Session-Based Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4635–4643.

[54] 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*. 131–138.

[55] Wayne Xin Zhao, Junhua Chen, Pengfei Wang, Qi Gu, and Ji-Rong Wen. 2020. Revisiting Alternative Experimental Settings for Evaluating Top-N Item Recommendation Algorithms. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2329–2332.

[56] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. 2021. Recbole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4653–4664.

[57] Yu Zheng, Chen Gao, Jianxin Chang, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2022. Disentangling Long and Short-Term Interests for Recommendation. In *Proceedings of the ACM Web Conference 2022*. 2256–2267.

[58] Yujia Zheng, Siyi Liu, Zekun Li, and Shu Wu. 2021. Cold-Start Sequential Recommendation via Meta Learner. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4706–4713.

[59] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-Enhanced MLP is All You Need for Sequential Recommendation. In *Proceedings of the ACM Web Conference 2022*. 2388–2399.