Improving Transformer-based Sequential Recommenders through Preference Editing

MUYANG MA, PENGJIE REN, ZHUMIN CHEN, and ZHAOCHUN REN,
Shandong University, China
HUASHENG LIANG, WeChat, Tencent, China
JUN MA, Shandong University, China
MAARTEN DE RIJKE, University of Amsterdam, The Netherlands

One of the key challenges in sequential recommendation is how to extract and represent user preferences. Traditional methods rely solely on predicting the next item. But user behavior may be driven by complex preferences. Therefore, these methods cannot make accurate recommendations when the available information user behavior is limited. To explore multiple user preferences, we propose a transformer-based sequential recommendation model, named MrTransformer (**M**ulti-p**r**eference **Transformer**). For training MrTransformer, we devise a *preference-editing*-based **self-supervised learning (SSL)** mechanism that explores extra supervision signals based on relations with other sequences. The idea is to force the sequential recommendation model to discriminate between common and unique preferences in different sequences of interactions. By doing so, the sequential recommendation model is able to disentangle user preferences into multiple independent preference representations so as to improve user preference extraction and representation.

We carry out extensive experiments on five benchmark datasets. MrTransformer with preference editing significantly outperforms state-of-the-art sequential recommendation methods in terms of Recall, MRR, and NDCG. We find that long sequences of interactions from which user preferences are harder to extract and represent benefit most from preference editing.

CCS Concepts: • **Information systems** → **Recommender systems;**

Additional Key Words and Phrases: Transformer-based sequential recommendation, self-supervised learning, user preference extraction and representation

This research was partially supported by the National Key R&D Program of China with grant No. 2020YFB1406704, the Natural Science Foundation of China (61972234, 61902219, 62072279, 62102234), the Key Scientific and Technological Innovation Program of Shandong Province (2019JZZY010129), the Natural Science Foundation of Shandong Province (ZR2021QF129), the Tencent WeChat Rhino-Bird Focused Research Program (JR-WXG-2021411), the Fundamental Research Funds of Shandong University, and the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl.

Authors' addresses: M. Ma, P. Ren (corresponding author), Z. Chen, Z. Ren, and J. Ma, Shandong University, Qingdao, China; emails: muyang0331@gmail.com, jay.ren@outlook.com, chenzhumin@sdu.edu.cn, zhaochun.ren@sdu.edu.cn, majun@sdu.edu.cn; H. Liang, WeChat, Tencent, Shenzhen, China; email: watsonliang@tencent.com; M. de Rijke, University of Amsterdam, Amsterdam, The Netherlands; email: m.derijke@uva.nl.

1046-8188/2023/04-ART71 $15.00
https://doi.org/10.1145/3564282

ACM Reference format:
Muyang Ma, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Huasheng Liang, Jun Ma, and Maarten de Rijke. 2023. Improving Transformer-based Sequential Recommenders through Preference Editing. *ACM Trans. Inf. Syst.* 41, 3, Article 71 (April 2023), 24 pages.
https://doi.org/10.1145/3564282

1 INTRODUCTION

Sequential recommendation (SR) methods aim to predict the next item that the user is most likely to interact with based on his or her past interactions, such as clicking on products or watching movies [10, 25, 42, 71]. One of the key challenges faced by SR approaches is to *extract and represent user preferences* from historical interaction sequences [47, 75]. Traditional studies typically learn a user embedding vector by encoding the user's overall preferences from his or her complex behavior sequence [12]. However, in some recommendation scenarios users' preferences may change slightly over a period of time [1, 3, 80]. Prior studies fail to make distinctions between different preferences by combining multiple preferences into a single vector [51]. To extract multiple user preferences for each sequence, recent research has proposed a multi-head attention mechanism on top of RNN-based methods [1, 2, 9] or some graph convolutional techniques [35, 59, 80]. However, to the best of our knowledge, it has not been explored in transformer-based methods so far. Therefore, we propose a transformer-based model named MrTransformer that incorporates a *preference identification* module into a transformer-based model, BERT4Rec [45]. We concatenate special tokens at the start of each interaction sequence. The encoded vector corresponding to each special token from the transformer represents a specific user preference. We guarantee that each special token attends to different parts of the interaction sequence by introducing a *preference coverage* mechanism.

To train an SR model, traditional methods typically rely only on predicting the next item [7, 41, 58, 83]. The supervision signals are these next items that come from the sequence itself [36, 37, 61]. This learning process is very similar to the language modeling task in **Natural Language Processing (NLP)**. For example, in "I like orange [. . .]," the next word has a high probability of being "juice." This is reasonable because the co-occurrence probability of "orange" and "juice" is higher than other words in natural grammatical structures. But in a recommendation scenario, the order of items in the interaction sequence is not restricted by similar grammatical structures. Therefore, we need to explore new supervision signals suitable for recommendation scenarios to guide model learning in addition to predicting the next item.

Self-Supervised Learning (SSL) can automatically generate a supervision signal to learn representations of the data or to automatically label a dataset [52, 55], which has made great progress in **Computer Vision (CV)** [48, 76] and NLP [6, 8, 18, 19]. Similarly, it has been introduced to SR as well. These methods can enhance the learned representations of users and items and help alleviate the data sparsity of cold-start users/items by exploring the intrinsic correlations among sequences of interactions [28, 63, 70, 73]. Existing SSL methods for SR can be divided into three categories: generative, contrastive, and adversarial. Generative methods aim to reconstruct the part or the whole original interaction sequence. Representative methods are **Auto-Regressive (AR)** and **Auto-Encoder (AE)** models, which aim to establish a joint probability distribution of this sequence or randomly mask some items and try to predict these masked items based on the remaining sequence. Contrastive methods focus on sampling negative items to guide the model to learn the difference between positive and negative target items by **Noise Contrastive Estimation (NCE)** or model the relationship between the local and global features of the input sequence by **Mutual Information Maximization (MIM)**. Adversarial methods introduce the generative adversarial network into the traditional next-item prediction learning task by designing generators and discriminators to capture the context information, which can capture the high-level semantic information. However, these studies propose new SSL signals by exploring a current interaction sequence itself but neglect its relations with other sequences. Moreover, so far SSL methods have not made a direct connection to user preference extraction and representation.

In this article, we propose a novel learning strategy, named ***preference editing*** **(PE)**, for training our basic model MrTransformer. It aims to explore supervision signals from the relations between different interaction sequences and focus on discriminating between common and unique preference representations. During the training stage, PE involves two operations to edit both user preferences: *preference separation* and *preference recombination*. The former separates the common and unique preference representations from the respective multiple preference representations. The latter swaps the common user preference representations so as to get recombined user preference representations for each interaction sequence. We first sample two interactive sequences randomly. We can obtain multiple preference representations P_1 and P_2 through MrTransformer for each sequence, respectively. Then we extract the common preference representations c_1 and c_2 and the unique preference representations u_1 and u_2 through the *preference separation* operation. After that, we combine c_2 and u_1 to get the recombined preferences P_1' and combine c_1 and u_2 to get the recombined preferences P_2' through the *preference recombination* operation. Based on the above process, we devise two types of SSL signals:

(1) We use the recombined preferences to predict the next item in both sequences.
(2) We require that the recombined preferences (e.g., P_1') are as similar as possible to the original preferences (e.g., P_1), and that the common preference representations are close to each other (i.e., $c_1 = c_2$).

By doing so, we force the preference extraction model to learn how to identify and edit user preferences so as to do better user preference extraction and representation.

For the whole model MrTransformer (PE), the training phase contains two stages (i.e., the pre-training stage and the fine-tuning stage). In the pre-training stage, we only use the SSL signals to train the model. Then we initialize the model with the pre-trained parameters and use the next-item recommendation supervision signal to guide the model learning.

To assess the full model MrTransformer (PE), we carry out extensive experiments on five benchmark datasets: Beauty, Sports, Toys of Amazon datasets, ML-100k, and Yelp datasets. The results show that MrTransformer with *preference editing* significantly outperforms state-of-the-art baselines on all datasets in terms of Recall, MRR, and NDCG. We also find that long sequences whose user preferences are harder to extract and represent benefit the most from preference editing.

To sum up, the main contributions of this work are as follows:

- We propose a transformer-based multi-preference extraction model MrTransformer for SRs, which can produce multiple preference representations.
- We devise a novel self-supervised learning method *preference editing* for training MrTransformer, which explores supervision signals from the relations between sequences.
- The joint framework, MrTransformer, combines the basic SR model MrTransformer and the learning strategy *preference editing* to capture the commonness and uniqueness of different sequences.
- We demonstrate the effectiveness of MrTransformer and *preference editing* through extensive experiments on five benchmark datasets.

2 RELATED WORK

In this section, we survey related work from two categories: transformer-based SR and self-supervised learning for SR.

2.1 Transformer-based Sequential Recommendation

Various neural architectures or mechanisms have successfully been applied to the sequential recommendation task [64]. These include **Recurrent Neural Networks (RNNs)** [12, 21, 31, 46], **Convolutional Neural Networks (CNNs)** [49, 50], **Graph Neural Networks (GNNs)** [38, 60, 72], **Reinforcement Learning (RL)** [57, 69], copy mechanisms [40], and memory networks [4, 15, 56]. Transformer-based methods have recently been proven to be effective [16, 45, 68].

Kang and McAuley [16] propose SASRec, which introduces a self-attention mechanism (the most important component in the transformer) to SRs to identify important items from interactions. Several variants have been proposed to improve upon SASRec. Mi et al. [34] argue that traditional SRs use SASRec as a basic sequence representation extractor; they propose a continual learning setup with an adaptive distillation loss to update the recommender periodically as new data streams in. Li et al. [22] propose TiSASRec, a variant of SASRec, to explicitly model the timestamps of items in sequences and explore the influence of different time intervals on next-item prediction. Luo et al. [29] argue that even the same item can be represented differently for different users at the same timestep; they propose a collaborative self-attention network to learn sequence representations and predict the preferences of the current sequence by investigating neighborhood sequences. Sun et al. [45] adopt a bidirectional transformer to predict masked items in sequences based on the surrounding items. Wang et al. [54] equip the transformer with hyper-graph neural networks to capture dynamic representations of items across time and users. Xie et al. [68] propose three data augmentation approaches (crop, mask, and reorder) to pre-train a transformer-based model to get user and sequence representations and then fine-tune it on the SR task.

Previous work has also investigated how to combine auxiliary tasks or information with SR based on the transformer. Cho et al. [5] introduce multiple types of position embeddings by considering timestamp information and propose a self-attention-based model in which each attention head uses a different position embedding. Wu et al. [62] point out that previous studies ignore the temporal and context information when modeling the influence of a historical item to the current prediction; they propose a contextualized temporal attention mechanism to weigh the influence of historical interactions not only on what items to interact with but also when and how the interactions took place. Lin et al. [23] argue that modeling users' global preferences only based on their historical interactions is imperfect and the users' preference is uncertain; they propose a FISSA solution, which fuses item similarity models with self-attention networks to balance the local and global user preference representations by taking the information of the candidate items into account. Wu et al. [65] propose a personalized transformer model with a recent regularization technique called **stochastic shared embeddings (SSE)** [66] to overcome overfitting caused by simply adding user embeddings.

The studies listed above have proposed various transformer-based SR models. No previous work has considered how to identify users' multiple preferences behind the interaction sequence. In contrast, we extract multiple user preferences and represent them using distributed vectors.

2.2 Self-supervised Learning for Sequential Recommendation

Previous approaches to SR are typically trained by predicting the next interaction, which is prone to suffer from data sparsity [68, 70]. To mitigate this, some work explores SSL and derives self-supervised signals to enhance the learning of item and sequence representations. Inspired by the survey of existing work about SSL [27], we divide SSL for SR approaches into three categories.

The first category of SSL approaches are generative methods, which aim to reconstruct the part or whole of the original interaction sequence. One type of method models the user's interaction sequence in an auto-regressive fashion. The goal is to establish a joint probability distribution of this sequence, which can be decomposed into the product of conditional probability distributions. Each variable depends on the previous variable. The next-item prediction labels are derived from the piece of sequence itself, which can make use of contextual information, and the information flow is from left to right in most cases. One of the representative works is SASRec [16], which seeks to identify which items are "relevant" from a user's interaction sequence and use them to predict the next item by applying a self-attention mechanism. The other type of method models the user's interaction sequence in an auto-encoder fashion. It randomly masks some items in an interaction sequence and tries to predict these masked items based on the surrounding items. Sun et al. [45] adopt a BERT-like training scheme, which predicts the masked items in the sequence based on surrounding items. Instead of masking items, Yao et al. [70] propose to mask sparse categorical features of items; they mask or drop out some categorical feature embeddings to learn internal relations between two sets of categorical features. Yuan et al. [74] represent and transfer user representation models for serving downstream tasks where only limited data exists through a pre-train-fine-tune strategy; in the pre-training stage, they randomly mask a certain percentage of items in the sequence and then predict the masked items to get user preference representations.

The second category of SSL approaches are contrastive methods, which focus on sampling negative items to guide the model to learn the difference between positive and negative target items by noise contrastive estimation, or model the relationship between the local and global features of the input interaction sequence by mutual information maximization. One type of method mainly focuses on sampling negative items to guide the model to learn the difference between positive and negative target items by NCE. The input interaction sequence is augmented to obtain multiple views of this sequence, and the model can distinguish the different views by taking different negative examples. Zhou et al. [81] propose a fixed-size queue to store items' representations computed in previous batches and use the queue to sample negative examples for each sequence. Xie et al. [68] propose three data augmentation methods (crop, mask, and reorder); then they encode the sequence representation by maximizing the agreement between different augmented methods of the same sequence in the latent space. Another type of contrastive methods aims to model the relationship between the local and global features of the input sequence by MIM. Ma et al. [30] propose a sequence-to-sequence training strategy to mine extra supervision by looking at the longer-term future; they first use a disentangled encoder to obtain multiple representations of a given sequence and predict the representation of the future sub-sequence given the representation of the earlier sequence; sequence representations that are not from the same sequence as the earlier ones are considered as negative samples. Zhou et al. [82] devise four auxiliary self-supervised objectives to learn correlations among four types of data (item attributes, items, sub-sequences, and sequences) by utilizing the mutual information maximization principle. Xia et al. [67] model sequences as a hypergraph and propose a dual-channel hypergraph convolutional network to capture higher-order relations among items within sequences; during training, they maximize the mutual information between the sequence representations learned via the two channels; negative sampling SSL ensures that different channels of the same sample are similar.

The third category of SSL approaches consists of adversarial methods, which introduce the generative adversarial network into the traditional next-item prediction learning task. Adversarial methods mainly design generators and discriminators to capture the context or other information and trace how this information contributes to the recommendation. Zhao et al. [78] propose a seq2seq learning strategy for SR, which yields a sequence of items consistent with the user preference in sequence level rather than on next-item prediction. They also present adversarial oracular learning over the seq2seq recommendation, reducing the exposure bias in the auto-regressive style while improving the integrality in the recommended sequences. Ren et al. [43] propose a transformer-based generator taking user interaction sequences as input to recommend the possible next items, and multiple discriminators to evaluate the generated sub-sequence from the perspectives of different context information.

Compared with existing SSL strategies for SR, we mainly focus on how to devise self-supervision signals by investigating correlations between different sequences through preference editing, i.e., forcing the SR model to learn better representations by identifying the common and unique preferences for sequence pairs.

3 METHOD

We first formulate the sequential recommendation task. Then, we introduce our basic transformer-based multi-preference extraction model MrTransformer. Next, we describe our new SSL method *preference editing*. Together, preference editing and MrTransformer constitute our complete model MrTransformer (PE).

3.1 Task Definition

Let $\mathbb{I}$ denote the item set and $\mathbb{S}$ denote the set of interaction sequences, respectively, where $i \in \mathbb{I}$ denotes an item and $S \in \mathbb{S}$ denotes a sequence. The numbers of items and sequences are denoted as $|\mathbb{I}|$ and $|\mathbb{S}|$, respectively. Each sequence can be denoted as $S = [i_1, \ldots, i_\tau, \ldots, i_t]$, where i_τ refers to an item that was interacted with at timestep τ. Given S, the sequential recommendation (SR) task is to predict the next item that the user will interact with at timestep $t+1$ by computing the recommendation probabilities over all candidate items as follows:

$$P(i_{t+1}|S) \sim f(S), \tag{1}$$

where $P(i_{t+1}|S)$ denotes the probability of recommending the next item i_{t+1}, and $f(S)$ is the model or function to estimate $P(i_{t+1}|S)$.

3.2 MrTransformer

In this section, we will introduce our proposed transformer-based [53] multi-preference extraction model, MrTransformer. Unlike the basic BERT4Rec model, the core idea of MrTransformer is to mine multiple preferences behind the interaction sequence and then generate predictions based on the multiple preference representations.

MrTransformer's network structure is based on the transformer, and it consists of three main components:

(1) A sequence encoder
(2) A preference identification module
(3) A sequence decoder

The preference identification module identifies multiple preferences behind the current interaction sequence and represents them as distributed vectors. Unlike RNN-based methods and traditional transformer-based methods, we concatenate K special tokens ($[P_1], [P_2], \ldots, [P_K]$) at the start of each sequence. During sequence encoding, the special tokens can capture all sequence information; each of them can represent the sequence representation. The coverage mechanism (which will be introduced later) ensures that the representations learned by all special tokens are different; each special token refers to a specific user preference. Like other model parameters, the special tokens are initialized randomly. Next, we introduce these modules in detail.

Sequence Encoder. We define the processed sequence $S' = [[P_1], [P_2], \ldots, [P_K], i_1, \ldots, i_\tau, \ldots, i_t]$, which concatenates K special tokens at the start of the sequence S. It is worth noting that K represents the number of latent preferences for the whole sequence set, not for a particular sequence. In this module, we encode the processed sequence $S^{'}$ into hidden representations.

We first initialize the embedding matrix $\mathbf{E}$ of $S^{'}$, where $\mathbf{e}_{\mathbf{i}_\tau} \in \mathbb{R}^h$ represents the embedding for item i_τ, and h is the hidden size. Then we add the position embedding matrix $\mathbf{P}$ of $S^{'}$ to the embedding matrix, which can be defined as $\mathbf{E} = \mathbf{E} + \mathbf{P}$. After that, we feed the sequence of items into a stack of L bidirectional transformer encoder layers. Each layer iteratively revises the representation of all positions by exchanging information across some specific positions, which are controlled by the masking matrix at previous layers. Specifically, the transformer encoder layer is composed of several self-attention layers and feedforward layers, which are defined as

$$\begin{aligned}\text{Self-Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V},\\ \text{Feed-Forward}(x) &= (\text{ReLu}(x\mathbf{W}_1 + b_1))\mathbf{W}_2 + b_2,\end{aligned} \tag{2}$$

where $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}^K$, and $\mathbf{V} = \mathbf{X}\mathbf{W}^V$ are the linear transformations of the input representation matrix $\mathbf{X}$, and $\sqrt{d_k}$ is the scale factor to avoid large values of the inner product. $\mathbf{W}_1$, b_1, $\mathbf{W}_2$, and b_2 are trainable parameters. As for the sequence encoder, the process is defined as follows:

$$\mathbf{E}^{\mathbf{l}} = \text{Trm}(\mathbf{E}^{\mathbf{l-1}}, \mathbf{Mask_e}), \tag{3}$$

where Trm refers to a transformer encoder layer, $\mathbf{E}^{\mathbf{l}} \in \mathbb{R}^{(K+t)*h}$ is the representation matrix of $S^{'}$ at the lth layer, and $\mathbf{Mask_e}$ is the masking matrix. Specifically, each special token $[P_k]$ obtains information from all positions because it needs to capture the user's multiple preferences behind the whole sequence. Each special token can attend to every position, and items can only attend to items. From the top layer, we can obtain the representation $\mathbf{E}^{\mathbf{L}}$ of $S^{'}$.

Preference Identification. In this module, we calculate the distributed attention scores over all items for each special token. The process is defined as follows:

$$\mathbf{P}, \mathbf{A} = \text{Ident}(\mathbf{E}^{\mathbf{L}}, \mathbf{Mask_I}), \tag{4}$$

where Ident is also implemented by a transformer encoder layer, and $\mathbf{Mask_I}$ is the masking matrix for preference identification. To guarantee that the learned preferences are different from each other, each special token only attends to items. $\mathbf{P} \in \mathbb{R}^{K*h}$ is the representation matrix for the multiple preferences corresponding to the first K special tokens. We can also get the attention matrix $\mathbf{A}$ over all items for the special tokens.

To avoid that different special tokens focus on the same items and thus learn similar representations, we introduce a *preference coverage mechanism.* We maintain K coverage vectors c_τ^k ($k = 1, \ldots, K$). c_τ^k is the sum of attention distributions over items at previous timesteps by the special token $[P_k]$, which represents the degree of coverage that those items have received from the attention mechanism by $[P_k]$ so far:

$$c_\tau^k = \sum_{j=1}^{\tau-1} a_j^k, \tag{5}$$

where $\mathbf{a}^{\mathbf{k}} \in \mathbf{A}$ is a distributed attention vector over all items by $[P_k]$, and $\sum_i a_i^k = 1$. c_0^k is a zero vector, which denotes that, at the first timestep, none of the items have been covered. Correspondingly, we define a coverage loss to penalize repeatedly attending to the same items by different preferences:

$$L_{cov}(\theta) = \sum_{k=1}^{K} \sum_{\tau} \min\left(a_\tau^k, c_\tau^k\right). \tag{6}$$

Sequence Decoder. After the *preference identification* module, we get the representation for the multiple preferences. Different from existing methods that use item representations to predict the next item [14, 20, 44, 79], we use the learned multiple preference representations to do recommendation:

$$P(i_{t+1}|S) = \mathrm{softmax}(\mathbf{pW} + b). \tag{7}$$

We sum the multiple preference matrix $\mathbf{P}$ along the first dimension, i.e., $p = \sum_{k=1}^{K} P_{[k,:]}$, where $p \in \mathbb{R}^{h}$ and $P \in \mathbb{R}^{K*h}$. $\mathbf{W} \in \mathbb{R}^{h*|\mathbb{I}|}$ is the embedding matrix of all items, and b is the bias term.

Objective Functions. As with traditional SR methods [13, 26, 39, 56], our first objective is to predict the next item for each position in the input sequence. We employ the negative log-likelihood loss to define the recommendation loss as follows:

$$L_{rec}(\theta) = -\frac{1}{|\mathbb{S}|}\sum_{S\in\mathbb{S}}\sum_{i_\tau\in S}\log P(i_{\tau+1}|S), \tag{8}$$

where θ are all parameters of MrTransformer.

Finally, the coverage loss function, weighted by the hyperparameter α, is added to the recommendation loss function to yield the total loss function:

$$L(\theta) = L_{rec}(\theta) + \alpha L_{cov}(\theta), \tag{9}$$

where α is the weight of the coverage loss.

3.3 Preference Editing

In this subsection, we detail the *preference editing* learning strategy to mine relations among items with other interaction sequences.

As illustrated in Figure 1, we first sample two sequences S_x and S_y from the sequence set $\mathbb{S}$ randomly. Through the *preference identification* module, we get the multiple preference representations for each sequence (e.g., P_1^x, P_2^x, and P_3^x for the sequence S_x and P_1^y, P_2^y, and P_3^y for the sequence S_y). Then, the *preference separation* module forces the model to separate common (P_1^x, P_2^x and P_1^y, P_2^y) and unique (P_3^x and P_3^y) preference representations for the paired sequences. After that, the *preference recombination* module swaps the common preference representations so as to obtain the recombined preference representations for each sequence (e.g., P_1^y, P_2^y, P_3^x for the sequence S_x and P_1^x, P_2^x, P_3^y for the sequence S_y).

Next, we explain the above process in detail.

Preference Separation. For each pair of sequences S_x and S_y, we obtain the multiple preference representation matrices $\mathbf{P_x}$ and $\mathbf{P_y}$ through the *preference identification* module, where $\mathbf{P_x}, \mathbf{P_y} \in \mathbb{R}^{K*h}$. In order to measure the degree of similarity between these two representations, we calculate the similarity matrix $\mathbf{I} \in \mathbb{R}^{K*K}$ to consider their relations. Each element $I_{ij} \in \mathbb{R}$ is calculated as

$$I_{ij} = \mathbf{W_s}^{\mathrm{T}}\left[\mathbf{p_i}; \mathbf{p_j}; \mathbf{p_i} \odot \mathbf{p_j}\right], \tag{10}$$

where $\mathbf{p_i}$ and $\mathbf{p_j}$ are the vectors in the multiple preference representation matrices $\mathbf{P_x}$ and $\mathbf{P_y}$, respectively; $\odot$ is the element-wise multiplication calculation; $\mathbf{W_s} \in \mathbb{R}^{3h*1}$ is initialized randomly and optimized with the other model parameters simultaneously. Based on the similarity matrix $\mathbf{I}$, we calculate the attention matrices $\mathbf{A_x}$ and $\mathbf{B_y}$, which reflect the attention distribution of the preference representations of one sequence to the preference representation in another sequence:

$$\begin{aligned}\mathbf{A_x} &= \mathrm{softmax}_{row}(\mathbf{I}),\\ \mathbf{B_y} &= \mathrm{softmax}_{col}(\mathbf{I}),\end{aligned} \tag{11}$$

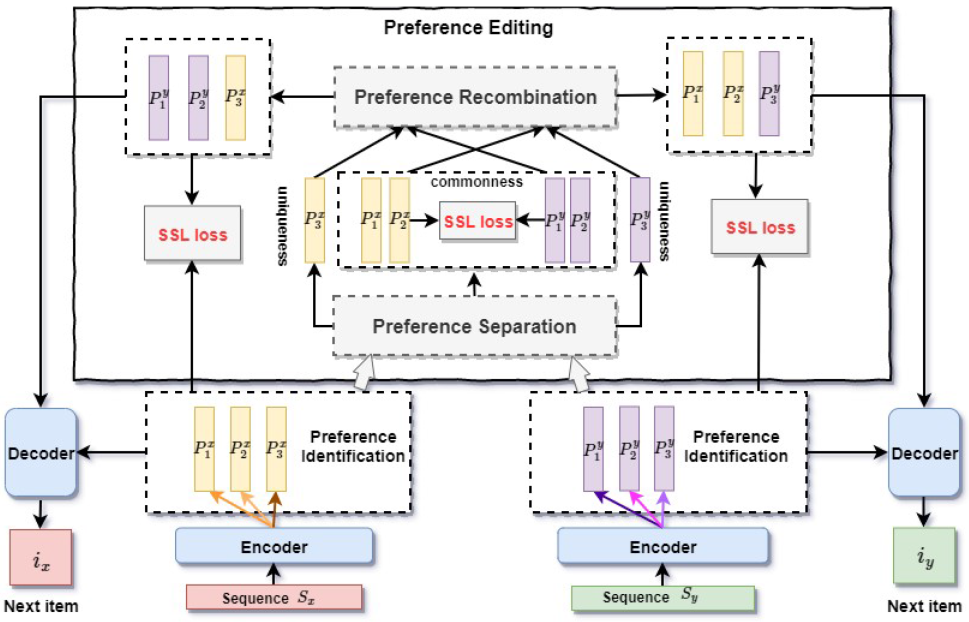

Fig. 1. Overview of *preference editing*. Section 3.3 contains a walk-through of this figure.

where softmax$_{row}$ and softmax$_{col}$ refer to performing a softmax calculation on the rows and columns, respectively. After that, the common and unique preference representations of each sequence are calculated as follows:

$$\begin{aligned} \mathbf{C}_\mathrm{x} &= \mathbf{B}_\mathrm{y} \odot \mathbf{P}_\mathrm{x}, \quad \mathbf{U}_\mathrm{x} = (1 - \mathbf{B}_\mathrm{y}) \odot \mathbf{P}_\mathrm{x}, \\ \mathbf{C}_\mathrm{y} &= \mathbf{A}_\mathrm{x} \odot \mathbf{P}_\mathrm{y}, \quad \mathbf{U}_\mathrm{y} = (1 - \mathbf{A}_\mathrm{x}) \odot \mathbf{P}_\mathrm{y}, \end{aligned} \tag{12}$$

where $\mathbf{C}_\mathrm{x}$ and $\mathbf{C}_\mathrm{y}$ refer to the common preference representations of each sequence, and $\mathbf{U}_\mathrm{x}$ and $\mathbf{U}_\mathrm{y}$ are the unique preference representations. These common preference representations ($\mathbf{C}_\mathrm{x}$ and $\mathbf{C}_\mathrm{y}$) and unique preference representations ($\mathbf{U}_\mathrm{x}$ and $\mathbf{U}_\mathrm{y}$) have the same shape, which is $\mathbb{R}^{K*h}$. $\mathbf{A}_\mathrm{x}$ and $\mathbf{B}_\mathrm{y}$ play the role of gate functions that filter the common and unique preference representations.

Preference Recombination. Through the *preference separation* module, we obtain the common and unique preference representations of the sequence pairs S_x and S_y. Then we swap the common preference representation to form a recombined representation for each sequence as follows:

$$\begin{aligned} \mathbf{P}'_\mathrm{x} &= \mathrm{add}[\mathbf{C}_\mathrm{y}; \mathbf{U}_\mathrm{x}; \mathbf{C}_\mathrm{y} \odot \mathbf{U}_\mathrm{x}], \\ \mathbf{P}'_\mathrm{y} &= \mathrm{add}[\mathbf{C}_\mathrm{x}; \mathbf{U}_\mathrm{y}; \mathbf{C}_\mathrm{x} \odot \mathbf{U}_\mathrm{y}], \end{aligned} \tag{13}$$

where the add operation means the matrix addition; the shapes of $\mathbf{P}'_\mathrm{x}$ and $\mathbf{P}'_\mathrm{y}$ are $\mathbb{R}^{K*h}$. Based on the recombined representation, we define two types of supervision signals for learning:

$$L_{SSL}(\theta) = L_{pred}(\theta) + L_{reg}(\theta). \tag{14}$$

$L_{pred}(\theta)$ is used to perform next-item prediction based on the recombined representations $\mathbf{P}'_\mathrm{x}$ and $\mathbf{P}'_\mathrm{y}$:

$$L_{pred}(\theta) = L^x_{rec} + L^y_{rec}, \tag{15}$$

where L^x_{rec} and L^y_{rec} are negative log-likelihood loss, which are the same as the calculation in Equation (8). L^x_{rec} and L^y_{rec} are used to predict the next item for each sequence using the recombined representation.

$L_{reg}(\theta)$ is a regularization term,

$$L_{reg}(\theta) = L_{reg}(\theta)^x + L_{reg}(\theta)^y + L_{reg}(\theta)^c, \tag{16}$$

where

$$\begin{aligned} L_{reg}(\theta)^x &= \frac{1}{K * h} \sum_{S_x, S_y \in \mathbb{S}} (\mathbf{P_x} - \mathbf{P'_x})^2, \\ L_{reg}(\theta)^y &= \frac{1}{K * h} \sum_{S_x, S_y \in \mathbb{S}} (\mathbf{P_y} - \mathbf{P'_y})^2, \\ L_{reg}(\theta)^c &= \frac{1}{K * h} \sum_{S_x, S_y \in \mathbb{S}} (\mathbf{C_x} - \mathbf{C_y})^2. \end{aligned} \tag{17}$$

$L_{reg}(\theta)^x$ and $L_{reg}(\theta)^y$ make sure that the recombined representation is close enough to the original preference representation. $L_{reg}(\theta)^c$ requires that the learned common representations are also close to each other.

The final training loss for MrTransformer (PE) is as follows:

$$L_{all}(\theta) = L(\theta) + L_{SSL}(\theta), \tag{18}$$

where $L(\theta)$ is the loss for preference extraction and recommendation (Equation (9)). $L_{SSL}(\theta)$ is the loss for preference editing (Equation (14)).

We use **MrTransformer** to refer to our basic model that models a user's multiple preferences based on transformer layers by using the preference identification module as detailed in Section 3.2. We write **MrTransformer (PE)** for MrTransformer pre-trained with the preference editing learning strategy described in this section.

3.4 Computational Complexity

The computational complexity of MrTransformer (PE) and other baselines is mainly due to the transformer layer that contains the self-attention layer and the feed-forward layer. The complexity is $O(n^2 * h + n * h^2)$, where n is the max sequence length and h is the hidden size. Compared with other baselines, our model contains the preference editing operation in the training stage; the complexity has increased by $O(K^2 * h)$, where K is the number of latent preferences and is much smaller than n. During testing, the complexity of our model is the same as other baselines.

4 EXPERIMENTAL SETUP

We seek to answer the following questions in our experiments:

(RQ1) What is the performance of MrTransformer (PE) compared to other methods? Does it outperform the state-of-the-art methods in terms of Recall, MRR, and NDCG on all datasets?
(RQ2) What is the effect of the preference editing learning strategy on the performance of MrTransformer (PE)? And how does it affect sequences of different lengths?
(RQ3) What is the effect of the preference coverage mechanism on the performance of MrTransformer (PE)?

Table 1. Statistics of the Datasets Used in This Article

Datasets	#users	#items	#actions	#min(len)	#max(len)	#avg(len)	#sparsity
Beauty	40,226	54,542	353,962	3	291	8.8	99.98%
Sports	25,598	18,357	296,337	3	294	6.3	94.50%
Toys	19,412	11,924	167,597	3	548	6.6	99.92%
ML-100k	943	1,349	99,287	17	646	103.2	92.19%
Yelp	30,431	20,033	316,354	3	348	8.4	99.94%

(RQ4) How does the hyperparameter K (the number of assumed latent preferences) affect the performance of MrTransformer (PE)?

4.1 Datasets

We conduct experiments on five datasets in different domains:

- **Amazon Beauty, Sports, Toys**: These datasets are obtained from Amazon product review datasets crawled by McAuley et al. [32]. We select three subcategories of Beauty (40,226 users and 54,542 items), Sports (25,598 users and 18,357 items), and Toys (19,412 users and 11,924 items).[1]
- **MovieLens**: This is a popular benchmark dataset for recommendation evaluation. We adopt a well-established version, ML-100k (943 users and 1,349 items).[2]
- **Yelp**: This is a popular dataset for business recommendation (30,431 users and 20,033 items).[3] As it is very large, we only use the transaction records after January 1, 2019, like [82].

We follow the data processing given in [16, 45]. We convert the rating scores of reviews to the implicit feedback of 1 if the user has interacted with this item. Then we group the interactions by the same user and sort them according to the timesteps. Like [20, 33, 77], since we do not target cold-start recommendation, we filter out cold-start users who have fewer than five interactions and cold-start items with fewer than five interactions. Because each user has a lot of interactions in a long time, we limit the maximum length of each sequence to be 50. The statistics of datasets are shown in Table 1.

4.2 Evaluation Metrics

We adopt the leave-one-out recommendation evaluation scheme [45]. We use the last item and the second-to-last item as the test set and validation set, and the rest as the training set. We concatenate the training and validation data to predict the last item in the test set. For example, assume that there is one user interaction sequence $S = [i_1, i_2, i_3, i_4, i_5, i_6]$. When we split the dataset, we regard $S_{train} = [i_1, i_2, i_3, i_4]$, $S_{valid} = [i_1, i_2, i_3, i_4, i_5]$, $S_{test} = [i_1, i_2, i_3, i_4, i_5, i_6]$. During training, we use $[i_1]$ to predict i_2, use $[i_1, i_2]$ to predict i_3, and use $[i_1, i_2, i_3]$ to predict i_4. During validation, we use $[i_1, i_2, i_3, i_4]$ to predict i_5. During testing, we use $[i_1, i_2, i_3, i_4, i_5]$ to predict i_6. Following [20, 32], we pair each ground-truth item with 100 randomly sampled negative items according to their popularities.

We report results using Recall@k, MRR@k, and NDCG@k with $k = 5, 10, 20$. Assuming that the tested item of user u is ranked r_u based on the predicted scores, the metrics are calculated as follows:

[1]http://jmcauley.ucsd.edu/data/amazon/.
[2]https://grouplens.org/datasets/movielens/.
[3]https://www.yelp.com/dataset.

- Recall@k: The proportion of cases when the ground-truth item is among the top-k ranked items
- MRR@k: The average of reciprocal ranks of the ground-truth items, i.e., $MRR_u@k = \frac{1}{r_u}$ if $r_u \leq k$ and $MRR_u@k = 0$ otherwise
- NDCG@k: A position-aware metric that assigns larger weights on higher ranks, i.e., $NDCG_u@k = \frac{1}{\log_2(1+r_u)}$ if $r_u \leq k$ and $NDCG_u@k = 0$ otherwise

4.3 Methods Used for Comparison

To assess the effectiveness of MrTransformer (PE), we compare it with the following methods:

- **POP** ranks items in the training set based on their popularity and always recommends the most popular items [11].
- **BPR-MF** is a commonly used matrix factorization method. We apply it for sequential recommendation by representing a new sequence with the average latent factors of items appearing in the sequence so far [12].
- **Item-KNN** computes an item-to-item similarity matrix and recommends items that are similar to the actual item. Regularization is included to avoid coincidental high similarities [24].
- **SASRec** [16] uses a left-to-right self-attention-based model to capture users' sequential behavior.
- **SSE-PT** [65] is a personalized transformer-based model with a regularization technique called SSE [66] to overcome overfitting caused by simply adding user embeddings.
- **FISSA** [23] uses the SASRec model to get the user preference representations, with a transformer-based layer that fuses item similarity to model the global user preference representation; a gating module balances the local and global representations.
- **BERT4Rec** [45] models the user preference representation with a bidirectional transformer network; during training, it randomly masks some items in the sequence and predicts these items jointly conditioned on their left and right context; during testing, it only masks the last item to do recommendations.
- **S3-rec** [82] adopts SASRec as the base model and devises four auxiliary self-supervised objectives; it pre-trains the sequential recommender model using attribute, item, subsequence, and sequence by utilizing mutual information maximization; it fine-tunes the parameters according to the SR task; we do not consider attribute characteristics as they are not supported by the datasets and only use the masked item prediction and sequence prediction loss functions.
- **CL4SRec** [68] proposes three data augmentation methods (crop, mask, and reorder); then it encodes the sequence representation by maximizing the agreement between different augmented methods of the same sequence in the latent space for pre-training the sequence representations.

Other SSL-based methods [e.g., 63, 70, 73, 74, 81] have only been proposed for general, cross-domain, or social recommendation tasks; hence, we omit comparisons against them. We also exclude the SSL-based method in [30], as the authors found flaws in their experimental setup. As for the SSL-based method in [67], they adopt a different experimental setting of next-session recommendation to model the sequence graph representation; instead, we use the next-item recommendation setting that does not satisfy their hypergraph construction assumptions.

4.4 Implementation Details

We implement POP, BPR-MF, and Item-KNN using Tensorflow. For SASRec,[4] SSE-PT,[5] FISSA,[6] BERT4Rec,[7] and S3-rec,[8] we use the code provided by the authors. We use hyperparameters as reported or suggested in the original papers. We implement CL4SRec with Pytorch.

We implement MrTransformer with Tensorflow. All parameters are initialized using a truncated normal distribution in the range $[-0.02, 0.02]$. We set the hidden size to 64 and dropout probability to 0.5. We set the number of transformer layers to 2 and the number of attention heads to 2. For MrTransformer (PE), the training phase contains two stages (i.e., the pre-training and the fine-tuning stage). The learned parameters in the pre-training stage are used to initialize the embedding layers and transformer layers in the fine-tuning stage. In the pre-training stage, we only use the SSL loss (as defined in Equation (14)) to train MrTransformer. In the fine-tuning stage, we use the recommendation loss (as defined in Equation (9)) to train MrTransformer. We try different settings for the number of latent preferences K, the analysis of which can be found in Section 5.4. Here we set $K = 1$ for the "Yelp" dataset, $K = 3$ for the "Beauty" and the "ML-100k" datasets, $K = 9$ for the "Sports" dataset, and $K = 11$ for the "Toys" dataset. We train the model using the Adam optimizer [17]; we set the learning rate as 1e-4, $\beta_1 = 0.99, \beta_2 = 0.999$, l_2 weight decay of 0.01, with linear decay of the learning rate. We also apply gradient clipping with range $[-5, 5]$ during training. To speed up training and convergence, we use a mini-batch size of 256. We test the model performance on the validation set for every epoch. All models are trained on a GeForce GTX TitanX GPU.

For sampling sequence pairs in the learning of preference editing, we sample 20 sequences from the sequence set for each sequence randomly.

5 RESULTS AND ANALYSIS

5.1 Overall Performance of MrTransformer (PE)

To answer RQ1, we report on the performance of MrTransformer (PE) and the baseline methods in terms of Recall, MRR, and NDCG. See Table 2. We obtain the following insights from the results.

First, MrTransformer (PE) achieves the best results on all datasets in terms of all metrics. It outperforms strong transformer-based SR methods (SSE-PT, FISSA) and SSL-based methods (S3-Rec and CL4SRec). Specifically, concerning the transformer-based SR methods on the "Beauty" dataset, the increase over FISSA is 31.55% in terms of MRR@5. As for the SSL-based SR methods, the increase over S3-Rec is 11.91% in terms of Recall@5. On the "ML-100k" dataset, the increase over SSE-PT is 24.39% in terms of Recall@5, while the increase over S3-Rec is 19.13% in terms of NDCG@5. On the "Yelp" dataset, the increase over the transformer-based SR method FISSA is 2.80% in terms of NDCG@10. The increase over the SSL-based SR method BERT4Rec is 8.17% in terms of Recall@20. To sum up, MrTransformer (PE) consistently and in many cases significantly outperforms all the compared methods on these datasets. The improvements are mainly because MrTransformer (PE) achieves better preference extraction and representation by discriminating the common and unique items of multiple preferences between sequences with the help of preference editing. We will analyze the preference editing learning strategy in more depth in Section 5.2.

Second, the differences in performance between MrTransformer (PE) and the baselines on the "ML-100k" dataset are larger than on other datasets. This is related to the characteristics of

[4] https://github.com/kang205/SASRec.
[5] https://github.com/wuliwei9278/SSE-PT.
[6] http://csse.szu.edu.cn/staff/panwk/publications/FISSA/.
[7] https://github.com/FeiSun/BERT4Rec.
[8] https://github.com/RUCAIBox/CIKM2020-S3Rec.

Table 2. Experimental Results on Five Datasets

	Metric	POP	BPR-MF	Item-Knn	SASRec	SSE-PT	FISSA	BERT4Rec	S3-Rec	CL4SRec	MrTransformer (PE)
Beauty	Recall@5	0.0315	0.1390	0.1660	0.1934	0.1959	0.2116	0.2137	0.2173	0.2029	**0.2432**
	Recall@10	0.0628	0.2152	0.2029	0.2653	0.2813	0.3079	0.2964	0.2909	0.3086	**0.3231**
	Recall@20	0.1280	0.3292	0.2695	0.3724	0.3901	0.4289	0.3971	0.3926	0.4199	**0.4354***
	MRR@5	0.0147	0.0755	0.1259	0.1112	0.1199	0.1198	0.1299	0.1348	0.1355	**0.1576***
	MRR@10	0.0187	0.0855	0.1308	0.1220	0.1286	0.1325	0.1408	0.1445	0.1467	**0.1681**
	MRR@20	0.0230	0.0933	0.1352	0.1291	0.1327	0.1408	0.1477	0.1514	0.1543	**0.1758***
	NDCG@5	0.0188	0.0912	0.1359	0.1436	0.1491	0.1424	0.1506	0.1552	0.1573	**0.1788**
	NDCG@10	0.0288	0.1157	0.1477	0.1633	0.1683	0.1734	0.1773	0.1789	0.1847	**0.2045***
	NDCG@20	0.0449	0.1443	0.1644	0.1823	0.1944	0.2039	0.2026	0.2044	0.2127	**0.2327***
Sports	Recall@5	0.0313	0.0376	0.1512	0.1622	0.2102	0.2133	0.2124	0.2072	0.2166	**0.2421**
	Recall@10	0.0624	0.0819	0.1927	0.2788	0.3212	0.3180	0.3105	0.3250	0.3381	**0.3691***
	Recall@20	0.1364	0.1735	0.2260	0.3908	0.4492	0.4592	0.4464	0.4817	0.5036	**0.5318**
	MRR@5	0.0139	0.0167	0.0449	0.0913	0.1180	0.1262	0.1227	0.1122	0.1194	**0.1363**
	MRR@10	0.0178	0.0224	0.0547	0.1042	0.1344	0.1367	0.1356	0.1276	0.1353	**0.1530***
	MRR@20	0.0227	0.0285	0.1224	0.1105	0.1464	0.1486	0.1449	0.1384	0.1467	**0.1642***
	NDCG@5	0.0181	0.0218	0.0783	0.1102	0.1407	0.1325	0.1449	0.1356	0.1434	**0.1622**
	NDCG@10	0.0279	0.0359	0.1132	0.1532	0.1780	0.1726	0.1764	0.1734	0.1824	**0.2032***
	NDCG@20	0.0464	0.0587	0.1550	0.1704	0.2126	0.2157	0.2106	0.2128	0.2240	**0.2442**
Toys	Recall@5	0.0348	0.0487	0.0986	0.2411	0.2940	0.3127	0.2980	0.3110	0.3118	**0.3420***
	Recall@10	0.0697	0.0970	0.2344	0.3417	0.4001	0.4062	0.3878	0.4094	0.4141	**0.4366***
	Recall@20	0.1443	0.1995	0.2360	0.4407	0.5344	0.5134	0.5058	0.5282	0.5455	**0.5565***
	MRR@5	0.0161	0.0226	0.0388	0.1530	0.1753	0.1934	0.1932	0.1964	0.1975	**0.2308***
	MRR@10	0.0206	0.0288	0.0562	0.1630	0.1882	0.2098	0.2050	0.2096	0.2110	**0.2434**
	MRR@20	0.0256	0.0356	0.0665	0.1707	0.1960	0.2200	0.2131	0.2177	0.2200	**0.2516**
	NDCG@5	0.0207	0.0290	0.0534	0.1749	0.2047	0.2254	0.2192	0.2249	0.2259	**0.2584**
	NDCG@10	0.0318	0.0443	0.0965	0.2177	0.2380	0.2653	0.2481	0.2567	0.2588	**0.2890***
	NDCG@20	0.0505	0.0699	0.1325	0.2274	0.2708	0.3025	0.2778	0.2866	0.2919	**0.3192***
ML-100k	Recall@5	0.0965	0.1866	0.1845	0.2948	0.3017	0.2585	0.3003	0.3012	0.3245	**0.3753***
	Recall@10	0.1431	0.3138	0.3276	0.4746	0.4688	0.4492	0.4662	0.4761	0.5133	**0.5186***
	Recall@20	0.2396	0.4655	0.4835	0.6548	0.6723	0.6409	0.6506	0.6755	**0.6925**	0.6713
	MRR@5	0.0484	0.0989	0.0958	0.1753	0.1748	0.1311	0.1707	0.1513	0.1765	**0.2104***
	MRR@10	0.0549	0.1154	0.1152	0.1984	0.2046	0.1555	0.1926	0.1748	0.2015	**0.2290**
	MRR@20	0.0613	0.1258	0.1257	0.2117	0.2183	0.1690	0.2058	0.1897	0.2140	**0.2395***
	NDCG@5	0.0602	0.1203	0.1176	0.2035	0.2106	0.1624	0.2028	0.1881	0.2131	**0.2509**
	NDCG@10	0.0755	0.1610	0.1642	0.2655	0.2599	0.2230	0.2562	0.2448	0.2739	**0.2967**
	NDCG@20	0.0996	0.1992	0.2032	0.3088	0.2993	0.2718	0.3033	0.2985	0.3193	**0.3352***
Yelp	Recall@5	0.0474	0.0500	0.0499	0.4070	0.3884	0.4520	0.4561	0.4224	0.4483	**0.4677***
	Recall@10	0.0960	0.0983	0.1080	0.5691	0.5609	0.6153	0.6138	0.5926	0.6258	**0.6523***
	Recall@20	0.1897	0.1970	0.2278	0.7414	0.7684	0.7757	0.7621	0.7625	0.8118	**0.8244***
	MRR@5	0.0227	0.0230	0.0224	0.2265	0.2261	0.2690	**0.2703**	0.2439	0.2515	0.2630
	MRR@10	0.0290	0.0307	0.0298	0.2476	0.2492	**0.2974**	0.2914	0.2666	0.2771	0.2876
	MRR@20	0.0353	0.0365	0.0380	0.2592	0.2686	0.3002	**0.3017**	0.2783	0.2895	0.2996
	NDCG@5	0.0288	0.0291	0.0291	0.2699	0.2663	0.3140	**0.3163**	0.2880	0.3027	0.3136
	NDCG@10	0.0443	0.0447	0.0476	0.3242	0.3224	0.3631	0.3673	0.3430	0.3647	**0.3733***
	NDCG@20	0.0677	0.0691	0.0777	0.3632	0.3789	0.4089	0.4048	0.3860	0.4098	**0.4168***

Boldface indicates the best results in terms of the corresponding metrics. Significant improvements over the best baseline results are marked with * (t-test, $p < .05$). We set K = 1 for the "Yelp" dataset, K = 3 for the "Beauty" and the "ML-100k" datasets, K = 9 for the "Sports" dataset, and K = 11 for the "Toys" dataset.

different datasets. From Table 1, we can observe that the average sequence length of the "ML-100k" dataset is much larger than other datasets, the number of items is less than other datasets, and it's denser than other datasets. Therefore, we speculate that there are more extensive and diverse preferences behind a user's interaction sequences on the "ML-100k" dataset. Since MrTransformer (PE) is especially adept at extracting multiple preferences and capturing their commonness and uniqueness through preference editing, its advantages are more obvious on "ML-100k."

Third, on most datasets the improvements in terms of NDCG and MRR tend to be larger than those in Recall. This demonstrates that MrTransformer (PE) is beneficial to the ranking of the list of recommendations. Compared with the improvements of the best baseline over the second best, the largest increase of CL4SRec over S3-Rec on the "Beauty" dataset achieves 6.18% in terms of Recall@10. On the "ML-100k" dataset, the largest increase of CL4SRec over SSE-PT achieves the

Table 3. Recall, MRR, and NDCG of MrTransformer without Preference Editing (Top) and with Preference Editing (Bottom) on all Datasets

Datasets	Recall			MRR			NDCG		
	@5	@10	@20	@5	@10	@20	@5	@10	@20
MrTransformer									
Beauty	0.2220	0.2989	0.4070	0.1438	0.1540	0.1614	0.1632	0.1879	0.2151
Sports	0.2069	0.3225	0.4829	0.1146	0.1297	0.1408	0.1374	0.1744	0.2149
Toys	0.3023	0.3955	0.5224	0.1969	0.2092	0.2178	0.2231	0.2530	0.2849
ML-100k	0.3436	0.4920	0.6648	0.1929	0.2119	0.2230	0.2315	0.2784	0.3209
Yelp	0.4403	0.6086	0.7771	0.2609	0.2834	0.2951	0.3053	0.3597	0.4023
MrTransformer (PE)									
Beauty	0.2432	0.3231	0.4354	0.1576	0.1681	0.1758	0.1788	0.2045	0.2327
Sports	0.2421	0.3691	0.5318	0.1363	0.1530	0.1642	0.1622	0.2032	0.2442
Toys	0.3420	0.4366	0.5565	0.2308	0.2434	0.2516	0.2584	0.2890	0.3192
ML-100k	0.3753	0.5186	0.6713	0.2104	0.2290	0.2395	0.2509	0.2967	0.3352
Yelp	0.4677	0.6523	0.8244	0.2630	0.2876	0.2996	0.3136	0.3733	0.4168

improvement of 7.55% in terms of Recall@5; the improvements of MrTransformer (PE) on Recall is already considered large. Generally, SSL-based SR methods outperform transformer-based SR methods, which indicates that SSL is an effective direction for SRs by introducing more supervision signals to improve representation learning. Moreover, the SSL-based method S3-Rec outperforms other SSL-based methods, e.g., BERT4Rec, on almost all datasets. This is because S3-Rec also leverages sequence-level self-supervised signals besides the item-level signals, which further confirms that self-supervised signals are very helpful for improving the recommendation performance.

5.2 Effect of Preference Editing

To answer RQ2, we conduct an ablation study to analyze the effects of the preference editing (PE) learning strategy. We compare MrTransformer (PE) with MrTransformer where MrTransformer is only trained with the recommendation loss (Equation (8)) and the preference coverage loss (Equation (6)). The results are shown in Table 3.

On all datasets, we can observe that the differences between MrTransformer and MrTransformer (PE) are significant. The results decrease when comparing MrTransformer (PE) vs. MrTransformer, i.e., by removing preference editing. The results drop by more than 18.15%pt in terms of MRR@5 on the "Sports" dataset, with similar results on the "Beauty" and "Toys" datasets. And the results drop by more than 4.80%pt in terms of Recall@5 on the "ML-100k" dataset. On the "Yelp" dataset, the results drop by more than 3.70%pt in terms of NDCG@20, respectively. This demonstrates that preference editing is effective by forcing MrTransformer to learn common and unique items between sequences. At the same time, we see that the MRR and NDCG of MrTransformer are slightly higher than those of strong baselines such as BERT4Rec and FISSA, while the Recall values are comparable on most datasets. This is in line with expectations, because MrTransformer is also a transformer-based method, and without preference editing, its improvements are limited.

Next, we analyze the performance on three datasets (i.e., "Sports" as a representative of the Amazon datasets, "ML-100k," "Yelp") to understand the effects of preference editing on sequences with different lengths. Specifically, we divide the sequences into four groups according to their length in terms of the number of items interacted with: $[20, 30)$, $[30, 40)$, $[40, 50]$, $[50, \infty)$. The results are shown in Tables 4 through 6.

Table 4. Recall, MRR, and NDCG of MrTransformer without Preference Editing (top) and with Preference Editing (Bottom) of Different Sequence Lengths on the "Sports" Dataset

Length	Number	Recall			MRR			NDCG		
		@5	@10	@20	@5	@10	@20	@5	@10	@20
MrTransformer										
[20, 30)	5,062	0.2176	0.3387	0.4846	0.1267	0.1424	0.1525	0.1492	0.1879	0.2246
[30, 40)	858	0.2172	0.3252	0.4521	0.1317	0.1455	0.1543	0.1528	0.1871	0.2192
[40, 50]	241	0.2018	0.3211	0.4678	0.1233	0.1392	0.1497	0.1425	0.1810	0.2185
[50, ∞)	178	0.2125	0.3375	0.4560	0.1295	0.1430	0.1520	0.1486	0.1822	0.2153
MrTransformer (PE)										
[20, 30)	5,062	0.2263	0.3413	0.4963	0.1347	0.1495	0.1602	0.1573	0.1940	0.2331
[30, 40)	858	0.2496	0.3589	0.4958	0.1445	0.1586	0.1683	0.1705	0.2089	0.2457
[40, 50]	241	0.2247	0.3394	0.4908	0.1376	0.1523	0.1625	0.1591	0.1956	0.2334
[50, ∞)	178	0.2313	0.3438	0.4938	0.1523	0.1665	0.1770	0.1718	0.2073	0.2403

The highest scores (per metric and method) are underlined.

Table 5. Recall, MRR, and NDCG of MrTransformer without Preference Editing (top) and with Preference Editing (Bottom) of Different Sequence Lengths on the "ML-100k" Dataset

Length	Number	Recall			MRR			NDCG		
		@5	@10	@20	@5	@10	@20	@5	@10	@20
MrTransformer										
[20, 30)	157	0.3872	0.6011	0.7645	0.1949	0.2232	0.2354	0.2424	0.3113	0.3554
[30, 40)	117	0.4220	0.5871	0.7706	0.2457	0.2670	0.2798	0.2888	0.3414	0.3880
[40, 50]	80	0.4545	0.6590	0.7840	0.2464	0.2713	0.2827	0.3032	0.3628	0.4034
[50, ∞)	589	0.2772	0.4176	0.5857	0.1529	0.1712	0.1827	0.1835	0.2284	0.2708
MrTransformer (PE)										
[20, 30)	157	0.4335	0.6127	0.7861	0.2430	0.2668	0.2788	0.2927	0.3516	0.3954
[30, 40)	117	0.4403	0.6238	0.7798	0.2441	0.2694	0.2793	0.2927	0.3528	0.3896
[40, 50]	80	0.4785	0.6785	0.8095	0.2333	0.2544	0.2668	0.2924	0.3462	0.3953
[50, ∞)	589	0.3032	0.4228	0.5875	0.1604	0.1772	0.1887	0.1955	0.2350	0.2767

The highest scores (per metric and method) are underlined.

From Tables 4 through 6, we see that MrTransformer (PE) outperforms MrTransformer in terms of most metrics over all sequence length groups on all three datasets. This verifies the effectiveness of preference editing. On the "Sports" dataset, MrTransformer (PE) achieves the largest increase when the sequence length is between 30 and 40. It gains 14.91% in terms of Recall@5. On the "ML-100k" dataset, when the sequence length is between 20 and 30, MrTransformer (PE) achieves the largest increase. It gains 24.67% in terms of MRR@5. This indicates that the gains brought by preference editing are bigger for sequences that are less than 50 in length. We also see that on different datasets the best performance is achieved for different-length groups. It achieves the best performances in the groups of [30, 40), [40, 50), [20, 30) on the "Sports," "ML-100k," and "Yelp" datasets, respectively. We believe that this is because of the varied length distributions on different datasets. From Table 1, we can see that the average length of "Sports" and "Yelp" datasets is shorter than that of the "ML-100k" dataset. Hence, the biggest improvements are achieved for longer sequences with a length of no more than 50 on the "ML-100k" dataset.

Table 6. Recall, MRR, and NDCG of MrTransformer without Preference Editing (Top) and with Preference Editing (Bottom) of Different Sequence Lengths on the "Yelp" Dataset

Length	Number	Recall			MRR			NDCG		
		@5	@10	@20	@5	@10	@20	@5	@10	@20
MrTransformer										
$[20, 30)$	1,736	<u>0.4730</u>	<u>0.6405</u>	<u>0.8035</u>	<u>0.2842</u>	<u>0.3065</u>	<u>0.3178</u>	<u>0.3310</u>	<u>0.3851</u>	<u>0.4263</u>
$[30, 40)$	679	0.4326	0.5879	0.7606	0.2482	0.2693	0.2814	0.2938	0.3444	0.3882
$[40, 50]$	327	0.4512	0.6017	0.7694	0.2653	0.2845	0.2871	0.3104	0.3562	0.3971
$[50, \infty)$	397	0.3756	0.5405	0.7135	0.2171	0.2393	0.2512	0.2563	0.3098	0.3534
MrTransformer (PE)										
$[20, 30)$	1,736	<u>0.4840</u>	<u>0.6637</u>	<u>0.8139</u>	<u>0.2950</u>	<u>0.3193</u>	<u>0.3300</u>	<u>0.3418</u>	<u>0.4001</u>	<u>0.4385</u>
$[30, 40)$	679	0.4437	0.6371	0.8035	0.2569	0.2832	0.2947	0.3028	0.3658	0.4078
$[40, 50]$	327	0.4675	0.6234	0.7792	0.2661	0.2864	0.2948	0.3155	0.3655	0.4045
$[50, \infty)$	397	0.3946	0.5757	0.7703	0.2309	0.2547	0.2679	0.2714	0.3297	0.3786

The highest scores (per metric and method) are underlined.

Table 7. Recall, MRR, and NDCG of MrTransformer (PE) with and without Coverage Mechanism on Three Representative Datasets

MrTransformer (PE)		Recall			MRR			NDCG		
		@5	@10	@20	@5	@10	@20	@5	@10	@20
Beauty	−coverage	0.2169	0.2913	0.4012	0.1421	0.1520	0.1591	0.1606	0.1846	0.2111
	+coverage	**0.2432**	**0.3231***	**0.4354***	**0.1576**	**0.1681**	**0.1758***	**0.1788***	**0.2045***	**0.2327***
Toys	−coverage	0.3251	0.4191	0.5428	0.2176	0.2301	0.2385	0.2443	0.2746	0.3057
	+coverage	**0.3420**	**0.4366***	**0.5565**	**0.2308***	**0.2434***	**0.2516**	**0.2584***	**0.2890**	**0.3192***
ML-100k	−coverage	0.3351	0.4984	0.6702	0.1916	0.2131	0.2248	0.2270	0.2796	0.3227
	+coverage	**0.3753***	**0.5186**	**0.6716**	**0.2104***	**0.2290***	**0.2395***	**0.2509***	**0.2967***	**0.3352***

Boldface indicates the better results. Significant improvements over MrTransformer without coverage mechanism results are marked with * (t-test, $p < .05$).

Finally, when the sequence length is beyond 50, the performance drops sharply on all three datasets. This reveals the limitations of MrTransformer and MrTransformer (PE) for very long sequences.

5.3 Effect of Preference Coverage

To answer RQ3, we remove the coverage mechanism from MrTransformer (PE) and keep the other settings unchanged. The results of MrTransformer (PE) with and without the coverage mechanism are shown in Table 7.

As in Section 5.2, we select another three datasets ("Beauty," "Toys," and "ML-100k") for verification. On all datasets, the results of MrTransformer (PE) with preference coverage are significantly higher than MrTransformer (PE) without preference coverage in terms of most metrics. On the "Beauty" dataset, adding preference coverage results in an increase of 10.91% and 10.78% in terms of Recall@10 and NDCG@10, respectively. On the "ML-100k" dataset, adding preference coverage achieves an increase of 11.99% and 9.81% in terms of Recall@5 and MRR@5. This is because without preference coverage, different preferences might attend to the same items, resulting in redundant preferences and low coverage of the whole sequence.

Table 8. Results of MrTransformer (PE) on the "Beauty" Dataset with Different Values of K

K	Recall			MRR			NDCG		
	@5	@10	@20	@5	@10	@20	@5	@10	@20
1	0.2267	0.3068	0.4185	0.1430	0.1566	0.1642	0.1660	0.1918	0.2198
3	**0.2432**	**0.3231**	**0.4354**	**0.1576**	**0.1681**	**0.1758**	**0.1788**	**0.2045**	**0.2327**
5	0.2263	0.3057	0.4217	0.1432	0.1536	0.1615	0.1638	0.1893	0.2184
7	0.2250	0.3043	0.4176	0.1422	0.1567	0.1644	0.1657	0.1913	0.2197
9	0.2294	0.3013	0.4170	0.1450	0.1570	0.1637	0.1638	0.1917	0.2215
11	0.2237	0.3051	0.4217	0.1441	0.1523	0.1633	0.1642	0.1904	0.2197
15	0.2248	0.3021	0.4115	0.1438	0.1550	0.1625	0.1648	0.1895	0.2170
20	0.2234	0.3071	0.4173	0.1437	0.1546	0.1621	0.1634	0.1903	0.2180

Boldface indicates the value of K for which the highest performance is achieved for a given metric.

The effects of the preference coverage vary on different datasets. For example, compared with the other datasets, adding preference coverage only results in minor increases of 2.52% and 6.06% in terms of Recall@20 and MRR@5, respectively, on the "Toys" dataset. A possible reason for these differences in effect is that the characteristics of the datasets are different, where the number of items interacted with by users is smaller and the data is more sparse in the "Toys" dataset than in the "ML-100k" dataset. This may result in little similarity between items. In this case, the attention distribution in the transformer itself is not very concentrated to begin with, so the increase caused by the coverage mechanism is not so large.

MrTransformer (PE) without preference coverage performs worse than some of the baselines on the "Beauty" dataset. This demonstrates the importance and complementary effect of preference coverage for preference modeling. Comparing the results of MrTransformer in Tables 7 and 2, we see that removing the preference editing strategy results in worse performance, in general, than removing the preference coverage mechanism. This demonstrates that the preference editing strategy plays a leading role in improving recommendation performance.

5.4 Impact of the Hyperparameter K

Recall that MrTransformer (PE) concatenates K special tokens to represent user preferences. Note that K represents latent user preferences for the whole dataset, not for a particular sequence. To answer RQ4 and study how K affects the recommendation performance of MrTransformer (PE), we compare different values of K while keeping other settings unchanged. We consider $K \in \{1, 3, 5, 7, 9, 11, 15, 20\}$.

There is no uniform setting of K that achieves the best performance on all metrics on all datasets. Tables 8 through 12 show that the best values of Recall, MRR, and NDCG are achieved when $K = 3$ on the "Beauty" and "ML-100k" datasets, $K = 9$ on the "Sports" dataset, $K = 11$ on the "Toys" dataset, and $K = 1$ on the "Yelp" dataset. This reflects that user preferences on the "Beauty," "ML-100k," and "Yelp" datasets are more centralized than those on the "Sports" and "Toys" datasets. This does not conflict with our findings in Section 5.1 that the improvements of MrTransformer (PE) on the "ML-100k" dataset are larger than those on other datasets and the advantages of MrTransformer (PE) are more obvious on "ML-100k", as K represents the number of latent preferences for all sequences in the whole dataset in general, not for a particular sequence. MrTransformer (PE) still works well when $K = 1$. This reflects that user preferences on some datasets are diverse, but sometimes users do have only one preference behind an interaction sequence on other datasets. We calculate the cosine similarity between different learned user preferences. On the "Beauty" dataset, there are three user preferences (P_1, P_2, and P_3). The similarity between P_1 and P_2 is 0.26,

Table 9. Results of MrTransformer (PE) on the “Sports” Dataset with Different Values of K

K	Recall			MRR			NDCG		
	@5	@10	@20	@5	@10	@20	@5	@10	@20
1	0.2339	0.3401	0.4812	0.1375	0.1514	0.1610	0.1613	0.1954	0.2309
3	0.2238	0.3334	0.4840	0.1307	0.1451	0.1554	0.1537	0.1889	0.2268
5	0.2283	0.3375	0.4905	0.1324	0.1468	0.1573	0.1560	0.1912	0.2297
7	0.2284	0.3397	0.4956	0.1334	0.1480	0.1586	0.1569	0.1926	0.2318
9	**0.2421**	**0.3691**	**0.5318**	**0.1363**	**0.1530**	**0.1642**	**0.1622**	**0.2032**	**0.2442**
11	0.2270	0.3349	0.4865	0.1335	0.1476	0.1579	0.1566	0.1912	0.2293
15	0.2285	0.3427	0.4965	0.1334	0.1485	0.1589	0.1618	0.1936	0.2322
20	0.2175	0.3317	0.4845	0.1259	0.1410	0.1514	0.1485	0.1853	0.2236

Boldface indicates the value of K for which the highest performance is achieved for a given metric.

Table 10. Results of MrTransformer (PE) on the “Toys” Dataset with Different Values of K

K	Recall			MRR			NDCG		
	@5	@10	@20	@5	@10	@20	@5	@10	@20
1	0.3215	0.4166	0.5356	0.2156	0.2282	0.2363	0.2419	0.2725	0.3025
3	0.3230	0.4177	0.5389	0.2171	0.2296	0.2379	0.2434	0.2739	0.3044
5	0.3202	0.4162	0.5446	0.2142	0.2269	0.2357	0.2405	0.2715	0.3038
7	0.3257	0.4243	0.5466	0.2160	0.2290	0.2374	0.2432	0.2750	0.3037
9	0.3123	0.4069	0.5305	0.2044	0.2169	0.2254	0.2312	0.2616	0.2928
11	**0.3420**	**0.4366**	**0.5565**	**0.2308**	**0.2434**	**0.2516**	**0.2584**	**0.2890**	**0.3192**
15	0.2997	0.3982	0.5317	0.1913	0.2043	0.2143	0.2182	0.2499	0.2835
20	0.3006	0.3968	0.5208	0.1981	0.2108	0.2193	0.2236	0.2546	0.2857

Boldface indicates the value of K for which the highest performance is achieved for a given metric.

the similarity between P_1 and P_3 is 0.37, and the similarity between P_2 and P_3 is 0.24. While on the “Yelp” dataset the similarity between P_1 and P_2 is 0.77, the similarity between P_2 and P_3 is 0.81. The result shows that user preferences are diverse on some datasets, like “Beauty,” while they are more concentrated on others, like “Yelp.” It is also in line with the number of preferences that users reflect in an interaction sequence in most cases. The lowest scores are obtained when K is set to a very large value (e.g., $K = 15, 20$) on most datasets.

Importantly, although the value of K can affect recommendation performance, the influence is limited. On the “Beauty” dataset, the largest gap between the best and the worst performance is 5.80% and 7.23% in terms of Recall@20 and NDCG@20, respectively.

6 CONCLUSIONS AND FUTURE WORK

We have proposed a transformer-based model named MrTransformer and introduced a novel SSL strategy, i.e., *preference editing*. Different from traditional SR methods, which use the next item of the sequence as the supervision signal to train the model, we have designed SSL signals between interaction sequences for the SR scenario. We have conducted extensive experiments and analyses on five benchmark datasets to show the effectiveness of MrTransformer and preference editing.

Main Findings. Our experimental results demonstrate that MrTransformer with preference editing significantly outperforms state-of-the-art methods in terms of Recall, MRR, and NDCG. We find that the differences between MrTransformer (PE) and other baselines in different datasets are caused by the characteristics of different datasets. MrTransformer (PE) is especially adept at extracting multiple preferences and capturing their commonness and uniqueness through preference

Table 11. Results of MrTransformer (PE) on the "ML-100k" Dataset with Different Values of K

K	Recall			MRR			NDCG		
	@5	@10	@20	@5	@10	@20	@5	@10	@20
1	0.3595	0.5143	0.6808	0.1975	0.2188	0.2297	0.2375	0.2896	0.3297
3	**0.3753**	**0.5186**	0.6716	**0.2104**	**0.2290**	**0.2395**	**0.2509**	**0.2967**	**0.3352**
5	0.3521	0.5068	**0.6903**	0.1991	0.2193	0.2319	0.2367	0.2862	0.3325
7	0.3383	0.4973	0.6787	0.1948	0.2160	0.2289	0.2303	0.2817	0.3280
9	0.3521	0.4942	0.6861	0.1961	0.2154	0.2289	0.2346	0.2809	0.3297
11	0.3351	0.4942	0.6585	0.1940	0.2150	0.2265	0.2287	0.2799	0.3216
15	0.3298	0.4835	0.6702	0.2030	0.2211	0.2341	0.2322	0.2823	0.3294
20	0.3552	0.5121	0.6935	0.2010	0.2214	0.2338	0.2308	0.2790	0.3217

Boldface indicates the value of K for which the highest performance is achieved for a given metric.

Table 12. Results of MrTransformer (PE) on the "Yelp" Dataset with Different Values of K

K	Recall			MRR			NDCG		
	@5	@10	@20	@5	@10	@20	@5	@10	@20
1	**0.4677**	**0.6523**	**0.8244**	0.2630	0.2876	0.2996	0.3136	**0.3733**	**0.4168**
3	0.4500	0.6284	0.8039	**0.2787**	**0.2918**	**0.3040**	**0.3154**	0.3630	0.4075
5	0.4447	0.6173	0.7877	0.2541	0.2771	0.2889	0.3012	0.3570	0.4000
7	0.4514	0.6300	0.8012	0.2602	0.2841	0.2960	0.3075	0.3652	0.4086
9	0.4385	0.6148	0.7885	0.2490	0.2725	0.2846	0.2958	0.3528	0.3968
11	0.4587	0.6361	0.8055	0.2592	0.2830	0.2948	0.3086	0.3660	0.4089
15	0.4125	0.5870	0.7634	0.2295	0.2528	0.2651	0.2747	0.3312	0.3758
20	0.4354	0.6147	0.7918	0.2488	0.2727	0.2851	0.2949	0.3529	0.3978

Boldface indicates the value of K for which the highest performance is achieved for a given metric.

editing, when the user preferences are changeable in some cases. MrTransformer (PE) is beneficial to the ranking of the list of recommendations compared to the Recall metric. Moreover, preference editing is effective by forcing MrTransformer to learn common and unique items between sequences, as the results decrease when comparing MrTransformer (PE) and MrTransformer, i.e., by removing preference editing. Both MrTransformer and MrTransformer (PE) have limitations on modeling very long sequences. Furthermore, we observe that the model MrTransformer (PE) is not sensitive to hyperparameter K, which represents latent user preferences for the whole dataset.

Broader Implications. We believe that the model MrTransformer (PE) has further uses in the context of recommender systems, including cold-start scenarios and privacy-preserving recommendations. What's more, this SSL method is not limited to recommendation tasks but can also be used in NLP tasks to mine the relationship between adjacent sentences by exploring the commonness and uniqueness of them.

Limitations and Future Work. At the same time, MrTransformer also has limitations. Importantly, it is not able to effectively explore multiple user preferences from very long sequences. In addition, we sample 20 sequences for each sequence randomly. More effective sampling algorithms can be designed to improve the performance. MrTransformer can be advanced and extended in several directions. First, rich side information can be taken into consideration for user preference extraction and representation, as well as for SSL. Second, variants of MrTransformer can be applied to other recommendation tasks by introducing other information, such as conversational recommendations.

REPRODUCIBILITY

To facilitate the reproducibility of the results obtained in this article, we share (i) the datasets, (ii) code, and (iii) parameter files used in this article at https://github.com/mamuyang/MrTransformer.

ACKNOWLEDGMENTS

We thank our editor and reviewers for their constructive feedback that helped us to improve the paper.

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. 2015. Optimal greedy diversity for recommendation. In *The 24th International Joint Conference on Artificial Intelligence*. 1742–1748.

[2] Wanyu Chen, Pengjie Ren, Fei Cai, Fei Sun, and Maarten de Rijke. 2020. Improving end-to-end sequential recommendations with intent-aware diversification. In *The 29th Conference on Information and Knowledge Management*. 175–184.

[3] Wanyu Chen, Pengjie Ren, Fei Cai, Fei Sun, and Maarten de Rijke. 2021. Multi-interest diversification for end-to-end sequential recommendation. *ACM Transactions on Information Systems* 40 (2021), 1–30.

[4] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *The 11th Conferences on Web-inspired Research Involving Search and Data Mining*. 108–116.

[5] Sung Min Cho, Eunhyeok Park, and Sungjoo Yoo. 2020. MEANTIME: Mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In *The 14th ACM Conference on Recommender Systems*. 515–520.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *The North American Chapter of the Association for Computational Linguistics*. 4171–4186.

[7] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *The 11th ACM Conference on Recommender Systems*. 152–160.

[8] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *The Conference on Empirical Methods in Natural Language Processing*. 1615–1625.

[9] Cheng Guo, Mengfei Zhang, Jinyun Fang, Jiaqi Jin, and Mao Pan. 2020. Session-based recommendation with hierarchical leaping networks. In *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 1705–1708.

[10] Lei Guo, Li Tang, Tong Chen, Lei Zhu, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2021. DA-GCN: A domain-aware attentive graph convolution network for shared-account cross-domain sequential recommendation. In *The 30th International Joint Conference on Artificial Intelligence*. 2483–2489.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *The 26th Web Conference*. 173–182.

[12] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *The 4th International Conference on Learning Representations*.

[13] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *The 10th ACM Conference on Recommender Systems*. 241–248.

[14] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *The 12th Conferences on Web-inspired Research Involving Search and Data Mining*. 573–581.

[15] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM Conference on Research and Development in Information Retrieval*. 505–514.

[16] Wangcheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *The 18th International Conference on Data Mining*. 197–206.

[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *The 3rd International Conference on Learning Representations.*

[18] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *The 8th International Conference on Learning Representations.*

[19] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *The 58th Association for Computational Linguistics.* 7871–7880.

[20] Chenliang Li, Xichuan Niu, Xiangyang Luo, Zhenzhong Chen, and Cong Quan. 2019. A review-driven neural model for sequential recommendation. In *The 28th International Joint Conference on Artificial Intelligence.* 2866–2872.

[21] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *The 26th Conference on Information and Knowledge Management.* 1419–1428.

[22] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *The 13th Conferences on Web-inspired Research Involving Search and Data Mining.* 322–330.

[23] Jing Lin, Weike Pan, and Zhong Ming. 2020. FISSA: Fusing item similarity models with self-attention networks for sequential recommendation. In *The 14th ACM Conference on Recommender Systems.* 130–139.

[24] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 1 (2003), 76–80.

[25] Chang Liu, Xiaoguang Li, Guohao Cai, Zhenhua Dong, Lifeng Shang, and Hong Zhu. 2021. Noninvasive self-attention for side information fusion in sequential recommendation. In *The 35th Conference on Artificial Intelligence.*

[26] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-term attention/memory priority model for session-based recommendation. In *The 24th ACM International Conference on Knowledge Discovery and Data Mining.* 1831–1839.

[27] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[28] Zhuang Liu, Yunpu Ma, Yuanxin Ouyang, and Zhang Xiong. 2021. Contrastive learning for recommender system. abs/2101.01317.

[29] 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 *The 29th International Joint Conference on Artificial Intelligence.* 2591–2597.

[30] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled self-supervision in sequential recommenders. In *The 26th ACM International Conference on Knowledge Discovery and Data Mining.* 483–491.

[31] Muyang Ma, Pengjie Ren, Yujie Lin, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. π-Net: A parallel information-sharing network for shared-account cross-domain sequential recommendations. In *The 42nd International ACM Conference on Research and Development in Information Retrieval.* 685–694.

[32] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *The 38th International ACM Conference on Research and Development in Information Retrieval.* 43–52.

[33] Wenjing Meng, Deqing Yang, and Yanghua Xiao. 2020. Incorporating user micro-behaviors and item knowledge into multi-task learning for session-based recommendation. In *The 43rd International ACM Conference on Research and Development in Information Retrieval.* 1091–1100.

[34] Fei Mi, Xiaoyu Lin, and Boi Faltings. 2020. Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *The 14th ACM Conference on Recommender Systems.* 408–413.

[35] Zhiqiang Pan, Fei Cai, Wanyu Chen, Chonghao Chen, and Chen Honghui. 2022. Collaborative graph learning for session-based recommendation. *ACM Transactions on Information Systems* 40 (2022), 1–26.

[36] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. An intent-guided collaborative machine for session-based recommendation. In *The 43rd International ACM Conference on Research and Development in Information Retrieval.* 1833–1836.

[37] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking item importance in session-based recommendation. In *The 43rd International ACM Conference on Research and Development in Information Retrieval.* 1837–1840.

[38] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. 2020. GAG: Global attributed graph neural network for streaming session-based recommendation. In *The 43rd International ACM Conference on Research and Development in Information Retrieval.* 669–678.

[39] Massimo Quadrana, Alexandros Karatzoglou, Balzs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *The 11th ACM Conference on Recommender Systems.* 130–137.

[40] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2019. RepeatNet: A repeat aware neural recommendation machine for session-based recommendation. In *The 33rd Conference on Artificial Intelligence*. 4806–4813.

[41] Pengjie Ren, Zhaochun Ren, Fei Sun, Xiangnan He, Dawei Yin, and Maarten de Rijke. 2020. NLP4REC: The WSDM 2020 workshop on natural language processing for recommendations. In *The 13th Conferences on Web-inspired Research Involving Search and Data Mining*. 907–908.

[42] Ruiyang Ren, Zhaoyang Liu, Yaliang Li, Wayne Xin Zhao, Hui Wang, Bolin Ding, and Ji-Rong Wen. 2020. Sequential recommendation with self-attentive multi-adversarial network. In *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 89–98.

[43] Ruiyang Ren, Zhaoyang Liu, Yaliang Li, Wayne Xin Zhao, Hui Wang, Bolin Ding, and Ji-Rong Wen. 2020. Sequential recommendation with self-attentive multi-adversarial network. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 89–98.

[44] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *The 12th Conferences on Web-inspired Research Involving Search and Data Mining*. 555–563.

[45] 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 *The 28th Conference on Information and Knowledge Management*. 1441–1450.

[46] Peijie Sun, Le Wu, and Meng Wang. 2018. Attentive recurrent social recommendation. In *The 41st International ACM Conference on Research and Development in Information Retrieval*. 185–194.

[47] Peijie Sun, Le Wu, Kun Zhang, Yanjie Fu, Richang Hong, and Meng Wang. 2020. Dual learning for explainable recommendation: Towards unifying user preference prediction and review generation. In *The 29th Web Conference*. 837–847.

[48] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A. Efros. 2019. Unsupervised domain adaptation through self-supervision. In *arXiv preprint arXiv:1909.11825*.

[49] Yang Sun, Fajie Yuan, Min Yang, Guoao Wei, Zhou Zhao, and Duo Liu. 2020. A generic network compression framework for sequential recommender systems. In *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 1299–1308.

[50] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *The 11th Conferences on Web-inspired Research Involving Search and Data Mining*. 565–573.

[51] Yu Tian, Jianxin Chang, Yannan Niu, Yang Song, and Chenliang Li. 2022. When multi-level meets multi-interest: A multi-grained neural model for sequential recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[52] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. In *arXiv preprint arXiv:1807.03748*.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[54] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item recommendation with sequential hypergraphs. In *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 1101–1110.

[55] Meng Wang, Weijie Fu, Xiangnan He, Shijie Hao, and Xindong Wu. 2020. A survey on large-scale machine learning. *IEEE Transactions on Knowledge and Data Engineering* 34 (2020), 2574–2594.

[56] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A collaborative session-based recommendation approach with parallel memory modules. In *The 42nd International ACM Conference on Research and Development in Information Retrieval*. 345–354.

[57] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, Shaozhang Niu, and Jimmy Huang. 2020. KERL: A knowledge-guided reinforcement learning model for sequential recommendation. In *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 209–218.

[58] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Defu Lian. 2021. A survey on session-based recommender systems. *ACM Computing Surveys (CSUR)* 54, 7 (2021), 1–38.

[59] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *WWW*. 878–887.

[60] 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 *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 169–178.

[61] Yufei Wen, Lei Guo, Zhumin Chen, and Jun Ma. 2018. Network embedding based recommendation method in social networks. In *WWW*. 11–12.

[62] Jibang Wu, Renqin Cai, and Hongning Wang. 2020. Déjà vu: A contextualized temporal attention mechanism for sequential recommendation. In *The 29th Web Conference*. 2199–2209.

[63] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2020. Self-supervised graph learning for recommendation. In *arXiv preprint arXiv:2010.10783*.

[64] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2021. A survey on neural recommendation: From collaborative filtering to content and context enriched recommendation. *arXiv preprint arXiv:2104.13030* (2021).

[65] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *The 14th ACM Conference on Recommender Systems*. 328–337.

[66] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James L. Sharpnack. 2019. Stochastic shared embeddings: Data-driven regularization of embedding layers. In *The 33rd Conference on Neural Information Processing Systems*. 24–34.

[67] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2021. Self-supervised hypergraph convolutional networks for session-based recommendation. In *AAAI*. 4503–4511.

[68] Xu Xie, Fei Sun, Zhaoyang Liu, Jinyang Gao, Bolin Ding, and Bin Cui. 2020. Contrastive learning for sequential recommendation. *arXiv preprint arXiv:2010.14395* (2020).

[69] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. 2020. Self-supervised reinforcement learning for recommender systems. In *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 931–940.

[70] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix X. Yu, Aditya Krishna Menon, Lichan Hong, Ed H. Chi, Steve Tjoa, Jieqi Kang, and Evan Ettinger. 2020. Self-supervised learning for deep models in recommendations. In *arXiv preprint arXiv:2007.12865*.

[71] Wenwen Ye, Shuaiqiang Wang, Xu Chen, Xuepeng Wang, Zheng Qin, and Dawei Yin. 2020. Time matters: Sequential recommendation with complex temporal information. In *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 1459–1468.

[72] Feng Yu, Yanqiao Zhu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2020. TAGNN: Target attentive graph neural networks for session-based recommendation. In *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 1921–1924.

[73] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *arXiv preprint arXiv:2101.06448*.

[74] Fajie Yuan, Xiangnan He, Karatzoglou Alexandros, and Liguang Zhang. 2020. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 1469–1478.

[75] Fajie Yuan, Xiangnan He, Haochuan Jiang, Guibing Guo, Jian Xiong, Zhezhao Xu, and Yilin Xiong. 2020. Future data helps training: Modeling future contexts for session-based recommendation. In *The 29th Web Conference*. 303–313.

[76] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. 2019. S4L: Self-supervised semi-supervised learning. In *International Conference on Computer Vision*. 1476–1485.

[77] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level deeper self-attention network for sequential recommendation. In *The 28th International Joint Conference on Artificial Intelligence*. 4320–4326.

[78] Pengyu Zhao, Tianxiao Shui, Yuanxing Zhang, Kecheng Xiao, and Kaigui Bian. 2020. Adversarial oracular seq2seq learning for sequential recommendation. In *IJCAI*. 1905–1911.

[79] Lin Zheng, Naicheng Guo, Weihao Chen, Jin Yu, and Dazhi Jiang. 2020. Sentiment-guided sequential recommendation. In *The 43rd International ACM Conference on Research and Development in Information Retrieval*. 1957–1960.

[80] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. 2021. Disentangling user interest and conformity for recommendation with causal embedding. In *Proceedings of the Web Conference 2021*. 2980–2991.

[81] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2020. Contrastive learning for debiased candidate generation in large-scale recommender systems. In *arXiv preprint arXiv:2005.12964*.

[82] Kun Zhou, Hui Wang, WayneXin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *The 29th Conference on Information and Knowledge Management*. 1893–1902.

[83] Fuzhen Zhuang, Yingmin Zhou, Fuzheng Zhang, Xiang Ao, Xing Xie, and Qing He. 2017. Cross-domain novelty seeking trait mining for sequential recommendation. In *The 26th Web Conference*. 881–882.

Received 20 July 2021; revised 28 August 2022; accepted 30 August 2022