World Wide Web
https://doi.org/10.1007/s11280-023-01163-1

Towards more effective encoders in pre-training for sequential recommendation

Ke Sun[1] · Tieyun Qian[1] · Ming Zhong[1] · Xuhui Li[2]

Received: 20 January 2023 / Revised: 3 March 2023 / Accepted: 8 March 2023

Abstract
Pre-training emerges as a new learning paradigm in natural language processing and computer vision. It has also been introduced into sequential recommendation in several seminal studies for alleviating data sparsity issue. However, existing methods adopt the bidirectional transformer as the encoder which suffers from two drawbacks. One is *insufficient intention modeling* since the transformer architecture is suitable for extracting distributed consumption intention but cannot well catch users' concentrated and occasion consumption intentions. The other is *information leakage* caused by foreseeing the future item in advance during the bidirectional encoding process. To address these problems, we propose to construct more effective encoders in pre-training for sequential recommendation. Specifically, we first decouple the original bidirectional process in transformer structure into two unidirectional processes which can avoid the information leakage problem and capture the distributed consumption intention. We then employ the locality-aware convolutional neural networks (CNNs) with narrow receptive field for concentrated consumption modeling. We also introduce a random shuffle strategy to empower CNN with the ability of modeling the occasion consumption. Experiments on five datasets demonstrate that our method improves the performance of various types of downstream sequential recommendation models to a large extent, and it also generates the overall better performance than the state-of-the-art self-supervised pre-training methods.

Keywords Sequential recommendation · Self-supervised learning · Pre-training · Encoder

✉ Tieyun Qian
qty@whu.edu.cn

Ke Sun
sunke1995@whu.edu.cn

Ming Zhong
clock@whu.edu.cn

Xuhui Li
lixuhui@whu.edu.cn

[1] School of Computer Science, Wuhan University, Wuhan, China

[2] School of Information Management, Wuhan University, Wuhan, China

1 Introduction

Recommender systems, which provide users with customized experiences in the era of big data, have played a vital role in people's daily life. When people surf on the internet, their interests are dynamic and constantly evolving over the time. This intrinsic nature draws great research attention to sequential behavior patterns and sequential recommendation. Classic methods in sequential recommendation often adopt Markov chain to model user's sequential behaviors [1, 2]. Later on, many neural models are proposed to extract the sequential patterns. Typical methods include recurrent neural networks (RNN) [3–6], attention and self attention based architectures [7–9].

The big obstacle in sequential recommendation is the sparsity issue in the user-item interaction sequence. A good number of methods have been developed to tackle this problem, mainly by incorporating context information such as brand, descriptions, category, and time [10–13]. However, all these methods require the additional context information which might be unavailable in many cases. Recently, pre-trained models (PTMs) have become a mainstream in natural language processing and computer vision since they can learn universal representations and provide a better model initialization for the target task which usually has small data [14]. Among various PTMs, BERT [15] is the most prevalent method with self-supervised learning task whose goal is to predict the masked words in a sentence given the rest words. The success of PTMs owes much to the large-scale data and diverse self-supervised learning tasks. These two characteristics are homogeneous to the nature of multi-domain learning [16, 17] and multi-task learning [18–20] in recommendation, which have already been proved to be beneficial either in academia or industry. Inspired by this, researchers start to consider pre-training technique in the recommendation field.

PTMs also shed light on sequential recommendation since the interaction sequence resembles the sentence in natural language. Several pioneering studies including BERT4Rec [21], SSI [22], and S3Rec [23] borrow the idea in BERT by randomly masking items in a sequence and then recovering these masked items. Such a pre-training scheme provides valuable self-supervised context signal [24] and has been proved to be beneficial for addressing the sparsity problem.

Despite their effectiveness, existing BERT-style pre-training sequential recommendation (PTSR) methods fail to catch users' diverse consumption intentions. Indeed, there are mainly three types of users' consumption intentions: *periodicity consumption* [25, 26], *specific intention consumption* [4, 27], and *occasion consumption* [2, 9]. (1) The periodicity consumption refers the purchase patterns in daily life such as repurchasing daily necessities periodically. For example, a user tends to buy a shampoo again after using up the old one. Note that we exclude periodicity consumption in this paper due to the property of the data used in our study, and we only explore the remaining consumption types[1]. (2) The specific intention consumption denotes that a user's sequential behaviors are towards a specific intention. Depending on whether the transactions occur in a very short time interval or are distributed in a relatively long time period, we further categorized the specific intention consumption into two sub-types including *concentrated consumption* and *distributed consumption*. For example, in Figure 1, u_1 has a distributed consumption intention in buying phone-related accessories, and u_2 has a concentrated consumption intention in rock music and thus ordersfour songs of

[1] On the Amazon and LastFM datasets used in our experiments, we do not observe sufficient periodical repeat item records in Amazon, and most repeat music records of LastFM are in a loop style. Though the periodical repeat category records can be observed in both datasets, modeling them requires extra information and is unfair to other baseline methods along this line.

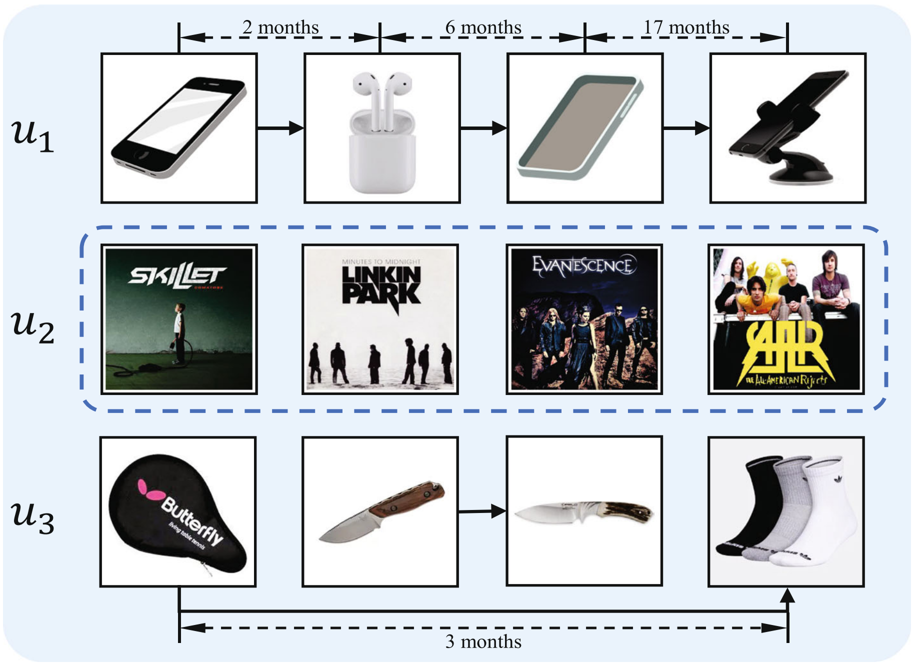


Figure 1 Examples for illustrating different consumption intentions. u_1 and u_3 are from Amazon dataset, and u_2 from LastFM dataset

different rock bands. Note that, different from the periodicity consumption in which items might also be distributed in a relatively long time period, there is no repurchase action in the distributed consumption. For instance, items bought by u_1 are actually different products. (3) The occasion consumption denotes that a user's purchase behaviors are random without any plan and cannot be observed in a close relation with their neighboring behaviors. For example, we find u_3 in Figure 1 buys the athletic socks which are not relevant to the surrounding items but have a weak connection with the faraway table tennis bag.

To have a close look at the BERT-style PTSR methods, we summarize three major steps in these methods in Figure 2. Step 1. Designing data preparation strategies for generating different data views. Step 2. Choosing a commonly used sequence data encoder (most probably a transformer). Step 3. Performing self-supervised learning tasks. The contributions of existing methods mainly lie in steps 1 and 3, and they often simply employ a bidirectional transformer as the sequence encoder in step 2. Such an encoder can well extract distributed consumption intention via the self-attention mechanism in BERT. However, the wide receptive field of the original transformer is not suitable for modeling the concentrated and occasion consumptions, since it is not focused enough and may introduce irrelevant noises. We term this *the insufficient intention modeling problem*. In addition, the simple application of the bidirectional encoder in BERT does not well fit the nature of sequential recommendation task, since we are unable to see the future item in advance in recommendation. This scenario is different from that in original cloze task in BERT for natural language understanding. We term this *the information leakage problem*.

It can be seen that both the insufficient intention modeling and information leakage problems are relevant to the encoder in step 2. In view of this, we dive into the design of sequential

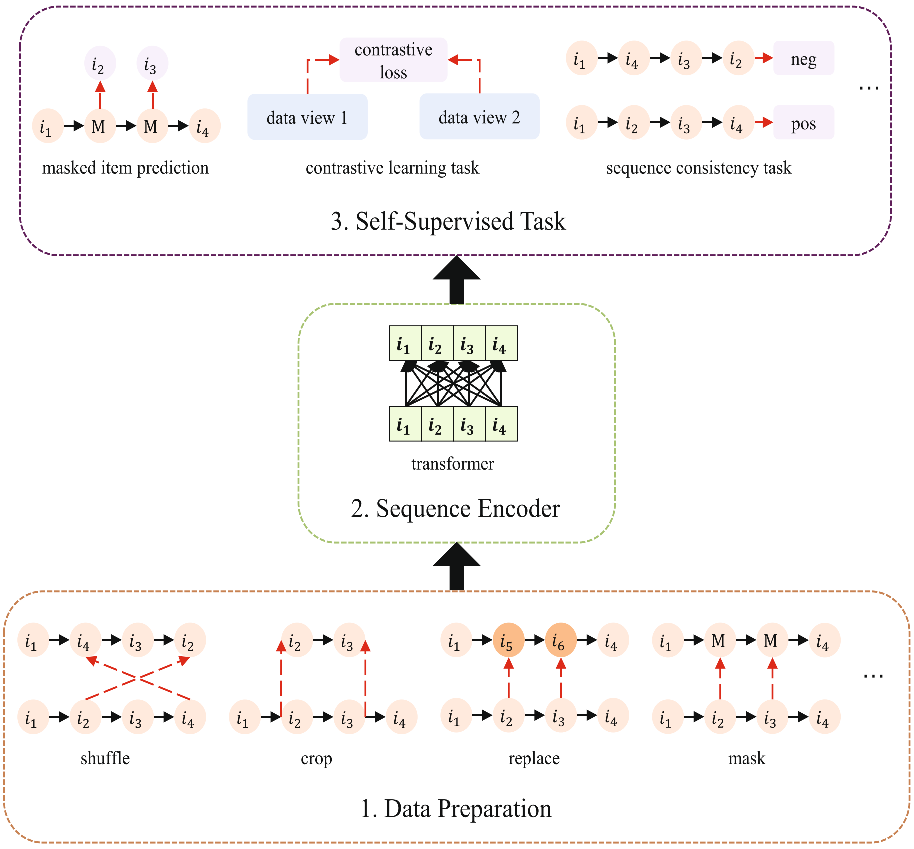


Figure 2 Three major steps of BERT-style pre-training sequential recommendation (PTSR) methods

encoder in this study and propose a self-supervised pre-training framework EEPT for sequential recommendation. More specifically, we first *employ a unidirectional transformer for addressing the information leakage problem in distributed consumption modeling* by decoupling the original bidirectional process into two separate forward and backward processes. We achieve this goal by two lower and upper triangular mask matrices, which can avoid the awareness of the future or historical records during pre-training. Such a decoupling operation is the first attempt to adapt transformer architecture to a specific pre-training task like sequential recommendation. Moreover, we *employ two convolutional neural networks (CNNs) for concentrated consumption and occasion consumption modeling*. The narrow receptive field of CNN can reduce the probability of introducing irrelevant noises, and it naturally fits the task of extracting concentrated consumption as it can perceive neighboring items. However, the narrow receptive field also prevents CNN from introducing novel items which seem to be irrelevant to the user's overall intention. This is exactly what an occasion consumption reflects. In light of this, we make a further improvement and introduce a random shuffle strategy over the input sequence, so that CNN can access faraway potential clues for occasion consumption.

We conduct extensive experiments on five real world sequential datasets, four from Amazon and one from LastFM. We obtain a set of well pre-trained item embeddings and use them to initialize the embedding layer of various types of downstream sequential models. The experimental results prove that our model can steadily improve the performance of downstream models to a large extent. It also significantly outperforms existing self-supervised pre-training methods in most cases or achieves comparable performance in the rest few cases.

2 Related work

2.1 Sequential recommendation

Sequential recommendation models seek to capture the transition patterns in users' behavior sequences which reflect users' consumption intentions. Early methods mainly rely on the Markov chain (MC) assumption, where the user's next choice always depends on several previous interactions [1, 2, 28, 29]. The MC-based methods are unable to capture complex consumption patterns, especially the occasion consumption since the neighboring records might be irrelevant.

Recently, many deep learning based models have been proposed and have achieved good performance owing to the non-linear expressive power of neural networks. In the literature, recurrent neural networks (RNNs), as well as their variants [3, 4], are first introduced in this field to characterize the temporal dependency. Later on, more methods [10–12, 30, 31] are presented to incorporate context information to address the data sparsity issue. The attention and self-attention mechanisms [32] are also the prominent architectures in deep sequential models [7–9, 33–36]. They are able to build dependencies between any items without distance limit and thus capture higher-order transitions. Recently, graph neural network has also attracted wide attention and many advanced methods have successfully applied it to various fields like traffic prediction [37, 38], text modelling [39], and cross-domain recommendation [40]. It also fits the sequential recommendation which enriches item or user representations from the global perspective [41, 42]. Other prevalent architectures include gate mechanisms [12, 43], memory networks [44–48], and CNNs [49–51].

In general, the common drawback in RNN-based methods is that they put more emphasis on the local context and lack the ability of modeling distributed and occasion consumption which require more global contexts. Meanwhile, the self-attention mechanism is weak in concentrated and occasion consumption intention extraction since it may introduce irrelevant noises. CNN is effective in concentrated consumption modeling but it is used directly for transition patterns modeling in the previous studies [49, 50]. In contrast, we introduce CNN into the pre-training stage as an additional encoder. We also present a shuffle strategy for equipping CNN with the ability of accessing faraway potential clues.

2.2 Pre-training and self-supervised learning

Pre-training is an emerging paradigm in natural language processing and computer vision. It learns valuable knowledge from huge data and benefits diverse downstream tasks which can alleviate the sparsity problem in small downstream data. Due to seldom need for labeled data, self-supervised learning tasks are most suitable for pre-training methods conducted on the huge datasets. Typical self-supervised learning tasks in natural language processing include cloze task and next sentence prediction [15]. BERT [15] is one of the most prevalent PLMs with the cloze task whose goal is to predict the masked words in a sentence.

Inspired by the merit of finding latent knowledge from unlabeled data and alleviating the sparsity problem, researchers have brought the pre-training paradigm with self-supervised learning into the sequential recommendation field [22, 23, 52–54]. Most of methods mainly focus on the design of self-supervised tasks. The widely adopted one is the masked item prediction task [21, 24]. Besides, mutual information maximization is also used to build relations among multiple views of the same input data [22, 23]. Recently, some works [53, 54] also propose to augment short sequences by generating pseudo-prior items with self-supervised learning tasks, which would help solve the cold-start issue.

Despite their achievements, existing PTSR methods fail to effectively model diverse consumption patterns. Our proposed method is distinctive in that it contains both the transformer and CNN encoders for capturing various types of consumption intentions, and it also successfully tackles the information leakage problem by decoupling the original bidirectional transformer into a unidirectional one.

3 Proposed method

In this section, we introduce our proposed self-supervised pre-training method EEPT, which stands for the **E**ffective **E**ncoders in **P**re-**T**raining for sequential recommendation.

3.1 Problem formalization

Let U and I denote the sets of users and items in the recommendation system respectively. Each user $u \in U$ has a chronologically-ordered behavior sequence $S_u = \left\{i_1, i_2, ..., i_{|S_u|}\right\}$, where $i \in I$. At the pre-training stage, we aim to learn a set of universal item representations with self-supervised tasks. At the fine-tuning stage, we use the learned representations to initialize the embedding layer of downstream sequential recommendation models. The notations used in this article are listed in Table 1.

Table 1 List of notations

Notation	Description
U, I	the set of all users, items
u, i	a user and an item
S_u	the behavior sequence of user u
$\boldsymbol{E}$	the embedding matrix of items
$\boldsymbol{S}$	the sequence representation of S
$[MASK]$, $[INI]$, $[CLS]$, $[SEP]$	the special indicator tokens in EEPT
$\boldsymbol{M^l}, \boldsymbol{M^u}$	the mask matrices w.r.t forward and backward processes
$\boldsymbol{S^l}_f, \boldsymbol{S^l}_b$	the forward and backward outputs at the lth layer in the distributed consumption module
$\boldsymbol{C}, \boldsymbol{C}'$	the outputs of concentrated consumption module and occasion consumption module
$\boldsymbol{F^k}$	the kth filter in CNN
$\boldsymbol{W}, \boldsymbol{b}$	the model parameters to be trained

3.2 The overall pre-training framework

Different from previous PTSR approaches, our proposed EEPT method focuses on modeling diverse consumption intentions of sequential data in the encoders. Figure 3 shows the overall framework of our EEPT.

Compared with existing PTSR methods, as shown in Figure 2, our proposed EEPT has the similar three-step backbone but pays more attention to the sequence encoder. Specifically, the data preparation step generates sequence data samples, w.r.t. different learning tasks. The input sequence samples are encoded by the middle sequential encoder step. Diverse consumption patterns are well explored in this step. The outputs from sequence encoder are used for the final self-supervised task learning step.

3.3 Data preparation

In our framework, three types of sequence samples are required, w.r.t different self-supervised learning tasks.

Masked sequence sample It is for the masked item prediction task. Given a behavior sequence S, we randomly replace items with a mask token $[MASK]$ with a probability p. For example, the original sequence $\{i_1, i_2, i_3, i_4\}$ is probably changed to $\{i_1, i_2, [MASK], i_4\}$.

Shuffled sequence sample This type of sample is utilized for occasion consumption. We randomly shuffle the original order of a masked sequence, excluding the masked item. Considering the masked sequence example $\{i_1, i_2, [MASK], i_4\}$, an instance of the second type could probably be $\{i_2, i_4, [MASK], i_1\}$.

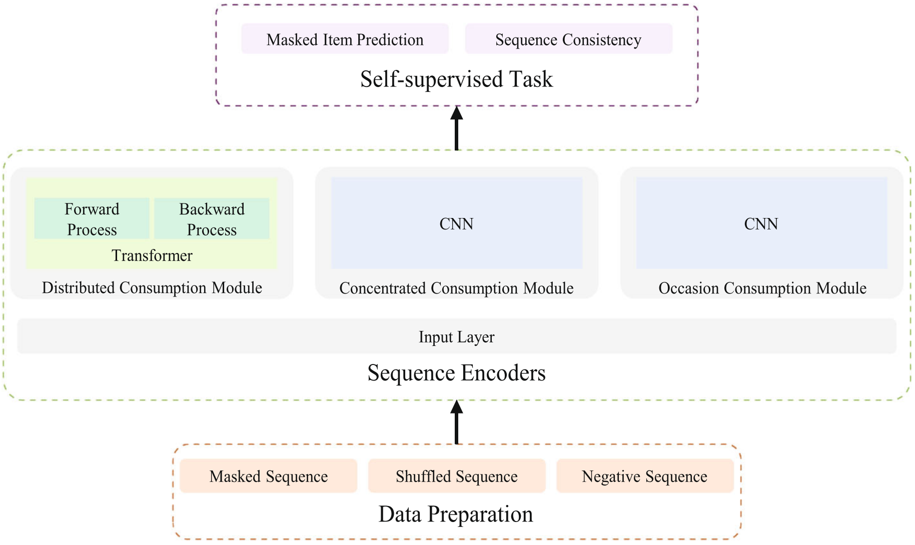


Figure 3 The overall framework of our proposed EEPT model

Negative sequence sample It is for the sequence consistency task, and treated as the negative sample against the masked sequence sample. We keep the relative order of items visited in a single day unchanged, and shuffle the order of different days. A specific example corresponding to $\{i_1, i_2, [MASK], i_4\}$ could be $\{[MASK], i_4, i_1, i_2\}$, assuming that sub-sequences $\{i_1, i_2\}$ and $\{i_3, i_4\}$ occur in two different days.

After obtaining various types of samples, we further insert indicator tokens $[INI]$ and $[CLS]$ at the beginning position of a sequence, and $[SEP]$ at the end place. $[INI]$ is for indicating the sequence consistency. $[CLS]$ and $[SEP]$ denote the beginning and end of a sequence [15]. Taking $\{i_1, i_2, [MASK], i_4\}$ as an example for illustration, the final output sequence sample is $\{[INI], [CLS], i_1, i_2, [MASK], i_4, [SEP]\}$. For the sake of simplicity, we use S to denote the generated sequence data sample in the rest of paper.

3.4 Sequence encoders

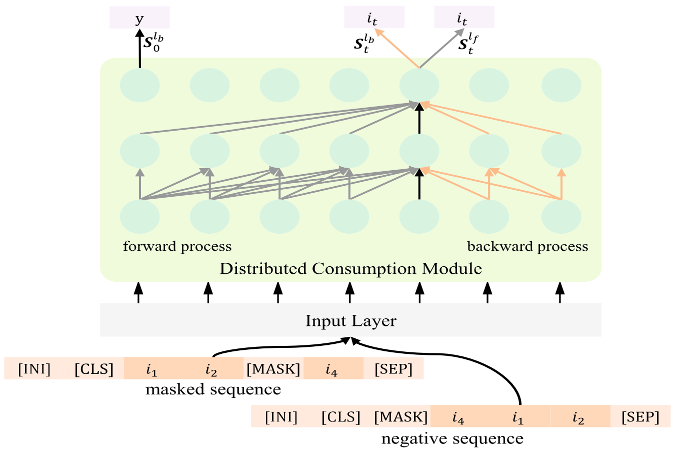
3.4.1 Input layer

At this stage, we project each sequence sample into a low dimensional dense latent matrix [22, 23]. Given a sequence sample S, the corresponding sequence representation $\boldsymbol{S} \in \mathbb{R}^{|S| \times d}$ is generated by looking-up the embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|I'| \times d}$, where d denotes the vector dimension size, and $| I' |$ equals to the size of item set I plus four special indicator tokens. Mention that the pre-trained $\boldsymbol{E}$ will be further used to initialize the embedding layer in downstream models.

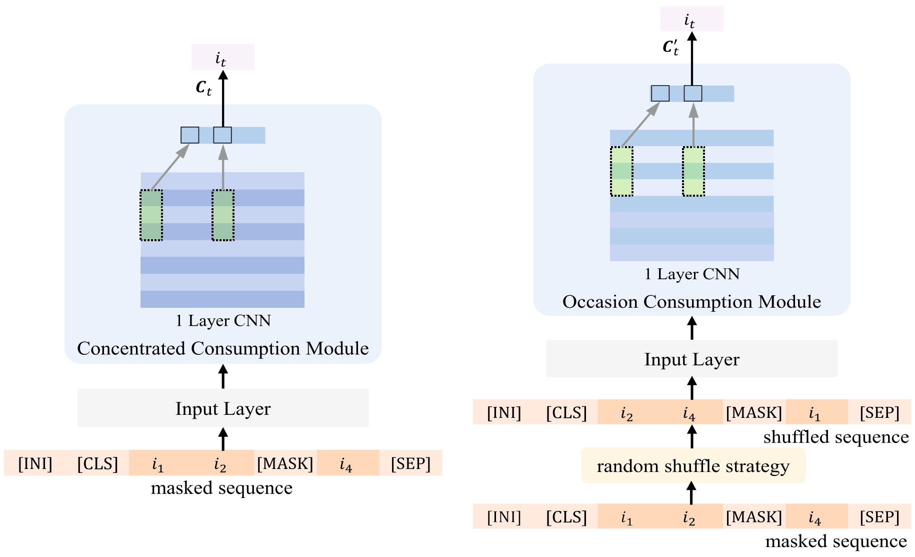
3.4.2 Distributed consumption module

The distributed consumption belongs to the specific intention consumption, where several transactions towards a specific intention are scattered in the sequence. One key property of the distributed consumption is the quite long time interval of transactions. For example, it takes about eight months for u_1 to buy the phone case after the cell phone in Figure 1. To cope with this case, existing PTSR methods rely on the transformer with wide receptive field. We follow their solutions and also introduce transformer to extract the distributed consumption as shown in Figure 4a, which takes as input the masked sequence sample and the negative sequence sample. However, the bidirectional nature of the original transformer is not suitable for our task. The direct application of bidirectional transformer may lead to the information leakage, since we are unable to foresee the future in advance in recommendation. To address this problem, we decouple the bidirectional process in the original transformer into two separate forward and backward unidirectional processes. The former helps learn the forward transition patterns e.g. the phone to the headset. While the latter helps exploit the reverse backward transition patterns e.g. the phone car mount to the phone. It is worth noting that the reverse transition pattern is also of great value, and helps promote the understanding of real data [53].

The prevalent basic transformer architecture [32] is made up of two stacked sub-layers sequentially: *Multi-head Self-Attention Layer* and*Position-wise Feed-Forward Layer*. We



(a)


i_t
c_t
1 Layer CNN
Concentrated Consumption Module
Input Layer
[INI] [CLS] i_1 i_2 [MASK] i_4 [SEP]
masked sequence
i_t
c'_t
1 Layer CNN
Occasion Consumption Module
Input Layer
[INI] [CLS] i_2 i_4 [MASK] i_1 [SEP]
shuffled sequence
random shuffle strategy
[INI] [CLS] i_1 i_2 [MASK] i_4 [SEP]
masked sequence

(b) (c)

Figure 4 The details of sequence encoder in EEPT. There are three modules w.r.t. different consumption patterns: (a) Distributed consumption module, (b) Concentrated consumption module, (c) Occasion consumption module

first introduce the basic attention mechanism of a scaled dot-product operation in the multi-head self-attention layer.

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V} \tag{1}$$

where $\boldsymbol{Q} \in \mathbb{R}^{L_q \times d}$, $\boldsymbol{K} \in \mathbb{R}^{L_k \times d}$, and $\boldsymbol{V} \in \mathbb{R}^{L_v \times d}$ denote the keys, queries, and values, respectively. In our case, they are projected from the same sequential representation matrix $\boldsymbol{S}$ with different trainable matrices, and $L_q = L_k = L_v =\mid S \mid$. This means that the vectors at the same row in $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are corresponding to the same item. However, each query in $\boldsymbol{Q}$ is able to reach all keys in $\boldsymbol{K}$. In other words, the output is a weighted sum of values from both the history and the future, which brings about the *information leakage problem*.

Then the input is projected into different spaces, leading to a multi-head self-attention:

$$\mathrm{MH}(\boldsymbol{S}^l) = [head_1; head_2; ...; head_h]\boldsymbol{W}^{\boldsymbol{O}}, \tag{2}$$

$$head_i = \mathrm{Attention}(\boldsymbol{S}^l \boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{K}}, \boldsymbol{S}^l \boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{Q}}, \boldsymbol{S}^l \boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{V}}), \tag{3}$$

where $\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{K}} \in \mathbb{R}^{d \times \frac{d}{h}}$, $\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{Q}} \in \mathbb{R}^{d \times \frac{d}{h}}$, $\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{V}} \in \mathbb{R}^{d \times \frac{d}{h}}$, and $\boldsymbol{W}^{\boldsymbol{O}} \in \mathbb{R}^{d \times d}$ are trainable matrices. h is the number of heads. $\boldsymbol{S}^l$ is the input to the $(l+1)$-th transformer layer, and it is equal to the original sequential representation $\boldsymbol{S}$ when $l = 0$. In a word, the multi-head self-attention layer generates the output via a linear operation by the weighted sum of vectors. After that, a position-wise feed-forward layer is followed with a non-linear operation, which is defined as:

$$\boldsymbol{S}^{l+1} = \mathrm{LayerNorm}(\boldsymbol{H}^l + \mathrm{FFN}(\boldsymbol{H}^l)), \tag{4}$$

$$\mathrm{FFN}(\boldsymbol{H}^l) = \mathrm{GLEU}(\boldsymbol{H}^l \boldsymbol{W}_1^F + \boldsymbol{b}_1^F)\boldsymbol{W}_2^F + \boldsymbol{b}_2^F, \tag{5}$$

$$\boldsymbol{H}^l = \mathrm{LayerNorm}(\boldsymbol{S}^l + \mathrm{MH}(\boldsymbol{S}^l)), \tag{6}$$

where $\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{F}} \in \mathbb{R}^{d \times d}$, $\boldsymbol{b}_{\boldsymbol{i}}^{\boldsymbol{F}} \in \mathbb{R}^{d \times 1}$ are parameters trained during learning process. GLEU$(\cdot)$ is the Gaussian Error Linear Unit activation function, and LayerNorm$(\cdot)$ is a classic normalization operation defined in [55] for accelerating and stabilizing the training process:

$$\mathrm{LayerNorm}(\boldsymbol{x}) = \alpha \odot \frac{\boldsymbol{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{7}$$

where $\odot$ represents the element-wise product operation, μ and σ are the mean and variance of $\boldsymbol{x}$, α and β are learned scaling factors and bias terms. ϵ is set to 1e-6 for preventing the denominator from being 0.

Until now, the original bidirectional transformer well models distributed consumption from the large-scale sequence context. However, the straight utilization of original bidirectional transformer is not feasible in recommendation task, since it may cause the information leakage problem. Thus we propose to decouple the original bidirectional process into two independent unidirectional forward and backward processes.

In detail, we achieve this goal by introducing two mask matrices $\boldsymbol{M_l}, \boldsymbol{M_u} \in \mathbb{R}^{|S| \times |S|}$, w.r.t forward and backward processes, which resemble the lower and upper triangular matrices:

$$\boldsymbol{M_l} = \begin{bmatrix} -inf & -inf & \ldots & -inf & -inf \\ -inf & 1 & \ldots & -inf & -inf \\ \ldots & \ldots & \ldots\ldots & \ldots \\ -inf & 1 & \ldots & 1 & -inf \\ -inf & 1 & \ldots & 1 & 1 \end{bmatrix}, \tag{8}$$

$$\boldsymbol{M_u} = \begin{bmatrix} -inf & 1 & \ldots 1 & 1 \\ -inf & 1 & \ldots 1 & 1 \\ \ldots & \ldots & \ldots\ldots & \ldots \\ -inf & -inf & \ldots 1 & 1 \\ -inf & -inf & \ldots -inf & 1 \end{bmatrix}, \tag{9}$$

where $-inf$ denotes negative infinity. The intuition behind $\boldsymbol{M_l}$ or $\boldsymbol{M_u}$ is that the output at a position t only perceives input items before or after the position t in a sequence, except the first indicator $[INI]$. Specifically, 1 in $\boldsymbol{M_l}$ indicates preserving the corresponding weight while $-inf$ indicates neglecting the weight. For instance, the t-th row in $\boldsymbol{M_l}$ means the perception of positions from 1 to t, excluding $[INI]$ at the first position. In our method, $[INI]$ is aware of the whole input sequence and will be utilized for maintaining the sequence consistency. Therefore, we set the first column of the mask matrix to $-inf$ and erase the weight of $[INI]$ when calculating the outputs at other positions in the same sequence. This would prevent the outputs at other positions from the awareness of the whole sequence. With $\boldsymbol{M_l}$ and $\boldsymbol{M_u}$, we transform the basic attention mechanism in Eq. 1 into following forward and backward attention mechanisms:

$$\text{Attention}_f(\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{QK}^T \odot \boldsymbol{M_l}}{\sqrt{d}})\boldsymbol{V}, \tag{10}$$

$$\text{Attention}_b(\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{QK}^T \odot \boldsymbol{M_u}}{\sqrt{d}})\boldsymbol{V} \tag{11}$$

By replacing the original attention mechanism in Eq. 3 with Attention$_f(\cdot)$ and Attention$_b(\cdot)$, we can obtain two unidirectional sequence representations $\boldsymbol{S}^{l_f}$ and $\boldsymbol{S}^{l_b}$ through Eq. 4 at layer l. These generated unidirectional vectors will finally be utilized for the masked item prediction task, and the information leakage problem can also be prevented.

Overall, we develop a unidirectional transformer based module for distributed consumption modeling, and we argue it has three main advantages. (1) It is transformer-based and owns the ability to model items in a fusion way without distance limit, so that the large-scale context in distributed consumption is considered. (2) It outputs the final representation in a unidirectional generative way, which is better suitable to sequential item recommendation task, because the item hidden representation vector from any position in $\boldsymbol{S}^{l_f}$ or $\boldsymbol{S}^{l_b}$ only receives the information from its history or future and thus avoids the information leakage problem. (3) The output is position-aware. Swapping the positions of any input two items will lead to the fluctuation of output. It is the reason why we do not add the additional position vector in the input layer, which is different from previous approaches [23, 52].

3.4.3 Concentrated consumption module

In concentrated consumption, items purchased in a short time period are towards a specific intention. Different from distributed consumption, people order the products in a quite short time span, which is similar to a basket purchase. An example is u_2 in Figure 1, who intends to enjoy rock music, and orders four songs in a single day. In the concentrated consumption circumstance, it is rational to infer the user's latent intention from her local context. An intuitive choice for concentrated consumption modeling is the convolutional neural network, which can well capture the local relations.

As shown in Figure 4b, we employ a 2D convolutional layer, which consists of filters with the fixed kernel size 3×1, over the masked sequence representation $\boldsymbol{S} \in \mathbb{R}^{|S| \times d}$. In our method, we fix the filter kernel size to 3×1 for two reasons. Firstly, for a target item in a transaction sequence, a filter with the kernel size 3×1 enables CNN layer to perceive that item's at least one nearest neighbor item in both the left and right sides. Secondly, a relatively smaller kernel size, i.e., 3×1 in our method, can largely prevent CNN layers from perceiving irrelevant items. Specifically, given n filters $\boldsymbol{F}^k \in \mathbb{R}^{3 \times 1}$, $1 \leq k \leq n$, we can generate a processed sequence representation $\boldsymbol{C} \in \mathbb{R}^{|S| \times d}$ through the following equations:

$$\boldsymbol{C}^k_{t,j} = \phi(\boldsymbol{S}_{t-1:t+1,j} \odot \boldsymbol{F}^k), \tag{12}$$

$$\boldsymbol{C}_{t,j} = \max(\boldsymbol{C}^1_{t,j}, \boldsymbol{C}^2_{t,j}, ..., \boldsymbol{C}^n_{t,j}), \tag{13}$$

where n is the number of filters, $\odot$ denotes the inner product operator, and ϕ denotes the ReLU activation function. Obviously, the output at a position t is only aware of its neighbor inputs at $t-1$ and $t+1$, thus it well fuses the local context information for concentrated consumption extraction.

3.4.4 Occasion consumption module

In reality, people might buy a product randomly without any plan, which is irrelevant to the nearby transactions. This phenomenon occasionally occur in our daily life for complex reasons. Some are due to the periodicity of consumables and some due to the curiosity of people to try new products. Fortunately, users still prefer to choose items consistent with their interests even if these items are irrelevant to recent purchase records. For instance, the athletic socks bought by u_3 in Figure 1 have a weak connection with the faraway table tennis bag. It indicates that u_3 is a sports fan, and it is reasonable to build a bridge between athletic socks and table tennis bag, ignoring the middle knives.

To achieve this goal, we utilize convolutional neural network again instead of transformer, as shown in Figure 4c which takes as input the masked sequence sample. This is because CNN has a small receptive field and it has low probability to absorb noise compared with a transformer encoder. However, the local receptive field prevents CNN from accessing distant clues in a long sequence. To tackle this issue, we propose a random shuffle strategy over the input sequence, so that distant items can be reached by CNN indirectly. Given a masked sequence S where the t-th item i_t has been replaced with $[MASK]$, we keep the position of $[MASK]$ unchanged and randomly shuffle the positions of other items, resulting a new shuffled sequence S' where the t-th position is still $[MASK]$. Through this way, a distant relevant item can probably be moved to the target item i_t's neighborhood and be modeled by CNN layers. By looking-up the embedding matrix, we first turn the shuffled sequence S' into a representation $\boldsymbol{S}' \in \mathbb{R}^{|S| \times d}$. Then a CNN layer, which consists of filters with the fixed kernel size 3×1, is applied to $\boldsymbol{S}'$. Here the filter kernel size is also fixed to 3×1 for similar reasons in the concentrated consumption module. On one hand, we encourage CNN layers to perceive the target item's at least two neighbors in the shuffled input sequence, so that the distant relevant item can be reached if it is the target item's neighbor in the shuffled sequence. On the other hand, the relatively smaller kernel size can also prevent CNN layers from bringing noises in the shuffled sequence. Formally, given n filters $\boldsymbol{F}^k \in \mathbb{R}^{3 \times 1}$, $1 \leq k \leq n$, the occasion consumption hidden matrix $\boldsymbol{C}'$ is calculated as follows:

$$\boldsymbol{C}'^k_{t,j} = \phi(\boldsymbol{S}'_{t-1:t+1,j} \odot \boldsymbol{F}^k), \tag{14}$$

$$\boldsymbol{C}'_{t,j} = \max(\boldsymbol{C}'^{1}_{t,j}, \boldsymbol{C}'^{2}_{t,j}, ..., \boldsymbol{C}'^{n}_{t,j}), \tag{15}$$

Both $\boldsymbol{C}'$ from the occasion consumption module and $\boldsymbol{C}$ from the concentrated consumption module are utilized for the masked item prediction w.r.t different consumption patterns.

3.5 Self-supervised learning

In EEPT, we consider two self-supervised tasks at the pre-training stage. One is the cloze task for the masked item prediction, and the other is the sequence consistency task which discriminates the positive sequence samples from negative ones.

For the cloze task, supposing the item i_t in S is masked, we use four output hidden vectors $\boldsymbol{S}_t^{l_f}$, $\boldsymbol{S}_t^{l_b}$, $\boldsymbol{C}_t$, and $\boldsymbol{C}'_t$ for the masked item prediction. Taking $\boldsymbol{S}_t^{l_f}$ as an example, the output distribution over all items is calculated as:

$$P(i \mid \boldsymbol{S}_t^{l_f}) = \text{softmax}(\boldsymbol{S}_t^{l_f} \boldsymbol{W}^{\boldsymbol{p}} + \boldsymbol{b}^{\boldsymbol{p}}), \tag{16}$$

where $\boldsymbol{W}^{\boldsymbol{p}}$ and $\boldsymbol{b}^{\boldsymbol{p}}$ are trainable parameters. Then we define the corresponding masked item prediction objective as:

$$\mathcal{L}(\boldsymbol{S}_t^{l_f}) = \log(P(i = i_t \mid \boldsymbol{S}_t^{l_f})) \tag{17}$$

Considering all hidden outputs from sequence encoders in the same way, the entire masked item prediction objective function can be formulated as follows:

$$\mathcal{L}_{MLM} = \lambda_1(\mathcal{L}(\boldsymbol{S}_t^{l_f}) + \mathcal{L}(\boldsymbol{S}_t^{l_b})) + \lambda_2(\mathcal{L}(\boldsymbol{C}_t) + \mathcal{L}(\boldsymbol{C}'_t)), \tag{18}$$

where λ_i is the hyper-parameter for balancing transformer and CNN based encoders.

For the second task, i.e., maintaining sequence consistency, we adopt a binary classifier over $\boldsymbol{S}_0^{l_b}$ to discriminate the positive masked sequence samples from negative ones. $\boldsymbol{S}_0^{l_b}$ is the hidden representation of indicator $[INI]$ from the distributed consumption module. It not only perceives all transactions in a sequence, but also maintains the relative sequential order. Based on $\boldsymbol{S}_0^{l_b}$, the binary classification loss objective is formulated as:

$$\mathcal{L}_{SC} = y\log(\text{MLP}(\boldsymbol{S}_0^{l_b})) + (1 - y)\log(1 - \text{MLP}(\boldsymbol{S}_0^{l_b})), \tag{19}$$

where $y = 1$ if the sequence sample S is a masked sequence sample and $y = 0$ if S is a negative sequence sample. MLP$(\cdot)$ is a multi-layer perceptron neural network binary classifier.

Eventually, we add $\mathcal{L}_{MLM}$ and $\mathcal{L}_{SC}$ together, resulting in the final loss objective:

$$\mathcal{L} = \mathcal{L}_{MLM} + \lambda_3\mathcal{L}_{SC} \tag{20}$$

We maximize $\mathcal{L}$ at the pre-training stage, and obtain a set of item embeddings $\boldsymbol{E}$ which is well pre-trained and preserves latent characteristics of real sequence data.

3.6 Inference

EEPT is purely a pre-training architecture and we only conduct two self-supervised tasks at the pre-training stage rather than the inference one. The prediction is conducted by the downstream models. That is to say, the outputs of the three components are utilized for pre-training our EEPT to get the item embedding matrix $\boldsymbol{E}$.

For the inference step, i.e., predicting the next item, it is conducted by the downstream sequential recommendation methods like SASRec [9] and TiSASRec [35] at the fine-tuning stage after we obtain the pre-trained $\boldsymbol{E}$. To be more clear, we take SASRec as the downstream model and describe how the inference step is conducted with our pre-trained method.

1. We first pre-train EEPT and obtain the item embedding matrix $\boldsymbol{E}$ during pre-training.
2. At the fine-tuning stage, we use the pre-trained $\boldsymbol{E}$ to initialize the item embedding matrix $\boldsymbol{M}$ of SASRec and fine-tune SASRec with its own next item prediction objective.
3. Finally, we conduct the inference with SASRec and calculate the probability score $r_{i,t}$ of item i being the next item given historical visited item sequence: $r_{i,t} = \boldsymbol{F}_t \boldsymbol{M}_i$, where $\boldsymbol{F}_t$ is the representation of historical sequence and $\boldsymbol{M}_i$ is the embedding of item i, according to SASRec.

4 Experiment

To justify the superiority of our proposed pre-training method, we conduct experiments on various downstream sequential recommendation models by using the learned universal item embedding matrix $\boldsymbol{E}$. We not only compare our method with *downstream sequential base models*, but also compare it with the state-of-art *self-supervised pre-training methods*.

4.1 Datasets

We evaluate EEPT on five public real-world datasets. Four of them are sub-datasets from Amazon: *Video Games* (Video), *Toys and Games* (Toys), *Cell Phones and Accessories* (Cell), and *Sports and Outdoors* (Sports). The last one is a music artist recommendation dataset *LastFM* (LastFM). Table 2 summarizes the statistics of five datasets. For each user from these datasets, the most recently visited item is considered as the test item while the second one is for validation, and all other items are for training.

4.2 Evaluation protocol

We evaluate the performance of pre-training methods on the downstream sequential recommendation models. Two conventional Top-k evaluation metrics are employed to quantitatively evaluate the model performance: *Hit Ratio* (Hit) and *Normalized Discounted Cumulative Gain* (NDCG). k is from {5, 10}. Hit ratio measures the proportion of cases thatthe desired

Table 2 Statistics of the evaluation datasets

Datasets	#users	#items	#records	sparsity
Video	24,303	10,619	230,961	99.91%
Toys	19,412	11,793	166,196	99.93%
Cell	27,878	10,329	193,228	99.93%
Sports	35,598	18,236	294,878	99.95%
LastFM	1,805	12,523	48,671	99.78%

item is among the Top-k item list. NDCG concerns about the quality of ranking list. The formulations of Hit@k and NDCG@k are:

$$\text{Hit@}k = \frac{1}{\mid U \mid} \sum_{u \in U} I(r_u \leq k) \tag{21}$$

$$\text{NDCG@}k = \frac{1}{\mid U \mid} \sum_{u \in U} \frac{I(r_u \leq k)}{\log_2(r_u + 1)} \tag{22}$$

where r_u is the ranking of a test item for user u, and $I(\cdot)$ denotes the indicator function. For the sake of efficient evaluation, we adopt the leave-one-out strategy [23, 35], and rank the ground-truth item with 99 randomly sampled negative items.

4.3 Baselines

We compare our model with following four state-of-the-art *self-supervised pre-training methods*.

S3Rec [23]: It is a pioneer self-supervised pre-training sequential recommendation method. It utilizes mutual information maximization objective to extract relations between different data views. Note that we remove the attribute information for a fair comparison.

BERT4Rec [21]: This method introduces BERT for sequential recommendation. Masked item prediction task is firstly employed in this work.

SSI [22]: This method develops three self-supervised consistency tasks for pre-training, and fine-tunes the downstream model with knowledge distillation (KD) technology.

SSI- [22]: This is a variant of **SSI** by removing the knowledge distillation technology at the fine-tuning stage. Instead, we initialize the embedding layer of downstream models with learned embedding matrix. Note that there are three learned embedding matrices in SSI-, and we adopt the average result of them for the initialization.

All the pre-training methods (including ours) are evaluated on six representative *downstream base sequential recommendation methods*, termed as base SR methods:

GRU4Rec [3]: This is a classic RNN-based method for session-based recommendation. It directly applies Gated Recurrent Unit (GRU) on sequential behavior data.

NARM [7]: This method combines both RNN and attention mechanism for modeling users' global and local preferences respectively.

SASRec [9]: It is a widely used benchmark, which merely utilizes the powerful self-attention based transformer.

TiSASRec [35]: It is a time interval aware variant of SASRec, which further improves the performance of SASRec.

DuoRec [56]: This method introduces a contrastive regularization for item embedding learning based on a model-level augmentation and a positive sampling strategy.

ContraRec [57]: This method employs contrastive learning to model not only the correlations between the historical sequence and the target item but also contrast signals hidden in user interaction sequences.

4.4 Implementation details

At the pre-training stage, we use self-supervised tasks to learn model parameters. For baseline pre-training methods, we utilize the source code of S3Rec and BERT4Rec provided by the authors, and we implement SSI and SSI- with Pytorch since their codes are not available. The embedding dimension d is set to 64 and the mask probability p is 0.15 for all pre-training methods including EEPT. Other parameters of baselines are set following the original papers. For our EEPT, we set the batch size to 128. The self-attention layers and the attention heads are set to 8. λ_1, λ_2 and λ_3 are 1.0, 0.5, and 0.5 for Amazon sub-datasets, and 1.0, 1.0, and 0.1 for LastFM. We employ the AdamW optimizer to optimize our model with learning rate 0.001. The AdamW weight decay is 0.0, 0.1, 0.1, 0.2, and 0.0 for Video, Toys, Cell, Sports, and LastFM, respectively. The number of pre-training epochs is set to 100.

At the fine-tuning stage, we incorporate pre-trained item embeddings into downstream models, except the knowledge distillation based method SSI. For downstream sequential recommendation methods except DuoRec, we use the implementations provided by Rechorus [26]. For DuoRec, we implement it with Pytorch. We report the average scores of four group results, which are generated by running pre-training models two times and then downstream models two times.

Table 3 Performance comparison with GRU4Rec as the base method

Dataset	Metric	GRU4Rec					
		Base	+S3Rec	+BERT4Rec	+SSI	+SSI-	+EEPT
Video	Hit@5	54.23	54.29	$\underline{59.55}$	58.68	59.30	**61.79***
	Hit@10	68.22	68.20	72.02	71.02	$\underline{72.10}$	**73.85***
	NDCG@5	39.83	39.92	$\underline{45.41}$	44.29	44.77	**47.48***
	NDCG@10	44.36	44.43	$\underline{49.45}$	48.48	48.91	**51.39***
Toys	Hit@5	31.82	31.90	$\underline{36.95}$	34.36	34.49	**39.35***
	Hit@10	43.96	44.07	$\underline{48.39}$	46.21	46.55	**50.86***
	NDCG@5	22.18	22.12	$\underline{26.68}$	24.42	24.51	**28.44***
	NDCG@10	26.09	26.05	$\underline{30.38}$	28.24	28.40	**32.53***
Cell	Hit@5	41.31	41.89	$\underline{45.05}$	44.04	43.61	**48.72***
	Hit@10	54.41	55.42	$\underline{58.22}$	56.56	57.19	**61.28***
	NDCG@5	29.64	30.01	$\underline{32.89}$	32.46	31.56	**36.39***
	NDCG@10	33.86	34.38	$\underline{37.15}$	36.50	35.95	**40.45***
Sports	Hit@5	32.36	34.01	36.70	36.95	$\underline{37.19}$	**40.16***
	Hit@10	45.21	47.14	49.73	49.77	$\underline{50.34}$	**53.39***
	NDCG@5	22.54	23.73	26.01	$\underline{26.41}$	26.30	**28.88***
	NDCG@10	26.68	27.97	30.22	30.54	$\underline{30.60}$	**33.15***
LastFM	Hit@5	27.56	$\underline{36.09}$	32.24	33.99	33.72	**36.69**
	Hit@10	37.48	$\underline{47.14}$	43.36	45.52	44.69	**49.05***
	NDCG@5	19.51	$\underline{26.13}$	22.98	24.92	24.47	**26.52**
	NDCG@10	22.70	$\underline{29.69}$	26.58	28.64	28.00	**30.50**

Table 4 Performance comparison with NARM as the base method

Dataset	Metric	NARM					
		Base	+S3Rec	+BERT4Rec	+SSI	+SSI-	+EEPT
Video	Hit@5	54.32	55.22	59.71	58.43	59.76	**62.27***
	Hit@10	68.16	68.70	72.22	70.91	72.34	**74.29***
	NDCG@5	39.98	40.80	45.57	44.29	45.31	**47.89***
	NDCG@10	44.46	45.17	49.63	48.33	49.39	**51.79***
Toys	Hit@5	31.79	32.31	36.87	34.02	34.73	**39.36***
	Hit@10	43.95	44.66	48.63	45.80	46.92	**50.98***
	NDCG@5	22.03	22.48	26.64	24.14	24.56	**28.85***
	NDCG@10	25.95	26.47	30.44	27.93	28.49	**32.60***
Cell	Hit@5	41.22	41.66	45.59	44.06	44.17	**49.11***
	Hit@10	54.71	54.99	58.64	56.55	57.67	**61.68***
	NDCG@5	29.65	29.95	33.49	32.59	31.96	**36.78***
	NDCG@10	33.99	34.25	37.71	36.62	36.32	**40.85***
Sports	Hit@5	32.64	34.26	37.19	36.83	37.15	**40.68***
	Hit@10	45.61	47.28	50.26	49.53	50.48	**53.82***
	NDCG@5	22.75	23.96	26.45	26.26	26.30	**29.38***
	NDCG@10	26.93	28.15	30.66	30.36	30.60	**33.61***
LastFM	Hit@5	28.18	35.88	34.88	35.99	33.89	**37.59***
	Hit@10	37.92	47.58	45.86	46.57	45.66	**49.99***
	NDCG@5	20.13	26.02	25.35	26.41	24.93	**27.32***
	NDCG@10	23.26	29.80	28.88	29.82	28.72	**31.35***

4.5 Experimental results

The overall results with respect to six downstream sequential base models are respectively shown in Tables 3, 4, 5, 6, 7, and 8. In each table, we highlight the best results in boldface and underline the second best ones. $\star$ and $*$ indicate the statistical significance for $p < 0.01$ and $p < 0.05$ comparing the best results to the second best in each table, respectively. We also use † to denote the highest score in each dataset among all methods in six tables. We have the following observations. Note that **+** denotes using the corresponding pre-training method while **Base** means no pre-training.

Firstly, on all five datasets and six base methods, our proposed pre-training method can significantly and consistently promote the performance of base SR methods. For example, EEPT brings about 7.9 percentage average growth of Hit@5 over all datasets to GRU4Rec. Meanwhile, by applying our pre-training method, the performance deviations among various downstream sequential models become very small. Taking the Cell dataset as an example, with NARM, TiSASRec, and DuoRec as the base methods, EEPT produces three similar NDCG@5 scores 36.78, 37.14, and 37.27. In contrast, the base results without any pre-training are 29.65, 33.31, and 34.36, showing a big gap. These results prove that our EEPT is

Table 5 Performance comparison with SASRec as the base method

Dataset	Metric	SASRec					
		Base	+S3Rec	+BERT4Rec	+SSI	+SSI-	+EEPT
Video	Hit@5	59.34	56.96	60.75	61.45	60.38	**62.30***
	Hit@10	71.14	69.65	72.48	73.47	72.73	**74.09***
	NDCG@5	45.61	42.94	47.03	47.19	46.13	**47.82***
	NDCG@10	49.44	47.05	50.83	51.09	50.14	**51.65***
Toys	Hit@5	36.10	34.99	37.92	40.18	39.68	**40.25**
	Hit@10	45.53	44.68	48.49	51.18	50.50	**51.45**
	NDCG@5	27.58	25.92	28.62	29.90	29.38	**30.01**
	NDCG@10	30.62	29.04	32.03	33.45	32.88	**33.63**
Cell	Hit@5	44.47	42.14	46.27	48.80	48.18	**49.05***
	Hit@10	56.02	54.24	58.38	60.88	60.53	**61.05**
	NDCG@5	33.58	30.99	34.68	36.76	36.10	**37.17***
	NDCG@10	37.31	34.89	38.59	40.67	40.10	**41.05***
Sports	Hit@5	34.10	42.64	38.37	**40.45**	37.97	40.38
	Hit@10	45.64	46.07	50.56	**52.98**	51.21	52.97
	NDCG@5	24.77	31.54	27.75	**29.34**	27.12	29.33
	NDCG@10	28.50	35.42	31.68	**33.39**	31.40	**33.39**
LastFM	Hit@5	32.68	32.36	35.34	**40.98***†	38.62	38.18
	Hit@10	43.88	42.20	46.10	**51.40***	49.73	49.57
	NDCG@5	23.91	23.89	26.07	**30.07***†	28.61	28.63
	NDCG@10	27.53	27.05	29.54	**33.43***	32.21	32.31

not sensitive to the various neural architectures of the base SR methods. They also demonstrate the generalization of the pre-trained item representations generated by our EEPT.

Secondly, compared with four self-supervised pre-training baselines, EEPT shows significant performance improvements in most cases and achieves comparable performance in the rest few cases. This proves the superiority of our well designed sequence encoders, which capture the latent consumption patterns in data. We also note that SSI achieves the best results on LastFM dataset using SASRec, TiSASRec, and ContraRec as the base SR methods. The reason might be that LastFM has a larger average sequence length than other datasets, which makes the teacher model in SSI provide more crucial sequential knowledge at the fine-tuning stage. However, SSI requires three extra redundant pre-trained teacher models and the complicated KD technique to help the base model learn from the teachers. This also explains that SSI outputs the largest scores for Sports dataset in some cases. In contrast, our EEPT, with a single pre-trained model, still achieves competitive results at a low cost of only using the pre-trained item embedding $\boldsymbol{E}$. This further shows the strong point of EEPT.

Thirdly, among the pre-training baselines, SSI, SSI-, and BERT4Rec are competitive pre-training methods, since they inherit the transformer's merit of modeling distributed consumption. We find that SSI performs better than SSI- with SASRec, TiSASRec, and

Table 6 Performance comparison with TiSASRec as the base method

Dataset	Metric	TiSASRec					
		Base	+S3Rec	+BERT4Rec	+SSI	+SSI-	+EEPT
Video	Hit@5	59.72	57.41	60.91	61.58	60.95	**62.93***
	Hit@10	71.45	70.16	72.64	73.71	73.38	**74.76***
	NDCG@5	45.95	43.31	47.14	47.29	46.62	**48.44***
	NDCG@10	49.75	47.44	50.95	51.23	50.65	**52.28***
Toys	Hit@5	36.37	34.93	37.94	40.09	39.63	**40.36**
	Hit@10	45.74	44.53	48.55	51.18	50.63	**51.60**
	NDCG@5	27.69	25.75	28.58	29.85	29.35	**30.08**
	NDCG@10	30.71	28.85	32.00	33.43	32.91	**33.71**
Cell	Hit@5	44.24	42.64	46.32	48.73	48.16	**48.97**
	Hit@10	55.87	54.66	58.42	**61.06**	60.48	61.04
	NDCG@5	33.31	31.54	34.72	36.71	36.03	**37.14***
	NDCG@10	37.07	35.42	38.63	40.69	40.02	**41.04***
Sports	Hit@5	34.60	33.82	38.40	40.33	38.07	**40.38**
	Hit@10	46.21	46.28	50.57	52.83	51.16	**53.10**
	NDCG@5	25.07	23.86	27.75	29.28	27.20	**29.39**
	NDCG@10	28.81	27.87	31.67	33.31	31.42	**33.49**
LastFM	Hit@5	31.96	31.63	35.18	**40.58***	38.17	38.66
	Hit@10	43.09	41.47	45.93	**51.94**†*	49.25	49.54
	NDCG@5	23.46	23.13	25.82	**30.01***	28.17	28.74
	NDCG@10	27.04	26.29	29.32	**33.70**†*	31.75	32.26

ContraRec as the base methods, but worse with GRU4Rec, NARM, and DuoRec. This could be explained from two issues. On one hand, GRU4Rec and NARM are based on RNN, which is incompatible to the attention mechanism, thus hurting the performance of SSI. On the other hand, DuoRec includes negative samples to the whole item set, making the knowledge distillation from SSI inefficient. S3Rec damages the base performance in many cases because the mutual information maximization objective in S3Rec leads the model into an overfitting state, especially when tackling sparse datasets after removing the attribute information.

4.6 In-depth analysis

We provide a detailed analysis on whether EEPT works well.

4.6.1 Ablation study

We conduct ablation experiments with the representative downstream base SR method TiSASRec. There are three types of ablations in total.

Table 7 Performance comparison with DuoRec as the base method

Dataset	Metric	DuoRec					
		Base	+S3Rec	+BERT4Rec	+SSI	+SSI-	+EEPT
Video	Hit@5	60.48	61.30	61.99	61.01	62.35	**63.87**†*
	Hit@10	71.54	72.60	73.20	71.85	73.49	**74.98**†*
	NDCG@5	47.44	47.90	48.64	47.79	49.02	**50.42**†*
	NDCG@10	51.03	51.57	52.27	51.30	52.62	**54.01**†*
Toys	Hit@5	36.10	36.83	39.37	37.07	39.29	**41.35**†*
	Hit@10	45.35	46.00	49.39	46.72	49.45	**51.68**†*
	NDCG@5	28.20	28.70	30.51	28.85	30.29	**32.00**†*
	NDCG@10	31.18	31.65	33.74	31.96	33.57	**35.35**†*
Cell	Hit@5	44.92	45.02	47.04	45.18	47.06	**48.87***
	Hit@10	56.12	56.55	58.56	56.50	58.66	**60.38***
	NDCG@5	34.36	34.26	35.98	34.49	36.06	**37.27***
	NDCG@10	37.97	37.98	39.71	38.15	39.81	**40.98***
Sports	Hit@5	35.66	35.43	38.52	36.24	**40.69**	40.67
	Hit@10	46.85	47.40	50.23	47.60	**52.91***	52.62
	NDCG@5	26.17	26.00	28.49	26.70	29.99	**30.40***
	NDCG@10	29.78	29.85	32.26	30.37	33.94	**34.26***
LastFM	Hit@5	34.20	30.79	39.65	32.83	**40.77**	40.38
	Hit@10	44.62	41.55	50.49	42.92	**51.06**	50.65
	NDCG@5	25.45	22.44	29.30	24.55	**30.23**	29.68
	NDCG@10	28.83	25.90	32.78	27.78	**33.57**	32.96

(a) Removing a module or component from EEPT:

- **-Distributed Consumption Module** denotes the removal of distributed consumption module. Note the sequence consistency task is still maintained.
- **-Concentrated Consumption Module** denotes the removal of concentrated consumption module.
- **-Occasion Consumption Module** denotes the removal of occasion consumption module.
- **-CNN-based Modules** denotes the removal of both CNN-based concentrated consumption module and occasion consumption module.

(b) Removing one unidirectional process from the distributed consumption module:

- **-Forward Process** denotes the removal of the forward process.
- **-Backward Process** denotes the removal of the backward process.

(c) Removing the sequence consistency task:

- **-Sequence Consistency** denotes the removal of the sequence consistency task.

Table 8 Performance comparison with ContraRec as the base method

Dataset	Metric	ContraRec					
		Base	+S3Rec	+BERT4Rec	+SSI	+SSI-	+EEPT
Video	Hit@5	61.40	57.52	57.29	61.36	59.73	**62.83***
	Hit@10	72.98	69.93	69.57	73.24	71.92	**74.61***
	NDCG@5	47.52	43.45	43.42	47.31	45.54	**48.45***
	NDCG@10	51.28	47.48	47.40	51.16	49.49	**52.27***
Toys	Hit@5	38.37	34.70	36.31	39.63	38.29	**40.04***
	Hit@10	47.68	44.65	47.29	50.44	49.50	**51.41***
	NDCG@5	29.13	25.57	26.67	**29.67**	28.09	29.64
	NDCG@10	32.13	28.79	30.20	33.17	31.72	**33.32**
Cell	Hit@5	46.34	42.03	43.90	48.16	47.43	**49.74**†*
	Hit@10	58.04	53.30	56.33	60.06	59.64	**62.01**†*
	NDCG@5	35.34	30.89	32.51	36.39	35.65	**37.47**†*
	NDCG@10	39.13	34.53	36.54	40.24	39.59	**41.44**†*
Sports	Hit@5	38.18	34.25	36.24	40.62	37.74	**41.20**†*
	Hit@10	49.96	46.49	48.24	53.04	50.74	**53.75**†*
	NDCG@5	27.84	24.36	25.94	29.71	26.89	**30.14**†*
	NDCG@10	31.65	28.31	29.80	33.72	31.08	**34.19**†*
LastFM	Hit@5	35.42	33.45	35.49	**40.48***	37.87	38.73
	Hit@10	46.25	42.30	46.93	**51.50***	47.68	50.09
	NDCG@5	26.18	24.81	26.29	**29.88***	28.33	28.93
	NDCG@10	29.65	27.66	29.99	**33.45***	31.49	32.60

The results on five different datasets are shown in Tables 9, 10, 11, 12, and 13, respectively. Overall, EEPT experiences an obvious performance drop on all datasets without the distributed consumption module. This shows the transformer's powerful capability of large-

Table 9 Ablation study on Video dataset with TiSASRec as the base method

Method	Video			
	Hit		NDCG	
	k=5	k=10	k=5	k=10
-Distributed Consumption	57.43	69.27	44.22	48.05
-Concentrated Consumption	62.83	74.70	48.30	52.16
-Occasion Consumption	62.48	74.36	47.97	51.82
-CNN-based Modules	62.10	74.09	47.88	51.76
-Forward Process	62.20	74.16	48.07	51.95
-Backward Process	61.84	73.71	47.61	51.45
-Sequence Consistency	62.78	74.43	48.34	52.12
EEPT	**62.93**	**74.76**	**48.44**	**52.28**

Table 10 Ablation study on Toys dataset with TiSASRec as the base method

Method	Toys			
	Hit		NDCG	
	k=5	k=10	k=5	k=10
-Distributed Consumption Module	35.54	45.03	27.42	30.48
-Concentrated Consumption Module	40.24	51.44	29.98	33.60
-Occasion Consumption Module	<u>40.49</u>	51.48	29.96	33.51
-CNN-based Modules Module	40.05	50.69	30.01	33.46
-Forward Process	37.97	48.02	28.85	32.09
-Backward Process	38.90	49.37	29.10	32.49
-Sequence Consistency	**40.86**	**51.74**	**30.51**	**34.03**
EEPT	40.36	<u>51.60</u>	<u>30.08</u>	<u>33.71</u>

scale dependency modeling. In addition, CNN-based concentrated and occasion consumption modules also contribute to EEPT in most cases because CNN is able to model complex latent consumption patterns. An interesting phenomenon is that -Occasion Consumption is slightly worse than -Concentrated Consumption. This infers that people are not always influenced by the nearby purchases in a sequence, but also the transactions faraway.

The sequence consistency task does not constantly work well on five datasets. One reason might be that it is difficult to judge whether or not a long sequence is in its original order using a single classifier.

The removal of any forward or backward process results in inferior performance. As a building block in distributed consumption module, each unidirectional process plays a crucial role in EEPT. They captures the valuable forward and backward transition patterns w.r.t history and future contexts.

The performance degrades more with the removal of the backward process than with the removal of the forward process in most cases. This is because that the fine-tuning of all downstream sequential recommendation methods follows the same forward process at the pre-training stage. That is to say, the downstream models are still able to learn the knowledge of the forward process even if the forward process is removed from the pre-training procedure. On the contrary, the backward process only exists in the pre-training stage. Thus the removal

Table 11 Ablation study on Cell dataset with TiSASRec as the base method

Method	Cell			
	Hit		NDCG	
	k=5	k=10	k=5	k=10
-Distributed Consumption Module	43.95	55.26	33.35	37.01
-Concentrated Consumption Module	<u>48.86</u>	**61.04**	36.91	40.84
-Occasion Consumption Module	48.76	<u>61.03</u>	36.79	40.76
-CNN-based Modules	48.56	60.58	36.78	40.66
-Forward Process	46.79	58.29	35.65	39.36
-Backward Process	46.27	57.78	35.22	38.95
-Sequence Consistency	48.83	61.00	<u>36.99</u>	<u>40.93</u>
EEPT	**48.97**	**61.04**	**37.14**	**41.04**

Table 12 Ablation study on Sports dataset with TiSASRec as the base method

Method	Sports			
	Hit		NDCG	
	k=5	k=10	k=5	k=10
-Distributed Consumption Module	38.01	50.33	27.79	31.77
-Concentrated Consumption Module	**40.78**	**53.58**	**29.69**	**33.82**
-Occasion Consumption Module	40.77	53.50	29.65	33.76
-CNN-based Modules	40.69	53.53	29.65	33.80
-Forward Process	39.37	51.71	28.61	32.60
-Backward Process	38.97	51.48	28.55	32.58
-Sequence Consistency	40.76	53.47	29.54	33.64
EEPT	40.38	53.10	29.39	33.49

of the backward pass may lead to a larger performance decrease. This phenomenon actually also proves the importance of our proposed backward process at the pre-training stage from another point of view.

4.6.2 Combination strategy study

EEPT uses four output hidden vectors from three separate modules for the masked item prediction independently during the pre-training phase. Therefore, we investigate whether or not it achieves better results by combining the outputs of these modules together. Specifically, three strategies are considered to combine the four output vectors, i.e., $\boldsymbol{S}_t^{l_f}$, $\boldsymbol{S}_t^{l_b}$, $\boldsymbol{C}_t$, and $\boldsymbol{C}'_t$:

- **Mean-Pooling**: It calculates the average vector of these four vectors.
- **Max-Pooling**: It selects the maximum value for each index from these four vectors.
- **FC Layer**: It introduces a fully connected layer over the concatenation of these four vectors.

Through above strategies, the four vectors are merged into a resultant vector which will be utilized for the cloze task.

To examine the effectiveness of these combination strategies, we conduct pre-training with the above strategies and show the downstream performance with TiSASRec as the base method on Video and Toys datasets. The results are presented in the Table 14. Clearly,

Table 13 Ablation study on LastFM dataset with TiSASRec as the base method

Method	LastFM			
	Hit		NDCG	
	k=5	k=10	k=5	k=10
-Distributed Consumption Module	30.14	38.57	22.28	25.01
-Concentrated Consumption Module	36.92	48.47	27.63	31.38
-Occasion Consumption Module	37.45	47.75	27.78	31.12
-CNN-based Modules	37.23	**49.86**	27.38	31.45
-Forward Process	34.14	43.60	25.34	28.41
-Backward Process	34.56	44.32	25.31	28.47
-Sequence Consistency	**38.66**	49.76	28.73	**32.33**
EEPT	**38.66**	49.54	**28.74**	32.26

Table 14 The results of three combination strategies on Video and Toys datasets with TiSASRec as the base method

Dataset	Model	Hit		NDCG	
		k=5	k=10	k=5	k=10
Video	EEPT	62.93	74.76	48.44	52.28
	Mean-Pooling	58.70	70.72	45.26	49.15
	Max-Pooling	59.79	72.16	45.60	49.61
	FC Layer	58.22	70.29	44.76	48.66
Toys	EEPT	40.36	51.60	30.08	33.71
	Mean-Pooling	38.74	49.06	29.34	32.67
	Max-Pooling	37.07	47.68	27.43	30.86
	FC Layer	35.62	45.78	26.70	29.97

the results show that all combination strategies hurt the downstream recommendation performance. This phenomenon might be due to the incompatibility among these modules. In EEPT, different modules handle different types of input sequences, and these sequence types reflect totally different latent patterns. For example, $\boldsymbol{S}_t^{lf}$ models the historical sequence while $\boldsymbol{C}'_t$ models the shuffled sequence. Thus a simple ensemble of different modules weakens the ability of each module and brings performance decrease of the downstream recommendation. Considering this, EEPT conducts the cloze task for different output vectors independently so as to avoid the negative effects of combining different modules.

4.6.3 Impacts of unidirectional processes

We examine whether two unidirectional processes in distributed consumption module works better than one bidirectional process in the original transformer. We introduce a variant EEPT$_B$ by replacing distributed consumption module in EEPT with a normal transformer. Experiments are conducted with TiSASRec as the base method. The results are shown in Table 15. It can be seen that all scores suffer from sharp decreases on Amazon datasets, which clearly demonstrates that two forward and backward processes not only model the consumption patterns, but also help avoid the information leakage problem. On the contrary, the performance on the music data LastFM increases in all metrics with a normal transformer. This is because that the absolute sequential order is more relax and the information leakage problem is not severe in LastFM. The users from LastFM are likely to visit pieces of music simultaneously. An example is u_2 in Figure 1, who orders four pieces of rock music in a single day. It is better to model both history and future contexts using the original bidirectional process than two independent unidirectional processes for a dataset like LastFM.

4.6.4 Impacts of amount of pre-training data

In the real-world applications, recommendation systems are likely to suffer from the insufficient data problem. To investigate the sensitivity of EEPT to the data size, we only use 50% of the full data for pre-training, and present the fine-tuning results on the sequential model TiSASRec. Note that an extra dropout layer is added after embedding lookup operation for preventing overfitting in EEPT. We mainly compare EEPT with the best pre-training baseline model SSI. From the results shown in Figure 5, it is obvious that EEPT still promotes the performance of downstream base method TiSASRec consistently, and outperforms the state-of-the-art pre-training baseline SSI substantially. This implies the adaptability of EEPT under the circumstance of insufficient data. Another interesting phenomenon is that SSI hinders

Table 15 Performance comparison between two unidirectional processes and one bidirectional process with TiSASRec as the base method

Dataset	Method	Hit		NDCG	
		k=5	k=10	k=5	k=10
Video	EEPT	62.93	74.76	48.44	52.28
	EEPT$_B$	61.75↓	73.69↓	47.39↓	51.26↓
Toys	EEPT	40.36	51.60	30.08	33.71
	EEPT$_B$	38.42↓	49.75↓	28.33↓	31.98↓
Cell	EEPT	48.97	61.04	37.14	41.04
	EEPT$_B$	47.71↓	59.88↓	35.83↓	39.76↓
Sports	EEPT	40.38	53.10	29.39	33.49
	EEPT$_B$	39.34↓	51.73↓	28.53↓	32.52↓
LastFM	EEPT	38.66	49.54	28.74	32.26
	EEPT$_B$	40.41↑	50.60↑	29.48↑	32.77↑

the fine-tuning performance of TiSASRec on Video. It is probably because the teachers in SSI do not learn soundly on the deficient Video data, and provides misleading signals to the downstream sequential model.

4.6.5 Impacts of history sequence length

We compare the performance of EEPT and SSI on users with different history sequence length m at the fine-tuning stage. We choose TiSASRec as the base method and conduct experiments on Video and Sports. The results are shown in Figure 6. Concretely, the users are divided into five groups based on the number of history sequence length m. The first group contains users having 4 or less behaviors, while users having 8 or more ones are in the last group. The performance result of each group is evaluated separately. As expected, SSI and EEPT bring positive effects on all groups, demonstrating the self-supervised signals indeed make important contributions. When comparing EEPT with SSI, we find that our EEPT always generates the best scores on the group with no more than 4 historical behaviors. This proves that EEPT can better deal with the insufficient data problem in sequential recommendation than SSI, which is a desired property in low-resource scenarios.

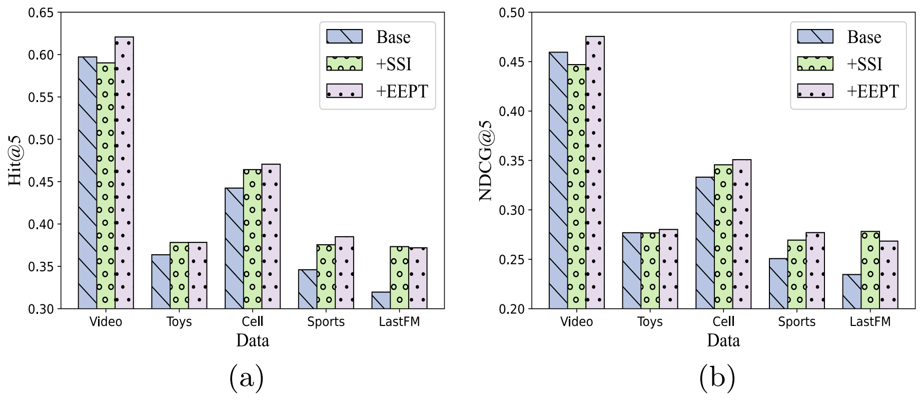


Figure 5 The fine-tuning results with TiSASRec as the base method, with only 50% of full data for pre-training. (a) Hit@5. (b) NDCG@5

Table 16 Parameter numbers of different models

Dataset	S3Rec	BERT4Rec	SSI	EEPT
Video	0.8M	0.8M	5.4M	1.8M
Toys	0.9M	0.9M	5.9M	2.0M
Cell	0.8M	0.8M	5.3M	1.8M
Sports	1.3M	1.3M	8.4M	2.8M
LastFM	0.9M	0.9M	4.9M	2.0M

4.7 Computational cost analysis

This section investigates the computational cost of pre-training methods, in terms of the model size, the epoch number for pre-training and fine-tuning.

Analysis on Model Size We first study the pre-training model size and list the trainable parameter numbers of different methods in Table 16. S3Rec and BERT4Rec have the lowest parameter costs, since they directly choose transformer as the sequence encoder. SSI assigns an independent BERT to each self-supervised task, and consequently it is about three times larger than EEPT. This analysis proves the effectiveness of EEPT, which achieves the best performance with much lower parameter cost.

Impacts of Pre-training Epoch We draw the performance curves corresponding to the pre-training epoch number in Figure 7. TiSASRec is the downstream base method, and Video Games and LastFM are selected as the representative datasets for this experiment. Apparently, the curves are nearly following the same trend regardless of different datasets. The score continuously rises at the first 50 epochs and then enters a plateau after the 60th epoch. This phenomenon provides a more convincing evidence that our model learns latent data features from self-supervised signals within a small number of epochs.

Impacts of Fine-tuning Epoch We further investigate the performance comparison with respect to different fine-tuning epoch numbers on Toys and Cell datasets, where TiSASRec is the base method. The curves are shown in Figure 8. Obviously, EEPT boosts the downstream sequential models not only in recommendation accuracy, but also the training efficiency. It only takes TiSASRec about 45 epochs to converge, while the base model takes

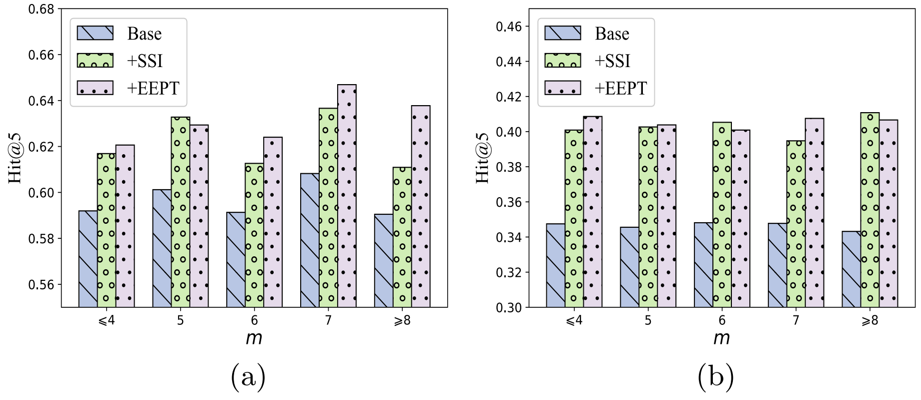


Figure 6 The performance of EEPT and SSI with different sequence length m on two datasets. TiSASRec is the base method. (a) Video. (b) Sports

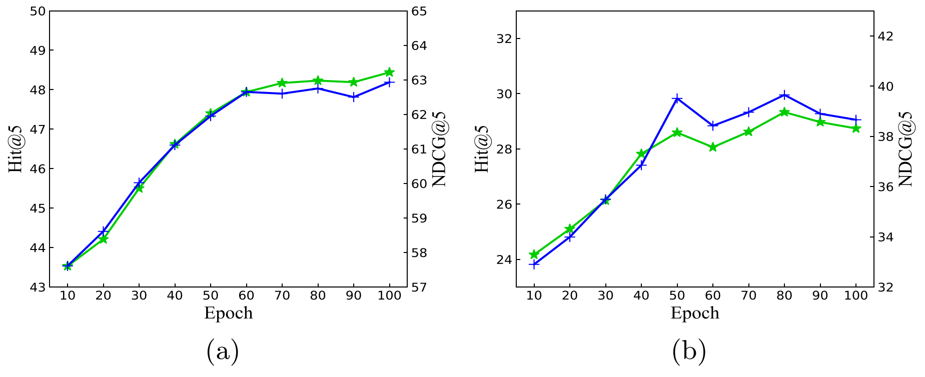


Figure 7 The impacts of different pre-training epoch numbers with TiSASRec as the base method on two datasets. (a) Video. (b) LastFM

about 65 epochs. Although the final results of SSI are very close to ours, its fine-tuning procedure is very time-consuming. This is because SSI uses knowledge distilling technique, and spends over 3 times than EEPT on model updates for one epoch.

4.8 Case analysis

We conduct a case study with TiSASRec as the downstream model to show whether our pre-training method learns the actual consumption types. In specific, we select three real sequence examples from Amazon dataset, and each of them reflects a specific consumption intention. We compare the attention scores assigned by TiSASRec to historical items, which measure the importance of each historical item to the predicted item. For each historical interaction, we present the attention score assigned by the last layer of TiSASRec. Furthermore, we show the categories of all items released by the raw dataset. The results are shown in Figure 9.

As can be seen, for all of example, TiSASRec with pre-trained embeddings correctly predicts the next ground truth item, while the original TiSASRec using randomly initialized embeddings makes wrong prediction.

The example *a* is a distributed consumption type sequence, where the transactions are distributed in a relatively long time period (14 months in total). Clearly, all historical purchases

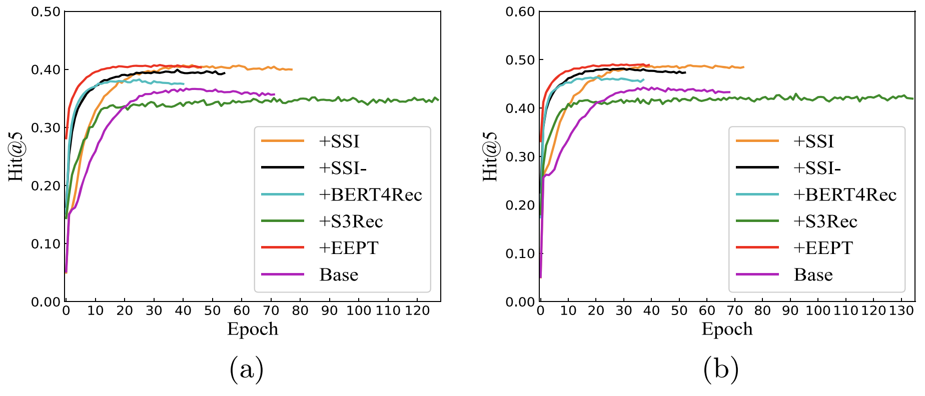


Figure 8 The impacts of different fine-tuning epoch numbers with TiSASRec as the base method on two datasets. (a) Toys. (b) Cell

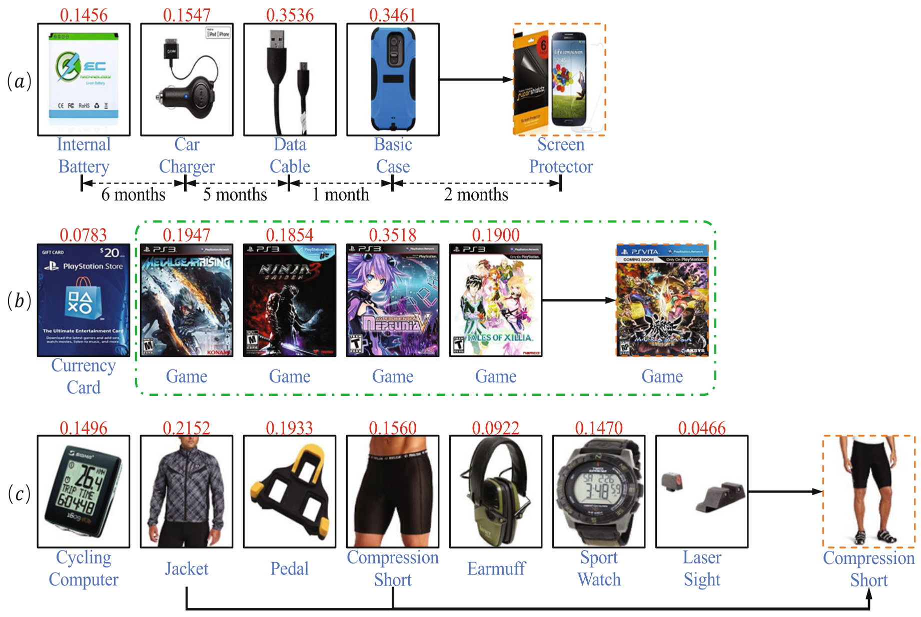


Figure 9 Three real sequence examples from Amazon datasets. We present historical items (marked in black rectangles) and the next ground truth items (marked in orange rectangles). Furthermore, we present attention scores of historical items (marked in red) and the categories of items (marked in blue)

are for the same intention of buying phone-related accessories. Despite of the long time interval, our pre-training method prompts TiSASRec to pay more attention to the internal battery (score of 0.1456) and the car charger (score of 0.1547). This proves that our method helps capture the distributed consumption.

The example b reflects a concentrated consumption intention. Items in the green dashed rectangle all belong to the game category and are bought in a single day. It is observed that the game items in the rectangle gain larger attention scores than the irrelevant item outside the rectangle. Thus we can conclude that EEPT also help handle the concentrated consumption.

As for the example c, the target Compression Short item is not relevant to the surrounding items but have connections with the faraway Jacket and Compression Short items, showing an occasion consumption intention. With our pre-trained embeddings, TiSASRec finds these hidden correlations and assigns higher attention scores to the relevant items, i.e., 0.2152 to the Jacket and 0.1560 to the Compression Short. This proves that our method enables downstream methods to model the occasion consumption.

5 Conclusion

In this paper, we propose a novel self-supervised pre-training method EEPT for sequential recommendation. Different from previous methods, our EEPT focuses on developing sequence encoders and extracts diverse user consumption patterns. Specifically, we develop the unidirectional transformer encoders and two CNN encoders, which are responsible for modeling the distributed consumption, the concentrated consumption, and the occasion consumption intentions, respectively. The pre-trained item embeddings are well learned and beneficial for diverse downstream sequential recommendation models. Extensive experiments on five real datasets have shown the superiority of EEPT over the base downstream methods and pre-training baselines.

Acknowledgements This work was supported by a grant from the National Natural Science Foundation of China (NSFC) project (No. 62276193) and a grant from the Key Laboratory of Satellite Information Intelligent Processing and Application Technology (No. 2022-ZZKY-JJ-16-01). It was also supported by the Joint Lab. on Credit Sci. and Tech. of CSCI-Wuhan University.

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Ke Sun and Tieyun Qian. The first draft of the manuscript was written by Ke Sun and revised by Tieyun Qian. Ming Zhong and Xuhui Li helped perform the analysis with constructive discussions. All authors read and approved the final manuscript.

Funding This work was supported by a grant from the National Natural Science Foundation of China (NSFC) project (No. 62276193). It was also supported by a grant from the Key Laboratory of Satellite Information Intelligent Processing and Application Technology (No. 2022-ZZKY-JJ-16-01).

Declarations

Financial interests The authors declare they have no financial interests.

Non-financial interests The authors declare they have no non-financial interests.

References

1. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th International Conference on World Wide Web, pp. 811–820 (2010)
2. He, R., McAuley, J.: Fusing similarity models with markov chains for sparse sequential recommendation. In: Proceedings of the 2016 IEEE 16th International Conference on Data Mining, pp. 191–200 (2016)
3. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. arXiv:1511.06939 (2015)
4. Yu, F., Liu, Q., Wu, S., Wang, L., Tan, T.: A dynamic recurrent model for next basket recommendation. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 729–732 (2016)
5. Sun, K., Qian, T., Yin, H., Chen, T., Chen, Y., Chen, L.: What can history tell us? In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 1593–1602 (2019)
6. Sun, K., Qian, T., Chen, T., Liang, Y., Nguyen, Q.V.H., Yin, H.: Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence, pp. 214–221 (2020)
7. Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., Ma, J.: Neural attentive session-based recommendation. In: Proceedings of the 26th ACM International Conference on Information and Knowledge Management, pp. 1419–1428 (2017)
8. Liu, Q., Zeng, Y., Mokhosi, R., Zhang, H.: Stamp: short-term attention/memory priority model for session-based recommendation. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1831–1839 (2018)
9. Kang, W.-C., McAuley, J.: Self-attentive sequential recommendation. In: Proceedings of the 2018 IEEE 18th International Conference on Data Mining, pp. 197–206 (2018)
10. Liu, Q., Wu, S., Wang, D., Li, Z., Wang, L.: Context-aware sequential recommendation. In: Proceedings of the 2016 IEEE 16th International Conference on Data Mining, pp. 1053–1058 (2016)
11. Zhu, Y., Li, H., Liao, Y., Wang, B., Guan, Z., Liu, H., Cai, D.: What to do next: Modeling user behaviors by time-lstm. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 3602–3608 (2017)
12. Zhao, P., Zhu, H., Liu, Y., Xu, J., Li, Z., Zhuang, F., Sheng, V.S., Zhou, X.: Where to go next: A spatio-temporal gated network for next poi recommendation. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, pp. 5877–5884 (2019)

13. Sun, K., Qian, T., Chen, X., Zhong, M.: Context-aware seq2seq translation model for sequential recommendation. Inform. Sci. **581**, 60–72 (2021)
14. Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., Huang, X.: Pre-trained models for natural language processing: A survey. Science China Technological Sciences, 1–26 (2020)
15. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018)
16. Sheng, X.-R., Zhao, L., Zhou, G., Ding, X., Dai, B., Luo, Q., Yang, S., Lv, J., Zhang, C., Deng, H., et al.: One model to serve all: Star topology adaptive recommender for multi-domain ctr prediction. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management, pp. 4104–4113 (2021)
17. Hao, X., Liu, Y., Xie, R., Ge, K., Tang, L., Zhang, X., Lin, L.: Adversarial feature translation for multi-domain recommendation. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2964–2973 (2021)
18. Tang, H., Liu, J., Zhao, M., Gong, X.: Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In: 14th ACM Conference on Recommender Systems, pp. 269–278 (2020)
19. Wang, H., Chang, T.-W., Liu, T., Huang, J., Chen, Z., Yu, C., Li, R., Chu, W.: Escm2: Entire space counterfactual multi-task model for post-click conversion rate estimation. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 363–372 (2022)
20. Zhang, Q., Liu, J., Dai, Y., Qi, Y., Yuan, Y., Zheng, K., Huang, F., Tan, X.: Multi-task fusion via reinforcement learning for long-term user satisfaction in recommender systems. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 4510–4520 (2022)
21. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 1441–1450 (2019)
22. Yuan, X., Chen, H., Song, Y., Zhao, X., Ding, Z., He, Z., Long, B.: Improving sequential recommendation consistency with self-supervised imitation. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence, pp. 3321–3327 (2021)
23. Zhou, K., Wang, H., Zhao, W.X., Zhu, Y., Wang, S., Zhang, F., Wang, Z., Wen, J.-R.: S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management, pp. 1893–1902 (2020)
24. Yuan, F., He, X., Jiang, H., Guo, G., Xiong, J., Xu, Z., Xiong, Y.: Future data helps training: Modeling future contexts for session-based recommendation. In: Proceedings of the 29th International Conference on World Wide Web, pp. 303–313 (2020)
25. Wang, C., Zhang, M., Ma, W., Liu, Y., Ma, S.: Modeling item-specific temporal dynamics of repeat consumption for recommender systems. In: Proceedings of the 28th International Conference on World Wide Web, pp. 1977–1987 (2019)
26. Wang, C., Zhang, M., Ma, W., Liu, Y., Ma, S.: 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, pp. 109–118 (2020)
27. Bai, T., Nie, J.-Y., Zhao, W.X., Zhu, Y., Du, P., Wen, J.-R.: An attribute-aware neural attentive model for next basket recommendation. In: Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1201–1204 (2018)
28. Cheng, C., Yang, H., Lyu, M.R., King, I.: Where you like to go next: Successive point-of-interest recommendation. In: Proceedings of the 23th International Joint Conference on Artificial Intelligence (2013)
29. Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., Cheng, X.: Learning hierarchical representation model for nextbasket recommendation. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 403–412 (2015)
30. Yu, Z., Lian, J., Mahmoody, A., Liu, G., Xie, X.: Adaptive user modeling with long and short-term preferences for personalized recommendation. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 4213–4219 (2019)
31. Beutel, A., Covington, P., Jain, S., Xu, C., Li, J., Gatto, V., Chi, E.H.: Latent cross: Making use of context in recurrent recommender systems. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining, pp. 46–54 (2018)
32. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Proceedings of the Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)

33. Zhou, C., Bai, J., Song, J., Liu, X., Zhao, Z., Chen, X., Gao, J.: Atrank: An attention-based user behavior modeling framework for recommendation. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, pp. 4564–4571 (2018)
34. Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C., Zhu, X., Gai, K.: Deep interest evolution network for click-through rate prediction. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, pp. 5941–5948 (2019)
35. Li, J., Wang, Y., McAuley, J.: Time interval aware self-attention for sequential recommendation. In: Proceedings of the 13th ACM International Conference on Web Search and Data Mining, pp. 322–330 (2020)
36. Yuan, E., Guo, W., He, Z., Guo, H., Liu, C., Tang, R.: Multi-behavior sequential transformer recommender. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1642–1652 (2022)
37. Wang, S., Zhang, M., Miao, H., Peng, Z., Yu, P.S.: Multivariate correlation-aware spatio-temporal graph convolutional networks for multi-scale traffic prediction. ACM Trans. Intell. Syst. Technol. **13**(3), 1–22 (2022)
38. Wang, S., Cao, J., Yu, P.: Deep learning for spatio-temporal data mining: A survey. IEEE Trans. Knowl. Data Eng. **34**(8), 3681–3700 (2020)
39. Peng, H., Li, J., Wang, S., Wang, L., Gong, Q., Yang, R., Li, B., Philip, S.Y., He, L.: Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification. IEEE Trans. Knowl. Data Eng. **33**(6), 2505–2519 (2019)
40. Ma, M., Ren, P., Chen, Z., Ren, Z., Zhao, L., Liu, P., Ma, J., de Rijke, M.: Mixed information flow for cross-domain sequential recommendations. ACM Trans. Knowl. Disc. Data **16**(4), 1–32 (2022)
41. Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: Session-based recommendation with graph neural networks. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, pp. 346–353 (2019)
42. Xia, X., Yin, H., Yu, J., Wang, Q., Cui, L., Zhang, X.: Self-supervised hypergraph convolutional networks for session-based recommendation. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence, pp. 4503–4511 (2021)
43. Ma, C., Kang, P., Liu, X.: Hierarchical gating networks for sequential recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 825–833 (2019)
44. Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., Zha, H.: Sequential recommendation with user memory networks. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining, pp. 108–116 (2018)
45. Tan, Q., Zhang, J., Liu, N., Huang, X., Yang, H., Zhou, J., Hu, X.: Dynamic memory based attention network for sequential recommendation. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence, pp. 4384–4392 (2021)
46. Pi, Q., Bian, W., Zhou, G., Zhu, X., Gai, K.: Practice on long sequential user behavior modeling for click-through rate prediction. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2671–2679 (2019)
47. Huang, J., Ren, Z., Zhao, W.X., He, G., Wen, J.-R., Dong, D.: Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In: Proceedings of the 12th ACM International Conference on Web Search and Data Mining, pp. 573–581 (2019)
48. Yu, B., Li, X., Fang, J., Tai, C., Cheng, W., Xu, J.: Memory-augmented meta-learning framework for session-based target behavior recommendation. World Wide Web **26**(1), 233–251 (2023)
49. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining, pp. 565–573 (2018)
50. Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J.M., He, X.: A simple convolutional generative network for next item recommendation. In: Proceedings of the 12th ACM International Conference on Web Search and Data Mining, pp. 582–590 (2019)
51. Xu, C., Zhao, P., Liu, Y., Xu, J., S. Sheng, V.S.S., Cui, Z., Zhou, X., Xiong, H.: Recurrent convolutional neural network for sequential recommendation. In: Proceedings of the 28th International Conference on World Wide Web, pp. 3398–3404 (2019)
52. Lin, Y., Wan, H., Guo, S., Lin, Y.: Pre-training context and time aware location embeddings from spatial-temporal trajectories for user next location prediction. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence, pp. 4241–4248 (2021)
53. Liu, Z., Fan, Z., Wang, Y., Yu, P.S.: Augmenting sequential recommendation with pseudo-prior items via reversely pre-training transformer. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1608–1612 (2021)

54. Jiang, J., Luo, Y., Kim, J.B., Zhang, K., Kim, S.: Sequential recommendation with bidirectional chronological augmentation of transformer. arXiv:2112.06460 (2021)
55. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv:1607.06450 (2016)
56. Qiu, R., Huang, Z., Yin, H., Wang, Z.: Contrastive learning for representation degeneration problem in sequential recommendation. In: Proceedings of the 15th ACM International Conference on Web Search and Data Mining, pp. 813–823 (2022)
57. Wang, C., Ma, W., Chen, C.: Sequential recommendation with multiple contrast signals. ACM Transactions on Information Systems (2022)