User Comprehensive Representation via Cross Multi-source Behavior Pre-training on Mobile Terminal

Chengqi Yang*
University of Chinese Academy of Sciences
Beijing, China
yangchengqi18@mails.ucas.ac.cn

Zijun Zhou*
OPPO Research Institute
Shenzhen, China
zhouzijun1@oppo.com

Yuqing Yuan†
OPPO Research Institute
Shenzhen, China
yuanyuqing@oppo.com

Chanfei Su
OPPO Research Institute
Shenzhen, China
suchanfei@oppo.com

Xiaoyun Mo
OPPO Research Institute
Shenzhen, China
moxiaoyun@oppo.com

Jun Wang
OPPO Research Institute
Shenzhen, China
wangjun7@oppo.com

Xiang Ao†
University of Chinese Academy of Sciences
Beijing, China
aoxiang@ucas.ac.cn

ABSTRACT

User representation pre-training aims to capture users' interests based on extensive unlabeled user behaviors, avoiding data sparsity issues when training task-specific models. In this paper, we investigate the user representation pre-training on mobile terminals, whose behavior data fosters two imperative differences from existing research. First, the user behaviors come from multiple sources. That is because a mobile terminal supplier can not only access detailed user interactions within the pre-installed Apps but also listen to coarse actions of third-party Apps such as installations, launches, or uninstalls. Second, the user intentions may consist of cross-sequence activities. It means behaviors related to a particular intention may successively occur in behavior sequences from different sources. To this end, we propose the **C**ross **M**ulti-source Behavior **P**re-**T**raining **M**odel (**CM-PTM** for short), in which several hierarchical cascaded mask-then-predict proxy tasks are devised. CM-PTM first predicts in which sequence the user's next behavior will occur, and then sequentially predicts coarse-to-fine actions at the App level. Following such a design can better carve the diversity and complexity of user behaviors on mobile terminals. Extensive experiments on six real-world large-scale datasets demonstrate that our model can effectively capture users' comprehensive interests and provide significant improvements on different downstream tasks.

*Both authors contributed equally to this research.
†Corresponding authors.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

User Behavior Modeling, Pre-training Model, Multi-sequence Modeling, Cross-sequence Modeling

ACM Reference Format:
Chengqi Yang, Zijun Zhou, Yuqing Yuan, Chanfei Su, Xiaoyun Mo, Jun Wang, and Xiang Ao. 2024. User Comprehensive Representation via Cross Multi-source Behavior Pre-training on Mobile Terminal. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With the advent of the digital age, significant transformations have occurred in people's lifestyles, work dynamics, and social interactions. Intelligent mobile phones have evolved into essential accessories for everyone. On a daily basis, individuals engage in numerous activities on their mobile terminals, such as socializing, online shopping, and watching short videos. These behaviors, due to serving as indicators of user-specific characteristics or interests, provide a natural endogenous data source for modeling user representations.

User representation modeling [20] refers to learning a dense vector representing users' characteristics and interests based on their historical behaviors. It can be applicable to various downstream tasks related to users, such as user profile prediction, personalized recommendation, and click-through rate prediction [18, 31, 45]. Recently, inspired by the success of pre-training models (also known as PTM) in NLP [8, 34], CV [2, 3], fraud transactions detection [17] etc, some studies have begun to train models with large-scale unlabeled historical user behavior data by self-supervised learning techniques and then fine-tune models with labeled data in downstream tasks. For instance, some methods [33, 42] aligned user behavior with text, using masked-item-prediction and next-item-prediction as pre-training tasks. Additionally, some studies [4, 19, 38] proposed contrastive self-supervised learning at user level, training the model by maximizing the similarity between the original and the augmented views. Models trained with pre-training paradigms not only enhance the transferability of user representations but also effectively avoid the problem of data sparsity when few labeled data are available [10, 43].

Despite the remarkable success of previous studies, we observe they predominantly focused on users' behaviors within a single source or App [4, 38, 40, 42]. While the data on mobile terminals are very different. A mobile terminal supplier, in general, might be able to listen to user data from multiple sources, which typically fall into two categories. The first category encompasses activities within the pre-installed Apps on the phone. Detailed user interactions within these Apps can be accessed by the mobile terminal supplier. The second category involves activities within third-party apps. Though it lacks detailed information about user activities within these third-party Apps, these phone manufacturers are also able to get knowledge about coarse actions like App installations, launches, or uninstalls.

Apart from multi-source data, another very special characteristic of user behavior data on mobile terminals is cross-sequence activities. For example, Fig. 1 illustrates the real user behaviors on a mobile terminal extracted from our dataset. The process of this user downloading *Honor of Kings* spans two sequences, i.e. Third-party Apps and Appstore. Similarly, the process of this user becoming interested in *Zombie Frontier* but ultimately not downloading it spans the sequence of Third-party Apps, Appstore and Browser. Certainly, in addition to crossing sequences, there are also user behaviors crossing different Apps within a single sequence, such as the process of this user recharging games, which is coordinated with previous user representation modeling studies.

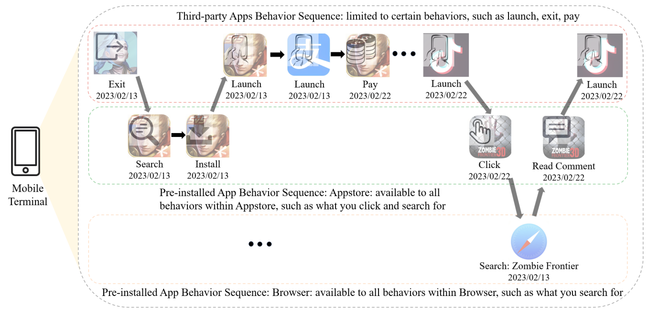
Third-party Apps Behavior Sequence: limited to certain behaviors, such as launch, exit, pay
Exit 2023/02/13
Launch 2023/02/13
Launch 2023/02/13
Pay 2023/02/22
Launch 2023/02/22
Launch 2023/02/22
Search 2023/02/13
Install 2023/02/13
Click 2023/02/22
Read Comment 2023/02/22
Mobile Terminal
Pre-installed App Behavior Sequence: Appstore: available to all behaviors within Appstore, such as what you click and search for
Search: Zombie Frontier 2023/02/13
Pre-installed App Behavior Sequence: Browser: available to all behaviors within Browser, such as what you search for

Figure 1: A user's behaviors on a mobile terminal. There are three kinds of sequences. First sequence represent the user's behavior in third-party Apps, the second sequence and the third sequence represent the user's behavior in Appstore and browser respectively.

Through this example, we can summarize the main challenges of modeling multi-source behavior sequences for user representation on mobile terminal as follows.

Unlike single-sequence scenarios, user behaviors denoting the same intention may transcend a single sequence. In other words, a complete intention of the user may be scattered in different sequences, resulting in inter-sequence associations. At the same time, these inter-sequence associations will also weaken the correlation between adjacent behaviors within the same sequence. However, this correlation is imperative for existing generative PTMs [33, 42]. Therefore, a new surrogate self-supervised learning task is is urgently demanded.

To this end, we propose the **C**ross **M**ulti-source Behavior **P**re-**T**raining **M**odel (**CM-PTM** for short). Though our model still follows the mask-then-predict framework, we design several hierarchical cascaded proxy tasks. Specifically, based on a user's historical behavior on the mobile terminal, CM-PTM first predicts in which sequence the user's next behavior will occur, and then sequentially predicts the category of App, App ID, and user operation type on this App. We believe these proxy tasks can better carve the diversity and complexity of user behaviors on mobile terminals. Our contributions are as follows:

- To the best of our knowledge, we are the first to propose a cross-sequence cascaded multi-task prediction pre-training framework that comprehensively considers the integrity of user behavior across the entire mobile terminal.
- We conduct extensive experiments on six real-world downstream tasks and the results demonstrate the effectiveness of our proposed approach.

2 RELATED WORK

In this section, we will briefly review several works related to ours, including user representation pre-training and sequence modeling.

2.1 User Representation Pre-training

The pre-training methods, e.g., BERT [6] , GPT [25] and ELMo [22], can achieve excellent performance while significantly reducing training costs and the budget of downstream tasks. Additionally, these models can leverage unlabeled data for self-supervised training, mitigating the sparsity issue in labeled data. In recent years, some studies have also begun to model user representations using pre-training paradigms. The mainstream user representation pre-training methods can be divided into two categories: generative methods and discriminative methods.

2.1.1 Generative methods. Generative methods usually disrupt user's behavior sequence and then attempt to reconstruct the disrupted sequence based on surrounding behaviors. Commonly used generative strategies include masked behavior prediction [24] and next behavior prediction [42]. [33] concurrently chose these two tasks to train user representations and achieved excellent performance. Recently, [7] proposed a user representation learning approach for multi-time scale distribution prediction, forecasting the distribution of user behavior over a future time period.

2.1.2 Discriminative methods. Discriminative methods usually apply explicit or implicit data augmentation and contrastive learning to learn user representations. Commonly used method is to maximize the similarity between the original sequence and the augmented sequence. There are various approaches for performing explicit data augmentation, such as dropout [8], mask-and-fill [1], crop [30], mask and substitute [19]. However, the sequence view obtained through these data augmentation methods may not be semantically consistent with the original view [40]. Additionally, methods based on implicit data may add distinct noise into the feature space [4, 23, 40].

2.2 Sequence Modeling

2.2.1 Single-sequence Modeling. Early works on user behavior modeling [16, 27] overlooked the order of these behaviors. With [28] proposing to use Markov chain (MC) and Markov Decision Process (MDP) to sequentially model user behavior, sequence modeling gradually developed. Since then, numerous methods based on MDP [10, 26], RNN[11] and CNN [41] are applied for sequence modeling. Recently, due to the powerful sequence modeling capabilities demonstrated by Transformer [32], approaches based on attention mechanism [13] and Transformer Encoder [40] have become mainstream.

2.2.2 Multi-sequence Modeling. Apart from single sequence modeling, there have been studies focusing on multi-sequence modeling. [15] adopted multi-scale behavior sequence generated from different granularities of web page structures to detect online fraud. In recommender systems, [12, 37, 39] have found that utilizing diverse behaviors such as click, cart, and purchase can more comprehensively encompass users' interests and jointly model these sequences in various ways. For example, some studies aim to facilitate interactions between different behavior sequences from the perspective of sequence modeling using attention mechanism [44] or dynamic methods [5] while others combine sequence modeling and graphs to capture more comprehensive correlations between different user behaviors [35].

However, data from previous studies tend to be single-source, of the same granularity, and focus on using multi-sequence modeling for specific tasks. Our data comes from multiple sources at different granularities, and we combine multi-sequence modeling with pre-training paradigm for general user representations.

3 METHODOLOGY

In this section, we introduce our proposed CM-PTM framework for user representation pre-training. In our experiment, we utilize five sequences as input data for the model, including the behaviors of all third-party GameApps on mobile terminal, the user's click behaviors in Appstore, search behaviors in Appstore, click behaviors in Gamecenter and search behaviors in Gamecenter. Fig. 2 illustrates the overall architecture of our model. Considering the relatively sparse behaviors of Appstore and Gamecenter, we first concatenate the click and search behaviors in Appstore into one sequence based on timestamps and perform the same operation on Gamecenter. After concatenating, three sequences are obtained. Then an attention mechanism is used to interact the embeddings of these three sequences and output three corresponding user representations for the three sequences. Each user representation will be individually used to complete pre-training tasks, predicting the user's next behavior in the order of sequence, App category, App ID and operation type. We will provide a detailed explanation of the entire process later.

3.1 Problem Statement

Before delving into the details of the model, we first provide a formal description for the problem. Assuming that we have collected a substantial amount of user behavior data, including five different types of behaviors: GameApp operation, Appstore click, Appstore search, Gamecenter click and Gamecenter search. Formally, we define $\mathcal{U} (u \in \mathcal{U})$ to denote the set of users, and $\mathcal{I}_k (k = 1, 2, ..., 5)$ to respectively denote the sets of mentioned five types of behaviors above. For each user u, the set of his or her behaviors can be presented as $\mathcal{S}_u = \{\mathcal{S}_u^1, \mathcal{S}_u^2, \mathcal{S}_u^3, \mathcal{S}_u^4, \mathcal{S}_u^5\}$, where $\mathcal{S}_u^k = \left\{x_1^k, x_2^k, ..., x_{n_k}^k\right\}$ ($x_i^k \in \mathcal{I}_k, k = 1, 2, ..., 5$) represents a behavior sequence composed of behaviors of the same type.

Formally, our user model can be expressed as:

$$\boldsymbol{u} = \mathcal{F}_\theta(\mathcal{S}_u), \tag{1}$$

where $\boldsymbol{u} \in \mathbb{R}^{d_u}$ denotes the user representation learned by our model. θ represents the model parameters.

3.2 Embedding layer

The input to our model is the set of five sequences, denoted as $\mathcal{S}_u = \{\mathcal{S}_u^1, \mathcal{S}_u^2, \mathcal{S}_u^3, \mathcal{S}_u^4, \mathcal{S}_u^5\}$. Through our statistical analysis of the real world data, we found that the majority of users have a relatively low number of click and search behaviors in Appstore and Gamecenter. To avoid data sparsity, we mix the click and search behaviors in Appstore, as well as the click and search behaviors in Gamecenter based on the timestamp of behavior. Then we truncate sequences longer than l and pad sequences shorter than l, l is a hyper-parameter. The entire process can be expressed with the following formula:

$$\boldsymbol{E}_u^1 = \text{Embedding}(\text{Padding}_l(\text{Truncate}_l(\mathcal{S}_u^1))), \tag{2}$$

$$\boldsymbol{E}_u^2 = \text{Embedding}(\text{Padding}_l(\text{Truncate}_l(\text{Mix}(\mathcal{S}_u^2, \mathcal{S}_u^3)))), \tag{3}$$

$$\boldsymbol{E}_u^3 = \text{Embedding}(\text{Padding}_l(\text{Truncate}_l(\text{Mix}(\mathcal{S}_u^4, \mathcal{S}_u^5)))), \tag{4}$$

where $\boldsymbol{E}_u^k = \left\{\boldsymbol{e}_1^k, \boldsymbol{e}_2^k, ..., \boldsymbol{e}_l^k\right\}, k = 1, 2, 3$, $\boldsymbol{e}_i^k \in \mathbb{R}^d$ denotes the embedding of one behavior or padding behavior.

Considering the importance of element positions in sequence modeling [32], for each $\boldsymbol{e}_i^k \in \boldsymbol{E}_u^k, i = 1, 2, ..., l$ and $k = 1, 2, 3$, we set:

$$\boldsymbol{e}_i^k = \boldsymbol{e}_i^k + \text{PositionEmbedding}(\boldsymbol{e}_i^k), \tag{5}$$

3.3 User Model

In subsection 3.2, we have obtained behavior embeddings $\boldsymbol{E}_u = \left\{\boldsymbol{E}_u^k\right\}, k = 1, 2, 3$ for each user $u \in \mathcal{U}$. To obtain the user representation, we use attention mechanism for modeling the serialized embeddings $\boldsymbol{E}_u$.

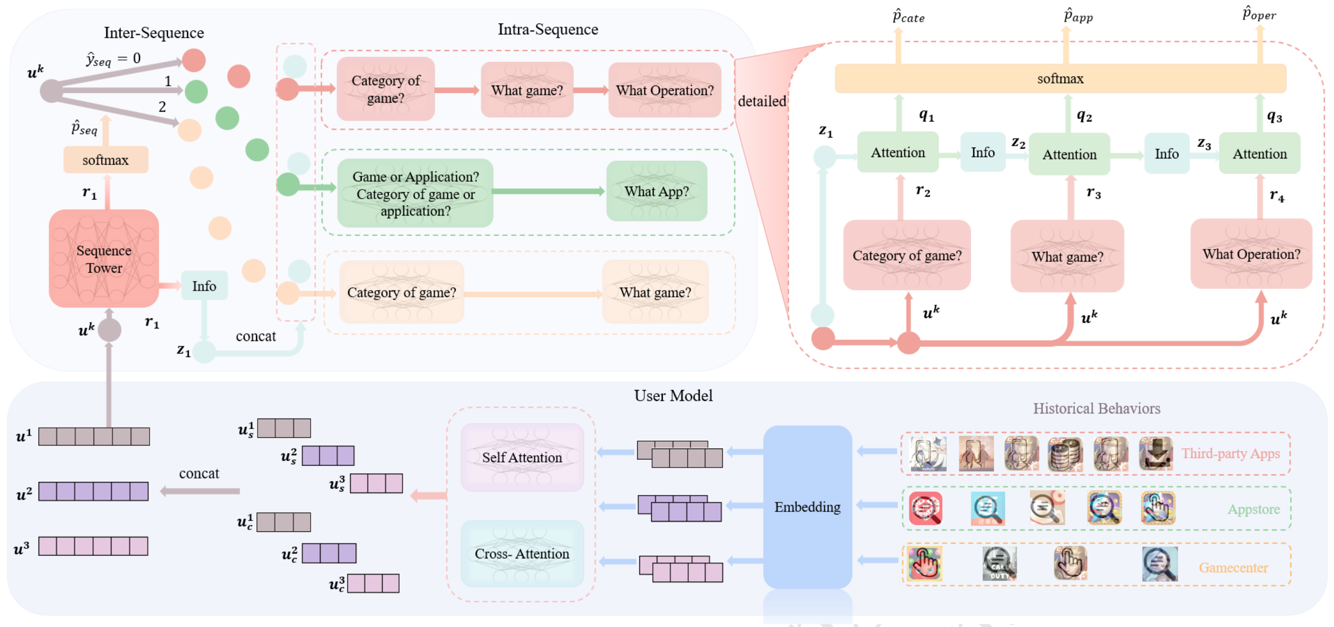

Figure 2: The overall architecture of our proposed CM-PTM. The lower part of this figure represents user model, generating user representations based on user's historical behaviors in third-party GameApps, Appstore and Gamecenter. The left side of the upper part illustrate our cross-sequence cascaded multi-task framework, where the three different colors of red, green and yellow represent that u^k needs to be divided into different task groups according to the results of sequence prediction. The right side uses third-party GameApps behavior sequence as an example to show our cascaded structure in detail.

3.3.1 Self-Attention Representation. For each $E_u^k(k=1,2,3)$, we use self-attention to calculate its self hidden state S^k as follows:

$$S^k=[s_1^k,s_2^k,...,s_l^k]=\text{Self\_Attention}(E_u^k), \tag{6}$$

Based on Eq (6) , self-attention representations can be calculated as follows:

$$\boldsymbol{u}_s^k=\sum_{i=1}^{l}\alpha_i^k s_i^k, \tag{7}$$

where $\alpha_i^k=\frac{exp(\boldsymbol{q}^\top s_i^k)}{\sum_{j=1}^{l}exp(\boldsymbol{q}^\top s_j^k)}$, $s_i^k\in\mathbb{R}^d$, $\boldsymbol{q}$ is a learnable query vector extracted from E_u^k.

3.3.2 Cross-Attention Representation. To enhance the model's cross-prediction capabilities, we proactively employ cross-attention to facilitate interactions among the information from different sources. It uses one sequence to provide query, while the other two sequences jointly provide key and value. For each $E_u^k(k=1,2,3)$, cross-attention representation can be calculated as follows:

$$\begin{aligned}C^k&=[c_1^k,c_2^k,...,c_l^k]=\text{Cross\_Attention}(Q=\phi_0(E_u^k),\\ K&=\phi_1(\text{concat}(E_u^j|j\neq k)),V=\phi_2(\text{concat}(E_u^j|j\neq k)))\end{aligned} \tag{8}$$

where concat$(E_u^j|j\neq k)$ means concatenating vectors along the second dimension, taking $k=1$ for example, concat$(\{E_u^2,E_u^3\})\in\mathbb{R}^{l\times 2d}$. ϕ_0,ϕ_1,ϕ_2 respectively represent the vector wise network function for extracting Q,K,V from sequence, and

$$\boldsymbol{u}_c^k=\sum_{i=1}^{l}\beta_i^k c_i^k, \tag{9}$$

where $\beta_i^k=\frac{exp(\boldsymbol{q}^\top c_i^k)}{\sum_{j=1}^{l}exp(\boldsymbol{q}^\top c_j^k)}$, $c_i^k\in\mathbb{R}^d$.

3.3.3 User Representation. For pre-training tasks, since we aim to predict the next behavior for each sequence individually, the representation of user $u\in\mathcal{U}$ is:

$$\boldsymbol{u}^k=\text{Concat}(\boldsymbol{u}_s^k,\boldsymbol{u}_c^k), \tag{10}$$

where $k=1,2,3$.

For downstream tasks, the representation of user $u\in\mathcal{U}$ is:

$$\boldsymbol{u}=\text{Concat}(\boldsymbol{u}_s^1,\boldsymbol{u}_s^2,\boldsymbol{u}_s^3,\boldsymbol{u}_c^1,\boldsymbol{u}_c^2,\boldsymbol{u}_c^3) \tag{11}$$

3.4 Cross-Sequence Multi-task Cascade Prediction

User model provides three user representations $\{\boldsymbol{u}^1,\boldsymbol{u}^2,\boldsymbol{u}^3\}$ used for pre-training tasks. Each $\boldsymbol{u}^k(k=1,2,3)$ is individually used to predict the user's next behavior. In order to learn users' heterogeneous behavioral habits and deepen our comprehensive understanding of users, we propose a cross-sequence multi-task cascade prediction approach. Overall, we divide the predicting process into two stages, inter-sequence and intra-sequence. Our first proxy task is sequence prediction (T_1), predicting in which sequence the user's next behavior will occur based on user representation (inter-sequence). The next step is to predict specific behaviors (intra-sequence). Considering that directly predicting specific App is challenging, and achieve too low accuracy might adversely affect training user representations, we design multi-level category proxy tasks as the second set of proxy tasks (T_2) before predicting specific App. In our dataset, each App has a first-level category, a second-level category, and a third-level category. For example, the categories of *Sky: Children of Light* are: game ⟶ action adventure game ⟶ adventure game; *Tiktok*: application ⟶ video play ⟶ short videos. If T_1 predicts that the user's next behavior will occur in Appstore, then we will predict these three categories step by step. If it is predicted that the user's next behavior will occur in Gamecenter or third-party GameApps, we will start prediction directly from second-level category, because the Apps that the user interacts with in Gamecenter or third-party GameApps must be games. After T_2, our third proxy task is to predict App (T_3). Based on the results of T_2, the candidate pool for T_3 will be greatly reduced, thereby reducing the difficulty of this task. Our last proxy task is to predict the user's operation type on the App (T_4). We believe that users' click and search behaviors in Appstore and Gamecenter all reflect the user's positive attitude to this App, so we don't distinguish whether the user specifically performs clicks or searches. However, user behaviors towards third-party GameApps include launching, installing, paying, and uninstalling. These behaviors may reflect various user attitudes. Therefore, we also designed an operation type prediction task for the behaviors of third-party GameApps.

It is evident in this proxy task chain that the prediction of the previous task is helpful for the next one. Therefore, on top of multi-task, inspired by [36], we establish a task-cascaded structure. Then we provide the formula representation of our cross-sequence multi-task cascade prediction.

3.4.1 *Inter-sequence.*

When given a $\boldsymbol{u}^k(k = 1, 2, 3)$ as input, the model initially uses a sequence tower to implement T_1. An Info module is utilized to determine which information needs to be transmitted to the subsequent prediction task in order to achieve task cascading. Then, based on the sequence prediction results, the model steers all input $\boldsymbol{u}^k$ to different streams, select their subsequent prediction tasks and take $\boldsymbol{u}^k$ and the learned z_1 as joint inputs for Intra-sequence stage. This phase can be represented by formula:

$$\boldsymbol{r}_1 = \theta_{\text{seq}}(\boldsymbol{u}^k), \tag{12}$$

$$\hat{\boldsymbol{p}}_{seq} = \text{Softmax}(\boldsymbol{r}_1), \tag{13}$$

$$z_1 = \theta_{\text{info}}(\boldsymbol{u}^k) \tag{14}$$

where $\theta_{\text{seq}}(\cdot)$ represents the network of sequence tower used to implement T_1, we use MLP in this work. $\theta_{info}(\cdot)$ represents the network used to implement module Info, we use fully connected layers in this work.

3.4.2 *Intra-sequence.*

In this stage, each input $\boldsymbol{u}^k$ is associated with a set of specific prediction tasks and a corresponding z_1. Tasks in this stage also adopt a cascaded structure. Info module learns which information is selected to pass to the next prediction task, while Attention module comprehensively extracts the most relevant information from the output of previous task and the current task's prediction tower. This phase can be represented by formula:

$$\boldsymbol{r}_i = \theta_T^i(\boldsymbol{u}^k), i = 2, 3, 4 \tag{15}$$

$$\boldsymbol{q}_i = \text{Self\_Attention}(\boldsymbol{r}_{i+1}, \boldsymbol{z}_i), i = 1, 2, 3 \tag{16}$$

$$z_i = \theta_{\text{info}}(\boldsymbol{q}_{i-1}), i = 2, 3 \tag{17}$$

where $\theta_T^i(\cdot)$ represents the network of the i-th task tower, we similarly use MLP in this work.

3.5 Model Training

Let $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ represent the prediction task sets for GameApp, Appstore and Gamecenter in intra-sequence stage respectively. The model is trained under the cross entropy loss which can be described as below:

$$\mathcal{L}_{SEQ} = -\sum_{k=1}^{3}\sum_{i=0}^{2}(y_{seq}^k = i)\log(\hat{p}_{seq}^k) \tag{18}$$

$$\mathcal{L}_{BEH} = -\sum_{k=1}^{3}\sum_{T\in\mathcal{S}_k}\sum_{i=0}^{N(T)-1}(y_T^k = i)\log(\hat{p}_T^k) \tag{19}$$

where $N(T)$ represents the total number of categories for task T, $\hat{p}_T^k$ represents the prediction results of task T, the label of task category i is $i-1$.

The overall loss function of our pre-training tasks is:

$$\mathcal{L} = \lambda\mathcal{L}_{SEQ} + (1-\lambda)\mathcal{L}_{BEH}, \tag{20}$$

where λ is a hyper-parameter measuring the importance of sequence prediction task and behavior attributes prediction tasks. We will discuss the impact of choosing different values of λ on the model's performance in section 4.7.

4 EXPERIMENT

In this section, many experiments on large-scale real-word industrial datasets are conducted to investigate the effectiveness of our proposed CM-PTM.

4.1 Datasets

We will introduce our datasets used in the experiment and the details. All of the datasets are collected from **OPPO**, a mainstream smartphone manufacturer. Each user has been hashed into an anonymized ID to prevent privacy leakage. The datasets comprise user IDs and user behaviors collected from 180 days leading up to the sampling date. User behaviors are comprised of:

- GameApp: Launch, Install, Uninstall and Pay.
- Appstore: Search and Click.
- Gamecenter: Search and Click.

4.1.1 *Pre-training Dataset.*

To ensure an adequate and diverse dataset, we selected a one-week sampling interval for all of our datasets, ranging from 2023/07/13 to 2023/07/19. We randomly sampled pre-training users from active game users. This dataset contains approximately 1.95 million mobile users and their behaviors. Tab. 1 provides statistical information about it.

Table 1: Statistics of anonymized pre-training dataset.

Statistics	Pre-training dataset
# users	1,947,683
# apps	83,341
avg. # gameapp operation behaviors	144.93
avg. # appstore click behaviors	62.21
avg. # appstore search behaviors	22.60
avg. # gamecenter click behaviors	3.84
avg. # gamecenter search behaviors	1.12

4.1.2 Downstream Datasets. We selected six popular games, each of which represents a downstream task. For each downstream task, our goal is to predict whether a user will play this game. For training sets, sampling date is from 2023/07/20 to 2023/07/25. During this period, We sampled the downstream dataset of one game every day. Positive samples include users who played the specific game the day after sampling date, while negative samples include users who didn't play the game the day after sampling date. The positive-to-negative sample ratio is 1 : 2. The test sets were sampled from 2023/07/21 to 2023/07/26. The test set sampling date of each game is the day after its training set sampling date. Labels for test sets were determined by checking whether user played that game on the day following the sampling date.

More information about downstream scenarios and datasets will be introduced in detail in Section 4.3.

4.1.3 Details of Datasets.

(1) Our datasets include more than 80000 different GameApps. We numbered these Apps consecutively as 1, 2, 3, and so on based on their appearance frequency in the training set. To deduce the dimension of classification, we retained the top 5000 App IDs, while any IDs beyond 5000 were uniformly set to 5001.

(2) Considering that user's searching keywords are randomly input by users, we compared the search behavior keywords in Appstore and Gamecenter with App names to match each search behavior to a specific App. If the user's search keyword does not contain the name of a specific App, we discard that behavior. Approximately, 80% of search behaviors matched App names.

4.2 Hyper-parameters and Pre-training Implementation Details

Our pre-training model is implemented with PyTorch [21] under Linux environment. Adam [14] is selected as the optimizer. Considering the results of Optuna optimization and previous work[1, 40], we set the embedding dimension d as 64 and the dimension of user representation d_u as 384 respectively. For the transformer encoder, the number of Transformer heads and layers are both set as 2. The batch size, the learning rate, the maximum length of each sequence l and λ in Eq (20) are set to 512, $1e-4$, 256 and 0.5 respectively. We divide the training set and validation set for pre-training task in a 9 : 1 radio.

During the pre-training process, there might be cases where sequence prediction errors lead to predicted subsequent tasks that are not exactly the same as the actual subsequent tasks. For such cases, we only calculate the loss for matched tasks. Other prediction results do not contribute to the loss calculation.

4.3 Downstream Evaluation Tasks

We set up three different downstream scenarios, consisting of six downstream tasks $\mathcal{T}_1 \sim \mathcal{T}_6$. Tab. 2 records the statistical information for all downstream task datasets.

4.3.1 Recommend old games to new users. Old games are those that have been released for at least one month. This scenario refers to predicting whether a user who has never played this old game before will play this game. We select three popular old mobile games to test the effectiveness of our proposed pre-training method: *Mini World* ($\mathcal{T}_1$), *Eggy Party* ($\mathcal{T}_2$) and *Sky: Children of Light* ($\mathcal{T}_3$).

4.3.2 Recommend old games to old users. Old users refer to those who have played this game in the past six months, but have not played this game in the past two weeks. This scenario refers to predicting whether a old user will play this game again. We select a very famous game *Honor of Kings* ($\mathcal{T}_4$) to test the effectiveness of our method.

4.3.3 Recommend newly released game to new users. This scenario refers to predicting whether a user will play the newly released game. We select two popular recently released mobile games to test the effectiveness of our proposed pre-training method: *Crystal of Atlan* ($\mathcal{T}_5$) and *Journey to West: Burning Soul* ($\mathcal{T}_6$).

Table 2: Statistics of anonymized downstream datasets.

	# Training Set	# Test Set	# Test Positive Rate
$\mathcal{T}_1$	236,609	4,995,000	0.018%
$\mathcal{T}_2$	766.978	4,995,000	0.087%
$\mathcal{T}_3$	72,923	4,995,000	0.003%
$\mathcal{T}_4$	1,950,447	2,850,000	0.579%
$\mathcal{T}_5$	675,033	4,985,000	0.219%
$\mathcal{T}_6$	364,235	4,973,234	0.024%

4.4 Evaluation Metrics

In our experiments, two widely used metrics are selected ,namely, **AUC** and **R@P**$_N$, to measure the performance of our proposed pre-training method. The first metric AUC is defined as the area enclosed by the coordinate axis under ROC curve. Compared to the ROC curve, we can more directly distinguish which method performs better with AUC. The second metric R@P$_N$ indicates the **R**ecall rate when the **P**recision rate equals to N. Since we may target users with strong conversion intentions by delivering a large number of ads and messages, which incurs a substantial expense, a high precision is generally required. In our experiment, we set $N = 0.95$. The higher AUC and R@P$_N$ indicate the stronger risk discrimination ability of the model.

4.5 Baselines

In order to comprehensively evaluate the performance of our model, we compare it with representative generative pre-training methods and discriminative pre-training methods.

Table 3: Overall performance of various pre-training tasks on downstream tasks. The best results are bold.

Pre-training Method	$\mathcal{T}_1$		$\mathcal{T}_2$		$\mathcal{T}_3$		$\mathcal{T}_4$		$\mathcal{T}_5$		$\mathcal{T}_6$	
	AUC	R@P$_{0.95}$	AUC	R@P$_{0.95}$	AUC	R@P$_{0.95}$	AUC	R@P$_{0.95}$	AUC	R@P$_{0.95}$	AUC	R@P$_{0.95}$
None	0.7920	0.3164	0.7543	0.2382	0.8553	0.4268	0.8008	0.2569	0.8906	0.4698	0.9830	0.8546
Bert4Rec	0.8370	0.3579	0.8109	0.3836	0.8779	0.5786	0.8546	0.3485	0.8819	0.5054	**0.9917**	0.9034
PeterRec	0.8258	0.4681	0.8113	0.3376	0.8697	0.5792	0.8590	0.3607	**0.9049**	0.5103	0.9829	0.9201
PTUM	0.8135	0.4267	0.7926	0.3415	0.8676	0.5244	0.8263	0.2969	0.8911	0.4729	0.9822	0.8710
CLUE	0.8198	0.4476	0.8036	0.3791	0.8689	0.5183	0.8347	0.3126	0.8919	0.4793	0.9833	0.8839
CCL	0.8215	0.4563	0.8097	0.3814	0.8721	0.5253	0.8406	0.3263	0.8921	0.4907	0.9835	0.8864
AdaptSSR	0.8247	0.3969	0.8017	0.3152	0.8866	0.5548	0.8525	0.3336	0.8509	0.4252	0.9843	0.9190
CM-PTM	**0.8542**	**0.4829**	**0.8411**	**0.4091**	**0.8976**	**0.5915**	**0.8724**	**0.3817**	0.8956	**0.5176**	0.9859	**0.9226**

(a) Generative User Representation Pre-training Methods

- **PeterRec** [42] applies masked behavior prediction to the behavior sequence.
- **PTUM** [33] applies masked behavior prediction and next K behavior prediction to the behavior sequence.
- **Bert4Rec** [29] applies bidirectional self-attentive and a cloze task to model the behavior sequence.

(b) Discriminative User Representation Pre-training Methods

- **CLUE** [4] uses implicitly augmented views for contrastive pre-training.
- **CCL** [1] proposes a mask-and-fill data augmentation strategy, using Masked Language Modeling (MLM) task to obtain the augmented view, and uses contrastive learning to train the model.
- **AdaptSSR** [40] proposes an adaptive data augmentation self-supervised ranking method to model user representations.

4.6 Overall Performance of Downstream Tasks

The experimental results on various downstream tasks are presented in Tab. 3. From these results, we have several findings. First, the effects of generative methods are generally better than those of discriminative pre-training methods. This is because commonly used data augmentation methods may change some behaviors, and if these behaviors are key cross-sequence behaviors, their models will have inaccurate understanding of user behaviors. Second, both metrics of our CM-PTM are ahead of the current state-of-the-art models in four old game recommendation scenarios, AUC increased by 1.72%, 2.98%, 1.1% and 1.34% respectively. In new games recommendation scenarios, our CM-PTM is slightly inferior to one of other methods in AUC, but the R@P$_{0.95}$ still remains ahead. The experimental results demonstrate that our cross-sequence cascaded multi-task architecture enhances the performance of pre-training models in the vast majority of downstream tasks.

4.7 Effects of Major Hyper-parameter

In Eq (20), we set a hyper-parameter λ to measure the importance of sequence prediction task and other tasks. We select four representative downstream tasks from three different scenarios to investigate the impact of different values for λ on our experimental results. We set several different values of λ: 0.2, 0.25, 0.33, 0.5, 0.67, 0.75 and 0.8. Fig. 3 shows our experimental results. It is obvious that when recommending games to new users, whether it is an old game or a new game, setting $\lambda = 1$ is the best choice. However, when recommending games to old users, it may be necessary to increase the weight of the sequence prediction task appropriately to achieve better results. We speculate that it may be because if the user has already played this game, it indicates that the user has more or less potential interest in this type of game, so he or she may be more likely to learn about similar games in Appstore or Gamecenter than new users, and there will be more cross-sequence behaviors such as downloading $\rightarrow$ launching. Therefore, the sequence prediction task for old users will be more important.

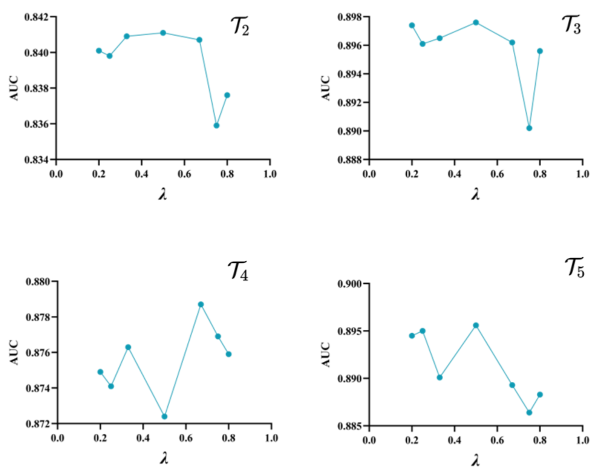

Figure 3: The impact of different hyper-parameter λ on AUC of various downstream tasks.

4.8 Effects of Different Pre-training Tasks

We experiment with several pre-training tasks and test their effectiveness on downstream task $\mathcal{T}_2$. We set dropout radio to 0.1, $K = 2, 3, 4$ and record the best result as the result of Next K item . Tab. 4 illustrates our test results, the experiment proves that next item prediction is the most effective pre-training task. We speculate that the accumulation of losses in the sequence prediction task results in the inferior performance of Next K item prediction compared to Next item prediction and the dropout method may lose some important cross-sequence behaviors, leading to a decrease in the performance of the model.

Table 4: The effect of different pre-training tasks.

Pre-training Task	AUC	R@P$_{0.95}$
None	0.7543	0.2382
Next K item	0.8367	0.3663
Next item	**0.8411**	**0.4091**
Masked item	0.8275	0.3523
Next K item + Dropout	0.8392	0.3815
Next item + Dropout	0.8398	0.3927

4.9 Ablation Study

We conduct a detailed ablation study to validate the effectiveness of each component in our method, i.e., sequence prediction, App category prediction, App prediction, operation prediction, multi-source behavior data and cascaded proxy tasks. We pre-train the user representation with our CM-PTM and its variant with one component removed. For experiment *w/o* data, we try the behaviors of using only third-party GameApps and the behaviors of only pre-installed Apps on mobile terminal, and choose the best result. For experiment *w/o* cascaded, we make all proxy tasks parallel.

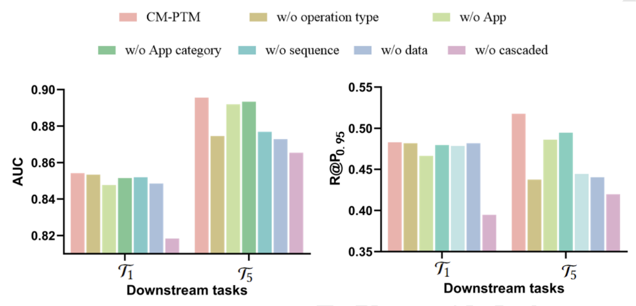

Figure 4: The AUC and R@P$_{0.95}$ performance with different components removed in our CM-PTM.

The results of the downstream tasks $\mathcal{T}_1$ and $\mathcal{T}_5$ are shown in Fig. 4. It shows that our cross-sequence task has increased the AUC of two downstream tasks by 0.22% and 1.87% respectively and increased the R@P$_{0.95}$ of these two downstream tasks by 0.44% and 7.32% respectively. Moreover, excluding any proxy task will lead to a decrease in both performance metrics of the model. This confirms the rationality of our cross-sequence multi-task cascade architecture.

4.10 Case Study

By tracing downstream prediction results and corresponding user behaviors, we discovered some interesting cases in our downstream scenarios, which demonstrate the effectiveness of our approach.

User A's third-party GameApp behavior sequence shows that his or her recent 87 game-related behaviors only include *Minecraft*, *Genshin Impact* and *Mini World*. These games are all sandbox games, which offer players a large degree of freedom in exploring, interacting with, and manipulating the game world. However, this user has clicked on *Honor of Kings* twice in Appstore in the past six months. The downstream result shows that this user has played *Honor of Kings* again. This example shows that joint modeling of multi-source sequences can comprehensively capture users' interests.

Additionally, from the historical behavior of user B, we found that this user once performed such a series of behaviors on *Crystal of Atlan* → click in Appstore → download → launch,..., launch → exit → Appstore search → Gamecenter search. When *Journey to West: Burning Soul* was released, which was similar to *Crystal of Atlan*, our downstream result shows that this user played it. This example shows that cross-sequence modeling of multi-source sequences can better capture users' interests.

4.11 Online Simulation

Finally, an online simulation is conducted. We randomly sample nearly one million users over entire two-week period (ranging from 2024/01/02 to 2024/01/16). Then We use streaming data to simulate online experiment to compare our proposed CM-PTM against DeepFM [9], which is maturely run in the real system. Tab. 5 shows our experiment results. Compared with the original model DeepFM, our pre-training model CM-PTM achieve great performance.

Table 5: Online simulation results.

Methods	AUC	R@P$_{0.95}$
DeepFM	0.8064	0.3549
DeepFM + CM-PTM	0.8132	0.3657
Improvement	0.68%	1.08%

Based on the online simulation results, we estimated the improvement benefits that our model can bring in each downstream task, as shown in Tab. 6.

Table 6: Estimated benefits that CM-PTM can bring. For $\mathcal{T}_4$, we estimate the number of payments, since we can't obtain the specific payment amount.

	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$	$\mathcal{T}_5$	$\mathcal{T}_6$
Benefits per recalled user	¥17.38	¥70.60	¥119.39	2.38	¥14.08	¥13.92

5 CONCLUSION

In this paper, we proposed CM-PTM to generate comprehensive user representations, designing multiple cascaded proxy tasks. We pre-trained our CM-PTM with large-scale real-world dataset and validated the effectiveness of our method on six different downstream tasks. The experimental results exhibit that our CM-PTM is superior to existing methods.

REFERENCES

[1] Shuqing Bian, Wayne Xin Zhao, Kun Zhou, Jing Cai, Yancheng He, Cunxiang Yin, and Ji-Rong Wen. 2021. Contrastive curriculum learning for sequential user behavior modeling via data augmentation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3737–3746.

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems* 33 (2020), 9912–9924.

[3] Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 15750–15758.

[4] Mingyue Cheng, Fajie Yuan, Qi Liu, Xin Xin, and Enhong Chen. 2021. Learning transferable user representations with sequential behaviors via contrastive pre-training. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 51–60.

[5] Junsu Cho, Dongmin Hyun, Dong won Lim, Hyeon jae Cheon, Hyoung-iel Park, and Hwanjo Yu. 2023. Dynamic multi-behavior sequence modeling for next item recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 4199–4207.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Chilin Fu, Weichang Wu, Xiaolu Zhang, Jun Hu, Jing Wang, and Jun Zhou. 2023. Robust User Behavioral Sequence Representation via Multi-scale Stochastic Distribution Prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 4567–4573.

[8] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821* (2021).

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[10] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 191–200.

[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[12] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 659–668.

[13] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

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

[15] Wangli Lin, Li Sun, Qiwei Zhong, Can Liu, Jinghua Feng, Xiang Ao, and Hao Yang. 2021. Online credit payment fraud detection via structure-aware hierarchical recurrent neural network.. In *IJCAI*. 3670–3676.

[16] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[17] Can Liu, Yuncong Gao, Li Sun, Jinghua Feng, Hao Yang, and Xiang Ao. 2022. User Behavior Pre-training for Online Fraud Detection. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3357–3365.

[18] Hu Liu, Jing Lu, Xiwei Zhao, Sulong Xu, Hao Peng, Yutong Liu, Zehua Zhang, Jian Li, Junsheng Jin, Yongjun Bao, et al. 2020. Kalman filtering attention for user behavior modeling in ctr prediction. *Advances in Neural Information Processing Systems* 33 (2020), 9228–9238.

[19] Zhiwei Liu, Yongjun Chen, Jia Li, Philip S Yu, Julian McAuley, and Caiming Xiong. 2021. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv preprint arXiv:2108.06479* (2021).

[20] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 596–605.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[22] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. arXiv preprint. *arXiv preprint arXiv:1802.05365* (2018).

[23] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. 2022. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*. 813–823.

[24] Zhaopeng Qiu, Xian Wu, Jingyue Gao, and Wei Fan. 2021. U-BERT: Pre-training user representations for improved recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4320–4327.

[25] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[26] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[28] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, 9 (2005).

[29] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[30] Qinghui Sun, Jie Gu, XiaoXiao Xu, Renjun Xu, Ke Liu, Bei Yang, Hong Liu, and Huan Xu. 2022. Learning Interest-oriented Universal User Representation via Self-supervision. In *Proceedings of the 30th ACM International Conference on Multimedia*. 7270–7278.

[31] Jie Tang, Limin Yao, Duo Zhang, and Jing Zhang. 2010. A combination approach to web user profiling. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 5, 1 (2010), 1–44.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[33] Chuhan Wu, Fangzhao Wu, Tao Qi, Jianxun Lian, Yongfeng Huang, and Xing Xie. 2020. Ptum: Pre-training user model from unlabeled user behaviors via self-supervision. *arXiv preprint arXiv:2010.01494* (2020).

[34] Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. 2021. R-drop: Regularized dropout for neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 10890–10905.

[35] Yiqing Wu, Ruobing Xie, Yongchun Zhu, Xiang Ao, Xin Chen, Xu Zhang, Fuzhen Zhuang, Leyu Lin, and Qing He. 2022. Multi-view multi-behavior contrastive learning in recommendation. In *International Conference on Database Systems for Advanced Applications*. Springer, 166–182.

[36] Dongbo Xi, Zhen Chen, Peng Yan, Yinger Zhang, Yongchun Zhu, Fuzhen Zhuang, and Yu Chen. 2021. Modeling the sequential dependence among audience multi-step conversions with multi-task learning in targeted display advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3745–3755.

[37] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Xiyue Zhang, Hongsheng Yang, Jian Pei, and Liefeng Bo. 2021. Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4486–4493.

[38] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 1259–1273.

[39] Hongrui Xuan, Yi Liu, Bohan Li, and Hongzhi Yin. 2023. Knowledge Enhancement for Contrastive Multi-Behavior Recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 195–203.

[40] Yang Yu, Qi Liu, Kai Zhang, Yuren Zhang, Chao Song, Min Hou, Yuqing Yuan, Zhihao Ye, Zaixi Zhang, and Sanshi Lei Yu. 2023. AdaptSSR: Pre-training User Model with Augmentation-Adaptive Self-Supervised Ranking. *arXiv preprint arXiv:2310.09706* (2023).

[41] 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 *Proceedings of The Web Conference 2020*. 303–313.

[42] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 1469–1478.

[43] Lei Zheng, Chaozhuo Li, Chun-Ta Lu, Jiawei Zhang, and Philip S Yu. 2019. Deep distribution network: Addressing the data sparsity issue for top-n recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1081–1084.

[44] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. Atrank: An attention-based user behavior modeling framework for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[45] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference*

on knowledge discovery & data mining. 1059–1068.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009