Pre-training General User Representation with Multi-type APP Behaviors

Yuren Zhang[1] , **Min Hou**[2 *] , **Kai Zhang**[1 *] , **Yuqing Yuan**[3] , **Chao Song**[3] , **Zhihao Ye**[3] , **Enhong Chen**[1] , **Yang Yu**[1]
[1]State Key Laboratory of Cognitive Intelligence,
University of Science and Technology of China
[2]Hefei University of Technology
[3]OPPO Research Institute
{yurenz, yflyl613}@mail.ustc.edu.cn, hmhoumin@gmail.com, {kkzhang08, cheneh}@ustc.edu.cn,
{yuanyuqing, songchao12, yezhihao3 }@oppo.com

Abstract

In numerous user-centric services on mobile applications (apps), accurately mining user interests and generating effective user representations are paramount. Traditional approaches, which often involve training task-specific user representations, are becoming increasingly impractical due to their high computational costs and limited adaptability.

This paper introduces a novel solution to this challenge: the Multi-type App-usage Fusion Network (MAFN). MAFN innovatively pre-trains universal user representations, leveraging multi-type app behaviors to overcome key limitations in existing methods. We address two primary challenges: 1) the varying frequency of user behaviors (ranging from low-frequency actions like (un)installations to high-frequency yet insightful app launches); and 2) the integration of multi-type behaviors to form a cohesive representation. Our approach involves the creation of novel pre-training tasks that harness self-supervised signals from diverse app behaviors, capturing both long-term and short-term user interests. MAFN's unique fusion approach effectively amalgamates these interests into a unified vector space, facilitating the development of a versatile, general-purpose user representation. With a practical workflow, extensive experiments with three typical downstream tasks on real-world datasets verify the effectiveness of our approach.

1 Introduction

Recent years have witnessed the rapid development of mobile technology, leading to the widespread integration of mobile apps into people's daily lives, including online shopping, daily commuting, and entertainment [Radosavljevic *et al.*, 2016]. The fast-growing user behaviors on mobile apps (e.g., launch, (un)install, retention) encompasses valuable insights into user interests [Lu *et al.*, 2014]. This phenomenon presents both opportunities and challenges for user modeling, which plays a pivotal role in providing user-centric mobile services. Traditional methods that learn task-specific user embeddings using the end-to-end paradigm face practical challenges, as they often require substantial expert efforts and are hard to generalize to other scenarios [Zhao *et al.*, 2016; Zhang *et al.*, 2020]. Therefore, a crucial demand asks to develop an approach to mine app usage data and generate general-purpose user representations that can support various tasks. To this end, our study focuses on pre-training universal user representations based on multi-type app behaviors.

Towards this goal, we identify the following two primary challenges. **Firstly**, the diverse frequency of user behavior necessitates the design of appropriate pre-training tasks. Existing studies mainly concentrate on a single type of user behavior or behaviors with similar frequencies, such as *purchase* and *cart* [Cheng *et al.*, 2021; Gu *et al.*, 2021], and commonly adopt a unified modeling approach [Jin *et al.*, 2020; Xu *et al.*, 2023]. However, user behaviors in app scenarios exhibit substantial frequency differences. While app launches occur frequently throughout the day, (un)installation behaviors typically take place over longer periods, spanning weeks or even longer. To achieve a comprehensive understanding of mobile users, it is essential and challenging to explore tailored pre-training tasks for collaborative modeling these multi-type app behaviors with varying frequencies. **Secondly**, further exploration is required to develop an effective approach that can integrate users' multi-aspect interests in an adaptive manner during the pre-training stage. Traditional vector fusion methods may yield sub-optimal performance due to the distinct semantic spaces of users' long- and short-term preferences. Consequently, finding a practical solution for user interests fusion to form a cohesive user representation remains a substantial challenge.

To address these challenges, we propose the Multi-type App-usage Fusion Network (MAFN) for learning general user representation. Specifically, in light of the rapid evolution observed in app **launch** behaviors [Li *et al.*, 2020], we first design the Short-term Interest Module (SIM) to capture users' recent preferences related to high-frequency app launches. Then, we introduce the Long-term Interest Module (LIM) to extract users' long-term demands based on the less frequent but more informative app **(un)installation** and **retention** (which apps are currently installed on the phone).

*Min Hou and Kai Zhang are corresponding authors.

To effectively guide the learning of each module, we also propose several customized self-supervised pre-training tasks that align with the data characteristics. Furthermore, we devise the Interest Fusion Module (IFM) in conjunction with a contrastive learning task to integrate the multi-aspects interests at the user level in a unified vector space. To validate the effectiveness of our methods, we conduct the experiments with three downstream tasks on real-world datasets from a worldwide leading smartphone manufacturer. Experimental results demonstrate the effectiveness of our approach.

Our contributions can be summarized as three-fold:

- To address an important practical issue, we propose a novel framework named MAFN for collectively modeling multi-type app behaviors and extracting general user representation. To the best of our knowledge, this is the pioneering effort to pre-train universal user model based on multi-type app behaviors with varying frequencies.
- We design tailored modules and pre-training tasks to effectively capture users' multi-aspect interests. Our modeling ideas can inspire research on other user behaviors.
- We extensively evaluate our approach through experiments on three real-world tasks and compare it with competitive models, indicating its effectiveness.

2 Related Work

2.1 User Representation Learning

User representation learning aims to obtain informative vectors that can effectively represent users. Existing studies mainly focused on extracting features from single-type interaction data, and leveraging statistical and deep learning methods for this purpose [Yuan *et al.*, 2020; Zhang *et al.*, 2020; Wu *et al.*, 2020; Gu *et al.*, 2021]. For example, Wu *et al.* proposed two self-supervision tasks for task-specific user representation learning, which took the relatedness between past and future behaviors into account. Gu *et al.* considered behavioral consistency and proposed a self-supervised model to learn universal user embeddings. Recent efforts have also attempted to integrate multi-domain interaction for alleviating data sparsity [Qiu *et al.*, 2021; Mu *et al.*, 2022]. For example, Qiu *et al.* proposed the customized U-BERT to pre-train user representation by leveraging review interaction to bridge the gaps among different domains. Though related to these studies, our focus is on mining user interests from multi-type app behaviors with varying frequencies to benefit multiple real-world tasks. Existing methods are not directly applicable.

2.2 Multi-Behavior Learning

Multi-behavior Learning aims to leverage users multi-type behaviors to enhance the accuracy of predicting their target behavior, and has been widely explored and verified in practice [Wu *et al.*, 2022; Zhang *et al.*, 2023]. Scholars have proposed various models via deep learning techniques. For example, Zhou *et al.* designed a self-attention based framework to model feature interactions. Later on, GNNs are utilized to explore multi-hop user-item interactions. For example, MRIG [Wang *et al.*, 2020] and MBGCN [Jin *et al.*, 2020] learned distinct user interests using user-item interaction graphs, while Xia *et al.* followed a graph meta-learning paradigm to distill the behavior diversity for recommendations. To fight against the data sparsity and cold-start issues, Wu *et al.* and Xu *et al.* introduced self-supervised Contrastive Learning tasks into Multi-behavior Learning. Despite the existing progress, most previous studies are task-oriented, which constrains their practical value in serving various applications. In this paper, we focus on developing a user model for general use based on multi-type app usage.

2.3 Analysis of Mobile App Data

As mobile apps have become indispensable in our lives, a quantity of studies has been devoted to analyzing the app usage pattern. Several statistical-based works have illustrated the rich information contained in app usage[Li *et al.*, 2016; Tu *et al.*, 2018; Peltonen *et al.*, 2018; Li *et al.*, 2020; Li *et al.*, 2022; Yu *et al.*, 2020]. Recently, advanced efforts have attempted to build general user models based on users' app behaviors. For example, Zhang *et al.* extracted general-purpose user embeddings by a tailored Transformer-based AETN. Bian *et al.* learned user representation by the proposed macro-micro fusion network, which considers the semantic relevance of users' interaction both within and outside of apps. Liu *et al.* proposed DAUC to obtain task-optimized user clusters by leveraging app usage to transfer knowledge from active users to cold-start users. However, these works mainly focused on low-frequency app usage and overlooked the significance of high-frequency ones. In this paper, we analyse multi-type app behaviors with different frequencies for a more comprehensive understanding of user interests.

3 Problem Formulation

For learning general user representations, we consider three types of user-app interactions that are closely related to users' daily lives, namely **launch**, **(un)installation**, and **retention**. Formally, we define the set of users and apps as $\mathcal{U}$ ($u \in \mathcal{U}$) and $\mathcal{A}$ ($a \in \mathcal{A}$) respectively, where $|\mathcal{U}|$ and $|\mathcal{A}|$ represent the number of users and apps. The app behavior sequence of a user u can be presented as $\mathcal{S}_u = \{a_1, a_2, ..., a_n\}$. For each app behavior a_i, we also introduce its corresponding app category c_i and timestamp t_i to provide better characterization. To distinguish different types of behaviors, we utilize unique superscripts: a^{la} for launch, a^{in} for installation, a^{un} for uninstallation and a^{re} for retention. Therefore, our user model acts as a function $\mathcal{F}_\theta$ which takes these behavioral sequences as input and outputs low-dimensional user embeddings e_u as:

$$e_u = \mathcal{F}_\theta \left(S_u^{la}, S_u^{in}, S_u^{un}, S_u^{re} \right). \tag{1}$$

For simplification, we may omit the subscript u indicating a user in the rest of the paper when there is no confusion.

4 Methodology

This section introduces our MAFN framework for general user representation learning. As illustrated in Figure 1, MAFN consists of two phases. In the pre-training phase, we design customized self-supervised learning objectives to obtain universal user representations from three types of app

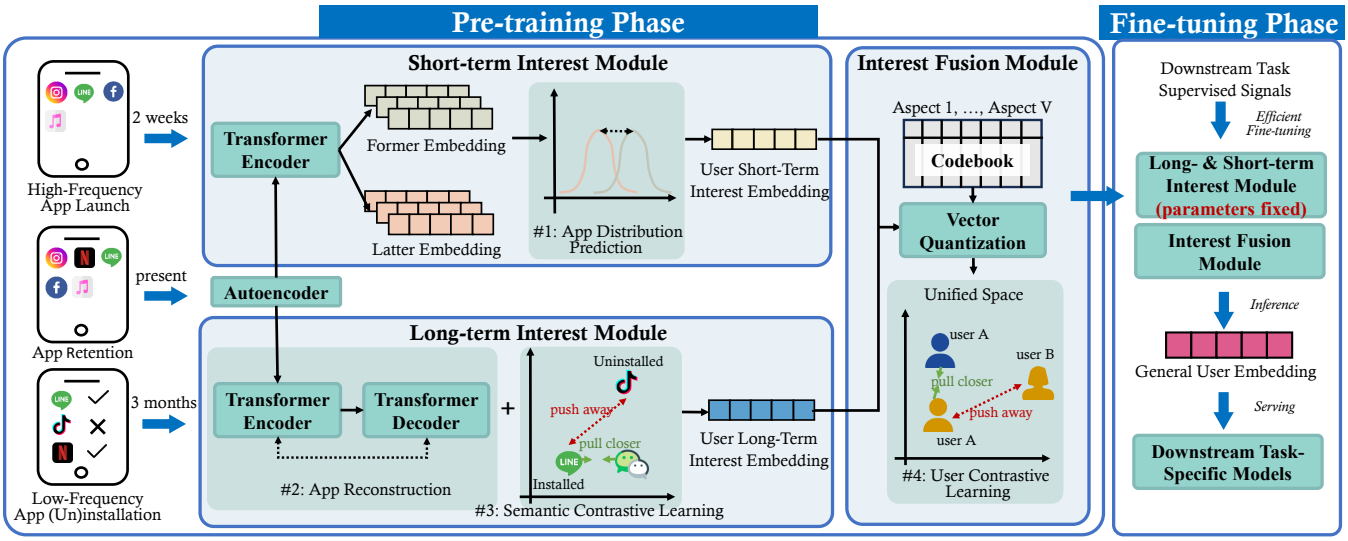


Figure 1: The two-phases architecture of our proposed MAFN. The framework is designed for generating general-purpose user representation based on multi-type mobile app behaviors, and serving multiple downstream mobile tasks. In the Pre-training phase, the Short- and Long-term Interest Modules leverage types of high- and low-frequency app behaviors to learn users' recent preferences and stable demands, respectively. The retention AutoEncoder models the apps' co-occurrence relations, and the Interest Fusion Module projects two aspects of users' interests embeddings onto unified vector space and dynamically combines them. In the Fine-tuning phase, MAFN performs efficient fine-tuning with downstream task supervised signals and generate high-quality user representations for specific mobile service.

behaviors with varying frequencies. Specifically, we design three key components to gain a deeper understanding of user intents. 1) Short-term Interest Module (SIM) models high-frequency app launch behaviors to capture users' recent preferences. 2) Long-term Interest Module (LIM) extracts users' stable demands from low-frequency (un)installation and retention behaviors. 3) Interest Fusion Module (IFM) combines users' multi-aspect interests dynamically. In the fine-tuning phase, user representations are adapted to arbitrary downstream tasks. In the following part, we first introduce three components of the pre-training phase in detail. After that, we elaborate on our two-stage schema.

4.1 Short-term Interest Module

App launch behavior is quite frequent in users' lives and indicates their recent intents. For example, if a user installed *Call of Duty Mobile* and opened it frequently in the last week, the conclusion that he/she recently likes FPS games is convincing. However, if he/she has only installed the game but rarely opened it, this conclusion appears unreliable. Motivated by this, we propose the SIM to capture users' short-term interests from high-frequency app launches. In the following subsection, we provide a detailed explanation of the architecture and the customized pre-training task employed in the SIM.

High-Frequency App Launch Modeling

Inspired by the Transformer architecture that succeeds in modeling temporal interaction data [Devlin *et al.*, 2018; Mu *et al.*, 2022; Seol *et al.*, 2022], we develop a BERT-based module to encode app launch behavior. Specifically, we first utilize the shared app embedding layer to encode each app a_i in user launch records into a dense vector $e_i^{la} = E(a_i^{la})$. We also incorporate category embedding c_i^{la} and timestamp embedding t_i^{la}, which can alleviate the gravely sparse of long-tailed apps and provide helpful time information, i.e., $e_i^{la} = e_i^{la} + c_i^{la} + t_i^{la}$. Since app retention serves as an intuitive indicator of a user's current habits, we concatenate the app launch embedding e_i^{la} with the app retention embedding e^{re}, which is generated from an AutoEncoder. Finally, we utilize multi-head self-attention $MHA(\cdot)$ and mean-pooling $MP(\cdot)$ operations to incorporate the representations of each behavior into the short-term interest embedding e^{short} as:

$$e^{short} = MP(FFN(MHA(\{e_1^{la}, e_2^{la}, ..., e_n^{la}; e^{re}\}))), \quad (2)$$

where $FFN(\cdot)$ represents a feed-forward network.

Pre-training Task #1: App Distribution Prediction

Some classical self-supervised learning tasks have succeeded in modeling sequential data, such as *mask word prediction* and its variants [Sun *et al.*, 2019]. However, app launch records often exhibit repetitions and noise due to users frequently launching the same apps and occasionally launching irrelevant ones. In this case, common pre-training tasks that aim to predict specific launched apps may not be appropriate, as the specific behaviors and their order may not carry meaningful information. Instead, focusing on capturing the overall statistics of the behaviors can be more beneficial. Therefore, we design the *app distribution prediction* pre-training task to accurately guide model learning, which aims to predict the occurrence of each app in the user's subsequent records.

Specifically, for user u with launch behavior records $\{a_1^{la}, a_2^{la}, ..., a_N^{la}\}$, we force the module to use his preceding N behaviors to predict the distribution of app occurrences in the subsequent K behaviors, *i.e.*, the number of times each app appears in $\{a_{N+1}^{la}, a_{N+2}^{la}, ..., a_{N+K}^{la}\}$. Given user's short-term interests embedding e^{short} generated from past N be-

haviors, the predicted app distribution can be presented as:

$$\hat{p}(a_i|u) = \frac{exp\left(e_i^T f_\phi(e^{short})\right)}{\sum_{a_j \in S'} exp\left(e_j^T f_\phi(e^{short})\right)}, \quad (3)$$

where $f_\phi(\cdot)$ is a fully connected layer with parameters ϕ. e_i^T is the transpose of embedding e_i^{la}, and i is the i^{th} app that occurs in the following sequence $S^{'}$. Then, we set the normalized occurrences of apps as the ground-truth, *i.e.*,

$$p(a_i|u) = \frac{\log\left(1 + \textit{count}(a_i)\right)}{\sum_{a_j \in S'} \log\left(1 + \textit{count}(a_j)\right)}, \quad (4)$$

where $count(a_i)$ is the number of occurrence of the app a_i in sequence $S^{'}$. With $\hat{p}(a_i|u)$ and $p(a_i|u)$, we can use the Kullback-Leibler divergence to measure the differences and define the pre-training loss function as:

$$\mathcal{L}_{dist} = D_{KL(p||\hat{p})} \propto -\sum p(a_i|u) \log \hat{p}(a_i|u). \quad (5)$$

4.2 Long-term Interest Module

In contrast to app launches, app (un)installation and retention behaviors occur less frequently but provide more extensive information. Hence, we design the LIM with a classical encoder-decoder structure to capture users' long-term demands. Considering the unique semantic relationship between app installation and uninstallation, we further introduce two customized self-supervised learning tasks to guide the user representation learning process.

Low-Frequency App (Un)installation Modeling

The users' app (un)installing behaviors often occur over weeks, characterized by low-frequency and uneven distribution over time. To better model users' long-term and cross-temporal interest dependencies, we employ a Transformer-based encoder for collectively modeling these behaviors. Specifically, we first encode app (un)installing behaviors via the shared app embedding layer $E(\cdot)$. Then, we introduce the type embeddings d to distinguish the installation and uninstallation, e.g., $e_i^{in} = E(a_i^{in}) + d^{in}$. Thereafter, we incorporate time embedding t_i, aiming to mitigate the impact of the uneven distribution of (un)installations. Finally, we leverage a Transformer encoder $\textit{TE}(\cdot)$ to accomplish the information aggregation and generate the user's long-term preference embedding e^{long}, denoted as:

$$e^{long} = \textit{TE}\left(\{e_1^{in}, ..., e_n^{in}; e_1^{un}, ..., e_m^{un}; e^{re}\}\right). \quad (6)$$

Pre-training Task #2: App Reconstruction

Motivated by the success in the pre-training language model [Devlin *et al.*, 2018], we initially introduce the *app reconstruction* task that aids the model in obtaining a comprehensive understanding of the behavioral sequences by reconstructing the original app sequence. We aim to formulate self-supervised learning objectives based on the consistency of user behavior. Formally, we leverage a Transformer decoder to reconstruct the (un)installation sequences with a softmax cross-entropy loss function as:

$$\mathcal{L}_{re} = -\frac{1}{|S|} \sum_{a_n \in S} \log P(a_n = a_n^*|S^n), \quad (7)$$

where S^n is the raw app (un)installation sequence, S is the predicted sequence, a_n^* is the ground-true app for each predicted app a_n, $P(a) = \textit{softmax}(f_\psi(e^{long}))$ is the predicted probability, and $f_\psi(\cdot)$ is a fully connected layer.

Pre-training Task #3: Semantic Contrastive Learning

There is a natural negative correlation between the app installation and uninstallation. Intuitively, users' app installation often reflects their preferences and needs, while uninstallation indicates their dislike or dissatisfaction. Such semantic correlation is highly valuable in improving the model's understanding of users [Zhang *et al.*, 2022]. Hence, we design a novel *Semantic contrastive learning* task to effectively leverage this correlation.

Specifically, we consider obtaining user representations from three distinct app sets: the set of apps the user has installed, the set of apps the user has never interacted with, and the set of uninstalled apps. We note those representations as u^{in}, u^{no} and u^{un}, respectively. Naturally, u^{in} are expected to be closer to u^{no} and be further away from u^{un}. Therefore, we regard (u^{in}, u^{no}) as the positive pair, considering them as different augmentations of user interests, and regard (u^{in}, u^{un}) as the in-batch negative pairs. Framed in the classical InfoNCE[Oord *et al.*, 2018] paradigm, we formulate the semantic loss function with temperature coefficient τ as:

$$\mathcal{L}_{sem} = -\log \frac{\exp\left(sim\left(u^{in}, u^{no}\right)/\tau\right)}{\sum \exp\left(sim\left(u^{in}, u^{un}\right)/\tau\right)}, \quad (8)$$

where $sim(\cdot)$ is dot-product for measuring similarity.

4.3 Interest Fusion Module

After obtaining the short- and long-term interest embeddings by SIM and LIM, how to fuse them properly is not trivial. Previous studies commonly adopt vector concatenation, weighted summation or gate network for fusion in similar contexts [Zhang *et al.*, 2020; Qiu *et al.*, 2021]. However, combining the two embeddings we generated becomes challenging due to their derivation from different modules, utilization of distinct data, and resulting in dissimilar representation spaces. To address this dilemma, we present a unified interests fusion module. Inspired by the Vector Quantization mechanism [Van Den Oord *et al.*, 2017; Liu *et al.*, 2021; Hou *et al.*, 2023], we create a discretized embedding space that is shared across short- and long-term interest embeddings. This discrete embedding space is capable of unifying different aspects of user interests through a finite number of embedding vectors, facilitating consistent representation fusion. Furthermore, we design a user-level contrastive learning task to enhance the fusion. In this subsection, we elaborate on the fusion approach and the training objective.

Unified Interests Fusion

Given the embeddings presenting users' multi-aspect interests, noted as e^I of interests $I \in \{long, short\}$, this module attempts to effectively fuse them to obtain general user representations. To address the disparity in vector spaces between the two types of embeddings derived from data with different frequencies, we intuitively project the short- and long-term interest embeddings into a shared discrete embedding

space. Specifically, we first construct a unified interests embedding table $E^c = \{e_1, e_2, ..., e_V\}$ of size V (named *codebook*), which represents V implicit types of users' interests. Then, we compute $v = \arg\min_{k \in V} ||e^I - e_k||_2$ to find the most relevant unified interests embedding $e_v \in E^c$ for each embedding e^I. After that, we project e^I into the unified embedding space by a Linear network f^I, and then combine it with e_v to obtain the transferred representation $\hat{e}^I$ as

$$\hat{e}^I = f^I(e^I) + sg(e_v - f^I(e^I)), \tag{9}$$

where $sg(\cdot)$ is the stop-gradient operator. In this way, the projected $\hat{e}^{long}$ and $\hat{e}^{short}$ are expected to lie in the same unified embedding space, containing both users' single-aspect interests and unified interests. Finally, we perform vector summation to generate final user representations as

$$e_u = \hat{e}^{long}_u + \hat{e}^{short}_u. \tag{10}$$

In the whole process, the *codebook* acts as a bridge, connecting user interests at different aspects within a shared discrete embedding space. We train the *codebook* jointly with the rest of our framework and fine-tune it with downstream tasks to achieve context-aware fusion of users' multi-aspect interests.

Pre-training Task #4: User Contrastive Learning

To learn a shared embedding space that is invariant to each input, we utilize a self-supervised contrastive learning task which encourages the model to focus more on the semantic aspect of the input. We maximize the mutual information between the high- and low-frequency app behavior sequences of the same user. For simplicity, we re-denote the representation of two kinds of sequences of the same user as $\boldsymbol{p}$ and $\boldsymbol{q}$, and then formulate the contrastive loss function as:

$$\mathcal{L}_{user} = -\mathop{\mathbb{E}}_{\mathcal{P}_l}\left[\log \frac{D_\omega(\boldsymbol{p}, \boldsymbol{q})}{D_\omega(\boldsymbol{p}, \boldsymbol{q}) + \sum_{\tilde{\boldsymbol{q}} \in \tilde{\mathbb{Q}}} D_\omega(\boldsymbol{p}, \tilde{\boldsymbol{q}})}\right], \tag{11}$$

where $\mathcal{P}_l$ represents the joint distribution of the two sequences, *i.e.*, $(\boldsymbol{p}, \boldsymbol{q}) \sim \mathcal{P}_l$. $\tilde{\boldsymbol{q}}$ denotes the negative sample randomly sampled from the different user's behavior sequence within one mini-batch. $D_\omega(\cdot, \cdot)$ is the discriminator parameterized by ω, we define it with parameter matrix W as:

$$D_\omega(\boldsymbol{p}, \boldsymbol{q}) = \exp\left(\boldsymbol{p}^\top \cdot \boldsymbol{W} \cdot \boldsymbol{q}\right). \tag{12}$$

The loss in Equation (11) is the categorical cross-entropy of classifying the positive sample correctly, which promotes the separation of samples from different latent classes, thereby improving the stability of the fused representation.

4.4 Training and Serving Scheme

Model Training. Our entire training procedure follows a classic two-stage paradigm, *i.e.*, pre-training and fine-tuning stages. At the pre-training stage, we first train the tailored modules to extract user interests via the well-designed pre-training tasks. Then, we train the adaptive fusion module to dynamically fuse different views of user intents. By integrating the loss functions, the overall objective is defined as:

$$\mathcal{L}_{pretrain} = \mathcal{L}_{dist} + \mathcal{L}_{re} + \mathcal{L}_{sem} + \mathcal{L}_{user}. \tag{13}$$

Statistics	Pre-training dataset
# of users	5,486,783
# of apps	10,404
avg. # of app launch behaviors	535
avg. # of app install behaviors	168
avg. # of app uninstall behaviors	174

Table 1: Statistics of the anonymized pre-training dataset.

Statistics	Ad	Attr	Player
# of users	2,771,647	4,878,698	360,658
# of apps	1,112	-	1
# of interactions	28,033,719	-	1,381

Table 2: Statistics of the three downstream datasets.

At the fine-tuning stage, we first initialize with the pre-trained parameters and then fine-tune the model with the supervised signals from downstream tasks. To achieve efficient fine-tuning, we keep most parameters fixed and only tune the fusion module, so as to learn context-aware representation.

Serving. In scenarios involving mobile apps, online serving time is often much more critical than offline training time. Hence, a practical way is to pre-train and fine-tune our model offline, store the generated user representations specific to downstream tasks, and provide these user embeddings to downstream models for online services.

Complexity Analysis. SIM and LIM utilize Transformer-based structures, resulting in a time complexity of $O(N^2)$, where N is the input sequence length. IFM involves clustering for user representations, resulting in a time complexity of $O(d \cdot V)$, where d is the embedding dimension. Commonly $N >> d \cdot V$, so the overall complexity is $O(N^2)$. In comparison to other sequential models, the complexity associated with our approach is competitive, and our practical workflow does not entail significant costs in downstream tasks.

5 Experiments

In this section, we report the extensive experimental results on real-world datasets and tasks, as well as the corresponding analysis and discussion of limitation.

5.1 Experimental Setup

Pre-training Dataset

We conduct experiments on industrial data from a worldwide leading smartphone manufacturer. We collect app data of users who agree to share their **anonymous** behavior data for improving service quality. More than 10 thousand popular apps and randomly sampled 5.5 million users are used for pre-training. We filter users and apps with fewer than five interactions, and collect app (un)installation actions over 3 months and app launch behaviors over 2 weeks. We randomly split out about 10% users for validation. The statistics of the processed data are summarized in Table 1. Note that our research process is tightly regulated for avoiding any disclosure of user privacy, so as to avoid possible *Ethical* issues.

Evaluation in Downstream Tasks

We evaluate our framework on three typical downstream applications, encompassing various commonly used downstream models. The tasks and settings are detailed as follows.

Task 1: Ad CVR Prediction. This task aims to predict the conversion rate (CVR), which is the probability that a user will become an active user after clicking on an advertisement. We consider it as a classification task, and train a two-layer MLP model to estimate the probability of conversion for each user based on the generated user embeddings.

Task 2: User Attribute Prediction. In this task, we try to predict the user gender based on the generated user embeddings. The dataset is constructed by selecting users with gender tags from the anonymized pre-training dataset. We train a DeepFM [Guo *et al.*, 2017] model for this task.

Task 3: Game Player Recall. This task aims to predict the chances of losing players returning to the game, so as to adopt appropriate strategies to recall game players. We regard it as a binary classification task, in which the returned players are positive cases and players who have not returned are negative cases. We train a GBDT-based [Friedman, 2001] model to predict the returning probability using the user embeddings and multiple statistical features.

Table 2 shows statistics of the downstream datasets.

Evaluation Settings

We combine the generated user embeddings with the original input of each task to evaluate the performance gains. For Task 1 and 2, we adopt the commonly used metrics including AUC, Recall and F1 score. For Task 3, considering the low player returning rate, we report the relative Hit ratio in the top 200,000 recalled players. The three datasets are all divided into train, valid and test sets in the ratio of 8:1:1. We repeat each experiment 5 times and report the average results.

Comparison Methods

We compare our methods with the following baselines.

- **Denoising AutoEncoder** [Vincent *et al.*, 2008] is widely applied for unsupervised representation learning.
- **Hierarchy LSTM** extracts sequential information using two-layer LSTMs[Graves and Graves, 2012].
- **BERT4Rec** [Sun *et al.*, 2019] models sequences via a bi-directional self-atentive model with a cloze objectve.
- **CLUE** [Cheng *et al.*, 2021] uses implicitly augmented views for contastive pre-training.
- **IDICL** [Sun *et al.*, 2022] performs contrastive pre-training via taking different time periods of the same behavior sequence as augmented views.
- **MBSSL** [Xu *et al.*, 2023] uses inter- and intra-behavior self-supervised learning tasks to capture the behavior correlations for multi-behavior recommendation.
- **AETN** [Zhang *et al.*, 2020] is tailored for generating general user embeddings based on app usage.
- **MFN** [Bian *et al.*, 2021] learns user representation with users' interaction both within and outside of apps.

These methods can be categorized into distinct sets: classic sequential models (AutoEncoder, LSTM and BERT4Rec), discriminative pre-training methods (CLUE, IDICL), multi-behavior learning approach (MBSSL), and user representation models designed for app scenarios (AETN, MFN).

5.2 Experimental Results

Table 3 presents the performance of all methods on three industrial downstream datasets. From the shown results, we make the following observations.

For the baseline methods, we can see that the user embeddings generated by all models are beneficial to downstream tasks, validating the practical significance of our goal. Among them, models that incorporate multiple behaviors (MBSSL, AETN and MFN) outperform those focusing on single behavior, demonstrating the importance of considering multi-type behaviors for an accurate understanding of users. Additionally, models designed for mobile scenarios (AETN, MFN) perform better than other approaches, highlighting the necessity of specialized modeling based on app usage.

For the proposed approach MAFN, it achieves the best performance on all datasets. On small-scale dataset with sparse supervised signals (task 3), the results indicate that the tailored modules can effectively extract users' intents from app usage, providing valuable information that helps alleviate the data sparsity in downstream tasks. On large-scale datasets (task 1, 2), the context-aware fusion module can be well fine-tuned to generate embeddings that contain relevant user interests based on the given context. Overall, these results show that the MAFN is an effective solution for obtaining general user representation based on app usage.

5.3 Ablation Study

We conduct a detailed ablation study to analyze how each of the proposed components affects final performance. Table 4 shows the performance of our default method and its six variants on *User Attribute Predicion* task. Similar conclusions can be drawn on other tasks, which are omitted here.

(1) *w/o LIM or SIM:* Without either of the two modules that capture users' specific aspect of interests, the variants exhibit significantly poorer performance compared to the MAFN. The results indicate both the importance of collectively modeling behaviors with varying frequencies, and the effectiveness of the LIM and SIM.

(2) *w/o IFM:* In these variants, we replace the fusion module with the vector concatenation, attention-based fusion and gate network fusion to obtain final user representations. The inferior performance of all these variants compared to MAFN across all evaluation metrics demonstrates that previous fusion methods, which overlook the dissimilar embedding space, can result in sub-optimal results.

(3) *w/o tailored pre-training tasks:* In these variants, we replace the pre-training tasks with widely-used mask behavior prediction tasks. The performance drops sharply, additionally showing the necessity of the proposed self-supervised tasks tailored to behavioral characteristics.

Model	Advertisement CVR Prediction			User Attribute Prediction			Game Player Recall
	AUC ↑	Recall@20% ↑	F1 ↑	AUC ↑	Recall@20% ↑	F1 ↑	HIT_r ↑
Base Model	0.6084	0.3130	0.1536	-	-	-	0.9363
Denosing Autoencoder	0.6163	0.3197	0.1547	0.5899	0.2925	0.3860	0.9385
Hierarchy LSTM	0.6174	0.3197	0.1550	0.5858	0.2545	0.3443	0.9370
BERT4Rec	0.6206	0.3258	0.1561	0.6153	0.2714	0.3612	0.9399
CLUE	0.6210	0.3204	0.1577	0.6205	0.2739	0.3641	0.9406
IDICL	0.6205	0.3195	0.1549	0.6147	0.2809	0.3628	0.9399
MBSSL	0.6225	0.3223	0.1577	0.6422	0.2912	0.3731	0.9392
AETN	0.6249	0.3300	0.1597	0.6448	0.2924	0.3881	0.9435
MFN	0.6247	0.3211	0.1583	0.6450	0.2809	0.3896	0.9442
MAFN	**0.6325**	**0.3359**	**0.1621**	**0.6619**	**0.2994**	**0.3978**	**0.9544**
Improv.	+1.2%	+1.7%	+1.5%	+2.6%	+2.4%	+2.1%	+1.1%

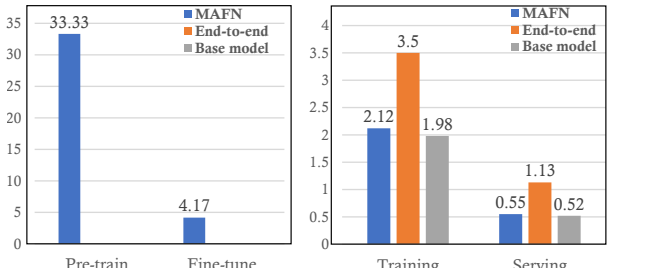
Table 3: Performance comparison of methods on real-world downstream tasks: Advertisement CVR Prediction, User Attribute Prediction and Game Player Recall. For each metric, we use bold and underline fonts to mark the best and second best performance, respectively. ***Improv.*** presents the relative improvement of MAFN over the best result of the baselines. Improvements are statistically significant with $\rho < 0.05$.

Variants	User Attribute Prediction		
	AUC ↑	Recall@20% ↑	F1 ↑
MAFN	**0.6619**	**0.2994**	**0.3978**
w/o LIM	0.6287	0.2759	0.3684
w/o SIM	0.6223	0.2744	0.3623
$Concat.$ Fusion	0.6580	0.2969	0.3941
$Attn.$ Fusion	0.6544	0.2897	0.3863
$Gate.$ Fusion	0.6515	0.2831	0.3821
$w/o\ \mathcal{L}_{dist}$	0.6391	0.2815	0.3737
$w/o\ \mathcal{L}_{re}$	0.6501	0.2922	0.3884
$w/o\ \mathcal{L}_{sem}$	0.6420	0.2902	0.3879
$w/o\ \mathcal{L}_{user}$	0.6601	0.2960	0.3938

Table 4: Ablation analysis on MAFN and its variants.

Service	Model	AUC	Recall@20%
Game App Recommendation	+AETN	+0.6%	+1.7%
	+MAFN	+1.1%	+2.5%

Table 5: Simulate online evaluation results.



(a) Offline Pre-training (*hours*). (b) Offline Training & Online Serving (*hours*).

Figure 2: Time cost analysis at different stages.

5.4 Simulate Online A/B Testing

To further verify the effectiveness of the output user embeddings, we simulate online feed recommendation A/B testing on industrial data for *Game App Recommendation* task using evaluation metrics AUC and Recall@20%. We evaluate the base model, model with AETN embeddings, and model with our MAFN embeddings. We split test traffic by user IDs evenly for the tested models. The improvement results compared with the base model are presented in Table 5, which further confirm the practical value of our approach.

5.5 Extra Cost and Run-Time Analysis

To further demonstrate the practical value of our method, we conduct a comparison with the base methods (*i.e.,* a downstream GBDT-based model that does not model users' historical behaviors and only uses statistical features) and traditional end-to-end method (*i.e.,* a BERT4Rec model that provides end-to-end training and inference based on **extra long** sequential records). Figure 2 illustrates the time cost of these three schemes at different stages on the *Game Player Recall* task with a larger user population. In this task, we trained the models on approximately 5 million users and made predictions for around 6 million users. The figure reveals that while MAFN incurs considerable time consumption during the offline pre-training and fine-tuning stages, it does not impose a significant burden on the model during the training and online service processes. This is because MAFN directly provides the generated user representations to downstream models, adding only 64-dimensional features. Consequently, the impact on downstream model with hundreds of dimensional features is minimal, yet the benefit is considerable (+1.1% on Recall@500,000). Though the end-to-end model offers similar benefits (+0.9% on Recall@500,000), its time consumption in the online service phase is nearly double compared to the base model, which hampers its practical application. Hence, in mobile scenarios where offline training time is not crucial, MAFN is a more suitable solution.

5.6 Case Study

To demonstrate our model's ability in understanding mobile users, we present a case study in Figure 3, where 3,000 users' embeddings are visualized via the t-SNE [Van der Maaten

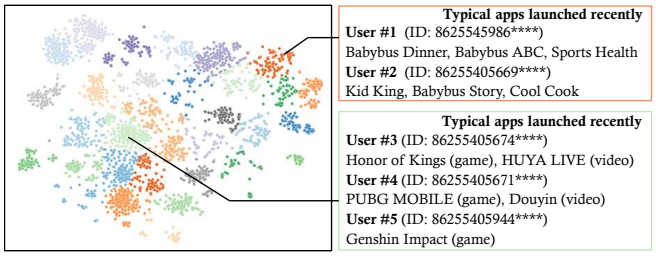
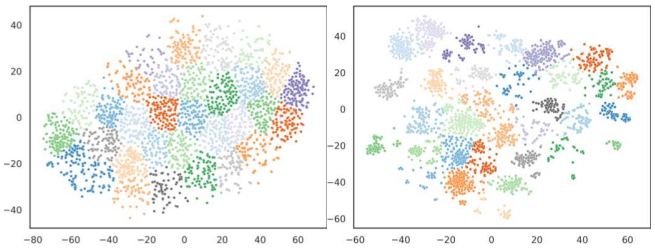


Figure 3: A case study of the generated user embeddings. We randomly sample five users from two clusters. The learned user representations can distinguish users with distinct preferences, and bring users with similar interests closer in the discrete embedding space.



(a) User embeddings from AETN. (b) User embeddings from MAFN.

Figure 4: Comparison of generated user embeddings.

and Hinton, 2008]. We found that the output user representations can distinguish users with different preferences, and bring users with similar interests closer in the discrete embedding space. For example, users #1 and #2, whose embeddings are close, both frequently accessed parenting-related apps, suggesting similar needs. Meanwhile, the other three neighboring users usually opened games, music, and video apps, indicating their shared intents in entertainment.

Figure 4 shows the distribution difference between the user representations generated by AETN and that by our MAFN. Upon observation, it can be noted that the user representations learned by AETN exhibit aggregation and lack distinctiveness, whereas the user embeddings learned by MAFN demonstrate a more discrete and dispersed pattern.

5.7 Implementation and Hyper-parameters

Our code is available at https://github.com/CGYR/MAFN.

Hyper-Parameters

Following previous works [Cheng *et al.*, 2021; Bian *et al.*, 2021], we set the dimension of user embedding and app embedding as 64. The learning rates are set to default settings for baselines and tuned to 0.01 for our model. In the utilized Transformer modules, the number of attention heads and layers are both set as 2. In the *app distribution prediction* task, where we use a user's preceding N behaviors to predict app occurrences in the next K behaviors, we set N and K as half of the sequence length to balance the difficulty in distilling information and making predictions. In the contrastive learning tasks (*i.e.,* pre-training tasks #3 and #4), the temperature coefficients are set to 0.1 and the in-batch negative sampling number for each sample are set as 31. The size of *codebook* (*i.e.,* the auxiliary embedding table for unifying users' multi-aspects interests) is set to 32. With respect to the sequence length distribution in our datasets, the max length of input sequences is set to 512 (*i.e.,* both the launch records and (un)install records will be truncated or padded to the length of 512). For downstream models, we adopt hyper-parameters similar to the practical models. The dimension of the intermediate layer in the two-layer MLP and DeepFM is set as 64. In GBDT-based model, the number of leaves is set to 53. For baseline models and our MAFN, the learning rate is either searched from $\{0.01, 0.02, 0.05, 0.1, 0.2, ..., 0.9\}$ or copied from previous works if provided.

Adjustments on Baselines

To ensure a fair comparison among different baseline models, we make moderate adjustments to baseline models to align them with our specific scenario. For models designed for learning single-type behavior (*i.e.,* Denosing Autoencoder, LSTMs, BERT4Rec, CLUE and IDICL), we train multiple copies of them to separately extract information from different types of app behavior sequences, and then concatenate these embeddings to obtain the final user representation. For models that consider multi-behaviors, we simply replace the original input behaviors with our app behaviors. Besides, for models that do not explicitly output user representations, we use the overall representation of user behavior sequences as user representations.

Training Settings

We use the Adam optimizer for model training. The batch size is set as 256 for both pre-training and fine-tuning. We implement all experiments with Python 3.7 and Pytorch 1.9.0 with CUDA 10.2 on NVIDIA Tesla V100 GPUs.

5.8 Limitations

A possible limitation is the additional time cost incurred during the offline pre-training and fine-tuning phases. In practice, many techniques can be invoked to speed up these processes, such as efficiency optimization [Singh, 2012], parallel computation and distributed parameter learning [Dean *et al.*, 2012]. Also, the Parameter-Efficient Fine-Tuning (PEFT) methods such as LoRA may accelerate model adaptation via only updating a small fraction of parameters.

6 Conclusions

In this paper, we presented a novel MAFN framework for pre-training general user representation based on multi-type app usage. Framed in the two-stage pre-train & fine-tune paradigm, we proposed several customized modules equipped with well-designed pre-training tasks to extract users' short- and long-term preferences from types of user-app interactions with varying frequencies. Further, we employed a unified fusion module to effectively combine users' multi-aspect interests and generate general user representations, which can efficiently adapt to downstream tasks. Besides, we provided a practical way to deploy our approach within feasible time in practice. Extensive experiments conducted on three real-world tasks have validated the effectiveness of our approach. In the future, we will further explore modeling for user behaviors across diverse mobile fields.

Ethical Statement

The user data we collected is from anonymous users who have opted into the **User Experience Improvement Program** (*i.e.,* users who have consented to share app data in order for mobile manufacturers to provide personalized services). Such program is globally recognized and widely used in APP scenarios as a user data sharing protocol, which strictly limits the scope of data usage, ensuring that users are informed and the data is used appropriately, thereby safeguarding user privacy and preventing any data breaches.

Acknowledgements

This research was partially supported by grants from the National Natural Science Foundation of China (U20A20229), Anhui Provincial Natural Science Foundation (No. 2308085QF229), the Fundamental Research Funds for the Central Universities (No. JZ2023HGQA0471, No. WK2150110034), and the OPPO joint research program. We furthermore thank the anonymous reviewers for their constructive comments.

References

[Bian *et al.*, 2021] Shuqing Bian, Wayne Xin Zhao, Kun Zhou, Xu Chen, Jing Cai, Yancheng He, Xingji Luo, and Ji-Rong Wen. A novel macro-micro fusion network for user representation learning on mobile apps. In *Proceedings of the Web Conference 2021*, pages 3199–3209, 2021.

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

[Dean *et al.*, 2012] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

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

[Friedman, 2001] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[Graves and Graves, 2012] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[Gu *et al.*, 2021] Jie Gu, Feng Wang, Qinghui Sun, Zhiquan Ye, Xiaoxiao Xu, Jingmin Chen, and Jun Zhang. Exploiting behavioral consistence for universal user representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4063–4071, 2021.

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[Hou *et al.*, 2023] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*, pages 1162–1171, 2023.

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

[Li *et al.*, 2016] Huoran Li, Wei Ai, Xuanzhe Liu, Jian Tang, Gang Huang, Feng Feng, and Qiaozhu Mei. Voting with their feet: Inferring user preferences from app management activities. In *Proceedings of the 25th international conference on world wide web*, pages 1351–1362, 2016.

[Li *et al.*, 2020] Tong Li, Mingyang Zhang, Hancheng Cao, Yong Li, Sasu Tarkoma, and Pan Hui. " what apps did you use?": Understanding the long-term evolution of mobile app usage. In *Proceedings of The Web Conference 2020*, pages 66–76, 2020.

[Li *et al.*, 2022] Tong Li, Tong Xia, Huandong Wang, Zhen Tu, Sasu Tarkoma, Zhu Han, and Pan Hui. Smartphone app usage analysis: datasets, methods, and applications. *IEEE Communications Surveys & Tutorials*, 2022.

[Liu *et al.*, 2021] Alexander H Liu, SouYoung Jin, Cheng-I Jeff Lai, Andrew Rouditchenko, Aude Oliva, and James Glass. Cross-modal discrete representation learning. *arXiv preprint arXiv:2106.05438*, 2021.

[Liu *et al.*, 2022] Bulou Liu, Bing Bai, Weibang Xie, Yiwen Guo, and Hao Chen. Task-optimized user clustering based on mobile app usage for cold-start recommendations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3347–3356, 2022.

[Lu *et al.*, 2014] Eric Hsueh-Chan Lu, Yi-Wei Lin, and Jing-Bin Ciou. Mining mobile application sequential patterns for usage prediction. In *2014 IEEE International Conference on Granular Computing (GrC)*, pages 185–190. IEEE, 2014.

[Mu *et al.*, 2022] Shanlei Mu, Yupeng Hou, Wayne Xin Zhao, Yaliang Li, and Bolin Ding. Id-agnostic user behavior pre-training for sequential recommendation. *arXiv preprint arXiv:2206.02323*, 2022.

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

[Peltonen *et al.*, 2018] Ella Peltonen, Eemil Lagerspetz, Jonatan Hamberg, Abhinav Mehrotra, Mirco Musolesi, Petteri Nurmi, and Sasu Tarkoma. The hidden image of mobile apps: geographic, demographic, and cultural factors in mobile usage. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–12, 2018.

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

[Radosavljevic *et al.*, 2016] Vladan Radosavljevic, Mihajlo Grbovic, Nemanja Djuric, Narayan Bhamidipati, Daneo Zhang, Jack Wang, Jiankai Dang, Haiying Huang, Ananth Nagarajan, and Peiji Chen. Smartphone app categorization for interest targeting in advertising marketplace. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 93–94, 2016.

[Seol *et al.*, 2022] Jinseok Jamie Seol, Youngrok Ko, and Sang-goo Lee. Exploiting session information in bert-based session-aware sequential recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2639–2644, 2022.

[Singh, 2012] Ajay Singh. An overview of the optimization modelling applications. *Journal of Hydrology*, 466:167–182, 2012.

[Sun *et al.*, 2019] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[Sun *et al.*, 2022] Qinghui Sun, Jie Gu, XiaoXiao Xu, Renjun Xu, Ke Liu, Bei Yang, Hong Liu, and Huan Xu. Learning interest-oriented universal user representation via self-supervision. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 7270–7278, 2022.

[Tu *et al.*, 2018] Zhen Tu, Runtong Li, Yong Li, Gang Wang, Di Wu, Pan Hui, Li Su, and Depeng Jin. Your apps give you away: distinguishing mobile users by their app usage fingerprints. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):1–23, 2018.

[Van Den Oord *et al.*, 2017] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[Van der Maaten and Hinton, 2008] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[Vincent *et al.*, 2008] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[Wang *et al.*, 2020] Wen Wang, Wei Zhang, Shukai Liu, Qi Liu, Bo Zhang, Leyu Lin, and Hongyuan Zha. Beyond clicks: Modeling multi-relational item graph for session-based target behavior prediction. In *Proceedings of the web conference 2020*, pages 3056–3062, 2020.

[Wu *et al.*, 2020] Chuhan Wu, Fangzhao Wu, Tao Qi, Jianxun Lian, Yongfeng Huang, and Xing Xie. Ptum: Pre-training user model from unlabeled user behaviors via self-supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1939–1944, 2020.

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

[Xia *et al.*, 2021] Lianghao Xia, Yong Xu, Chao Huang, Peng Dai, and Liefeng Bo. Graph meta network for multi-behavior recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 757–766, 2021.

[Xu *et al.*, 2023] Jingcao Xu, Chaokun Wang, Cheng Wu, Yang Song, Kai Zheng, Xiaowei Wang, Changping Wang, Guorui Zhou, and Kun Gai. Multi-behavior self-supervised learning for recommendation. *arXiv preprint arXiv:2305.18238*, 2023.

[Yu *et al.*, 2020] Yue Yu, Tong Xia, Huandong Wang, Jie Feng, and Yong Li. Semantic-aware spatio-temporal app usage representation via graph convolutional network. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(3):1–24, 2020.

[Yuan *et al.*, 2020] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 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*, pages 1469–1478, 2020.

[Zhang *et al.*, 2020] Junqi Zhang, Bing Bai, Ye Lin, Jian Liang, Kun Bai, and Fei Wang. General-purpose user embeddings based on mobile app usage. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2831–2840, 2020.

[Zhang *et al.*, 2022] Kai Zhang, Kun Zhang, Mengdi Zhang, Hongke Zhao, Qi Liu, Wei Wu, and Enhong Chen. Incorporating dynamic semantics into pre-trained language model for aspect-based sentiment analysis. *arXiv preprint arXiv:2203.16369*, 2022.

[Zhang *et al.*, 2023] Lei Zhang, Wuji Zhang, Likang Wu, Ming He, and Hongke Zhao. Shgcn: Socially enhanced heterogeneous graph convolutional network for multi-behavior prediction. *ACM Transactions on the Web*, 18(1):1–27, 2023.

[Zhao *et al.*, 2016] Sha Zhao, Gang Pan, Yifan Zhao, Jianrong Tao, Jinlai Chen, Shijian Li, and Zhaohui Wu. Mining user attributes using large-scale app lists of smartphones. *IEEE Systems Journal*, 11(1):315–323, 2016.

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