Optimizing Smartphone App Usage Prediction: A Click-Through Rate Ranking Approach

Yuqi Zhang
zhangyuqi2020@gmail.com
Harbin Institute of Technology
Harbin, China

Meiying Kang
mykang@stu.suda.edu.cn
Soochow University
Suzhou, China

Xiucheng Li*
lixiucheng@hit.edu.cn
Harbin Institute of Technology
Shenzhen, China

Yu Qiu
qiuyu5999446@gmail.com
Independent
Chengdu, China

Zhijun Li*
lizhijun_os@hit.edu.cn
Harbin Institute of Technology
Harbin, China

ABSTRACT

Over the past decade, smartphones have become indispensable personal mobile devices, experiencing a remarkable surge in software apps. These apps empower users to seamlessly connect with various internet services, such as social communication and online shopping. Accurately predicting smartphone app usage can effectively improve user experience and optimize resource utilization. However, existing models often treat app usage prediction as a classification problem, which suffers from issues of app usage imbalance and out-of-distribution (OOD) during deployment. To address these challenges, this paper proposes a novel click-through rate (CTR) ranking-based method for predicting app usage. By transforming the classification problem into a CTR problem, we can eliminate the negative impact of the app usage imbalance issue. To address the OOD issue during deployment, we generate the app click sequence and three types of discriminative features, which enable generalization on unseen apps. The app click sequence and the three types of features serve as inputs for training a CTR estimation model in the cloud, and the trained model is then deployed on the user's smartphone to predict the CTR for each installed app. The decision-making process involves ranking these CTR values and selecting the app with the highest CTR as the final prediction. Our method has been extensively tested with large-scale app usage data. The results demonstrate that our approach is able to outperform state-of-the-art methods, with improvements over 4.93% in top-3 accuracy and 6.64% in top-5 accuracy. It achieves approximately twice the accuracy in predicting apps with low usage frequencies in comparison to baseline methods. Our method has been successfully deployed on the app recommendation system of a leading smartphone manufacturer.

*Corresponding author.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671567

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing**.

KEYWORDS

Smartphone app; App usage prediction; Click-through rate; Ranking

ACM Reference Format:
Yuqi Zhang, Meiying Kang, Xiucheng Li, Yu Qiu, and Zhijun Li. 2024. Optimizing Smartphone App Usage Prediction: A Click-Through Rate Ranking Approach. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3637528.3671567

1 INTRODUCTION

The rapid development of mobile networks and the widespread adoption of smartphones have triggered an extraordinary surge in the number of software apps. According to a report by Statista [1], the number of apps in the Google Play Store has soared from 16,000 in December 2009 to 2,438,553 by December 2023. These apps have become indispensable tools in shaping our daily routines, such as social communication (e.g., WeChat, Facebook) and online shopping (e.g., Taobao, Amazon). Smartphone manufacturers are actively exploring various optimization strategies to address accompanying challenges, such as escalating resource demands and the growing complexity of app usage patterns. Among them, optimizing the smartphone app usage prediction approach can effectively improve user experience and optimize resource utilization [8]. For example, armed with knowledge of the next apps most likely to be used, smartphones can selectively preload resources related to predicted apps.

Many models have been developed to predict app usage in the community, mainly including the modeling of app click data and the integration of contextual information [5, 10–15, 17]. These models approach app usage prediction as a classification problem, identifying the app with the highest probability of classification among all possible labels. However, classification models encounter two challenges when dealing with a diverse range of apps: 1) App usage imbalance: Users typically exhibit a noticeable imbalance in their app usage, with a small subset of apps being heavily used. Consequently, classification models grapple with severe label imbalance,

excelling in predicting apps with high-frequency usage but struggling to accurately predict those with medium to low frequencies. In practice, smartphone manufacturers show a greater interest in the model's capability of accurately predicting apps across various usage frequencies for providing personalized app recommendations. 2) Out-of-distribution during deployment: Smartphone manufacturers usually train the app usage prediction model with all available app labels at hand in the cloud and then deploy it to edge devices. However, there may be unseen apps that emerge during deployment.

To address these issues, this paper proposes a novel method—Click-Through Rate Ranking-based Application Usage Prediction (CTR-RAD)—which transforms the classification problem into a click-through rate (CTR) ranking problem. It involves generating the app click sequence and three types of discriminative features as inputs for training a CTR estimation model in the cloud. The trained model is then deployed on the user's smartphone to predict the CTR for each installed app. The decision-making process entails ranking these CTR values, ultimately designating the app with the highest CTR as the final prediction. It is noteworthy that the CTR estimation model can be replaced with any user behavior sequence-based CTR estimation model. This interchangeability is attributed to the universality of the created app click sequence and three types of features (sequence feature, affinity feature, and context feature) across these models. CTR-RAD demonstrates the following two advantages: 1) Balanced performance across the entire spectrum of frequencies: Unlike classification models trained with heavily imbalanced app labels, the CTR estimation model is trained with only two labels (*click* or *not click*) for the target app, thereby mitigating the negative impact of the app usage imbalance issue. Additionally, we introduce a high-frequency app suppression strategy to further mitigate the imbalance among target apps. 2) Generalizing to unseen apps during deployment: We define app click sequence and three types of features that are sharable across different apps and enable generalization on unseen apps. Moreover, CTR-RAD is not tied to specific users and exhibits adaptability to new users.

In summary, our primary contributions are as follows: 1) We propose an innovative app usage prediction method based on click-through rate ranking, addressing the challenges encountered by traditional classification models. 2) Existing CTR estimation models are typically tailored for product recommendations, leaving a research gap in the field of app usage prediction. This paper fills this gap by introducing the universal app click sequence and three types of features adaptable to any user behavior sequence. 3) Through extensive evaluation using app usage data, the results demonstrate that CTR-RAD outperforms state-of-the-art methods, enhancing top-3 accuracy by 4.93% and top-5 accuracy by 6.64%. Moreover, compared to baseline methods, CTR-RAD achieves approximately double the accuracy in predicting apps with low usage frequencies. Notably, CTR-RAD has been successfully deployed on the app recommendation system of a leading smartphone manufacturer.

2 RELATED WORK

This section presents a brief overview of the existing literature on app usage prediction. The fundamental benchmark methods for predicting app usage typically include recognizing the most frequently used (MFU) and most recently used (MRU) apps [12]. In the initial stages of research, probabilistic models such as the Markov model [9] and the Bayesian model [2, 21] are commonly employed to model app click sequences. Meanwhile, early studies tend to enhance prediction performance by incorporating diverse contextual information from sensors [4, 12, 18, 20]. Shin et al. [12] conducted a comprehensive analysis of various sensor contextual features associated with app usage. Do et al. [4] further predict users' future locations and app usage by exploiting rich contextual information from smartphone sensors. Zhu et al. [20] propose leveraging context logs to mine users' personal context-aware preferences. Moreover, Zhao et al. [18] extract features related to human mobility from users' trajectories and integrate them with app usage sequences to train a classifier for predicting app usage. Thus, app usage prediction primarily relied on diverse probabilistic models and rich contextual information in the early stages.

To delve deeper into this field, researchers have employed a series of deep learning-based models [3, 7, 10, 11, 13–17] for modeling app usage data. AppUsage2Vec [17], as a classical deep learning-based app usage prediction model, incorporates an app-attention mechanism to quantify the impact of each app on the target app, along with a dual-DNN module for predicting app usage. Subsequent models predicting app usage based on deep learning can be broadly divided into two main categories. The first line of approaches, referred to as sequence-based methods, employs various forms of RNNs [7, 14, 15] to capture the temporal patterns underlying app click sequences. Lee et al. [7] introduce a GRU-based multi-task learning framework incorporating time and location contexts to enhance app usage predictions. Similarly, Xu et al. [15] and Xia et al. [14] use LSTM-based networks to model app click sequences while also considering time and location contexts.

Another line of work, known as graph-based methods, harnesses the graph embedding techniques to capture correlations among apps, locations, and time [3, 10, 11, 13, 16]. Chen et al. [3] construct three bipartite graphs to represent various relationships (app-location, app-time, and app-app type). They then introduce a heterogeneous graph embedding algorithm to map these graphs into a shared latent space. However, this approach neglects the dynamic nature of app usage. To overcome this limitation, Yu et al. [16] design a graph with nodes representing app, time, and location, where edges encapsulate their co-occurrence relations. They subsequently exploit a GCN-based model to learn a semantic-aware spatio-temporal representation. Simultaneously, to capture the dynamics of user interests over time, Ouyang et al. [10] model user app usage behaviors as a dynamic graph. They propose a dynamic usage graph network to acquire effective embeddings within this dynamic graph. Furthermore, Shen et al. [11] construct an attribute-aware directed graph and develop an attention-based aggregation model to characterize patterns in app usage.

Recent research has shown a trend towards the integration of sequence-based methods with graph-based methods. Wang et al. [13] present SGFNN, a pioneering model that seamlessly integrates these two domains and trains them in an end-to-end manner. In summary, existing approaches prioritize the exploration of probabilistic models and deep learning techniques, concurrently making efforts to incorporate diverse contextual information for the prediction of app usage. However, these methods treat app usage

Table 1: Examples of Events

EventType	Timestamp	Value	Date	UserID
App Click	1648893782448	{*app_name*}	20220402	42839
Screen	1648891954527	{*screen_on*; *screen_off*}	20220402	42839
Headset	1648950513848	{*headset_connected*; *headset_disconnected*}	20220403	141
WiFi	1648803122971	{*WiFi_connected*; *WiFi_disconnected*}	20220401	23
Install	1648803122971	{*install*; *uninstall*; *update*}	20220401	27955

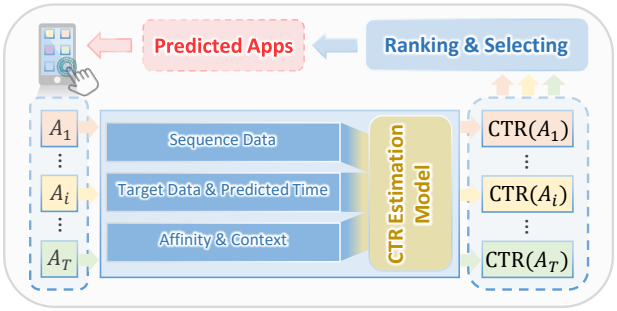


Figure 1: The overall framework of CTR-RAD.

prediction as a classification problem, which suffers from the issues of app usage imbalance and out-of-distribution during deployment.

3 METHOD

The overall framework of CTR-RAD is depicted in Figure 1. This section initiates by explaining the preparation of the input data for the CTR estimation model in Section 3.1. Subsequently, Section 3.2 introduces the training of the click-through rate estimation model using the constructed data. These data can be adapted to any user behavior sequence-based CTR estimation model. To exemplify, we employ a classic model—Deep Interest Evolution Network (DIEN) [19]. Finally, Section 3.3 ends with the prediction process of CTR-RAD.

3.1 Data Preparation

We adopt five types of events, namely, *App Click*, *Screen*, *Headset*, *WiFi*, and *Install*, to generate the input data. Each event is a tuple consisting of (EventType, Timestamp, Value, Date, UserID). Table 1 presents five instances of events with different types. The raw data is preprocessed to construct the app click sequence and three distinct types of features, serving as input for the click-through rate estimation model in Section 3.2.

3.1.1 App click sequence. Based on the *App Click* event, we initially collect the sequence of app clicks sorted by their timestamps in ascending order. Then, we generate training samples using a window size of M and a duration limit of D. This means that each training example consists of M apps and the cumulative duration of these apps within the window does not exceed D. In each training example, denoted as $a_{1:n+1} = (a_1, \ldots, a_{n+1})$, a_{n+1} represents the final clicked app, serving as the target for the model, while $a_{1:n}$ represents all preceding clicked apps, regarded as input. Each $a_{1:n+1}$ is accompanied by its corresponding click time sequence $t_{1:n+1} = (t_1, \ldots, t_{n+1})$, where t_{n+1} denotes the prediction time.

3.1.2 Sequence feature. Sequence features refer to the characteristics associated with the order of app clicks. We experimentally identified two valuable ones:

1) Time gap feature, which captures the duration between the app click and the prediction time. For every app click within the sequence $a_{1:n+1}$, a corresponding time gap feature is defined as $g_{1:n+1} = (g_1, \ldots, g_{n+1})$. Each time gap feature is calculated as follows:

$$g_i = t_{n+1} - t_i, 1 \le i \le n+1. \tag{1}$$

2) Screen feature, which captures the app's position in the sequence of clicks following the activation of the screen. For every app within the sequence $a_{1:n+1}$, a corresponding screen feature is defined as $s_{1:n+1} = (s_1, \ldots, s_{n+1})$. To determine the screen feature for each a_i in $a_{1:n+1}$, we first identify the time t_{screen_on} when the *Screen* event with a value of *screen_on* occurs, either at t_i or the closest time preceding t_i:

$$t_{screen\_on} = \max\{t \mid t \le t_i \text{ and } \mathit{Screen}(\text{t})=\mathit{screen\_on}\}. \tag{2}$$

Next, we obtain the sequence of app clicks occurring within the time interval $[t_{screen\_on}, t_i]$. Thus, s_i is equivalent to the length of the sequence within $[t_{screen\_on}, t_i]$:

$$s_i = |\{a_j \mid t_{screen\_on} \le t_j \le t_i\}|. \tag{3}$$

3.1.3 Context feature. Context feature refers to certain contextual information corresponding to the prediction time t_{n+1}. We experimentally identified three valuable ones:

1) Headset feature, denoted as $H(t_{n+1})$, which indicates the connection status of the headset at time t_{n+1}. This status is characterized by two values: *headset_connected* and *headset_disconnected*, as outlined in Table 1. We initially identify the time $t_{headset}$ when the values of the *Headset* event occur, either at t_{n+1} or the closest time preceding t_{n+1}:

$$t_{headset} = \max\{t \mid t \le t_{n+1} \text{ and } \mathit{Headset}(\text{t}) \text{ exists value}\}. \tag{4}$$

Then, the value of the *Headset* event at time $t_{headset}$ is defined as the headset feature $H(t_{n+1})$ at t_{n+1}.

2) WiFi feature, referred to as $W(t_{n+1})$, which indicates the connection status of WiFi at time t_{n+1}. This status contains two distinct values: *WiFi_connected* and *WiFi_disconnected*, as shown in Table 1. We first identify the time t_{WiFi} when the values of the *WiFi* event occur, either at t_{n+1} or the closest time preceding t_{n+1}:

$$t_{WiFi} = \max\{t \mid t \le t_{n+1} \text{ and } \mathit{WiFi}(\text{t}) \text{ exists value}\}. \tag{5}$$

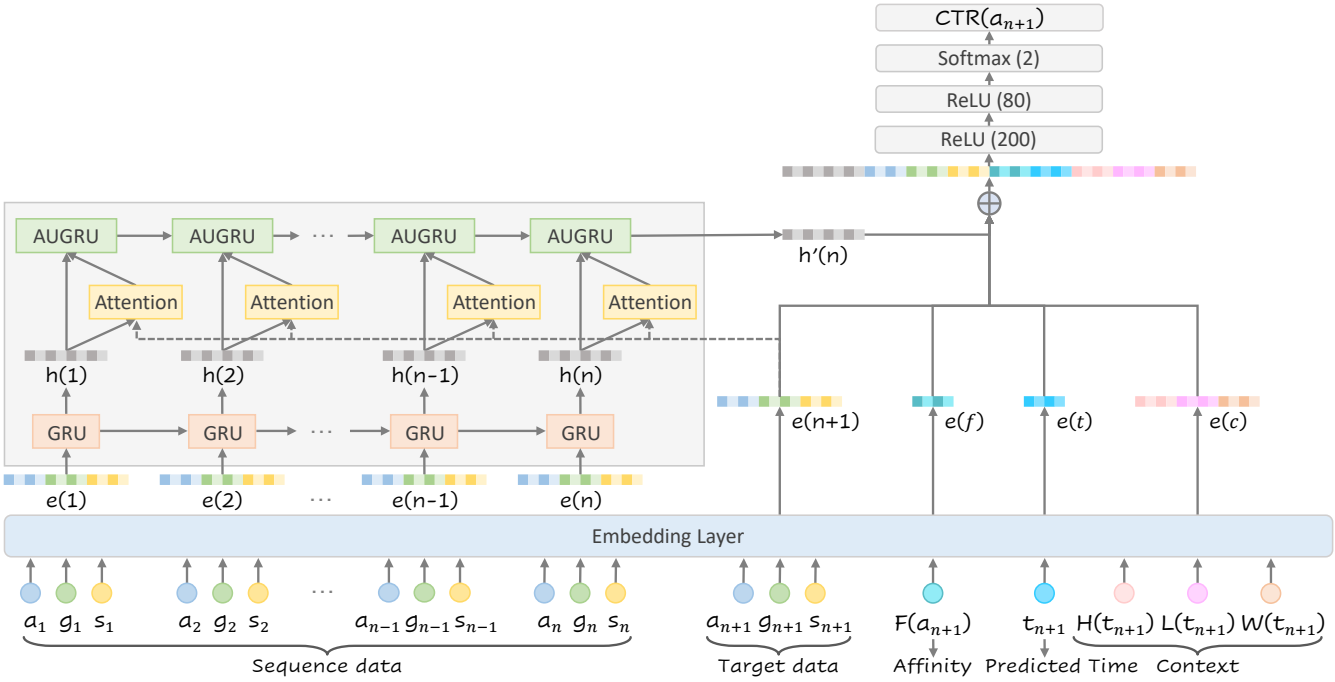


Figure 2: The framework of DIEN using the app click sequence and three types of features.

Then, the value of the *WiFi* event at time t_{WiFi} is defined as the WiFi feature $W(t_{n+1})$ at t_{n+1}.

3) Install feature, identified as $L(t_{n+1})$, which represents the values of the *Install* event at time t_{n+1}. We select two valid values for this event: *install* and *update*, as outlined in Table 1. Initially, we identify the time $t_{install}$ at which the values of the *Install* event occur, either at t_{n+1} or the closest time preceding t_{n+1}:

$$t_{install} = \max\{t \mid t \leq t_{n+1} \text{ and } Install(t) \text{ exists value}\}. \quad (6)$$

Then, the value of the *Install* event at time $t_{install}$ is defined as the the install feature $L(t_{n+1})$ at t_{n+1}.

3.1.4 Affinity feature. Affinity feature, described as $F(a_{n+1})$, which reflects the user's inclination towards the target app a_{n+1}. As users exhibit distinct preferences for different apps, we quantify the user's affinity for the target app a_{n+1} by employing the base-2 logarithm (log2) of the historical click counts associated with this app (Click_counts(a_{n+1})):

$$F(a_{n+1}) = \lfloor \log_2(\text{Click\_counts}(a_{n+1})) \rfloor. \quad (7)$$

3.2 Learning

In this section, we present the training process of DIEN [19], a classic user behavior sequence-based CTR estimation model. The framework of DIEN, using the constructed app click sequences and three types of features, is depicted in Figure 2. The input of DIEN includes sequence data ($a_{1:n}$, $g_{1:n}$, $s_{1:n}$), target data (a_{n+1}, g_{n+1}, s_{n+1}), affinity feature ($F(a_{n+1})$), prediction time (t_{n+1}), and context feature ($H(t_{n+1})$, $W(t_{n+1})$, $L(t_{n+1})$), with the output being the click-through rate for the target app a_{n+1}.

3.2.1 High-frequency app suppression. In the numerous available apps, only a small number of apps are used very frequently, leaving the majority with moderate or low frequencies. Consequently, there exists a pronounced imbalance in the distribution of the target app a_{n+1} within the training samples. To some extent, CTR-RAD partially alleviates the negative impact of imbalanced app distribution as the CTR estimation model is trained with only two labels (*click* or *not click*). To further mitigate this concern, we propose a strategy to suppress the number of training samples whose target apps have high-frequency usage. This suppression can be implemented using Equation 8:

$$P(a_{n+1}) = \left(\sqrt{\frac{w(a_{n+1})}{\tau}} + 1\right) \cdot \frac{\tau}{w(a_{n+1})}, \quad (8)$$

where $P(a_{n+1})$ denotes the probability of choosing samples with the target app a_{n+1} as positive samples. $w(a_{n+1})$ represents the usage frequency of the target app a_{n+1}, calculated using Equation 9:

$$w(a_{n+1}) = \frac{\text{Click\_counts}(a_{n+1})}{\text{All\_click\_counts}}, \quad (9)$$

where Click_counts(a_{n+1}) represents the historical click counts of a_{n+1}, and All_click_counts signifies the total click counts of all apps during the same historical period. Overall, the probability $P(a_{n+1})$ inversely correlates with the usage frequency $w(a_{n+1})$ of the target app.

3.2.2 Training. As illustrated in Figure 2, the embedding layer processes the sequence data $(a_{1:n}, g_{1:n}, s_{1:n})$ to produce a dense embedding representation denoted as $\mathbf{e}(1:n)^{n\times 3d}$, where d represents the embedding dimension. Simultaneously, the embedding layer processes the context feature $(H(t_{n+1}), W(t_{n+1}), L(t_{n+1}))$, generating an embedding representation $\mathbf{e}(c)^{3d}$. Similarly, the embedding layer encodes the target data $(a_{n+1}, g_{n+1}, s_{n+1})$ into the embedding representation $\mathbf{e}(n+1)^{3d}$, while $F(a_{n+1})$ is encoded as the embedding representation $\mathbf{e}(f)^{d}$, and t_{n+1} is represented as $\mathbf{e}(t)^{d}$.

Initially, the sequence data embedding $\mathbf{e}(1:n)^{n\times 3d}$ serves as the input to the sequential network in the user behavior sequence-based CTR estimation model. In DIEN, the sequence data embedding is fed into a GRU network, which produces hidden states $\mathbf{h}(1:n)^{n\times n_h}$ (where n_h represents the hidden size). Subsequently, the hidden states $\mathbf{h}(1:n)^{n\times n_h}$ and the target data embedding $\mathbf{e}(n+1)^{3d}$ are fed into an attention unit, which calculates attention scores $\alpha(1:n)$. The scores reflect the relationship between the target data and the sequence data, with higher scores indicating stronger relevance. Following this, the hidden states $\mathbf{h}(1:n)^{n\times n_h}$ and the attention scores $\alpha(1:n)$ are input to a GRU with attentional update gate (AUGRU) [19]. The AUGRU seamlessly integrates the attention mechanism with the GRU. The last hidden state of the AUGRU is denoted as $\mathbf{h}'(n)$.

Finally, the vectors $\mathbf{h}'(n)$, $\mathbf{e}(n+1)$, $\mathbf{e}(f)$, $\mathbf{e}(t)$, and $\mathbf{e}(c)$ are concatenated to form $\mathbf{x}$. The concatenated vector $\mathbf{x}$ is then fed into an MLP layer for the final prediction. Each training sample is labeled as $\mathbf{y} \in \{0, 1\}$, with 0 corresponding to a negative sample and 1 corresponding to a positive sample. The commonly used loss function in CTR estimation models is the negative log-likelihood function:

$$\mathcal{L} = -\frac{1}{N}\sum^{N}(\mathbf{y}\log p(\mathbf{x}) + (1-\mathbf{y})\log(1-p(\mathbf{x}))), \tag{10}$$

where N is the number of training samples, and $p(x)$, referred to as click-through rate, denotes the final output of the network, indicating the predicted probability of the user clicking on the target app.

3.3 Predicting

As illustrated in Figure 1, when the trained model in Section 3.2 is deployed on the user's smartphone, it anticipates the click-through rates ($C = \{CTR(A_1), \ldots, CTR(A_T)\}$) for each installed app ($\mathcal{A} = \{A_1, \ldots, A_T\}$, where T is the total number of apps on the user's smartphone). These CTR values are then ranked in descending order, and the top K ranked CTR values (TopK(C)) determine the final predictions:

$$\mathcal{P}_{\mathcal{A}} = \{A_i \mid 1 \le i \le T \text{ and } CTR(A_i) \in \text{TopK}(C)\}, \tag{11}$$

where $\mathcal{P}_{\mathcal{A}}$ represents the predicted top K apps.

4 EXPERIMENTS

In this section, we initially present an overview of the datasets, training setup, evaluation metrics, and baseline methods in Section 4.1. Then, we conduct a comprehensive evaluation of CTR-RAD using extensive app usage data in Section 4.2.

Table 2: Summary of Two-Version Datasets

Version	Period	Users	Apps
V1-data	2022/03/02-2022/04/27	50,000	76,460
V2-data	2023/02/05-2023/05/05	200,000	180,671

4.1 Datasets and Implementation Details

4.1.1 Datasets. The app usage data comprises two versions of datasets collected at different periods, denoted as V1-data and V2-data, respectively. The details of these two-version datasets are outlined in Table 2. Each app within the datasets is classified into one of 18 specific categories, including *Education*, *Games*, *Utilities*, *Entertainment*, *Health*, *News*, *Navigation*, *Social*, *Shopping*, *Music*, *Finance*, *Lifestyle*, *Sports*, *Video*, *Tools*, *Business*, *Travel*, and *Photography*. It is important to note that these datasets exclude any personally identifiable information. The user ID has been anonymized, and all user metadata has been removed.

4.1.2 Training setup. The hidden state size is set to 32, and the embedding size is set to 16. The model is trained for 100 epochs using the Adam optimizer [6] with a learning rate of 0.001, and a batch size of 200 is employed.

4.1.3 Evaluation metrics. We employ top-K accuracy as our evaluation metric, a widely adopted measure in the field of app usage prediction. Here, K denotes the number of predicted apps ($\mathcal{P}_{\mathcal{A}}$), as detailed in Section 3.3. The top-K accuracy is calculated as follows:

$$\text{top-K accuracy} = \frac{N(a_{n+1} \in \mathcal{P}_{\mathcal{A}})}{N_{\text{ALL}}}, \tag{12}$$

where N_{ALL} denotes the total number of test samples, and the numerator represents the count of correctly predicted test samples. A test sample is deemed correctly predicted if the target app a_{n+1} is within $\mathcal{P}_{\mathcal{A}}$. In our subsequent experiments, we primarily focus on top-1 accuracy (Top1), top-3 accuracy (Top3), and top-5 accuracy (Top5). It is noteworthy that smartphone manufacturers prioritize Top3 and Top5 metrics, as recommending 3 or 5 apps is more meaningful for users. To ensure fair evaluation, we conducted five experiments and then calculated the average as the final result.

4.1.4 Baselines. We compare CTR-RAD with the following baseline methods: 1) MFU (Most Frequently Used), which identifies the app most frequently used by a user. 2) MRU (Most Recently Used), which identifies the app most recently utilized by a user. 3) DNN [17], which uses two hidden layers for predicting the target app. 4) RNN-Attention [17], which employs an attention mechanism to learn the weights of the hidden states at each time step. A softmax activation is then applied to the weighted sum of these hidden states, which is connected to a fully connected layer to predict the target app. 5) AppUsage2Vec [17], which is a classical app usage prediction model that incorporates an app-attention mechanism and temporal context. 6) SGFNN [13], which is a pioneering model that integrates the sequence-based method with the graph-based method.

Table 3: Hyperparameter Study

P	V	V1-data			V2-data		
		Top1	Top3	Top5	Top1	Top3	Top5
τ	0.01	47.28	78.21	85.11	47.66	78.05	84.71
	0.05	48.28	78.65	85.35	48.63	78.48	84.89
	0.1	49.07	78.72	85.43	49.28	78.57	84.99
t-slot	15	48.84	78.73	85.43	49.12	78.62	85.04
	10	48.82	78.73	85.46	49.07	78.59	85.04
	5	48.93	78.73	85.43	49.15	78.59	85.01
g-slot	5	49.45	78.83	85.47	49.61	78.67	85.07
	2	49.85	79.01	85.59	50.03	78.80	85.14
	1	49.81	79.00	85.61	49.95	78.82	85.17
D_{train}	1	45.94	77.84	84.78	46.39	77.91	84.52
	7	49.07	78.72	85.43	49.28	78.57	84.99
	14	49.97	78.78	85.45	49.96	78.70	85.07
$D_{F(a_{n+1})}$	1	49.47	80.53	88.20	49.89	81.04	88.63
	7	49.55	80.76	88.59	49.90	81.19	88.89
	14	49.43	80.78	88.59	49.90	81.19	88.85
D	8	49.48	78.98	85.54	49.39	78.42	84.73
	15	49.07	78.72	85.43	49.28	78.57	84.99
	30	47.90	78.36	85.23	48.55	78.44	84.89
	60	47.13	77.99	85.05	47.81	78.14	84.81
M	8	48.67	78.61	85.21	48.78	78.33	84.66
	16	49.07	78.72	85.43	49.28	78.57	84.99
	32	48.88	78.64	85.37	49.12	78.51	84.94

4.2 Results

4.2.1 Hyperparameter study. As shown in Table 3, we perform a set of experiments to identify the optimal values for the hyperparameters $P = \{\tau$, t-slot, g-slot, $D_{\text{train}}, D_{F(a_{n+1})}, D, M\}$. We explore different values (V) for each hyperparameter, and the selected values for our subsequent experiments are highlighted in bold in Table 3. The details of each hyperparameter are outlined as follows: 1) τ: It is the hyperparameter of Equation 8, adjusting the extent of suppression for training samples with the high-frequency target app. A larger τ indicates a reduced degree of suppression. We opt for a value of 0.1. 2) t-slot: It denotes the granularity used for discretizing the prediction timestamp, measured in minutes. We select a value of 10 minutes. 3) g-slot: It represents the granularity used for discretizing the time gap feature, measured in minutes. We choose a value of 2 minutes. 4) D_{train}: It refers to the days of app usage data used for training. We observed that the results plateau when the number of days of app usage data increases to 14. Thus, we utilize 7 days of app usage data for training. Specifically, we use data from April 1st to April 7th, 2022, for training, and data from April 8th, 2022, for validation. To evaluate the model's generalization, we randomly select one day of test data from each of the V1-data and V2-data. 5) $D_{F(a_{n+1})}$: It denotes the historical days of app usage data utilized to derive the affinity feature. We can observe that the results plateau when the number of days extends to 14. Therefore, we use 7 historical days of app usage data, from March 25th to March 31st, 2022, to obtain the affinity feature. In all subsequent experiments, the user ID is set to 0. 6) D & M: We generate training samples by considering a window size of M with a duration limit of D (measured in minutes). Each training example consists of a total of M apps, ensuring that the combined duration of the apps within the window does not exceed D. Experimentally, we find that setting M to 16 and D to 15 maximizes the number of training samples, yielding optimal results.

Table 4: Results Across Different Features

	V1-data			V2-data		
	Top1	Top3	Top5	Top1	Top3	Top5
$a_{1:n}$	48.87	78.69	85.42	49.11	78.57	85.02
+headset	48.98	78.74	85.49	49.21	78.61	85.08
+WiFi	48.92	78.77	85.46	49.13	78.63	85.04
+install	48.95	78.69	85.43	49.15	78.57	85.04
+screen	49.93	78.87	85.56	50.13	78.70	85.12
+time gap	49.95	79.00	85.58	50.11	78.82	85.15
+sequence	50.76	79.17	85.71	50.88	78.94	85.27
+context	49.11	78.74	85.50	49.33	78.62	85.12
+affinity	49.50	80.74	88.59	49.90	81.24	88.94
CTR-RAD	**51.59**	**81.37**	**88.95**	**51.90**	**81.78**	**89.23**

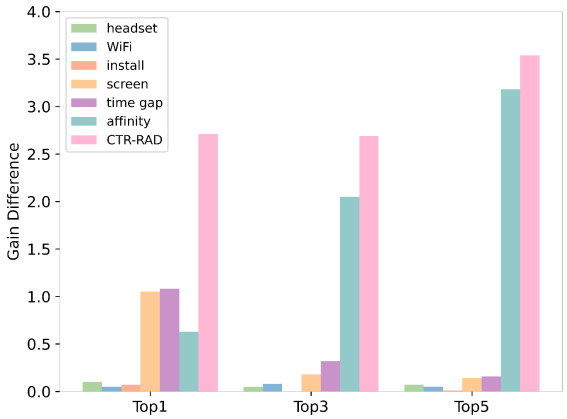


Figure 3: Gain differences across various features.

4.2.2 Ablation study. To highlight the impact of distinct features, we compare CTR-RAD with its variants, as presented in Table 4. Additionally, Figure 3 visually depicts the gain differences between using pure app click sequence input ($a_{1:n}$) and inputs incorporating each feature. Both results demonstrate that each feature improves performance to varying extents. Specifically, the context feature generally has a weak influence, while the sequence feature positively affects Top1, and the affinity feature notably enhances Top3 and Top5. Moreover, the combination of all features as input performs the best across all metrics.

To further investigate the performance contribution of feature interplay, we study the model performance changes using different feature combinations. Table 6 summarizes the results: '0' means

Table 5: Performance Comparison with Baselines

Methods	V1-data			V2-data		
	Top1	Top3	Top5	Top1	Top3	Top5
MFU	36.18	70.97	76.80	36.33	70.62	76.09
MRU	22.29	66.45	74.13	22.71	65.85	73.56
DNN [17]	48.30	76.70	82.58	48.48	76.26	82.00
RNN-Attention [17]	50.39	78.51	84.40	50.33	77.94	83.68
AppUsage2Vec [17]	42.27	74.95	81.90	42.65	74.56	81.22
SGFNN [13]	49.59	77.39	83.51	49.89	76.99	82.86
CTR-RAD (w/o features)	48.87	78.69	85.42	49.11	78.57	85.02
CTR-RAD	**51.59**	**81.37**	**88.95**	**51.90**	**81.78**	**89.23**
Improvement (%)	2.37	3.65	5.40	3.12	4.93	6.64

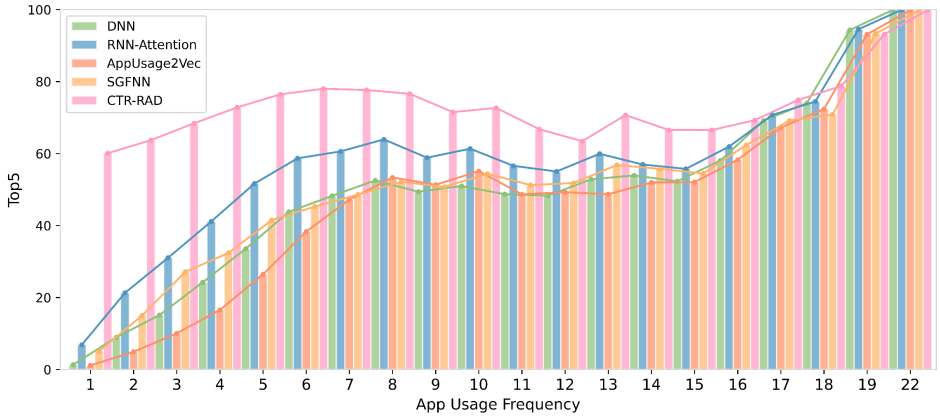


Figure 4: Performance comparison across various app usage frequencies.

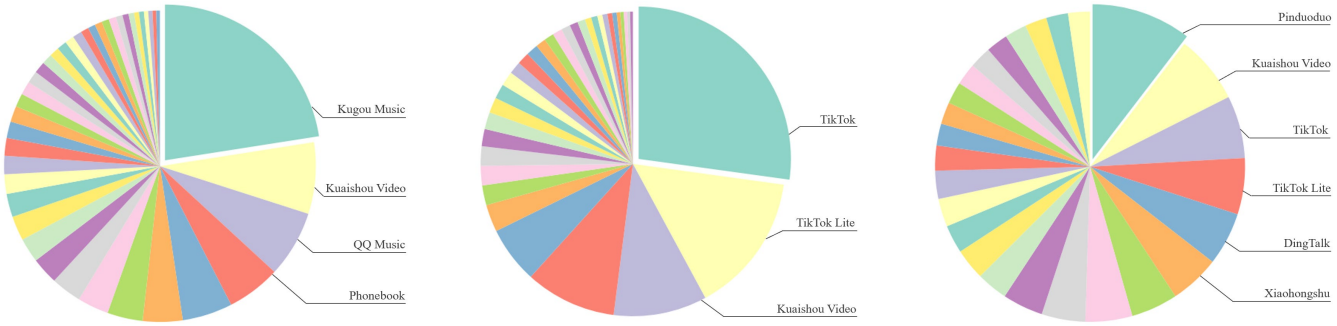
Table 6: Performance with Different Feature Combinations

Features	headset	WiFi	install	screen	time gap	affinity
headset	-	2	1	2	1	1
WiFi	2	-	0	2	1	2
install	1	0	-	1	2	2
screen	2	2	1	-	2	2
time gap	1	1	2	2	-	2
affinity	1	2	2	2	2	-

performance worse than individual features, '1' means performance better than a single feature, and '2' means superior performance compared to both individual features. We found that combining the screen feature with other features, excluding the install feature, consistently yields superior results. Furthermore, combining the install and affinity features demonstrates enhanced performance, potentially because the install feature offsets the lack of historical preference data for newly installed apps. However, combining the install feature with an unrelated WiFi feature yields worse results.

4.2.3 Results compared with baselines. In Table 5, we compare the performance of CTR-RAD with baseline methods across two datasets. The results demonstrate the superior performance of CTR-RAD in terms of Top1, Top3, and Top5 metrics, with particularly notable improvements in the Top3 and Top5 metrics. Specifically, CTR-RAD outperforms state-of-the-art methods by 4.93% in Top3 and 6.64% in Top5. Notably, CTR-RAD (w/o features), which relies solely on the app click sequence input, still outperforms baseline methods that incorporate additional features in terms of Top3 and Top5. Furthermore, we conduct a comparative analysis between CTR-RAD and baseline methods across apps with varying usage frequencies, as depicted in Figure 4. The x-axis represents the base-2 logarithm (log2) of the app usage frequency, calculated by Equation 9, where larger values indicate higher app usage frequency. By comparing the bars, it is evident that CTR-RAD surpasses the classification models, achieving approximately twice the accuracy in predicting low-frequency apps. By comparing the lines, it is observed that as the app frequency increases, the top-5 accuracy of CTR-RAD remains relatively stable across all frequencies. In summary, classification models excel in predicting high-frequency apps but face challenges in predicting medium to low-frequency ones. However, CTR-RAD demonstrates balanced performance across the entire spectrum of frequencies.

4.2.4 Case study. In this section, we conduct a comprehensive analysis of each feature, emphasizing how each one impacts CTR-RAD



(a) Headset (b) WiFi (c) Install

Figure 5: The analysis of context features.

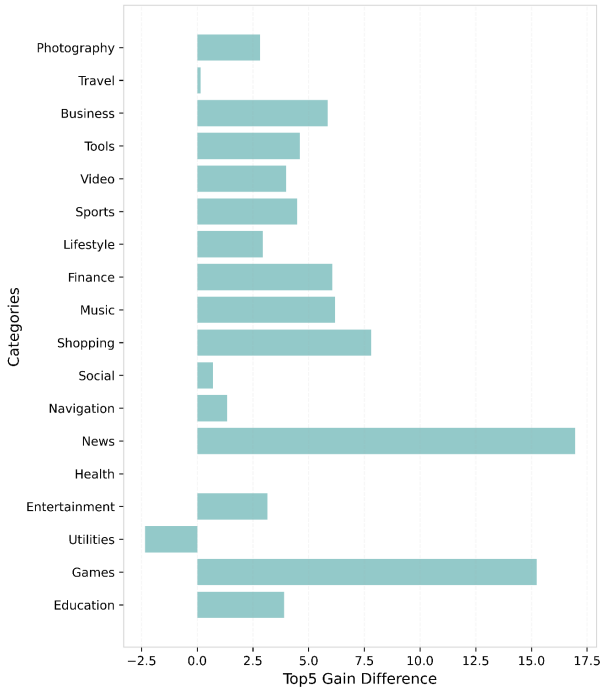


Figure 6: The Top5 gain difference of the affinity feature.

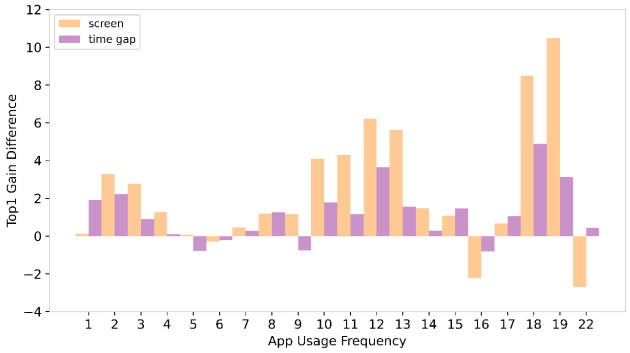


Figure 7: The Top1 gain difference of the sequence feature.

in predicting which apps, categories, and app usage frequencies. We compare the top 1 predicted results incorporating the context features $H(t_{n+1})$, $W(t_{n+1})$, and $L(t_{n+1})$ with those derived solely from the app click sequence input. Figure 5-(a) illustrates the percentage of apps that are accurately predicted by incorporating the $H(t_{n+1})$ feature, while being inaccurately predicted by only utilizing app click sequence input. It can be observed that *Music* or *Video* apps (e.g., *KuGou Music*) can be predicted more accurately when integrating the headset feature. This finding aligns with users' daily habits, indicating that they commonly wear headphones when using these apps.

Figure 5-(b) presents the percentage of apps accurately predicted by incorporating the $W(t_{n+1})$ feature, while being inaccurately predicted by solely relying on the app click sequence input. It can be found that *Video* apps (e.g., *TikTok*) can be predicted more accurately when integrating the WiFi feature. This aligns with users' habits as they commonly use WiFi for resource-intensive apps. Figure 5-(c) shows the percentage of apps accurately predicted by incorporating the $L(t_{n+1})$ feature, but inaccurately predicted by only relying on the app click sequence input. The results indicate that *Tool* or *Shopping* apps (e.g., *PinDuoDuo*, *DingTalk*) can be predicted more accurately when integrating the install feature. This is consistent with our daily habits because users often click on these apps immediately after installing them.

Figure 3 indicates that the affinity feature has a significant impact on Top3 and Top5. Thus, we compare the top 5 predicted results incorporating the affinity feature $F(a_{n+1})$ with those derived only from the app click sequence input. Figure 6 presents the difference in the top-5 accuracy between pure app click sequence input and inputs incorporated with the affinity feature. It demonstrates that *News* and *Games* apps can be predicted more accurately when integrating the affinity feature. These categories exhibit users' preferences; for example, some user groups prefer gaming apps, while others favor news-related apps.

Table 7: The cost against various dataset sizes

D_{train}	#Records	Model size (MB)	Training time (h)
1	660,506	15.6	0.92
7	4,349,482	15.7	2.28
14	8,827,062	16.0	3.95

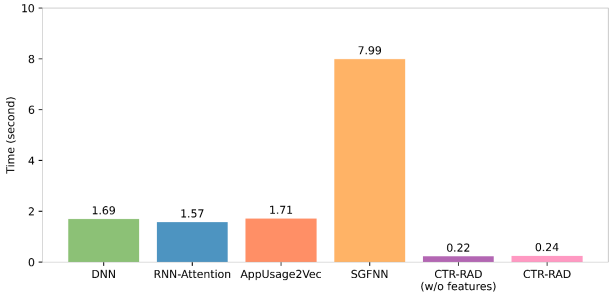


Figure 8: Inference times of different models.

Meanwhile, the sequence feature has a relatively large impact on Top1 compared to Top3 and Top5, as shown in Figure 3. Therefore, we compare the top 1 predicted results incorporating the sequence features $s_{1:n}$ and $g_{1:n}$ with those derived only from the app click sequence input. Figure 7 presents the difference in top-1 accuracy between pure app click sequence input and inputs incorporated with the sequence features across different app usage frequencies. It demonstrates that the sequence feature leads to stable improvements in predicting medium-frequency apps.

4.2.5 Complexity analysis. The algorithmic complexity of CTR-RAD is approximately O(RN), where R represents the number of records and N indicates the number of input apps. Through empirical analysis, as depicted in both Table 7 and Table 3, we compare the performance between D_{train}=7 and D_{train}=14. Despite noticeable increases in both model size and training time with D_{train}=14, we observe slight performance improvements. This suggests a favorable trade-off between model performance and training cost when D_{train}=7. Figure 8 illustrates empirical inference times (for 10,000 Records) of different models. Our approach, which exclusively predicts the CTR of the target app, exhibits notably lower inference time compared to baseline methods. Furthermore, although predicting CTRs for all apps installed on the user's phone (averaging around 60) is necessary, we can enhance efficiency by executing predictions in parallel.

4.2.6 Unseen apps prediction. During actual deployment, we allocate 300 feature values for potentially newly introduced apps. Based on our configuration experience, the number of new apps introduced per user generally remains within this threshold. However, despite allocating an equal number of feature values to the classification models, they still cannot predict unseen apps. Our model predicts the CTR of the target app using the target app itself as input. If the target app is newly introduced, our proposed method can still predict its CTR based on the input app click sequence (e.g., if the new app was clicked in the user's history) and features (e.g., if the new app was clicked immediately after the screen lights up). Table 8 presents the performance of CTR-RAD (trained on V1-data) in predicting those records whose target apps are newly introduced apps in V2-data, totaling 24,086. The results indicate that our approach accurately predicts these apps using solely app click sequences as input and further improves upon integrating additional features.

Table 8: Unseen apps prediction results

Records with unseen target apps	Top1	Top3	Top5
CTR-RAD (w/o features)	24.07	59.05	67.96
CTR-RAD	27.44	72.06	83.35

4.2.7 Limitation analysis. The development of CTR-RAD relies on initial features extracted and constructed from raw user behavioral data. Two potential limitations of our proposed method are: 1) the present feature engineering heavily relies on the expertise of professionals, and 2) parts of the extracted features are platform-specific, which may increase the cost of extending the model to new app platforms.

5 CONCLUSION

In this paper, we conclude that existing research commonly approaches app usage prediction as a classification problem, encountering challenges such as app usage imbalance and the out-of-distribution issue during deployment. To address these limitations, we introduce an innovative click-through rate ranking-based method for predicting app usage. Our approach involves using the app click sequence and three distinct types of features to train a CTR estimation model in the cloud. The trained model is then utilized to predict the CTR of each app installed on the user's smartphone. The decision-making process includes ranking these CTR values and selecting the app with the highest click-through rate as the final prediction. Extensive experiments and analysis demonstrate that our proposed method outperforms state-of-the-art methods, achieving approximately double the accuracy in predicting apps with low usage frequencies. Furthermore, this method has been successfully deployed in the app recommendation system of a leading smartphone manufacturer. In the future, we will further enhance our feature engineering by incorporating additional data, such as Points of Interest (POI) data and physical activity events (e.g., running, walking). We also aim to federate our method with other advanced click-through rate estimation models to achieve better prediction results.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 62206074 and Grant 62072137, in part by the Shenzhen College Stability Support Plan under Grant GXWD20220811173233001, and in part by the National Key R&D Program of China under Grant 2023YFB4503100.

REFERENCES

[1] 2024. *Number of available applications in the Google Play Store from December 2009 to December 2023.* Retrieved June 9, 2024 from https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[2] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting the next app that you are going to use. In *ACM International Conference on Web Search and Data Mining (WSDM)*. 285–294.

[3] Xinlei Chen, Yu Wang, Jiayou He, Shijia Pan, Yong Li, and Pei Zhang. 2019. CAP: Context-aware app usage prediction with heterogeneous graph embedding. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)* 3, 1 (2019), 1–25.

[4] Trinh Minh Tri Do and Daniel Gatica-Perez. 2014. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing (PMC)* 12 (2014), 79–91.

[5] Yonchanok Khaokaew, Mohammad Saiedur Rahaman, Ryen W White, and Flora D Salim. 2021. Cosem: Contextual and semantic embedding for app usage prediction. In *ACM International Conference on Information and Knowledge Management (CIKM)*. 3137–3141.

[6] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)* (2015).

[7] Younghoon Lee, Sungzoon Cho, and Jinhae Choi. 2019. App usage prediction for dual display device via two-phase sequence modeling. *Pervasive and Mobile Computing (PMC)* 58 (2019), 101025.

[8] Tong Li, Tong Xia, Huandong Wang, Zhen Tu, Sasu Tarkoma, Zhu Han, and Pan Hui. 2022. Smartphone app usage analysis: Datasets, methods, and applications. *IEEE Communications Surveys and Tutorials (COMST)* 24, 2 (2022), 937–966.

[9] Nagarajan Natarajan, Donghyuk Shin, and Inderjit S Dhillon. 2013. Which app will you use next? collaborative filtering with interactional context. In *ACM Conference on Recommender Systems (RecSys)*. 201–208.

[10] Yi Ouyang, Bin Guo, Qianru Wang, Yunji Liang, and Zhiwen Yu. 2022. Learning dynamic app usage graph for next mobile app recommendation. *IEEE Transactions on Mobile Computing (TMC)* (2022).

[11] Zhihao Shen, Xi Zhao, and Jianhua Zou. 2023. GinApp: An Inductive Graph Learning based Framework for Mobile Application Usage Prediction. In *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 1–10.

[12] Choonsung Shin, Jin-Hyuk Hong, and Anind K Dey. 2012. Understanding and prediction of mobile application usage for smart phones. In *ACM Conference on Ubiquitous Computing (UbiComp)*. 173–182.

[13] Yizhuo Wang, Renhe Jiang, Hangchen Liu, Du Yin, and Xuan Song. 2023. Sequence-Graph Fusion Neural Network for User Mobile App Behavior Prediction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. Springer, 105–121.

[14] Tong Xia, Yong Li, Jie Feng, Depeng Jin, Qing Zhang, Hengliang Luo, and Qingmin Liao. 2020. DeepApp: Predicting personalized smartphone app usage via context-aware multi-task learning. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 6 (2020), 1–12.

[15] Shijian Xu, Wenzhong Li, Xiao Zhang, Songcheng Gao, Tong Zhan, and Sanglu Lu. 2020. Predicting and recommending the next smartphone apps based on recurrent neural network. *CCF Transactions on Pervasive Computing and Interaction (CCF TPCI)* 2, 4 (2020), 314–328.

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

[17] Sha Zhao, Zhiling Luo, Ziwen Jiang, Haiyan Wang, Feng Xu, Shijian Li, Jianwei Yin, and Gang Pan. 2019. AppUsage2Vec: Modeling smartphone app usage for prediction. In *IEEE International Conference on Data Engineering (ICDE)*. IEEE, 1322–1333.

[18] Xiaoxing Zhao, Yuanyuan Qiao, Zhongwei Si, Jie Yang, and Anders Lindgren. 2016. Prediction of user app usage behavior from geo-spatial data. In *International ACM SIGMOD Workshop on Managing and Mining Enriched Geo-Spatial Data*. 1–6.

[19] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 33. 5941–5948.

[20] Hengshu Zhu, Enhong Chen, Hui Xiong, Kuifei Yu, Huanhuan Cao, and Jilei Tian. 2014. Mining mobile user preferences for personalized context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 4 (2014), 1–27.

[21] Xun Zou, Wangsheng Zhang, Shijian Li, and Gang Pan. 2013. Prophet: What app you wish to use next. In *ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp Adjunct)*. 167–170.