A Novel Context-Aware Mobile Application Recommendation Approach Based on Users Behavior Trajectories

KE ZHU\(^{1,2}\), YINGYUAN XIAO\(^{1,2}\), WENGUANG ZHENG\(^{1,2}\), XU JIAO\(^{3}\), AND CHING-HSIEN HSU\(^{4}\), (Senior Member, IEEE)

\(^{1}\)Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, Tianjin University of Technology, Tianjin 300384, China
\(^{2}\)Engineering Research Center of Learning-Based Intelligent System, Ministry of Education, Tianjin 300384, China
\(^{3}\)College of General Education, Tianjin Foreign Studies University, Tianjin 300204, China
\(^{4}\)Department of Computer Science and Information Engineering, Asia University, Taichung 41354, Taiwan

Corresponding authors: Yingyuan Xiao (yyxiao@tjut.edu.cn) and Wenguang Zheng (wenguangz@tjut.edu.cn)

This work was supported in part by the Major Research Project of National Nature Science Foundation of China under Grant 91646117, in part by the National Nature Science Foundation of China under Grant 61702368, and in part by the Natural Science Foundation of Tianjin, China under Grant 17JCYBJC15200, Grant 18JCQNJC00700, and Grant 61872084.

ABSTRACT
With the rapid development of mobile internet technology, mobile applications (apps) have been rapidly popularized. To facilitate users’ choice of apps, app recommendation is becoming a research hotspot in academia and industry. Although traditional app recommendation approaches have achieved certain results, these methods only mechanically consider the user’s current context information, ignoring the impact of the user’s previous related context on the user’s current selection of apps. We believe this has hindered the further improvement of the recommendation effect. Based on this fact, this paper proposes a novel context-aware mobile application recommendation approach based on user behavior trajectories. We named this approach CMARA, which is the initials acronym of the proposed approach. Specifically, 1) CMARA integrates the heterogeneous information of the target users such as the user’s app, time, and location, into users behavior trajectories to model the users’ app usage preferences; 2) CMARA constructs the context Voronoi diagram using the users’ contextual point and leverages the context Voronoi diagram to build a novel user similarity model; 3) CMARA uses the target user’s current contextual information to generate an app recommendation list that meets the user’s preferences. Through experiments on large-scale real-world data, we verified the effectiveness of CMARA.

INDEX TERMS
Collaborative filtering, app recommendation, voronoi diagram, behavior trajectories.

I. INTRODUCTION
The number of mobile applications (apps) has exploded with the development of the mobile internet, making it difficult for users to find apps that truly match their preferences and causing information overload problems. Although the app Store, Google Play and other mobile application stores have offered some app sorting and categorization methods for users, users still cannot quickly and conveniently find applications that truly address their needs in the face of a large number of applications. As the most successful method for solving the information overload problem, the recommender systems have been a popular research field, and a large number of recommendation methods have emerged.

Although app recommendation systems have achieved great success, since the app recommendation has its own characteristics, there are still some challenges in applying the app recommendation methods to practice. The main challenges of app recommendation are summarized as follows:
- To provide users with more personalized app recommendations, using users’ context information to recommend apps for users has become an important method for app recommendation [4], [5]. There is a large amount of heterogeneous information, such as users, apps, time and geography, in app recommendations, which makes app recommendation more difficult. How to integrate...
these large amounts of heterogeneous information into a unified model is a challenging task.

- Most current app recommendation methods mechanically consider users’ current contextual information, ignoring the historical contextual information; thus, app recommendation systems only provide similar recommendations based on similar current contexts, resulting in inaccurate recommendations. How to consider previous related context information in app recommendation is a difficult task.

- The recommendation results of context-based app recommendation systems are time-sensitive. Therefore, efficiency becomes an inevitable requirement of context-based app recommendation systems.

By analyzing the existing problems in previous research works, we found that few works focus on the relationship between the user’s app usage preferences and the previous related contextual information. The preferences of users for apps are the needs of users for a specific category of apps, so we observe users’ app usage category information and their previous context to obtain the correlation between previous context and users’ app preferences. We combed the data set, divide the time of a day into 00:00-07:00, 07:00-09:00, 09:00-11:30, 11:30-13:30, 13:30-17:30, 17:30-22:00, 22:00-24:00, 7 time slots. The data set contains a total of 31228 positions. We regard a position with a time slot as a context that contains temporal and spatial information. Obviously, our data set contains a total of 7*31228 contexts. We do the following work for each context. First, we statistic the users who have different preferences for apps in current context. Then we look for the previous contextual information of these users. We found that a large amount of previous context of users can explain why users have different app preferences under the same current context. For example, user A appears in the commercial street next to the CBD at 19:15, and user B also appears in the commercial street next to the CBD at 19:47. That is, user A and B have the same spatiotemporal contextual information. User A uses the movie ticketing app, and user B uses the dining app. That is, user A and user B have different preferences for app. The previous context of A is dining at home, and the previous context of B is work in the workplace. This obviously can explain that the user’s current choice of app (the user’s preference for app) is affected by the previous context. That is, user A watches a movie in the commercial street after eats, and user B eats in the commercial street after get off work. How to use the user’s historical app usage information and contextual information to determine the user’s app preference and recommend an app for the user under certain contextual conditions is the focus of our research work.

Based on the above considerations, we propose a context-aware mobile app recommendation approach based on user behavior trajectories, called CMARA. Specifically, 1) CMARA integrates the heterogeneous information of the target user, such as the user’s app, time, and location, into user behavior trajectories to model the user’s app usage preferences; 2) CMARA constructs a context Voronoi diagram using the user’s contextual point and leverages the context Voronoi diagram to build a user similarity model; 3) CMARA uses target users’ current contextual information to generate an app recommendation list that meets the user’s preferences.

CMARA uses user behavior trajectories to model the user’s app usage preference. The user behavior trajectory combines the time, geography, app usage information and the previous related contextual information of the target user, so the user behavior trajectory can well-model the user’s app usage preferences. To mine the user’s app usage preferences requires a collaborative filtering method, i.e., finding users similar to the target user and predicting the similar users’ app preferences as the preferences of the target user.

In the process of finding similar users, we need to solve the problem that how to calculate the similarity between two users. To solve this question, we convert the problem of calculating the similarity of two users in spatiotemporal conditions to the problem of calculating the similarity between the two users’ behavior trajectories. Calculating the similarity of two trajectories calculates the distance between trajectory points of the two trajectories; if the distance between the trajectory points of the two trajectories is smaller, the similarity of the two trajectories is higher.

To address the issue of calculating the distance of the trajectory points, this article innovatively proposes a user behavior trajectory similarity calculation method. Specifically, this method maps the trajectory to the context Voronoi diagram (CVD) and uses the property of the Voronoi diagram to calculate the similarity between the two users’ behavior trajectories.

The Voronoi diagram [1] is a spatial division of a finite point set. Each point in the set corresponds to the Voronoi region (Voronoi cell) that is closest to the point. All of the Voronoi regions do not have overlapping space and are seamlessly spliced together to fill the entire space determined by the point set. The Voronoi diagram is often used to solve problems such as the nearest neighbor query, path planning, image processing, and geographic information processing. The method of mapping the trajectory to the Voronoi diagram and calculating the similarity of the trajectory is based on the following considerations:

- In this problems, the users’ behavior trajectories are not the traditional trajectories but are constructed by the user’s contextual points and the app vector used by the user. In the process of finding similar user trajectories, we only need to determine the relative positional relationship (adjacent or nonadjacent) of the trajectory points to determine whether the two trajectories are similar [29]. The Voronoi diagram has such a property that if two Voronoi cells determined by two points have a common plane (or edge), the two points have a neighbor relationship. The property of the Voronoi diagram fits the need.
In the process of finding the similar users of the target user, it is necessary to calculate the similarity of any two trajectories, which will undoubtedly generate considerable computational overhead. The Voronoi diagram generates an index for the points, which records the neighbor points of those trajectory points. Therefore, when we look for similar trajectories in the Voronoi diagram, we only need to use the index to find those trajectories that have a common trajectory point or a neighbor trajectory point with the target trajectories. We disregard the trajectories that have no common trajectory point or the neighbor trajectory point, thereby greatly reducing the calculation time.

The main contributions of this article are summarized as follows:

- CMARA leverages the users’ behavior trajectories to integrate heterogeneous information such as users, apps and user contexts into a combined space to effectively model users’ app preferences.
- CMARA innovatively uses the users’ context point to design a 3-dimensional context Voronoi diagram and leverages the unique properties of the Voronoi diagram to design a novel and effective user similarity calculation method.
- CMARA uses tensor decomposition to dynamically predict the users’ app preferences and uses parallel tensor decomposition to effectively improve the real-time recommendation.
- We conduct comprehensive experiments by comparing CMARA with four benchmark methods over three real-world data sets.

**II. REALATE WORK**

The methods used by app recommenders focus on collaborative filter-based methods [14], [17], [18], [22], [28], [33]. The idea of the collaborative filter-based recommendation method is to recommend items that similar users like and have not been used by the target users. The collaborative filter-based recommendation method is divided into memory-based recommendation and model-based recommendation. The idea of the memory-based method is to create a similarity model for the user and recommend an item to a user according to the similarity model. In practical applications, the memory-based recommendation method requires explicit feedback information from users to items. If there is no explicit user feedback, the implicit user feedback information should be transformed into explicit feedback information according to certain rules. Feedback information is used as the user’s preference vector, and the user similarity model can be obtained by calculating the similarity of preference vectors among different users. [18] proposed a user similarity calculation method that relies on the user’s review content. [22] proposed an app recommendation approach by highlighting users’ functional requirements and calculating the user similarity by using the user’s functional requirements. [14] considered that a user’s app requirement is driven by short-term needs and constructing the user’s profile model by using the user’s app install sequence information. [17] proposed an app recommendation method based on an app’s permissions and using the app’s category information and the app’s permission information to construct the app’s similarity model. However, the effect of the similarity calculation between different user preferences is less effective when the user-item matrix data are too sparse, resulting in inaccurate recommendation consequences.

To overcome data sparseness, model-based recommendation methods have been proposed. The most popular method is based on matrix factorization recommendation [2], [4], [5], [9]–[12], [16], [17], [19]–[21], [23], [25]–[27], [30], [34], which decomposes the user-app, \( M = R_{m \times n} \) two-dimensional matrix into two \( M_1 = R_{m \times k} \), \( M_2 = R_{k \times n} \) low-dimensional matrices. The two low-dimensional vectors are multiplied to obtain the similarity matrix of the original matrix, and the elements of the similarity matrix represent the user’s preference for an app. Accordingly, apps with a high score in a similarity matrix are recommended to the users. [5] considered the effect of an app’s version information on recommendation and proposed a method to improve recommendation results by incorporating the app’s version information into the user-app matrix calculation. [9] proposed a hybrid matrix factorization method that considers an app’s features and the app user’s usage information. [10] incorporated the user’s app usage and download behavior information and the app’s popularity information into the user-app matrix calculation. Similar to [5], article [11] also considered that an app’s version information is an important factor that affects the recommendation result, but the difference is that they considered that the evolution of the app’s version information can affect the recommendation result, so they proposed an app recommendation method based on the app’s version evolution to improve the recommendation result. [12] proposed a kernel-based matrix factorization method that leverages an app’s categorical information to supplement the user-app matrix. [17], [19], [20] used the app’s permission information, the app’s description information and the app’s feature information to complete the user-app matrix to achieve the app recommendation purpose. The purpose of these methods is to decomposing the original matrix into two low-dimensional matrices simply to combat data sparsity. \( M_1 \) and \( M_2 \) are widely interpreted as implicit vector matrices of users and apps. \( k \) represents the dimension of the implicit vector. These works do not clarify the physical meaning of these low-dimensional implicit vectors and do not consider the impact of the user’s contextual factors on the recommendation results. In addition, [31], [32] leveraged the unique characteristics of the Voronoi diagram to constrain the geography of the target user and then generated the recommendation list. In addition, [36] used the image features of items to mine user preferences.

Furthermore, other works [3], [6]–[8], [13], [15], [35] considered contextual information to be an important factor affecting the recommendation results, and contextual factors
were added to the modeling process to improve the effectiveness of the recommendation. There are three main ideas in these methods. The first method [6], [8] adds context information as additional information into the user-app matrix to model, filters the original user-app matrix, obtains the small matrix under a specific situation, and then calculates the similarity or decomposes the matrix into a filtered matrix. The second method [13], [15] uses the situation information as the precondition information to obtain the probability of the user’s preferences for the app. The third idea [3], [7] extends the user-app two-dimensional matrix into a user-app-context three-dimensional tensor $T = \mathbb{R}^{m \times n \times l}$ and then decomposes it into tensors to complete the recommendation list. This approach is essentially the same as matrix decomposition, and we cannot identify the physical meaning of low-dimensional latent vectors.

### III. PRELIMINARIES

In this section, we present the formal definition of CMARA. This article uses the user app usage logs file to mine the users’ preferences. The format of the log files is shown in Table 1.

**Table 1. User’s app usage logs file.**

| App_id | Device_id | Time stamp | Longitude | Latitude |
|--------|-----------|------------|-----------|----------|
| 1778   | 3098      | 3487       | 115.05    | 30.25    |
| 2359   | 3098      | 3601       | 115.05    | 30.26    |
| 2147   | 3098      | 3654       | 114.96    | 30.39    |
| 1922   | 3098      | 3682       | 114.94    | 30.42    |
| 2147   | 3098      | 3689       | 114.94    | 30.42    |
| 1922   | 3098      | 3702       | 114.94    | 30.42    |

One record in table 1 indicates that a mobile device has used an app at a determined time and geographic location. Based on security considerations, we desensitize the data and assign a device_id and app_id to each mobile device and app to distinguish different users and apps. Since each user corresponds to a device, here we treat a device_id as a user.

**A. SYMBOLS**

- $U$: the set of all users.
- $A$: the app set.
- $C$: the set of context points. Each $c_i \in C$ is a point that is composed of time, longitude and latitude.

The notations and descriptions of the parameters used in CMARA are shown in Table 2.

**Table 2. The descriptions of symbols used in this article.**

| Symbols | Descriptions |
|---------|--------------|
| $u_m$   | The $m$-th user |
| $a_n$   | The $n$-th mobile app |
| $c_k$   | The $k$-th context point |
| $U$     | The set of target users |
| $A$     | The set of mobile apps |
| $C$     | The set of context points |
| $Vor(c_i)$ | The Voronoi cell determined by the $c_i$ |
| CVD     | The context Voronoi diagram |
| SPECV   | The two trajectories’ similarity point of expansion |
| CSP     | The common subsequence points of two trajectories |

According to Definition 1, for all the target users, we built user behavior trajectories that represent the user’s app usage preference. We use Figure 1 to illustrate the users’ behavior trajectories. There are two trajectories in Figure 1, $T_{u_1}$ and $T_{u_2}$, which represent two users using different apps at different context points. In our approach, we use the users’ behavior trajectories to find a user similar to the target user and then build the user’s behavior similarity model.

**Definition 1 (The User Behavior Trajectory):** The user behavior trajectory is a chronological sequence in time, where the points in sequence are composed of a context point.

We use a temporal sequence $T_{u_m} = \{(c_1^{u_m}, a_1^{u_m}), (c_2^{u_m}, a_2^{u_m}), \ldots, (c_n^{u_m}, a_n^{u_m})\}$ to denote a user $u_m$’s ($u_m \in U$) behavior trajectory where $(c_1^{u_m}, a_1^{u_m})$ is the trajectory point, and represents the target user $u_m$ uses an App $a_1 \in A$ at the context point $c_1 \in C$.

**Definition 2 (The Context Voronoi Diagram):** The context Voronoi diagram (CVD) is a 3-dimensional Voronoi diagram composed of a set of context points, where the context points are 3-dimensional points composed of time, longitude and latitude.

The CVD is a 3-dimensional Voronoi diagram composed of $C$, where $C$ is a set of context points in the users’ behavior trajectories of all the target users. We use $vor(C) = \{vor(c_n)|c_n \in C\}$ to denote the CVD, which is composed of $n$ Voronoi regions $vor(c_n)$ in the contextual environment space, $vor(c_n) = \{s \in C | dist(s, c_n) < dist(s, c_m), n \neq m\}$, $dist()$ is the Euclidean distance, and $s$ is a virtual point located in $vor(c_n)$.

We leverage a real-world 3-dimensional context point set to construct a context Voronoi diagram that satisfies Definition 2 and intercept a part of it, as shown in Figure 2. The points in the figure are context points where users use an app.

**Definition 3 (Similar Points of Expansion in the Contextual Voronoi Diagram):** For any two points $a$ and $b$ in the context Voronoi diagram, $vor(a)$ is expanded outward once to obtain a Voronoi diagram with a common plane of $vor(a)$, which is recorded as the expansion space of $vor(a)$. If point

**FIGURE 1. The behavior trajectories of two users, $T_{u_1}$ and $T_{u_2}$.**
b falls in this adjacent space, we regard point b as similar to point a, which is recorded as point b, as the similar point of expansion in the contextual Voronoi diagram (SPECV) of point a.

For the convenience of display, we illustrate Definition 3 with two points in a 2-dimensional Voronoi diagram, as shown in Figure 3. The adjacent space of $\text{vor}(a)$ is represented by a red box, b is located in the adjacent space of $\text{vor}(a)$, and a and b are similarity points.

IV. PROPOSED WORK

A. FRAMEWORK

In this section, we illustrate the CMARA framework.

Given the target users’ historical contextual information and the historical use of app information, our goal is to mine the target users’ preference for an app in a certain context. As in Figure 4, we first extract users’ behavior trajectories from users’ log files and then construct the user similarity model. Next, we use the target users’ similar users to construct the dynamic active area mining model. Third, we determine whether the target user’s current location is within the active area. If the target user’s current location is in the active area, we leverage the collaborative filtering method to predict the target user’s preference for apps; if not, we use dynamic preference prediction to mine user preferences. Finally, we obtain the user app use preferences under contextual conditions and accordingly generate app recommendation lists to recommend to users.

CMARA’s framework comprises four major components: i) user similarity model, ii) dynamic activity area mining model, iii) preference dynamic predicting and iv) app list generation model. Figure 4 illustrates the detailed CMARA framework.

- The user similarity model effectively integrates heterogeneous information such as users, apps, and context into a 3-dimensional Voronoi diagram to model the user’s app usage history and uses the context Voronoi diagram to find users with similar app usage behaviors in the time and geography dimensions of the target user. Compared with traditional methods, this module not only considers the interaction information between users and apps but also considers the impact of context information such as time and location on users’ app usage preferences, which makes our similarity calculation method more effective.

- The dynamic active area mining model mines the geographic area of a user’s app usage according to the app usage data of a user over a time slot. This module effectively mines users’ active area, which allows CMARA to obtain the users’ app usage preferences more accurately.

- The user app preference dynamic prediction dynamically predicts the user’s app usage preference over different time slots using three-dimensional tensors.

- The app list generation model personally generates a recommendation list for the target user based on his/her current location.

B. TIME WINDOW SELECTION

In this section, we present the detailed construction method to select the modeling time window.

When we observe real-world users using app data, we find that the time point of the user’s app usage presents the characteristics of time-point clustering. i.e., most users will use apps intensively during a period of time. For example, a user listens to music using a music app on the way to work and browses the news using the news app; when the user arrives at the workplace, the user no longer uses the mobile device during working hours. Therefore, when we select the time window, we cluster the target user’s app usage time-point and select the time period of the most recent time-point cluster as the time window.

We use Figure 5 to illustrate the above steps. The points in Figure 5 indicate that the user has used an app at a certain time, and these points show the characteristics of time aggregation. We cluster these points to obtain three clusters, each of which has a starting and ending time point. We select the target user’s most recent app usage time cluster as the starting and ending time for modeling, and the starting and ending time points of this cluster are $t_5$ and $t_6$. Our approach is based on the following considerations: the user’s app usage preferences dynamically change with the contextual information.
and related to the previous context information, we need to obtain the user’s latest dynamic contextual information to mine the user’s app usage preferences. Therefore, we cluster the target user’s app usage time points to obtain the user’s latest time contextual information and use the latest time period for modeling.

Through the above method, we obtain the modeling time window of the target user. In this time window, our goal is to obtain similar users of the target user. Next, we provide the construction approach of the user’s similarity model.

C. USER SIMILARITY MODEL

To find users similar to the target users, we design a user similarity model. The construction method of the user similarity model is as follows.

1) We first construct the context points plot, and the details are as follows:
   a) Determine a user set $U$ and a time window $T$, select all of the context point set $C$. The elements $c_k \in C$ are the context points where all target users $u_i \in U$ using the app are located.
   b) In the selected time window $T$, according to the habit of users using the apps, we select minutes as the split unit and split the time window $T$ into $L$ parts, each of which is a unit time window.
   c) For each time window, we randomly scatter $c_k$ on the 3-dimensional space with longitude as the $X$ axis, latitude as the $Y$ axis and time as the $Z$ axis.
   d) We traverse all unit time windows and repeat the above steps to finalize the construction of the context point plot.

2) Next, we leverage the context point set in the previous step as the point set for constructing CVD and construct a Voronoi cell for each point in the point set. The Voronoi cell meets the conditions of Definition 2. All of these Voronoi cells constitute CVD.

3) Then, we construct the user similarity model. The method details are as follows.
   a) According to Definition 1, we convert the user app usage log file data to the user behavior trajectory.
   b) We map all of the users’ behavior trajectories to the CVD according to the way that the contextual information points in the user behavior trajectory points correspond to the CVD points.
   c) In the CVD, if a user behavior trajectory has no CSP and no SPECV with the target user, we regard this user’s similarity as 0 to the target user; if a user behavior trajectory has CSP or SPECV with the target user, we calculate the similarity between the two users.
   d) We traverse all the target users and repeat the above steps to finalize the construction of the user similarity model.

In the above steps, the details of the similarity calculation method are as follows.

VOLUME 9, 2021
**The user similarity calculation method**: Assume \( T_{ui} \) and \( T_{uj} \) are two user behavior trajectories mapped to the CVD. For \( T_{ui} \) and \( T_{uj} \), create the common subsequence point set of two trajectory CSP\(_{ij}\): \( \{ c \in T_{ui}, c \in T_{uj} \} \); for the target user behavior trajectory \( T_{uj} \), create the \( c_{k}^{uj} \)’s SPECV set \( S_{k} : \{ c \in c_{k}^{uj} .SPECV \} \). We use the following equations to calculate the two user’s similarity:

\[
sim_{b}(T_{ui}, T_{uj}) = CS + SP \tag{1}
\]

\[
SP = \sum_{c_{l}^{ui} \in T_{ui}} \left( \frac{\sum_{c_{l}^{ui} \in S_{k}} \text{sim}_{a}(a_{k}^{ui}, a_{l}^{ui})}{|S_{k}|} \right) \tag{2}
\]

\[
CS = \frac{\sum_{c_{l}^{ui}, c_{l}^{uj} \in \text{CSP}_{ij}} \text{sim}_{a}(a_{l}^{ui}, a_{l}^{uj})}{|\text{CSP}_{ij}|} \tag{3}
\]

Based on the consideration that different users have different behavior trajectory lengths, we incorporate the directionality of similarity into the similarity calculation. In Equation 1, the order of \( T_{ui} \) and \( T_{uj} \) means the equation calculates the \( T_{ui} \)’s similarity for \( T_{uj} \).

The idea of similarity calculation is that the more CSP and SPECV of the two user behavior trajectories, and the more similar of the apps used on the context point, the more similar of the two users. In Equations 1, CS refers to the similarity of two trajectories’ CSP, and SP refers to the similarity of the two trajectories’ SPECV. Equation 2 calculates the similarity between of SPECV between \( T_{ui} \) and \( T_{uj} \), and Equation 3 calculates the similarity of CSP between \( T_{ui} \) and \( T_{uj} \). According to the Definition 1, the app \( a \) and the context point \( c \) is one-to-one correspondence, so the value of \( a_{l}^{ui} \) and \( a_{l}^{uj} \) in Equation 2 and 3 is determined by \( c_{l}^{ui} \) and \( c_{l}^{uj} \). \( \text{sim}_{a}(a_{l}^{ui}, a_{l}^{uj}) \) refers to the similarity of apps \( a_{l}^{ui} \) and \( a_{l}^{uj} \) used by the target user in the trajectory point, and we describe the detailed calculation step of \( \text{sim}_{a}(a_{l}^{ui}, a_{l}^{uj}) \) in section IV-D.

**FIGURE 6. Find similar users in the CVD.**

We use Figure 6 to show the relationship between two users’ behavior trajectories \( T_{ui} \) and \( T_{uj} \) in the CVD. For the convenience of display, we use a Voronoi diagram in 2 dimensions to represent the above similarity calculation method, as shown in Figure 7. In fact, our approach is implemented in a 3-dimensional context Voronoi diagram. We use \( T_{ui}, T_{uj} \), and \( T_{uj} \) to denote three trajectories in the Voronoi diagram. Assume the three users use the same apps, which means that the value of \( \text{sim}_{a}(a_{l}^{ui}, a_{l}^{uj}) \) is 1. We use \( \text{sim}_{b}(T_{ui}, T_{uj}) \) to calculate the \( T_{ui} \)’s similarity for \( T_{uj} \). One context point in \( T_{uj} \) is the SPECV of \( T_{ui} \); the result of \( \text{sim}_{b}(T_{ui}, T_{uj}) \) is approximately 0.056. Using the same method, \( \text{sim}_{b}(T_{uj}, T_{uj}) \) is approximately 0.316, and \( \text{sim}_{b}(T_{uj}, T_{uj}) \) is approximately 0.412.

We obtain the user similarity model using the above method.

**D. APP VECTOR SIMILARITY MODEL**

In this section, we propose the app similarity model construction method. The problems we need to solve are as follows: the user’s log file data do not include the user’s rating information for the apps, so we use the user’s app usage frequency, which we regard as the user’s pseudorating score to the app, to build the app vector similarity model. Specifically, we extract the user-app pseudorating matrix from the user’s app usage log file data, and through the matrix decomposition method to obtain the user vector \( U \times m \times d \) and the app vector \( A \times m \times d \), we use \( A \times m \times d \) as the vectors to evaluate similarities between any two apps. Through the Jaccard similarity coefficient to calculate the similarities between any two apps, we obtain the similarity of any two apps.

In the matrix factorization phase, we use \( r_{ij} \) to denote a user \( u_{i} \)’s pseudorating score for an app \( a_{j} \), and we use \( u_{i}^{T} \cdot a_{j} \) to denote the product of the target user’s vector and the target app’s vector. We use variance as the loss function. To prevent overfitting, we add a regularization term \( \lambda(||u_{i}||^{2} + ||a_{j}||^{2}) \).

Then, we need to optimize the following objective function:

\[
e_{ij}^{2} = (r_{ij} - u_{i}^{T} \cdot a_{j})^{2} + \lambda(||u_{i}||^{2} + ||a_{j}||^{2}) \tag{4}
\]

\[
J (u_{i}, a_{j}) = \min \sum_{i=1}^{m} \sum_{j=1}^{m} e_{ij}^{2} \tag{5}
\]

For this optimization problem, we use the gradient descent method to optimize the result. We have the gradient with
K. Zhu et al.: Novel Context-Aware Mobile Application Recommendation Approach Based on Users Behavior Trajectories

FIGURE 8. The convergence results with different \( \varepsilon \).

respect to \( a_j \) and \( u_i \) as:

\[
\frac{\partial J}{\partial a_j} = -2(r_{ij} - u_i^T a_j) a_j + 2\lambda u_i \quad (6)
\]

\[
\frac{\partial J}{\partial u_i} = -2(r_{ij} - u_i^T a_j) u_i + 2\lambda a_j \quad (7)
\]

We update the gradients \( a_j \) and \( u_i \) with the following equations:

\[
a_j = a_j + \varepsilon ((r_{ij} - u_i^T a_j) u_i - \lambda a_j) \quad (8)
\]

\[
u_i = u_i + \varepsilon ((r_{ij} - u_i^T a_j) a_j - \lambda u_i) \quad (9)
\]

We use Figure 10 to illustrate the above method. The points in the figure indicate the location where the target user’s similar user has been. Different colored points indicate increases when the value of \( \lambda \) is different under the fixed \( \varepsilon \) value. As shown in the two figures, when \( \varepsilon = 0.00001 \) and \( \lambda = 0.0003 \), the convergence speed of matrix decomposition is the fastest, so we choose \( \varepsilon = 0.00001 \) and \( \lambda = 0.0003 \) as the parameters of matrix decomposition model.

For the time complexity of the matrix factorization, although the time complexity of this part is high, due to the matrix factorization is deployed in the offline, the high complexity is acceptable. This part update every 90 days according to the [37]. Next, we use the app’s vector to calculate the similarity. We use \( \text{sim}_{a_i}(a_j) \) to denote the similarity of app \( a_i \) and app \( a_j \), which is calculated by the cosine similarity coefficient, which is represented as \( \text{sim}_{a_i}(a_j) = \frac{a_i \cdot a_j}{||a_i|| \cdot ||a_j||} \).

For all of the target user’s vectors, we calculate the similarity between them; finally, we obtain the app vector similarity model.

E. DYNAMIC ACTIVE AREA MINING MODEL

By observing the historical data of the user app, we find that the behavior of each user using the app in different time slots presents a geographic clustering phenomenon. In a certain time slot, due to the limited quantity of data used by the user, it is extremely difficult to mine the user’s app usage area. This paper uses users who have similar app usage behaviors in the two dimensions of time and geography to solve the above problems. The detailed steps of constructing the dynamic active area mining model are as follows:

1) we supplement the app usage data of similar users as the target user data and perform density clustering on the app usage locations of the target users and similar users in one time slot according to longitude and latitude.

2) A minimum circle coverage is then generated from the clusters obtained by clustering.

3) The area formed by these minimum circle covers is used as the target user’s app geographical area in this time slot.

For the values of parameter \( \varepsilon \) and \( \lambda \), we use the method of controlling variables to select the parameters that make the matrix factorization model converge quickly. The converge results are shown in Figure 8 and Figure 9. Figure 8 shows the convergence of loss as the number of iterations increases with different values of \( \varepsilon \) under the fixed \( \lambda \). When \( \varepsilon > 0.00001 \), the iterating of model training is not converging, so we choose the value within the range \( 0 < \varepsilon < 0.00001 \). Figure 9 shows the convergence of loss as the number of iterations
different clusters to which the point belongs. Each cluster is covered by a minimum circle coverage, which is indicated by a dashed line. The minimum circle coverage represents the target user’s active area. From the figure, we can determine whether the user’s current location is in the active area.

**F. USER APP PREFERENCE DYNAMIC PREDICTION**

CMARA generates the app recommendation list according to the current time and location of the target user. In the previous section, we obtain the target user’s active area. If the target user is not located in the active area at the current time, we need to predict the user’s app usage preference at the current time outside the active area. Each app has its own category, such as shopping and games. The category of an app usually implies the user’s app usage preference. In our algorithm, we refer to the categories of apps as user’s app usage preferences. The user’s app usage preference is reflected in the category of the app, so we need to predict the probability that the user will use the app category in the current geography and time. To solve the above problem, we decompose the 3-dimensional tensor composed of user-app usage preference-time to obtain the user’s app usage preference probability in the time slot and location. Our approach uses the method in [24] for tensor decomposition, which allows our system to update the user app preference model frequently without interrupting recommendations. Therefore, our algorithm has real-time performance.

**FIGURE 11.** User app preference dynamic prediction.

Specifically, our initial tensor is composed of time slots, users, and app preferences. The elements in the tensor are the number of users using a certain app category in a certain time slot. The time slot for constructing the tensor is the same as the time slot for constructing the context Voronoi diagram. We use Figure 11 to illustrate the principle of this process. We can extract the latent features of each user, time slot and preference by decomposing the tensor $X$ into three matrices $A$, $B$, $C$ and a core tensor $G$. As is shown in Equation 10.

$$X \approx G \times_A A \times_B B \times_C C = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_p \odot b_q \odot c_r$$ (10)

Here, $A \in \mathbb{R}^{I \times P}$, $B \in \mathbb{R}^{J \times Q}$, and $C \in \mathbb{R}^{K \times R}$ are the factor matrices. $G \in \mathbb{R}^{P \times Q \times R}$ is the core tensor.

**G. OBTAIN RECOMMENDATION LIST**

In this section, we use the idea of collaborative filtering to generate an app recommendation list for the target users. CMARA first determines whether the target user is in the active area at the current time.

1) If the target user’s current location is in the active area, CMARA leverages the collaborative filtering method to generate a top-$N$ recommendation app set, which is based on top-$K$ similar users obtained by our user similarity model. We use a real-world users’ app usage dataset to train CMARA and obtain the values of $K$ and $N$ that when $K = 10$ and $N = 5$, CMARA has the best recommendation effect.

The detailed steps of collaborative filtering adapted to our method are as follows: 1) CMARA uses the user similarity model to choose the top-$K$ similar users. 2) CMARA casts the apps that similar users used and the target users have not used as the candidate apps. 3) CMARA uses Equation 11 to calculate the recommendation rating of candidate apps. 4) CMARA selects the top-$N$ rating apps from the candidate apps as the recommendation list.

$$r_{u_j,a_m} = \sum_{u_i \in S(u_j)} sim_{b}(T_{u_i}, T_{u_j}) \cdot r_{u_i,a_m}$$ (11)

In Equation 11, $r_{u_j,a_m}$ reflects the recommendation rating of app $a_m$ to user $u_j$, $S(u_j)$ means the target user $u_j$ similar users set, $u_i$ is the target user $u_j$ similar user, $sim_{b}(T_{u_i}, T_{u_j})$ means the similarity between $u_i$ and $u_j$, and $r_{u_i,a_m}$ is the pseudorating of user $u_j$ to $a_m$.

2) If the target user’s current location is not in the active area, based on the result of user app preference dynamic prediction, CMARA obtains the target user’s app preference in the current time slot. CMARA selects the preferences with the top-$L$ probability and uses the top-$M$ popular app in each preference as the recommended candidate set. We train CMARA to obtain the best recommendation effect when $L = 5$ and $M = 12$.

**V. EXPERIMENTAL EVALUATION**

In this section, we experiment with the recommendation performance of our proposed method. Particularly, we focus on the following issues: i) what are the advantages of our proposed method in recommendation effect compared with other classical benchmark methods; ii) how can the proposed method improve the recommendation effect?

**A. EXPERIMENTAL DATASET**

Our experiment uses a dataset from the real world, which is presented in the form of a user’s app usage logs file. Each user’s app usage logs file contains the user’s identification and usage time stamp, and the app’s identification, and location information, which means that a user used an app at a particular time and place.

We clean the dataset, delete the users whose usage log files are less than 10 and delete the apps that are used
less than 5 times. Finally, we obtain a dataset that contains 8,198 users, 2,671 apps, 31,228 locations and 405,837 usage log files from 00:00 on May 1, 2018, to 24:00 on May 7, 2018. We use minute as the unit of segmentation to divide the time in the dataset into 10,080 copies. If considering the user-app 2-dimensional matrix, the sparsity of the dataset is 98.38%; if considering the user-app-context 3-dimensional tensor, the sparsity is 99.99%.

| App   | User       | Time stamp | Longitude | Latitude |
|-------|------------|------------|-----------|----------|
| Tik-tok | user1     | 2018/5/1 8:12 | 115.05    | 30.25    |
| Tik-tok | user1     | 2018/5/1 8:57 | 115.05    | 30.26    |
| Wechat | user1     | 2018/5/1 10:03 | 114.96    | 30.39    |
| Tik-tok | user2     | 2018/5/1 8:23 | 114.94    | 30.42    |
| Wechat | user2     | 2018/5/1 11:12 | 114.94    | 30.42    |
| Taobao | user2     | 2018/5/1 15:29 | 114.94    | 30.42    |

Table 3. An example of the dataset.

We use Table 3 to illustrate our dataset structure. One record in the table indicates that a user used an app at a certain time and location. We randomly divide the data set into three parts to form three data sets: Dataset 1, Dataset 2 and Dataset 3. We use the three datasets to evaluate the recommendation effect of CMARA.

B. BENCHMARK METHODS

We compare our method with the following recommendation methods:

- User-based collaborative filtering (UCF). This is a user-based CF method, where user similarity is calculated based on the users’ rating data. We adapt UCF to our problem by using pseudorating information to mine user preferences and generate the recommendation list. We leverage the pseudorating information to find the users similar to the target user and recommend apps that similar users have used and that the target user has not used to the target users.

- Time-order collaborative filtering (TOCF) [6]. TOCF is a collaborative filtering approach that considers the time-order condition and places the time of interaction between user and item into the user’s similarity calculation. We adapt TOCF to our problem, i.e., we use \( U(u) \) to denote the profile of a user \( u \), from \( U(u) \) to obtain an app sequence \( S(u) = (n(u, l_1), n(u, l_2), \ldots, n(u, l_m)) \) that reflects the usage history of \( u \) described in [6], then according to the TOCF method to obtain the recommendation list.

- Probabilistic matrix factorization (PMF). The PMF-based recommendation approach is a classic model-based method in the recommendation domain. We adapt PMF to our problem to obtain the app recommendation list. We use \( U = (u_1, u_2, \ldots, u_N) \) to denote the target user set, we use \( A = (a_1, a_2, \ldots, a_N) \) to denote a candidate app set, and obtain a user-app rating matrix from \( U \) and \( V \). We use the frequency of a target user using an app to denote the user’s rating scores to the app. Then, the PMF method is used to obtain the recommendation list.

\[ \text{Precision} = \frac{TP}{N} \quad \text{(12)} \]
\[ \text{Recall} = \frac{TP}{M} \quad \text{(13)} \]

\[ F_\alpha - \text{measure} \] balances the precision and recall, which is defined as:

\[ F_\alpha - \text{measure} = (1 + \alpha^2)^{-1} \frac{\text{Precision} \cdot \text{Recall}}{\alpha^2 \cdot \text{Precision} + \text{Recall}} \quad \text{(14)} \]

Here, we choose \( \alpha = 2 \), which indicates more emphasis on recall than precision.

C. EVALUATION MEASURES

In our experiments, we split the experimental data into training and test data. We use the training data to train CMARA and use the apps that users use in the test dataset actually used as the ground truth. We recommend an app list for target users, which leads to two evaluation metrics: precision and recall. We recommend a Top-\( N \) recommendation list to users, and the length of the recommendation list is \( N \), so we choose precision@\( N \) and recall@\( N \) to measure the proposed method. However, precision and recall are two indicators: one fall leads to another rise. To synthesize considering the two indicators, we use the \( F_\alpha \)-measure to measure the recommendation quality.

We use \( TP \) to denote the intersection between the recommendation list and ground truth; we use \( N \) to denote the length of the recommendation list, and we use \( M \) to denote the length of the ground truth. The Precision@\( N \) and Recall@\( N \) are defined as follows:

\[ \text{Precision} = \frac{TP}{N} \quad \text{(12)} \]
\[ \text{Recall} = \frac{TP}{M} \quad \text{(13)} \]

\[ F_\alpha - \text{measure} \] balances the precision and recall, which is defined as:

\[ F_\alpha - \text{measure} = (1 + \alpha^2)^{-1} \frac{\text{Precision} \cdot \text{Recall}}{\alpha^2 \cdot \text{Precision} + \text{Recall}} \quad \text{(14)} \]

Here, we choose \( \alpha = 2 \), which indicates more emphasis on recall than precision.

D. EXPERIMENT SET

- **Experiment 1**: We use CMARA and the benchmark methods to generate the recommendation list with different parameters on three datasets and use the evaluation measures precision@\( N \), recall@\( N \) and \( F_\alpha - \text{measure} \) to verify the recommendation effect of CMARA.

- **Experiment 2**: Because CMARA incorporates user behavior trajectory information when constructing the user similarity model, we need to evaluate the impact of user behavior trajectory information on the recommendation effect. We delete the user behavior trajectory.
information of CMARA, and the remaining part is called CMARA-B. We use the CMARA-B to generate a recommendation list and compare the recommendation effect of CMARA-B with CMARA.

**E. PERFORMANCE COMPARISONS**

1) In this section, we present the results of experiment 1 with different parameter values in Figure 12. We compared the precision@$N$, recall@$N$, $F$-measure of CMARA and benchmark methods in three datasets. $N$ is the length of the recommendation list. Figure 12(a) - 12(c), Figure 12(d) - 12(f) and 12(g) - 12(i) show the precision, recall and $F$-measure of the CMARA and benchmark methods’ recommendation results in three datasets, respectively. The experimental results show that the performance of UCF and PMF is poor. The main reason is that both methods only focus on mining user preference information, but they cannot effectively utilize the impact of contextual information on app usage preferences. In addition, the performance of UCF is the worst of all the methods because UCF has other problems in addition, the above: one problem is that the user rating vector does not accurately reflect the relationship between users, and the other problem is that the collaborative filtering method is sensitive to
data sparsity. PMF results are unacceptable because the matrix decomposition cannot clarify the physical meaning of the users and app latent vectors. Both TOCF and TF have good performance. TOCF uses the app usage time sequence to predict the probability that a target user uses an app at a certain time condition. TF constructs a tensor model of the user, the app category and the app category usage time. The commonality between TOCF and TF is that they both effectively utilize the impact of time contextual information on users’ choice of apps, which implies that contextual information is important for app recommendation. TOCF is better than TF. The reason is that the user app usage sequence information can not only make use of the user’s current time context information but also integrate the user’s historical time context information to improve the recommendation.

The performance of CMARA in precision, recall and comprehensive evaluation metric $F$-measure is always better than the benchmark methods mainly for the following reasons: CMARA leverages users’ historical and current contextual information and the app usage information to construct users’ behavior trajectories and effectively mines the users’ app usage preferences. The user behavior trajectories integrate user, app, historical geography, and temporal contextual information into a unified recommendation process. Additionally, we design a novel personalized user similarity calculation method that can quickly and effectively build a model by using the context Voronoi diagram. CMARA combines the user’s current spatiotemporal contextual information in the process of generating recommendation lists for target users.

2) We present the results of experiment 2, as shown in Figure 13. We chose precision@$N$ to measure the effect of the proposed similarity model and the baseline similarity model. As we can see from Figure 13, the recommendation effect of CMARA is superior to that of the CMARA-B method in three different datasets. This is because CMARA not only considers the influence of contextual factors but also considers the user’s app usage behavior information. The users’ behavior trajectories combine the time, geography, app usage information and historical context information of the target user, so the users’ behavior trajectories can well-model the users’ app usage preferences. Therefore, we can conclude that incorporating the users’ behavior trajectories in app recommendation can improve the recommendation effect.

3) We counted the frequency of the Top-20 popular apps used in two cities of Beijing and Shanghai in the dataset. At the same time, we statistic the frequency of CMARA’s recommendation of these Top-20 popular apps onto the two cities. The results of comparing as shown in Figure 14 and Figure 15. We can see from Figure 14 and Figure 15 that, the recommendation list of CMARA has the similar tend with the reality app usage data in the two cities. Furthermore, the recommendation list of CMARA has consideration for the apps located in the long tail, so that the apps in the long tail have more display opportunities, which is conducive to the app diversity of recommendation. This is because CMARA uses user behavior trajectories to construct the recommendation model, which is not only considers the users’ current contextual information, but
also comprehensively considers the users’ historical contextual information, making the model mines out more diversity preferences of users.

VI. CONCLUSION AND DISCUSSION

This paper proposes a novel context-aware mobile application recommendation approach based on user behavior trajectories, called CMARA. The novelty of CMARA is that i) it uses the users’ behavior trajectories to model target users’ app usage preferences, ii) it creatively constructs the context Voronoi diagram to mine the user app usage preferences and iii) it proposes a user similar construction method that maps the users’ behavior trajectories into the context Voronoi diagram to construct the user similarity model. The experimental results show that CMARA has excellent performance in the app recommendation under contextual conditions.

In our future work, we aim to optimize the user’s behavior similarity calculation method by deploying the method on a distributed computing platform to improve the speed of the approach. The proposed work does not consider the target app feature, which is an influencing factor that may improve the recommendation effect. In future work, we incorporate the features of apps into the modeling process to improve the recommendation effect. The proposed approach uses the 3-dimensional Voronoi diagram to build the user’s similarity model. In the real world, the contextual scatter points are connected by the road network. Therefore, in future work, we will construct a Voronoi diagram based on road network distance to further improve the user similarity model.

REFERENCES

[1] F. Aurenhammer, “Voronoi diagrams—A survey of a fundamental geometric data structure,” ACM Comput. Surv., vol. 23, no. 3, pp. 345–405, Sep. 1991.

[2] A. Mihel and R. R. Salakhutdinov, “Probabilistic matrix factorization,” in Proc. NIPS, 2008, pp. 1257–1264.

[3] A. Karatzoglou, L. Baltrunas, K. Church, and M. Böhmer, “Climbing the app wall: Enabling mobile app discovery through context-aware recommendations,” in Proc. CIKM, 2012, pp. 2527–2530.

[4] S. Chatzis, “Dynamic Bayesian probabilistic matrix factorization,” in Proc. AAAI, 2014, pp. 1731–1737.

[5] J. Lin, K. Sagiyaama, M.-Y. Kan, and T.-S. Chua, “New and improved: Modeling versions to improve app recommendation,” in Proc. SIGIR, 2014, pp. 647–656.

[6] Y. Xiao, P. Ai, C.-H. Hsu, H. Wang, and X. Jiao, “Time-ordered collaborative filtering for news recommendation,” China Commun., vol. 12, no. 12, pp. 53–62, Dec. 2015.

[7] V. Kuleshov, A. Chaganty, and P. Liang, “Tensor factorization via matrix factorization,” in Proc. AISTATS, 2015, pp. 507–516.

[8] H. Zhu, E. Chen, H. Xiong, K. Yu, H. Cao, and J. Tian, “Mining mobile user preferences for personalized context-aware recommendation,” ACM Trans. Intell. Syst. Technol., vol. 5, no. 4, pp. 1–27, Jan. 2015.

[9] X. Wu and Y. Zhu, “A hybrid approach based on collaborative filtering to recommending mobile apps,” in Proc. IEEE 22nd Int. Conf. Parallel Distrib. Syst. (ICPADS), Dec. 2016, pp. 8–15.

[10] L. Yezheng, D. Fei, J. Yuanchun, L. Xiao, and W. Qiudan, “A novel APPs recommendation algorithm based on APPs popularity and user behaviors,” in Proc. IEEE 1st Int. Conf. Data Sci. Cyberspace (DSC), Jun. 2016, pp. 584–589.

[11] D. Cao, L. Nie, X. He, X. Wei, J. Shen, S. Wu, and T.-S. Chua, “Version-sensitive mobile app recommendation,” Inf. Sci., vol. 381, pp. 161–175, Mar. 2017.

[12] C. Liu, J. Cao, and J. He, “Leveraging kernel incorporated matrix factorization for smartphone application recommendation,” in Proc. DASFAA, 2017, pp. 459–474.

[13] Y.-J. Han, S.-B. Park, and S.-Y. Park, “Personalized app recommendation using spatio-temporal app usage log,” Inf. Process. Lett., vol. 124, pp. 15–20, Aug. 2017.

[14] V. C. Cheng, L. Chen, W. K. Cheung, and C.-K. Fok, “A heterogeneous hidden Markov model for mobile app recommendation,” Knowl. Inf. Syst., vol. 57, no. 1, pp. 207–228, Oct. 2018.

[15] K. Zhu, L. Zhang, and A. Pattavina, “Learning geographical and mobility factors for mobile application recommendation,” IEEE Intel. Syst., vol. 32, no. 3, pp. 36–44, May 2017.

[16] H. Yin, L. Chen, W. Wang, X. Du, Q. V. H. Nguyen, and X. Zhuo, “Mobi-SAGE: A sparse additive generative model for mobile app recommendation,” in Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE), Apr. 2017, pp. 75–78.

[17] M. Peng, G. Zeng, Z. Sun, J. Huang, and G. Tian, “Personalized app recommendation based on app permissions,” World Wide Web, vol. 21, no. 8, pp. 1–16, 2017.

[18] F. Palomba, P. Salza, A. Ciummeleia, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, “Recommending and localizing change requests for mobile apps based on user reviews,” in Proc. IEEE/ACM 39th Int. Conf. Softw. Eng. (ICSE), May 2017, pp. 106–117.

[19] D. Cao, X. He, L. Nie, X. Wei, X. Hu, S. Wu, and T.-S. Chua, “Cross-platform app recommendation by jointly modeling ratings and texts,” ACM Trans. Inf. Syst., vol. 35, no. 4, pp. 1–27, Aug. 2017.

[20] T. Liang, L. Chen, X. Ying, P. S. Yu, J. Wu, and Z. Zheng, “Mobile application rating prediction via feature-oriented matrix factorization,” in Proc. IEEE Int. Conf. Web Services (ICWS), Jun. 2017, pp. 261–268.

[21] K. Zhu, Z. Liu, L. Zhang, and X. Gu, “A mobile application recommendation framework by exploiting personal preference with constraints,” Mobile Inf. Syst., vol. 2017, pp. 1–9, Jan. 2017.

[22] X. Xu, K. Dutta, A. Datta, and C. Ge, “Identifying functional aspects from user reviews for functionality-based mobile app recommendation,” J. Assoc. Inf. Sci. Technol., vol. 69, no. 2, pp. 242–255, Feb. 2018.

[23] H. Yin, W. Wang, L. Chen, X. Du, Q. V. H. Nguyen, and Z. Huang, “Mobi-SAGE-RS: A sparse additive generative model-based mobile application recommender system,” Knowl.-Based Syst., vol. 157, pp. 68–80, Oct. 2018.

[24] X. Jiao, Y. Xiao, W. Zheng, H. Wang, and Y. Jin, “R2SIGTP: A novel real-time recommendation system with integration of geography and temporal preference for next point-of-interest,” in Proc. World Wide Web Conf., May 2019, pp. 3560–3563.

[25] Y. Xiao, G. Wang, C.-H. Hsu, and H. Wang, “A time-sensitive personalized recommendation method based on probabilistic matrix factorization technique,” Soft Comput., vol. 22, no. 20, pp. 6785–6796, Oct. 2018.

[26] X. Jiao, Y. Xiao, W. Zheng, H. Wang, and C.-H. Hsu, “A novel next point-of-interest recommendation based on simulated user travel decision-making process,” Future Gener. Comput. Syst., vol. 100, pp. 982–993, Nov. 2019.

[27] X. Jiao, Y. Xiao, W. Zheng, L. Xu, and H. Wu, “Exploring spatial and mobility pattern’s effects for collaborative point-of-interest recommendation,” IEEE Access, vol. 7, pp. 158917–158930, 2019.

[28] Y. Xiao, J. Shi, W. Zheng, H. Wang, and C.-H. Hsu, “Enhancing collaborative filtering by user-user covariance matrix,” Math. Problems Eng., vol. 2018, pp. 1–9, Nov. 2018.
KE ZHU received the B.S. degree from Shandong Agriculture University, Taian, Shandong, China, in 2005. He was a Visiting Scholar with the School of Computing, National University of Singapore, from 2009 to 2010. He is currently a Professor with the School of Computer Science and Engineering, Tianjin University of Technology, Tianjin, China, in 2016, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. His research interests include personalized recommender systems and machine learning.

XU JIAO received the B.S., M.S., and Ph.D. degrees from the School of Computer Science and Engineering, Tianjin University, in 2003, 2010, and 2019, respectively. From 2003 to 2010, he was an Engineer with Tianjin Foreign Studies University, Tianjin. He is currently a Lecturer with the Faculty of Fundamental Courses, Tianjin Foreign Studies University, since 2010. His research interest includes personalized recommendation systems.

KE ZHU

WENGUANG ZHENG received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, Sichuan, China, in 2010, the M.S. degree from the National University of Singapore, in 2011, and the Ph.D. degree from the University of New South Wales, Sydney, Australia, in 2016. He is currently an Associate Professor with the School of Computer Science and Engineering, Tianjin University of Technology, Tianjin. His research interests include computer architecture, embedded systems, and machine learning.

CHING-HSIEN HSU (Senior Member, IEEE) is currently a Chair Professor and the Dean of the College of Information and Electrical Engineering, Asia University, Taiwan, and a Professor with the Department of Computer Science and Information Engineering, National Chung Cheng University. His research interests include high performance computing, cloud computing, parallel and distributed systems, big data analytics, and ubiquitous/pervasive computing and intelligence. He has published 200 papers in top journals such as IEEE TPDS, IEEE TSC, ACM TOMM, IEEE TCC, IEEE TETC, IEEE System, IEEE Network, top conference proceedings, and book chapters in these areas. He has been acting as an author/coauthor or an editor/co-editor of ten books from Elsevier, Springer, IGI Global, World Scientific, and McGraw-Hill. He is a Fellow of the Institution of Engineering and Technology.

YINGYUAN XIAO received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, China, in 2005. He was a Visiting Scholar with the School of Computing, National University of Singapore, from 2009 to 2010. He is currently a Professor with the School of Computer Science and Engineering, Tianjin University of Technology, China. His research interests include personalized recommender systems, advanced databases, and data mining. He has published more than 100 journals and conference papers in these areas, including IEEE TC, IEEE TPDS, Future Generation Computer Systems, Information Processing Letters, the Journal of Classification, Soft Computing, WWW, DASFAA, and DEXA. He has served as the Program Chair for WAIM 2013 International Workshop (International Workshop on Location-based Query Processing in Mobile Environments), the Publicity Chair for FSKD2009, and a Program Committee Member for a number of international conferences, including APWeb2011, APSCC2011, IEEE CloudCom2012, and ISI2013.

CHING-HSIEN HSU

XU JIAO

KE ZHU

References:

[29] S. Shang, L. Chen, Z. Wei, C. S. Jensen, K. Zheng, and P. Kalnis, “Trajectory similarity join in spatial networks,” in Proc. VLDB, 2017, pp. 1178–1189.

[30] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, “Temporal collaborative filtering with Bayesian probabilistic tensor factorization,” in Proc. SIAM Int. Conf. Data Mining, Apr. 2010, pp. 211–222.

[31] J. Das, S. Majumder, and P. Gupta, “Voronoi based location aware collaborative filtering,” in Proc. 3rd Nat. Conf. Emerg. Trends Appl. Comput. Sci., Mar. 2012, pp. 179–183.

[32] J. Das, S. Majumder, D. Dutta, and P. Gupta, “Iterative use of weighted Voronoi diagrams to improve scalability in recommender systems,” in Proc. PAKDD, 2015, pp. 605–617.

[33] S. Bai, L. Chen, and J. Wu, “Heterogeneous information network based ranking and clustering of mobile apps,” in Proc. WSDM, 2015, pp. 1–6.

[34] B. Yang, C. Wu, S. Sigg, and Y. Zhang, “CoCo (context vs. content): Behavior-inspired social media recommendation for mobile apps,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2016, pp. 1–6.

[35] M. J. Barranco, J. M. Nogueru, J. Castro, and L. Martinez, “A context-aware mobile recommender system based on location and trajectory,” Management Intelligent Systems (Advances in Intelligent Systems and Computing), vol. 171. Berlin, Germany: Springer, 2012.

[36] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, “What your images reveal: Exploiting visual contents for point-of-interest recommendation,” in Proc. 26th Int. Conf. World Wide Web, Apr. 2017, pp. 391–400.

[37] I. Blair. (2020). Mobile App Download and Usage Statistics. [Online]. Available: https://buildfire.com/app-statistics/#:˜:text=Research%20shows%20that%20an%20average%20of%20three%20apps%20are%20used%20in%20the%20United%20States%2C%20is%20spent%20using%20three%20apps

**K. Zhu et al.: Novel Context-Aware Mobile Application Recommendation Approach Based on Users Behavior Trajectories**