Incorporating contextual information into personalized mobile applications recommendation

Ke Zhu1,2 · Yingyuan Xiao1,2 · Wenguang Zheng1,2 · Xu Jiao3 · Chenchen Sun1,2 · Ching-Hsien Hsu4

Abstract

With the rise of the mobile Internet, the number of mobile applications (apps) has shown explosive growth, which directly leads to the apps data overload. Currently, the recommender system has become the most effective method to solve the app data overload. However, most of the existing recommended methods for apps ignore the app functional exclusiveness features, which makes it difficult to further improve the app recommendation performance. To solve this problem, we propose a personalized context-aware mobile app recommendation approach, called PCMARA. Specifically, (1) PCMARA explores the relationship between contextual information and function of apps and constructs the app contextual factors for app which represent the function of app. (2) PCMARA leverages the app contextual factor to design a novel app similarity model, which enable to effectively eliminate the adverse effects that ignore the app functional exclusiveness. (3) PCMARA comprehensively considers the contextual information of users and apps to generate a recommendation list for users based on the target users’ current time and location. We applied the PCMARA to a real-world dataset and conducted a large-scale recommendation effect experiment. The experimental results demonstrate the superiority of PMARA over the benchmark methods.

Keywords App recommendation · Gaussian mixture model · Item-based collaborative filter · Density clustering

1 Introduction

With the rise of the mobile Internet and the increase in the number of mobile devices (Li et al. 2020), the number of mobile applications (apps) has expanded dramatically. According to Statista1, as of the second quarter of 2020, the number of mobile apps is more than 2.7 million and 1.82 million apps in Google play and App store, respectively. The huge number of apps makes it difficult for people to find the apps that fit their preferences quickly and easily, causing app data overload problems. In recent years, as the most successful method for solving the app data overload problem, the recommender system (RS) has been a research hotspot (Pitt et al. 2021). The implementation of personalized apps recommendation has the following significance. (1) Personalized apps recommendation for users will help users find apps that truly meets their needs, so that users can have a better experience in the mobile Internet environment. (2) Personalized app recommendation can accurately target users for app developers, which help promotes app services and improve app services quality, thereby increasing mobile Internet revenue.

Among the traditional RS methods, the collaborative filtering (CF) based method is widely applied method (Zheng et al. 2010). There are two main CF-based methods, the user-based CF method and the item-based CF method. The CF-based method is based on the following assumption; users with similar items historical experiences will have similar items preferences. Based on this assumption, the CF-based method leverages the similarity between items or users to
mines users preferences. For instance, in book recommendation, if a user likes books related to machine learning, then according to the assumption of CF, the RS infers that the user also likes books related to deep learning, and make recommendation. There is a high probability that the user will accept such recommendations because he/she has preferences for books with similar computer science content, which is consistent with the assumption of item based CF. Hence, one of the important parts of the CF-based methods is the similarity calculation.

However, unlike the items in the traditional recommendation domain, such as film (Xiao et al. 2018), news (Xiao et al. 2015), POI (Jiao et al. 2019b) and so on, apps have functional exclusiveness features, i.e., if a user has downloaded an app with a certain function to meet his/her specific needs in a certain contextual condition, then the user is likely not interested in other apps with similar functions (Liu et al. 2016a). The functional exclusiveness features of the app make the item similarity calculation methods in the traditional recommender system invalid for apps. For example, if a user download an app with weather forecast function, according to the traditional CF method, the recommender system will predict the apps that with similar weather forecast function are also match the user preferences. Obviously, by downloading an app, the user has met his/her requirements for weather forecast; this user is no longer interested in other apps with weather forecast function. We conjecture that the ignorance of app functional exclusiveness in the process of app recommendation model construction is the reason that the recommendation effect is difficult to further improves. Therefore, we explore the app functional exclusiveness features and in anticipation of improving the performance of app recommendation in view of incorporating the contextual information.

We observed the user-app usage log file in the real world and visually analyzed the app usage location data in the form of a heat map. The heat map shows an app usage geographic distribution in a specific time slot and a specific geographical area. The geographical distribution is made up of density contours. Different density contours are represented by different colors.

Based on the observation of users’ use of app data in the real world, we find that within a certain geographic area and time slots, the usage location of the apps exhibits significant spatiotemporal aggregation phenomena. For example, Fig. 1, the app usage location heat map, shows the usage location of two apps under a specific time conditions, and the usage location of both apps shows specific spatiotemporal aggregation shapes. Moreover, some apps with different functions have overlapping phenomena of spatiotemporal aggregation. We obtain the following conclusions by observation:

- The phenomenon of app spatiotemporal aggregation reflects the strong correlation between the function of the app and the contextual information, i.e., the app with a certain function is suitable for the context that meets this function. For example, users use video apps to watch videos at home during nonworking hours but rarely use the video apps during work. This phenomenon occurs because the function of video apps is mainly entertainment, which is suitable for a leisure environment. This phenomenon provides a new idea for us to explore the context characteristics of apps.
- The phenomenon of the spatiotemporal aggregation overlapping between different functional apps also reflects the internal functional complementarity between apps. For example, users prefer to take photographs with a camera app at a tourist attraction and use the retouching tool app to beautify the photographs, and then, they share the photographs on the social app. The phenomenon of jointly using multiple app functions to meet users’ needs reflects the strong functional complementarity of apps in the contextual conditions.

In this paper, the spatiotemporal information is the contextual information. We named the phenomenon of apps’ spatiotemporal aggregation the app contextual factor which is related to the app function. We model the function of app by using the app contextual factor to integrate the app with the contextual information so that the app features can be expressed more abundantly and accurately.

To overcome the problem that ignores the app functional exclusiveness features in app recommendation, we adopt the following method. Based on the aforementioned conclusions, we can regard the functional complementarity of apps under the contextual conditions as a special kind of app similarity. Because we use the app contextual factor to model the function of apps, and the functional complementarity of apps is the similarity of apps in this paper, we can leverage the app contextual factor to build a novel app similarity model, which can makes up for the deficiency of traditional similarity calculation methods that do not consider app functional exclusiveness features.
To explore users’ app usage preferences, we use the assumption similar to the item-based collaborative filtering method, i.e., the apps with functional complementarity to the installed apps of the target users conform to the preferences of the target users. Specifically, we leverage the item-based collaborative filtering idea to find out the apps that have functional complementarity with the target users’ installed apps, which as the candidate recommended apps. Based on this consideration, we can reveal users’ preferences for similar app functions in specific contextual information to explore users’ app preferences more accurately.

In summary, to improve the performance of app recommendation, we proposed a personalized context-aware mobile application recommendation approach, called PCMARA.

PCMARA uses the app contextual factor to represent the spatiotemporal characteristics of apps and uses the app contextual factor to calculate the similarity of apps. Then, PCMARA leverages a user-app-context tensor model to mine target users’ app usage preferences. Finally, PCMARA combines the target user’s current context with the similarity of the apps and uses the item-based collaborative filtering technique to generate the app recommendation list.

The main contributions of this work are summarized as follows:

1. PCMARA effectively mines the geographical distribution characteristics of an app under spatiotemporal conditions and innovatively proposes a method for constructing an app contextual factor. By using app contextual factor, PCMARA reveals the relationship between app and contextual information.
2. PCMARA builds a novel app similarity model based on the app contextual factor. The app similarity model combines the spatiotemporal characteristics of the app and leverages the app functional complementarity to efficiently overcome the app functional exclusiveness.
3. Based on the proposed app similarity model and the target users’ current contextual information, PCMARA uses the item-based collaborative filtering idea to generate an app recommendation list for target users, which meets their app usage preferences.
4. Based on a real-world dataset of users use apps, we verified that the recommendation performance of PCMARA is superior to the benchmark methods through large-scale experiments.

In the rest of the paper, we first present the related work about app recommendations in Sect. 2. Then, we define the problem and symbols in Sect. 3. In addition, in Sect. 4, we introduce the PCMARA and describe the inference method and the recommendation approach. To test the proposed model, we conduct experiments for evaluation in Sect. 5. Finally, we present a discussion and conclusion in Sect. 6.

2 Related work

The main idea of an app recommendation method is based on collaborative filtering recommendation, i.e., to find similar users of the target user and recommend the apps of the similar users to the target user. For example, Wu and Zhu (2016) constructed a user-app feature matrix, used the matrix factorization method to obtain the app factor and then constructed an app similarity model to recommend apps for target users. Palomba et al. (2017) analyzed the sentence structure, semantics and sentiment of the user’s comment information on the app so that users can be classified. Zhu et al. (2017) defined the concept of a location block and uses the similarity of the location block when users use the app to generate a recommendation list for users. The location information of the user is employed to use the app to find similar users to the target user, thereby generating a recommendation list (Zheng et al. 2010). Hu et al. (2018) used user behavior logs to calculate user similarity. In addition, some research works used the app’s own characteristics, such as app review text, app version evolution and other information, to construct an app similarity model and generate a recommendation list for users (Bhandari et al. 2013; Chen et al. 2016; Frey et al. 2017; Hao et al. 2016; Hurtado et al. 2020; Kim et al. 2013; Liu and Wu 2016; Sun et al. 2017; Yang et al. 2016; Zhu et al. 2021; Yankov et al. 2013; Zhu et al. 2013).

In the process of constructing the similarity model, the user or app vector is always extremely sparse, which causes difficulty in constructing the similarity model. Therefore, to solve the problem of data sparseness, the model-based recommendation method has been proposed. Model-based recommendation usually uses additional information generated during the user’s interaction with the app, for example, the user’s time information, location information, comment information, etc., to build a recommendation model. For example, Cao et al. (2010) proposed a matrix factorization app recommendation method based on app popularity to counter the sparse user-app interaction information. In addition to the extensive research in the traditional field (Kou et al. 2014), matrix theory also has important applications in the field of recommender system (Lin et al. 2014), Cao et al. (2017), Yao et al. (2017) proposed app recommendation methods that use app version information. Liu et al. (2016b) proposed to mine the associations between user interactions and contexts captured by mobile devices, or behavior patterns for short, from context logs to characterize the habits of mobile users. Böhmer et al. (2013) presented a framework that allows for usage-centric evaluation considering different stages of application engagement.
Zheng et al. (2014) proposed a topic model-based similarity and recommendation algorithm. Liu et al. (2016a) developed a structural user choice model to learn fine-grained user preferences by exploiting the hierarchical taxonomy of apps as well as the competitive relationships among apps. He and Liu (2017) proposed a novel probabilistic model named the goal-oriented exploratory model (GEM), integrating exploratory behavior identification with personalized item recommendation. Zhu and Cao (2017) proposed the GTRM, a recommendation model that builds the top-N app list by optimizing the metric group-oriented mean average precision.

In addition, there are many works that have explored the contextual information when users interact with apps, such as the influence of time and location on the user’s preference for app selection (Liang et al. 2017). For example, Karatzoglou et al. (2012) proposed the Djinn model, a context-aware collaborative filtering algorithm for implicit feedback data that is based on tensor factorization. Han et al. (2017) proposed a probabilistic method to recommend apps appropriate to the current time and location of a user. The study in Böhmer et al. (2010) gives rise to the context-aware recommendation of apps. Davidsson and Moritz (2011) presented a context-aware recommender system for mobile applications that produces recommendations from the first use. Zhu et al. (2012) leveraged the app usage log file to build a personalized context-aware recommender system. Rodríguez-Hernández et al. (2016) described a framework that facilitates the development of context-aware recommendation systems for mobile environments. Zheng et al. (2014) proposed a broad learning approach for context-aware app recommendation with tensor analysis. Xu et al. (2019) proposed a neural approach that learns the embeddings of both users and apps and then predicts a user’s preference for a given app. Aiming at the problem of data imbalance in the recommendation system, Qiu et al. (2020a, b) gave effective solution. Bobadilla et al. (2020) explored the application of deep learning on recommendation. The use of opinion dynamics models has applications on recommendation (Zha et al. 2020).

3 Preliminaries

In this section, we give the formal description of the dataset and the symbols used in the PCMARA.

3.1 Problem definition

The goal of PCMARA is to build an app similarity model based on the historical location and time information of target users uses apps and leverage the current time and location information of target users to mine app usage preference of target users and generate an app recommendation list.

Through the above analysis and conclusion, to recommend the apps for users under the context condition, we need to solve the following problems: (1) How to define the app contextual factor. (2) How to measure the similarity between apps and construct the app similarity model. (3) How to combine the app similarity model with the user’s current context to generate a recommendation list for target users.

– For the first problem, (i) since the distribution of the app usage location has the spatial and temporal aggregation feature, the density of the location points in the distribution conforms to the Gaussian distribution, so we use the Gaussian distribution to describe the app. (ii) The app usage location distribution has multiple Gaussian density peaks. These Gaussian density peaks constitute the location distribution of the app. Therefore, we use a Gaussian mixture model to describe the app contextual factor. In Sect. 4, we explain the app contextual factor construction method in detail.

– For the second problem, after we obtain all app Gaussian mixture models, we simplify each app’s Gaussian mixture model into joint confidence ellipses that are located in the spatial and temporal environment. For the calculation of the similarity of any two apps, we calculate the intersection area of the joint confidence ellipses of the two apps in the same context and use the Jaccard similarity to obtain the app similarity model.

– For the third problem, (i) we first find apps similar to the apps installed by the target user according to the app similarity model to form an initial app recommendation list by using the item-based collaborating filter idea. (ii) Then, we look for the apps that match the target user’s current context information from the initial recommendation list to form the final app recommendation list.

To address the aforementioned challenges, we will give the specific implementation method of PCMARA in the next section.

3.2 Data description

We use a real-world user app usage log file to model the PCMARA. The log file is composed of App_id, User_id, Time stamp, Longitude and Latitude. Table 1 shows the data structure of the log file.

3.3 Symbols

To facilitate the statement of the content of this paper, we give the notations and descriptions of the symbols used in PCMARA, as shown in Table 2.
In this section, we will describe the PCMARA in detail. First, we introduce the system framework of PCMARA and then explain the key methods and algorithms used for each model in PCMARA.

### 4.1 Framework

In this paper, whether an app meets the target user’s preferences depends on the following two aspects: (1) whether the function of the app matches the target user’s current contextual information and (2) whether the function of the app with the target user installed app’s function is complementary. Therefore, PCMARA uses four major modules to bridge the above issues: the app contextual factor construction model, the app similarity model, user preferences to predict the model and the app recommendation list generation model. We illustrate the framework of PCMARA, as shown in Fig. 2, PCMARA framework.

1. The app contextual factor construction model uses the usage time and geographic location information to build an app context Gaussian mixture model (ACGMM) that matches the context characteristics of the app and uses

---

**Table 1** Users’ app usage log files

| App_id | User_id | Time stamp | Longitude | Latitude |
|--------|---------|------------|-----------|----------|
| 1778   | 3098    | 3487       | 115.05    | 30.25    |
| 2358   | 3098    | 3601       | 115.05    | 30.26    |
| 2147   | 3098    | 3654       | 114.96    | 30.39    |
| 1922   | 2107    | 3682       | 114.94    | 30.42    |
| 2147   | 3098    | 3689       | 114.94    | 30.42    |
| 1922   | 2107    | 3702       | 114.94    | 30.42    |

**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 |
| $T$     | The set of app usage times |
| $E_n$   | The app contextual factor of $a_n$ |
| $e_i^n$ | The $i$-th app confidence ellipse of $a_n$ |
| $L_m$   | The set of usage locations of app $a_m$ |
| $a_k^m$ | The proportion of the $k$-th cluster |
| $x_k^m$ | The center point of the $k$-th cluster |
| $a_k^m$ | The shape parameter of the $k$-th cluster |
the ACGMM’s parameters to represent the app contextual factor.
2. The app similarity construction method maps the app contextual factor to the influence range composed of joint confidence ellipses in a limited geographic area and uses the intersection area of the joint confidence ellipses to calculate the similarity between two apps.
3. The user preference prediction model leverages the tensor model constructed by the target users’ historical contextual information and app usage information to model the target users’ app usage preferences.
4. The app recommendation list generation model generates an app recommendation list for the target user based on the current contextual information and the installed apps of the target user.

The specific details of the four modules are as follows.

4.2 App contextual factor construction model

To represent the functional characteristics of the app, PCMARA uses the contextual information of the app to design the app contextual factor construction method. The detailed model construction steps are as follows:

1. We first divide one day into 6 time slots according to the habits of users using apps (0:00–7:00, 7:00–11:30, 11:30–13:30, 13:30–17:00, 17:00–19:00, and 19:00–23:59).
2. For an app in a certain fixed time slot, we extract all of this app usage location point in this time slot to form spatiotemporal scattered points of the app usage location.
3. We construct a Gaussian mixture model for the spatiotemporal scattered points of the app and then use a joint confidence ellipse to map the Gaussian mixture model to a plane. The joint confidence ellipse under this contextual condition is the app contextual factor of target app.

For step 3, we use the following three steps to build the app contextual factor.

4.2.1 Find the number of density peaks

First, PCMARA needs to determine the number of density peaks of a target app to provide parameter support for subsequent steps.

For a target app usage location point set, we first determine the number of density peaks. In this paper, the method to quickly find the number of cluster center points is employed (Zhou et al. 2018; Rodriguez and Liao 2014). The detailed steps of the method are shown as follows.

1. Given a usage location set of a target app \( a_m \) in a certain time slot: \( \mathcal{L}_m = \{l_1^m, l_2^m, \ldots, l_n^m\} \), we first calculate the coordinates distance between \( l_i^m \) and \( l_j^m \) for all locations, \( d_{ij}^m = \sqrt{(l_i^m - l_j^m)^2} \);
2. We sort all distances \( d_{ij}^m \) in increasing order to determine the 2\% \( d_{ij}^m \) and define it as \( d_{ij}^m \);
3. Then, we calculate the local density \( \rho_i^m \) of each point \( l_i^m \), where the specific calculation method of \( \rho_i^m \) is given as Eq. 1:

\[
\rho_i^m = \sum_{j \in I_S} \exp\left(-\frac{d_{ij}^m}{\sigma_j^m}\right)^2
\]

where \( I_S = \{1, 2, \ldots, n\} \) is the indicator set;
4. Calculate the distance \( \delta_i^m \) of each point \( l_i^m \), where the detailed calculation is given as Eq. 2;

\[
\delta_i^m = \min_{j \in I_S} d_{ij}^m, \quad \delta_i^m \neq \emptyset
\]

\[
\delta_i^m = \max_{j \in I_S} d_{ij}^m, \quad \delta_i^m = \emptyset
\]

\[
I_S = \{k \in I_S : \rho_k^m > \rho_i^m\}
\]

5. For all of the app usage location points \( l_i^m (\rho_i^m, \delta_i^m) \), \( i \in I_S \), calculate the \( \gamma_i^m = \rho_i^m \cdot \delta_i^m \) and sort all \( \gamma_i^m \) in descending order. Select \( \mathcal{K}_m \) as the number of cluster centers of \( a_m \)'s usage location point set, where the \( \mathcal{K}_m \)-th location point is the point before the \( \gamma_i^m \) value sharply decreases. \( \mathcal{K}_m \) is the number of the Gaussian components of the target app \( a_m \).

4.2.2 Construct the ACGMM

Then, we use the EM algorithm (Dempster et al. 1977) to construct the Gaussian mixture model of target app \( a_m \)'s usage location point set \( \mathcal{L}_m \) and record the model parameters. We use the following equations to present the Gaussian mixture model:

\[
p(L_m | \Theta^m) = \sum_{k=1}^{\mathcal{K}_m} \alpha_k^m \cdot \frac{1}{2\pi \sqrt{\sigma_k^m}} \exp(M)
\]

\[
M = \left(L_m - \mu_k^m\right)^T \sigma_k^{m-1} \left(L_m - \mu_k^m\right) - 2
\]

In Eqs. 3 and 4, \( L_m \) is the target app \( a_m \)'s usage location point set, \( \mathcal{K}_m \) is the number of Gaussian components, which is obtained in Sect. 4.2.1; \( \alpha_k^m \) is the proportion of the \( k \)-th cluster in the point set, and \( \sum_{k=1}^{\mathcal{K}_m} \alpha_k^m = 1 \). \( \mu_k^m \) is the cluster center position of the \( k \)-th cluster, and \( \sigma_k^m \) determines the shape of the \( k \)-th cluster. In this section, Algorithm 1 to train the parameters \( \Theta^m (\theta_k^m \in \Theta^m, \theta_k^m = (\alpha_k^m, \mu_k^m, \sigma_k^m)) \) of the Gaussian model is employed.
Algorithm 1: ACGMM construction

Input: Target app \(a_m\)'s usage location in a certain time slot: \(L^m=[l_1^m, l_2^m, \ldots, l_n^m]\); the number of cluster centers: \(K^m\); maximum number of iterations \(J\).

Output: The parameters of the model \(\Theta^m\)

1. Choose a random initial setting for the parameters \(\Theta^m\);
2. Initialization of joint distribution \(P(l_i; \Theta^m)\), and likelihood function \(P(z_i|l_i; \Theta^m)\);
3. for \(j = 1\) to \(J\) do
   4. Calculate the conditional probability expectations for the joint distribution:
      \[
      Q_i(z_i) = P(z_i|l_i; \Theta_j^m)
      \] (3)
   5. \(M(\Theta_j^m, \Theta_j^{m+1}) = \sum Q_i(z_i)\log P(l_i, z_i; \Theta_j^m)\) (4)
   6. Maximize \(M(\Theta_j^m, \Theta_j^{m+1})\) to obtain the \(\Theta_j^{m+1}\):
      \[
      \Theta_j^{m+1} = \arg \max_{\Theta_j^{m+1}} M(\Theta_j^m, \Theta_j^{m+1})
      \] (5)
   7. if \(\Theta_j^{m+1}\) has converged then
      8. the value of \(\Theta^m\) is \(\Theta_j^{m+1}\);
   else
      9. return to E-step to iterate.
   end
9 end

In Algorithm 1, \(z_i\) represents the latent variables of \(l_i\). \(Q_i(z_i)\) is the estimate of the \(z_i\), and \(\sum Q_i(z_i)=1\). We leverage the above method to get the context Gaussian mixture model of the app.

4.2.3 Construct app contextual factor

Finally, we use the joint confidence ellipse \(E_m\) to simplify the ACGMM of the target app \(a_m\) and map the joint confidence ellipse to the plane. The joint confidence ellipse of a target app is the app contextual factor of this target app.

For an ACGMM of a target app \(a_m\), we use a set \(E_m : \{e_1^m, e_2^m, \ldots, e_K^m\}\) to simplify the ACGMM of \(a_m\), where \(e_k^m (e_k^m \in E_m)\) is the confidence ellipse of the \(k\)-th density peak of the target ACGMM that was obtained in Sect. 4.2.2. The specific expression of \(e_k^m\) is shown in Eq. 5. Algorithm 2 describes the construction method of \(e_k^m\) in detail.

\[
E_m = \begin{cases} 
  x_k^m = x_k^m \cos \theta_k^m + x^{e_k^m} \sin \theta_k^m, & \theta_k^m \in (0, 2\pi) \\
  y_k^m = x_k^m \sin \theta_k^m + y^{e_k^m} \cos \theta_k^m, & x_k^m \in (0, 2\pi)
\end{cases}
\] (5)

For the convenience of display, we use Fig. 3, the \(k\)-th confidence ellipse of an app to illustrate an example regarding the \(k\)-th confidence ellipse \(e_k^m\) of \(a_m\). In Fig. 3, \(\mu_k^m\) represents the center point of the \(k\)-th ellipse; \(x_k^m\) and \(y_k^m\) are the long and short axis of the \(k\)-th ellipse, respectively; \(x^{e_k^m}\) and \(y^{e_k^m}\) are the coordinates of the point \(\mu_k^m\) on the X-axis and Y-axis, respectively; and \(\theta_k^m\) is the angle of the long axis and the line that is composed of a point in the ellipse and the \(\mu_k^m\).

It should be noted that the version and function of apps will evolve with the needs changes of users as time goes on, which will lead to change in the contextual factor of the apps. Therefore, it is necessary to update the app contextual factor. According to statistics of Buildfire\(^2\), 71% of app users

\(^2\)https://buildfire.com/app-statistics.
churn within the first 90 days of downloading an app. This metric means that the app developers should update their apps every 90 days at least to keeping users using their apps. According to this fact, PCMARA needs to recalculate the app contextual factor every 90 days in practical applications. By implementing the update strategy, the app contextual factors after update can still describe app features even though the passage of time.

4.3 App similarity construction method

The app similarity construction method is an important part of the PCMARA. We leverage the app contextual factor in a certain time slot to construct the app similarity model. The detailed app similarity model construction steps are given as Algorithm 3.

Algorithm 3 calculates the ratio of the intersection area and its union area between two app contextual factors and records the ratio as the similarity between the two apps. Figure 4/app similarity illustrates the similarity between two apps. The ratio of the area where the black horizontal line is located to the sum of the confidence ellipse areas of the two apps is the similarity of the two apps.

Algorithm 3: App similarity construction

Input: The set of all target app contextual factors: $SE : \{E_1, E_2, \ldots, E_m\}$; a set of random points $P : \{p_1, p_2, \ldots, p_n\}$

Output: The similarity between any two apps.

1. Initialize $n$;
2. the number of points in the intersection area of two apps: $IA = 0$;
3. the number of points in the total area of two apps: $TA = 0$;
4. for $E_i, E_j \in SE$ do
5. Randomly scatter the points in the set of $P$ on the finite plane containing $E_i$ and $E_j$;
6. for $p_k \in P$ do
7. if $p_k \in E_i \cap E_j$ then
8. $IA \leftarrow IA + 1$;
9. else
10. if $p_k \in E_i \cup E_j$ then
11. $TA \leftarrow TA + 1$;
12. end
13. end
14. calculate the similarity $s_{ij}$ between $E_i$ and $E_j$ with the following equation: $s_{ij} = \frac{IA}{IA + TA}$
15. end
16. end

Algorithm 3 uses the Monte Carlo idea (Kroese et al. 2014) to approximate obtain the similarity of two app contextual factors. The Monte Carlo method is widely used in the field of Statistics, economics, computational physics, and so on. It leverages a random method to estimate the probability of an event. Here, $P$ in Algorithm 3 is a uniformly distributed random points set and it is used to estimate $IA$ and $TA$. The ratio of the $IA$ to $(IA + TA)$ is the similarity the two apps. The more number of points in $P$, the more accurate calculation result, and at the same time, the calculation time consumption is more. In order to balance accuracy and efficiency, we use the same $P$ for all apps.

4.4 The user preference prediction model

In PCMARA, we need to predict the probability that the user will use the app in the current time. To solve the above problem, we decompose the three-dimensional tensor composed of the user-app-time to obtain the user’s app usage preference probability in the time slot. Our approach uses the same method with Jiao et al. (2019a) for tensor decomposition.

Specifically, our initial tensor is composed of time slots, users, and apps. The elements in the tensor are the frequency of users using a certain app in a certain time slot. The time slot for constructing the tensor is the same as the time slot for constructing the app contextual factor model. We use Fig. 5, user preference tensor factorization to illustrate the principle of this process. We can extract the latent features of each user, time slot and preference by decomposing the tensor $\chi$ into three matrices...
Incorporating contextual information into personalized mobile applications…

\[ A, B, \text{ and } C \text{ and a core tensor } G, \text{ as shown in Eq. 6.} \]

\[ \chi \approx G \times_1 A \times_2 B \times_3 C = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pq}a_{p}b_{q}c_{r} \quad (6) \]

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.

### 4.5 App recommendation list generation model

The app recommendation list generation model recommends to the target users apps that meet their preference and current contextual information. We know the target user’s current contextual information and the installed apps, and we use the following steps to obtain the app recommendation list.

1. According to the app similarity model, we look for apps that are similar to the target user’s installed apps according to the contextual factor to form a functionally sensitive candidate app set that conforms to the target user’s functional preferences for the apps. We use \( \lambda \) as the criterion for whether two apps are similar. If the similarity between an app and the target user’s installed apps is greater than \( \lambda \), then the app is selected into the functionally sensitive candidate app set. In this paper, \( \lambda \) is set to 0.6, which is determined experimentally in Sect. 5.5.5.

2. Then, we find the apps from the functionally sensitive candidate app set that match the target user’s preferences, according to the preference prediction model of the target user in this time slot to form the time-sensitive candidate app set. This app set conforms to the target user’s time preferences for the apps.

3. We leverage the target user’s current location contextual information to match the apps in the time sensitive candidate app set to form the final candidate app set. In this process, if the app contextual factor of the target app covers the target user’s current contextual information, the app is selected as the final app to be recommended. The final candidate app set conforms to the target user’s spatiotemporal and functional preferences for the apps.

4. Then, we rank the apps in the final candidate app set to form the app recommendation list. We use Algorithm 4 to rank apps in the final candidate app set. In Algorithm 4, the recommendation rate \( R_{im} \) is calculated as Eq. 7.

\[ R_{im} = \frac{a_{k}^{m} \sum_{a_{m} \in AS} S_{mn}}{d_{u_{i}, a_{m}}} \quad (7) \]

where \( a_{k}^{m} \) is the proportion of the \( k \)-th confidence ellipse of \( a_{m} \); \( AS \) is the set of installed apps in \( u_{i} \)’s mobile device; \( S_{mn} \) is the similarity between \( a_{m} \) and \( a_{n} \); \( d_{u_{i}, a_{m}} \) is the distance between \( u_{i} \)’s current location \( l_{u_{i}} \) and the

#### Algorithm 4: App ranking

**Input:** The target user \( u_{i} \)'s current contextual information, current time slot \( l_{u_{i}} \in T \) and current location \( l_{u_{i}} \in L \); the final candidate app set \( A \); the app contextual factor of all apps in \( A \).

**Output:** Ranking of apps in the recommendation list.

1. For \( a_{m} \in A \) in the recommendation list do
2. Extract \( a_{m} \)'s app contextual factor, which is in the target user \( u_{i} \)'s current time slot \( l_{u_{i}} \);
3. For the selected app contextual factor do
4. Extract the \( k \)-th confidence ellipse that \( l_{u_{i}} \) is located in;
5. Calculate the recommendation rate \( R_{im} \) of \( a_{m} \) recommended to \( u_{i} \) with Equation 7;
6. End
7. End
8. Sort all recommendation rates of target apps in descending order.

#### Table 3 The data structure of the dataset

| Dataset          | Beijing | Shanghai | Guangzhou |
|------------------|---------|----------|-----------|
| Duration (Hours) | 168     | 168      | 168       |
| App              |  637    |  667     |  629      |
| User             | 1972    | 1921     | 1967      |
| Record           | 45,707  | 46,973   | 44,367    |
| Location         | 20,582  | 21,693   | 20,117    |

\( a_{m} \)'s \( k \)-th confidence ellipse’s center point, and \( d_{u_{i}, a_{m}} = \sqrt{(\mu_{k} - l_{u})^{2}}. \)

5. Finally, we chose the top-\( N \) apps to form the recommendation list.

### 5 Experiments

In this section, we perform experiments on the recommendation performance of PCMARA. Particularly, we will focus on the following issues: (1) Compared with benchmark methods, what are the advantages of PCMARA in the recommendation effect? (2) What are the advantages of the proposed app contextual factor and app ranking algorithm for improving app recommendation?

#### 5.1 Experimental dataset

Our experimental data are the users’ app usage log files from real-world scenarios in three cities, Beijing, Shanghai and Guangzhou. The datasets contain a total of 5860 users, 701 apps, and 137,047 user-app usage log records. Each app is used at least 10 times, and each user has at least 10 app usage logs. Each user has an average of 23.39 logs. The detailed information of the dataset is shown in Table 3.
5.2 Benchmark methods

We compare PCMARA with the following recommendation methods:

- **User-based collaborative filtering (UserCF).** User CF is a user vector similarity-oriented collaborative filtering approach, where user similarity is calculated based on the users’ rating data. We adapt UserCF to our problem by using frequency information of users use apps to mine user preferences and generate the recommendation list. We leverage the frequency 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 target users.

- **Item-based collaborative filtering (ItemCF).** Item CF is an app vector similarity-oriented collaborative filtering approach. We adapt ItemCF to our problem with the following method. First, we matricize the dataset according to the frequency of users using apps to form a user-app usage matrix. Then, we extract the app vectors from the matrix and use the cosine similarity coefficient to build the app similarity model. Finally, we leverage the app similarity model to generate an app recommendation list for target users.

- **Matrix factorization (MF).** The MF recommendation approach is a classic model-based method in the recommendation domain. We input the user-app usage matrix and optimize the following equation to obtain a nonempty matrix:

\[
\min_{u, a} \sum_{i=1}^{n} \sum_{j=1}^{m} (r_{ij} - u_i a_j)^2 + \lambda_u \sum_{i=1}^{n} ||u_i||^2 + \lambda_v \sum_{j=1}^{m} ||a_j||^2
\]  

In Eq. 8, \(r_{ij}\) is the actual usage frequency information of app \(j\) by the user \(i\), and \(u_i, a_j\) are the latent vectors of the target matrix. The elements in the target matrix are the preference rating of target users for the apps. We obtain an app recommendation list according to the target matrix.

- **Singular Value Decomposition (SVD).** SVD approach is a matrix factorization-based (MF-based) recommendation method. The principle of SVD is the same as MF. Different from the MF method, to reduce the disturbance of matrix filling, SVD adds feedback bias for the latent vectors of users and items so that the model is more in line with the recommended reality. The aim of SVD is to compute the factors for the users and items. We input the user-app frequency rating matrix and optimize the following equation to obtain a nonempty matrix.

\[
\min_{b_i, b_u, q_i, p_u} \sum_{(u, i) \in K} (r_{ui} - \mu - b_i - b_u - q_i \cdot p_u)^2 + \\
\lambda_u \sum_{u} (b_u^2 + ||p_u||^2) + \sum_u (b_i^2 + ||q_i||^2)
\]  

In Eq. 9, \(\mu\) is the rating baseline, and \(b_u\) and \(b_i\) are the biases of the user \(u\) and app \(i\), respectively. \(q_i\) and \(p_u\) are the latent vectors of the target users and apps of the target matrix, respectively. The actual rating of app \(j\) by the user \(i\) is the \(r_{ui}\). The elements in the target matrix are the preference ratings of target users for the apps. We obtain an app recommendation list according to the target matrix.

- **Tensor factorization (TF).** The aim of the model is to compute the factors for the user \(U^{n \times d}\), item \(M^{r \times d}\) and context \(C^{x \times d}\) matrices using historical usage data. The TF approach is also an MF-based recommendation method. The principle of TF is the same as MF. Similar to the above two methods, we use the frequency of a target user using an app according to certain contextual information to denote the app usage information of a target user and construct a user-app-context tensor. Then, we use the TF method to obtain a recommendation list. Here, the elements in the tensor are the app usage information of a user using an app at a certain time.

5.3 Evaluation measures

In our experiments, we split the experimental data into training data and test data. We use the training data to train PCMARA and use the apps that users actually use in the test dataset 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 with a relationship that when one falls, it leads to the other rising. 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 and recall are defined as follows:

\[
\text{Precision} = \frac{TP}{N} \quad (10)
\]
\[
\text{Recall} = \frac{TP}{M} \quad (11)
\]
$F_{\alpha}$ — measure balances the precision and recall, which is defined as:

$$F_{\alpha} = \frac{\text{Precision} \times \text{Recall}}{\alpha^2 \times \text{Precision} + \text{Recall}}$$  \hspace{1cm} (12)$$

Here, we choose $\alpha=1$, which indicates the recall and precision has the same importance.

The PCMARA uses an app ranking algorithm to generate an app recommendation list, so we use the MAP metrics to measure the effect of the proposed app ranking algorithm. The MAP (mean average precision) calculates the average recommendation precision of each recommended app in the recommendation list. If the value of the MAP metrics is higher, it indicates that the sorted recommendation list is similar to the app actually used by the target user. The MAP is calculated as follows:

$$\text{MAP} = \frac{\sum_{u=1}^{U} \text{Ave}P(u)}{U}$$  \hspace{1cm} (13)$$

$$\text{Ave}P(u) = \frac{\sum_{n=1}^{N} (p(n) \times \text{rel}(n))}{\text{Number of relevance app}}$$  \hspace{1cm} (14)$$

In Eq. 13, $U$ is the target users set. $\text{Ave}P(u)$ is the precision of the recommendation list for the target user $u$, and the calculation method is shown in Eq. 14. In Eq. 14, $N$ is the length of the recommendation list for the target user $u$. $p(n)$ is the precision of the top-$n$ recommended apps in the recommendation list. $\text{rel}(n)$ indicates whether the $n$-th recommended app is relevant to the app actually used by the target user. If the $n$-th recommended app is the same as the app actually used by the target user, $\text{rel}(n) = 1$; otherwise, the value of $\text{rel}(n)$ is 0. In our dataset, a user only uses one app in a spatiotemporal context, so the value of “number of relevant apps” in Eq. 14 is 1. When the effect of the ranking algorithm is good, it will cause an increase in the value of the molecular part of Eq. 14, which in turn leads to a higher value of MAP. Therefore, we can use the MAP metric to measure the ranking algorithm of the model.

### 5.4 Experiment set

#### 5.4.1 Experiment 1: recommendation performance

We apply PCMARA and five benchmark methods to the dataset and generate a recommendation list. Then, we compare the recommendation effect on the three metrics of precision, recall and F-measure with different parameters.

#### 5.4.2 Experiment 2: similarity model performance

PCMARA uses the app contextual factor to build a new app similarity model. Therefore, we use the method of controlling variables to verify the performance of the proposed app similarity model on improving the recommendation effect. We extract the app vectors from the user-app usage matrix to calculate the app similarity and leverage the traditional cosine similarity to build a traditional app similarity model. We replace the proposed app similarity model with the traditional app similarity model and name it PCMARA-S. We compare the recommendation effects of PCMARA and PCMARA-S to verify the performance of the proposed app similarity model.

#### 5.4.3 Experiment 3: ranking algorithm performance

PCMARA uses a novel app ranking method to generate an app recommendation list. We use the MAP@N metric to compare the ranking results of the benchmark method with PCMARA to verify the improvement effect of the proposed ranking algorithm on recommendation.

#### 5.4.4 Experiment 4: the performance of different time slots

PCMARA takes into account both the spatiotemporal context information of target users and the app contextual factor when the target users use the app. We compare the recommendation performance of PCMARA with the benchmark methods in different time periods to verify the effect of the app contextual factor on improving app recommendation performance.

#### 5.4.5 Experiment 5: the influence of $\lambda$

PCMARA uses $\lambda$ to determine whether two apps are similar. We need to verify the influence of $\lambda$ on recommendation performance. We implement a comparison of the recommendation performance for precision@3 on datasets of three cities.

### 5.5 Performance comparisons

#### 5.5.1 Results of experiment 1

In this section, we present the results of experiment 1 with different parameter values in Fig. 6, recommendation performance results.

We compared the precision@N, recall@N, and F-score of PCMARA and benchmark methods in three cities. The abscissa in the figure is the length of the recommendation list. Figure 6(a–c), (d–f) and (g–i) shows the precision, recall and $F$-measure of the PCMARA and benchmark methods’ recommendation results in three cities, respectively.

The experimental results show that the performance of UserCF and ItemCF is poor. This is because, first, the two CF-based methods use user-app interaction information to mine user preference information while ignoring the users’ context information when using the app, which is an impor-
The comparison of recommendation performance between PCMARA and benchmark methods

Tant factor that affects the user’s app preference. Second, the matrix composed of users and apps is extremely sparse, which leads to the inability of the two models to accurately mine user app usage preferences. In addition, the performance of ItemCF is the worst of all the methods because there are other problems in ItemCF. In particular, because of the functional exclusiveness of apps, only app vectors are used to build the app similarity model which leads to app recommendation lists with similar functions that will not be selected by users, resulting in a poor recommendation effect.

MF, SVD and TF all show relatively good recommendation performance. Among them, the common point of MF and SVD is that they use the idea of matrix factorization to supplement the user-app sparse matrix so that nonsparse vectors can more accurately reflect user preferences. The recommendation performance of TF is always better than that of MF and SVD. This is because TF builds user-app-time context information in the three-dimensional tensor model, which not only considers the interaction information between the user and the app but also considers the context in which
Fig. 7  The comparison of recommendation performance between PCMARA and PCMARA-S

Fig. 8  The comparison of ranking performance between PCMARA and benchmark methods

the user uses the app. The information is integrated into the model building process, which effectively mines the user’s app usage preferences under specific time conditions.

PCMARA always outperforms the benchmark method in terms of precision, recall and the comprehensive evaluation F-score metric. The main reason is that PCMARA incorporates the time and geographic information of the users and apps when users are using the app into the construction of the user preference model, effectively mining the user’s app preferences. In addition, we designed a novel app similarity calculation method, which can effectively use app contextual factors to establish a similarity model. In the process of generating a recommendation list for the target user, PCMARA not only combines the user’s current temporal and spatial context information but also considers the contextual characteristics of the app itself, which effectively overcomes the functional exclusiveness problem of the app.

5.5.2 Results of experiment 2

We present the results of experiment 2, as shown in Fig. 7, results of ablation experiments. We chose precision@N to measure the effect of the proposed similarity model and the baseline similarity model. As seen in Fig. 7, the recommendation effect of PCMARA is superior to that of the PCMARA-S method in the datasets of three cities.

The reasons for the experimental results are as follows. First, the app contextual factor affects the app usage preferences of target users. The target users are more inclined to choose those apps that have appropriate app contextual factors. When constructing the app similarity model, the app contextual factor is incorporated in the model construction and will enable PCMARA to achieve better performance in mining the users’ app usage preferences. Second, PCMARA leverages the app functional complementarity to design the similarity model. The recommended apps that have functional complementarity with the target users are more likely to meet the target users’ app usage needs and preferences. Therefore, these apps are more favored by the target users.

5.5.3 Results of experiment 3

We present the results of experiment 3, as shown in Fig. 8, ranking performance. We chose MAP@N to measure the performance of the proposed ranking algorithm and the benchmark methods. As seen in Fig. 8, the recommendation effect of PCMARA using the app ranking algorithm is superior to that of the benchmark methods in the datasets of three cities.
The reasons for the experimental results are as follows. First, when designing the app ranking algorithm, we not only considered the influence of the target users’ spatiotemporal context information when using the app on the users app usage preferences but also considered the influence of the coverage of the target app contextual factor on the target users’ app usage preferences, which leads to PCMARA achieving better performance in mining users’ app usage preferences. Second, the app ranking algorithm of PCMARA uses Eq. 7 to quantitatively calculate the functional complementarity relationship between the target app and the apps installed by the target user, which provides accurate user preference calculation criteria for the PCMARA model so that PCMARA can mine the users’ app usage preferences more accurately.

5.5.4 Results of experiment 4
We present the results of experiment 4, as shown in Fig. 9, performance at different time slot. We chose precision@N to measure the recommendation performance of PCMARA and benchmark methods in different time slots. During the model design process of PCMARA, a day is divided into 6 time slots, and the abscissa in Fig. 9 represents different time slots. As seen in Fig. 9, the recommendation effect of PCMARA is superior to that of the benchmark methods in the datasets of three cities. Moreover, the recommendation performance of PCMARA in the time range of 7:00-19:00 has a great advantage over the benchmark methods.

The reasons for the experimental results are as follows. The recommendation results of UserCF, ItemCF, MF and SVD are unacceptable. This is because these four methods do not consider the spatiotemporal context information of target users using apps when constructing the model, which makes these methods unable to effectively recommend apps for target users in different time conditions. TF is relatively good in recommendation performance because TF considers the time contextual information and enriches the information dimensions required for the model construction. PCMARA performs best on recommendation performance in different time slots. PCMARA takes into account the time, geographic information and app contextual factor when the target users use apps to provide a more personalized and reasonable app recommendation list for target users in different time conditions.

5.5.5 Results of experiment 5
In this section, we present the results of experiment 5 with the datasets of three cities to determine the parameter values of λ in the app recommendation list generation model of PCMARA. The abscissa of Fig. 10 is from 0.1 to 0.9, which means that the value of λ is from 0.1 to 0.9 in PCMARA in the three datasets. As seen from the result, the performance of the recommendation shows a trend of first increasing and then decreasing as λ increases. When the value of λ is 0.6, the recommendation performance of PCMARA is the best. Hence, we choose the value of λ as 0.6 for PCMARA.
6 Discussion and conclusion

In this section, we first conduct further discussions of PCMARA and then give the conclusion and future work about PCMARA.

6.1 Impact of population density

The datasets in the experiments are cities with high population density. We need to discuss the performance of PCMARA deployed in low-population-density regions.

PCMARA mainly uses two parts to generate the app recommendation list. One part is the collaborative filtering part, which leverages app contextual factors to generate a candidate app list for users through item-based collaborative filtering ideas. The other part is the tensor factorization part, which uses the tensor data composed of user-app-time to generate a candidate app list for users through tensor factorization model.

When PCMARA is deployed in the low-population-density regions, although the user-app data may be more sparsity, it can still mine user’s app usage preference for the following reasons. (1) The app contextual factor constructed by rich logs file data can effectively reflect the app characteristics, so the collaborative filtering part can still mine user’s geographical preference of using apps under the sparse user-app data conditions. (2) The tensor factorization model has natural advantages in processing the sparse data (Jiao et al. 2019a). It leverages the tensor factorization technique to train a dense tensor which contains the same amount of information with the original tensor. Hence, when facing with the sparse data, the tensor factorization part can still effectively combat the adverse effects of sparse tensor on mining user preferences.

6.2 Impact of other additional information

PCMARA only uses the contextual information to explore the users app usage preferences. In this section, we give the discussions about the impact of other additional information on app recommendation performance.

The reasons that we only use the contextual information are as follows: (1) There are many factors that impact the app recommendation performance, such as app description, app version, contextual information, and so on. According to different impact factors, researchers have different research perspectives. Our work is from one of the perspectives, i.e., explore the impact of contextual information to user preferences for apps. The contextual information is the main impact factor in our work. (2) We explore the user preferences based on logs file which contains contextual information. The additional data collection from logs file has often certain difficulties, because the logs file only contains limited contextual information. Therefore, the proposed PCMARA model only uses contextual information.

While the limited information used, luckily for PCMARA model has achieved better app recommendation performance than existing methods under the contextual conditions, due to the consideration of app functional exclusiveness features. It should be noted that it does not mean that other additional information is not important for app recommendation. Contrary, it is foreseeable that comprehensive consideration of various additional information will improve the prediction performance of RS.

Incorporating the other additional information into the recommendation modeling is a challenging and meaningful research work (Kou et al. 2021). We intend to consider incorporating more additional information into the construction of the app recommendation model in our future work.

6.3 Conclusion and future work

This paper proposes a personalized context-aware mobile app recommendation approach, called PCMARA. The novelty of PCMARA is that (1) it uses the app contextual factor to model the function of the target app, (2) it creatively constructs the app functional similarity model, and (3) it provides an app recommendation list for target users that meet the preferences of target users’ current contextual information. The experimental results show that PCMARA has excellent performance in app recommendations under contextual conditions.

One of the most outstanding features that differentiates our approach from existing works is that PCMARA incorporates the contextual feature in the modeling. The app contextual factor reflects the app functional feature. Since the app contextual factor combines the contextual information and the functional information of apps, it can effectively avoid the adverse impact of app functional exclusiveness features on app recommendation. Incorporating the app contextual factor into app recommendation can achieve the purpose of improving the effect of app recommendation.

PCMARA leverages the item-based collaborating filtering idea to explore users preferences with location and leverages the tensor factorization to explore users preferences with time. In our future work, we aim to optimize the app contextual factor construction method and the tensor factorization method by deploying the methods on a distributed computing platform to improve the calculation speed of the approach.

PCMARA considers the contextual information of users and apps, and the sequential information of users using apps is also a feature that affects the app recommendation performance. Therefore, in the future work, we will consider incorporating user sequence features into the construction of the recommendation model to improve the recommendation performance.
Acknowledgements This work is supported by Tianjin “Project + Team” Key Training Project under Grant No. XC202022, the National Nature Science Foundation of China under Grant No. 61702368, and the Natural Science Foundation of Tianjin under Grant No. 18JQNJOC0700.

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by KZ, WZ and XJ. The first draft of the manuscript was written by KZ, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Human and animals rights This article does not contain any studies with human participants or animals performed by any of the authors.

References

Bhandari U, Sugiyama K, Datta A, Jindal R (2013) Serendipitous recommendation for mobile apps using item-item similarity graph. In: Asia information retrieval symposium, Springer, pp 440–451.

Bobadilla J, Lara-Cabrera R, González-Prieto Á, Ortega F (2020) Deep-fair: deep learning for improving fairness in recommender systems. arXiv preprint arXiv:2006.05255.

Böhmer M, Bauer G, Krüger A (2010) Exploring the design space of context-aware recommender systems that suggest mobile applications. In: 2nd Workshop on context-aware recommender systems, Citeseer, vol 5.

Böhmer M, Ganev L, Krüger A (2013) Appfunnel: a framework for usage-centric evaluation of recommender systems that suggest mobile applications. In: Proceedings of the 2013 international conference on Intelligent user interfaces, pp 267–276.

Cao D, Nie L, He X, Wei X, Shen J, Wu S, Chua TS (2017) Version-sensitive mobile app recommendation. Inf Sci 381:161–175.

Cao H, Bao T, Yang Q, Chen E, Tian J (2010) An effective approach for mining mobile user habits. In: Proceedings of the 19th ACM international conference on information and knowledge management, pp 1677–1680.

Chen N, Hoi SC, Li S, Xiao X (2016) Mobile app tagging. In: Proceedings of the ninth ACM international conference on web search and data mining, pp 63–72.

Davidsson C, Moritz S (2011) Utilizing implicit feedback and context to recommend mobile applications from first use. In: Proceedings of the 2011 workshop on context-awareness in retrieval and recommendation, pp 19–22.

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc: Ser B (Methodological) 39(1):1–22.

Frey RM, Xu R, Ammendola C, Moling O, Giglio G, Ilic A (2017) Mobile recommendations based on interest prediction from consumer’s installed apps-insights from a large-scale field study. Inf Syst 71:152–163.

Han YJ, Park SB, Park SY (2017) Personalized app recommendation using spatio-temporal app usage log. Inf Process Lett 124:15–20.

Hao Y, Wang Z, Xu X (2016) Global and personal app networks: Characterizing social relations among mobile apps. In: 2016 IEEE international conference on services computing (SCC), IEEE pp 227–234.

He J, Liu H (2017) Mining exploratory behavior to improve mobile app recommendations. ACM Trans Inf Syst (TOIS) 35(4):1–37.

Hu J, Liang J, Kuang Y, Honavar V (2018) A user similarity-based top-n recommendation approach for mobile in-application advertising. Expert Syst Appl 111:51–60.

Hurtado R, Bobadilla J, Gutiérrez A, Alonso S (2020) A collaborative filtering probabilistic approach for recommendation to large homogeneous and automatically detected groups. Int J Interact Multim Med Artif Intell 6:90–100.

Jiao X, Xiao Y, Zheng W, Wang H, Hsu CH (2019a) A novel next point-of-interest recommendation system based on simulated user travel decision-making process. Future Gener Comput Syst 100:982–993.

Jiao X, Xiao Y, Zheng W, Wang H, Jin Y (2019b) R2sigtp: A novel real-time recommendation system with integration of geography and temporal preference for next point-of-interest. In: The world wide web conference, pp 3560–3563.

Karatzoglou A, Baltrunas L, Church K, Böhmer M (2012) Climbing the app wall: enabling mobile app discovery through context-aware recommendations. In: Proceedings of the 21st ACM international conference on Information and knowledge management, pp 2527–2530.

Kim J, Kang S, Lim Y, Kim HM (2013) Recommendation algorithm of the app store by using semantic relations between apps. J Supercomput 65(1):16–26.

Kou G, Ergu D, Shang J (2014) Enhancing data consistency in decision matrix: adapting hadamard model to mitigate judgment contradiction. Eur J Oper Res 236(1):261–271.

Kou G, Xu Y, Peng Y, Shen F, Chen Y, Chang K, Kou S (2021) Bankruptcy prediction for smes using transactional data and two-stage multiobjective feature selection. Decis Support Syst 140:113429.

Kroese DP, Brereton T, Taimre T, Botov ZI (2014) Why the monte carlo method is so important today. Wiley Interdiscip Rev Comput Stat 6(6):386–392.

Li T, Kou G, Peng Y (2020) Improving malicious URLs detection via feature engineering: linear and nonlinear space transformation methods. Inf Syst 91:101494.

Liang T, He L, Lu CT, Chen L, Philip SY, Wu J (2017) A broad learning approach for context-aware mobile application recommendation. In: 2017 IEEE international conference on data mining (ICDM), IEEE, pp 955–960.

Lin J, Sugiyama K, Kan MY, Chua TS (2014) New and improved: modeling versions to improve app recommendation. In: Proceedings of the 37th international ACM SIGIR conference on research & development in information retrieval, pp 647–656.

Liu B, Wu Y, Gong NZ, Wu J, Xiong H, Ester M (2016a) Structural analysis of user choices for mobile app recommendation. ACM Trans Knowl Discov Data (TKDD) 11(2):1–23.

Liu CL, Wu XW (2016) Large-scale recommender system with compact latent factor model. Expert Syst Appl 64:467–475.

Liu Y, Du F, Jiang Y, Liu X, Wang Q (2016b) A novel apps recommendation algorithm based on apps popularity and user behaviors. In: 2016 IEEE first international conference on data science in cyberspace (DSC), IEEE, pp 584–589.

Palomba F, Salza P, Ciurumelea A, Panichella S, Gall H, Ferrucci F, De Lucia A (2017) Recommending and localizing change requests for mobile apps based on user reviews. In: 2017 IEEE/ACM 39th international conference on software engineering (ICSE), IEEE, pp 106–117.

Pitt S, van Meelis LM, Scaife E, Pitt J (2021) No app is an Island: collective action and sustainable development goal-sensitive design. Int J Interact Multim Artif Intell 6(5):24–33.

Qu J, Ji W, Rudes IJ, Gao H (2020a) Asynchronous sampled-data filtering design for fuzzy-affine-model-based stochastic nonlinear systems. In: IEEE transactions on cybernetics.
Qiu J, Wenqiang J, Hak-Keung L, Meng W (2020b) Fuzzy-affine-model based sampled-data filtering design for stochastic nonlinear systems. In: IEEE transactions on fuzzy systems
Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. Science 344(6191):1492–1496
Rodriguez-Hernández CM, Ilarri S (2016) Pull-based recommendations in mobile environments. Comput Stand Interfaces 44(1):185–204
Sun Z, Ji Z, Zhang P, Chen C, Qian X, Du X, Wan Q (2017) Automatic labeling of mobile apps by the type of psychological needs they satisfy. Telemat Inform 34(5):767–778
Wu X, Zhu Y (2016) A hybrid approach based on collaborative filtering to recommending mobile apps. In: 2016 IEEE 22nd international conference on parallel and distributed systems (ICPADS), IEEE, pp 8–15
Xiao Y, Ai P, Wang H, Hsu CH, Li Y (2015) Enrs: An effective recommender system using Bayesian model. In: International conference on database systems for advanced applications. Springer, pp 531–535
Xiao Y, Wang G, Hsu CH, Wang H (2018) A time-sensitive personalized recommendation method based on probabilistic matrix factorization technique. Soft Comput 22(20):6785–6796
Xu Y, Zhu Y, Shen Y, Yu J (2019) Leveraging app usage contexts for app recommendation: a neural approach. World Wide Web 22(6):2721–2745
Yang B, Wu C, Sigg S, Zhang Y (2016) Coco (context versus content): Behavior-inspired social media recommendation for mobile apps. In: 2016 IEEE global communications conference (GLOBECOM), IEEE, pp 1–6
Yankov D, Berkhin P, Subba R (2013) Interoperability ranking for mobile applications. In: Proceedings of the 36th international ACM SIGIR conference on research and development in information retrieval, pp 857–860
Yao Y, Zhao WX, Wang Y, Tong H, Xu F, Lu J (2017) Version-aware rating prediction for mobile app recommendation. ACM Trans Inf Syst (TOIS) 35(4):1–33
Zha Q, Kou G, Zhang H, Liang H, Chen X, Li CC, Dong Y (2020) Opinion dynamics in finance and business: a literature review and research opportunities. Financ Innov 6(1):1–22
Zheng V, Cao B, Zheng Y, Xie X, Yang Q (2010) Collaborative filtering meets mobile recommendation: a user-centered approach. Proc AAAI Conf Artif Intell 24:236–241
Zheng X, Ding W, Xu J, Chen D (2014) Personalized recommendation based on review topics. Serv Oriented Comput Appl 8(1):15–31
Zhou Y, Huang C, Hu Q, Zhu J, Tang Y (2018) Personalized learning full-path recommendation model based on LSTM neural networks. Inf Sci 444:135–152
Zhu H, Chen E, Yu K, Cao H, Xiong H, Tian J (2012) Mining personal context-aware preferences for mobile users. In: 2012 IEEE 12th international conference on data mining, IEEE, pp 1212–1217
Zhu H, Chen E, Xiong H, Cao H, Tian J (2013) Mobile app classification with enriched contextual information. IEEE Trans Mobile Comput 13(7):1550–1563
Zhu K, Zhang L, Pattavina A (2017) Learning geographical and mobility factors for mobile application recommendation. IEEE Intell Syst 32(3):36–44
Zhu K, Xiao Y, Zheng W, Jiao X, Hsu CH (2021) A novel context-aware mobile application recommendation approach based on users behavior trajectories. IEEE Access 9:1362–1375
Zhu N, Cao J (2017) Gtrm: A top-n recommendation model for smartphone applications. In: 2017 IEEE international conference on web services (ICWS), IEEE, pp 309–316

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.