IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INNOVATION AND APPLICATION,âOF INTELLIGENT PROCESSING OF DATA, INFORMATION AND KNOWLEDGE AS RESOURCES IN EDGE COMPUTING

Received June 26, 2019, accepted July 18, 2019, date of publication July 29, 2019, date of current version August 15, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2931756

Recommendations for Mobile Apps Based on the HITS Algorithm Combined With Association Rules

XIANGLIANG ZHONG[1], YIWEN ZHANG[1], DENGCHENG YAN[2], QILIN WU[3], YUAN TING YAN[1], AND WEI LI[1]

[1]School of Computer Science and Technology, Anhui University, Hefei 230601, China
[2]Institute of Physical Science and Information Technology, Anhui University, Hefei 230601, China
[3]School of Information Engineering, Chaohu University, Chaohu 238000, China

Corresponding author: Yiwen Zhang (zhangyiwen@ahu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872002, in part by the Anhui Key Research and Development Plan under Grant 201904a05020091, and in part by the Natural Science Foundation of Anhui Province of China under Grant 1808085MF197.

ABSTRACT With the increasing popularity of intelligent devices, the mobile apps market has exploded. Due to a large number of candidate app services, it has become very difficult for users to choose the mobile apps that he/she wants to install. Therefore, it is crucial to improve users' experience and make personalized recommendations. In some cases, the traditional recommendation methods can be convenient, but they still have some shortcomings, resulting in inaccurate recommendations in general. To address this issue, this paper proposes a method for mobile app recommendations that are based on the Hyperlink-Induced Topic Search (HITS) algorithm combined with association rules. This method integrates the authority and hub scores into the candidate applications through the download and rating information, and it not only considers the importance of mobile apps in association rules but also takes the reliability factor of users into account. Experiments with the Huawei application market datasets show that the proposed method significantly improves the recommendation accuracies compared with the traditional methods.

INDEX TERMS Recommender systems, app recommendation, association rules, data mining.

I. INTRODUCTION

With the widespread use of mobile Internet, mobile Apps have been increasing remarkably in the past decade. Global downloads of iOS and Google Play apps hit new highs in 2017. Although the rapid development of mobile Apps has brought huge economic benefits to the market, competition in the mobile Apps industry is still very intense [1], [2]. Therefore, helping users quickly find suitable applications has become the focus of the mobile Apps market. Both domestic and foreign applications markets use recommendation technologies to understand users' behaviors [3], [4].

A recommendation system [5]–[8] is an intelligent system that plays a central role in helping users select suitable apps for personalized recommendations. Currently, collaborative filtering (CF) [9] is the most classic recommendation algorithm [10], [11]. It includes memory-based CF and model-based CF. Model-based CF mainly includes user-based CF [12], [13] and item-based CF [14], [15].

Fig. 1 shows item-based CF. The main idea is to predict missing values by using similar users or similar services. An important part of collaborative filtering is how to choose the appropriate similarity calculation method. The two commonly used calculations are the Pearson correlation coefficient and cosine similarity. The collaborative filtering method is very effective when the user-service matrix is relatively dense. However, users usually only choose a few application services, resulting in a very sparse matrix. In this case, finding similar users or similar applications services with traditional collaborative filtering is difficult, so the prediction results are less accurate. In addition, when evaluating mobile Apps, some users may maliciously score some services, and these outliers will affect the experimental results.

Matrix factorization [16]–[19] is used to divide the users' score and the services into two matrices and predict a missing value by selecting the larger eigenvalue to reduce the dimension. Fig. 2 illustrates the basic method of

The associate editor coordinating the review of this manuscript and approving it for publication was Ying Li.

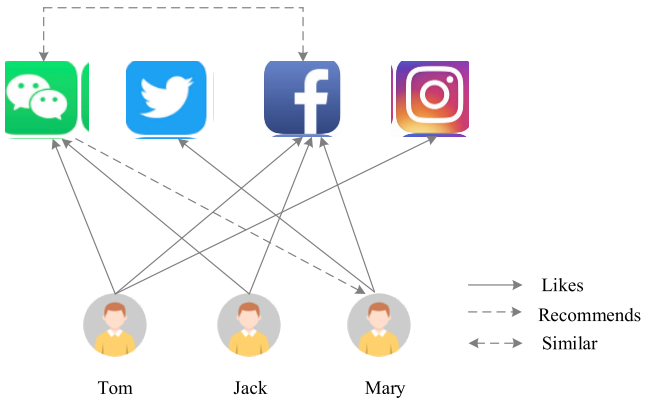


FIGURE 1. Illustration of collaborative filtering.

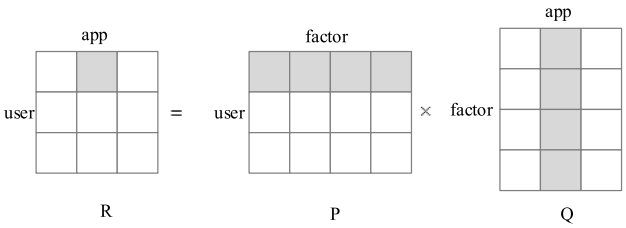


FIGURE 2. Illustration of matrix factorization.

matrix factorization. However, training the matrix decomposition model is complex, the recommended results are not very interpretable, and the recommended results are not accurate when there is a large amount of data [19]–[21]. The association rules algorithm [22], [23] is important in the field of data mining and is now widely used in recommended areas. The goal is to find hidden information and interesting patterns in the database. Frequent item sets are obtained through the calculation of item sets in the transaction database. Through pruning and connections between item sets, the association between item sets is mined, and the potential value in the transaction database is discovered. The method of associating rules requires multiple accesses of the transaction database, and, by default, the importance of different item sets is the same by default, the importance of item sets may be different, and mining a large number of associations is more complicated.

The rise of edge computing [24]–[30] brings new problems as well as improvements on the existing recommendation technologies. Traditional recommendation methods cannot be applied to the recommendation of mobile Apps. Firstly, the recommendation of the mobile app is more frequent than the recommendation of the traditional product. App updates quickly, and users may be interested in these changes [31]. As a software product, mobile phone has less acquisition cost and users can use different apps with high frequency [32]. Secondly, mobile app recommendation is influenced by environmental factors. Under different environmental conditions, users may have different experiences when using the same mobile app. Finally, the mobile app is a persistent consumer, and its use will affect the users' future app download behavior. When the user notices the recommended app and thinks that it is more attractive than the downloaded app, the user will download the new app and try out. Therefore, the recommendation of the mobile app is more personalized. To solve the problems in the tradition recommendation methods, this paper proposes a HITS-based weighted association rules calculation method. By calculating the apps' authority and the users' hub score, the amount of interest in the applications and the experience of the users are respectively obtained. The authority and the hub are combined with the association rules to mine the relationship between different applications. Finally, apps are recommended to the user. The main contributions of this paper are as follows:

1) We use the HITS algorithm to join the user scoring matrix instead of simply and iteratively using the directional relationship of the edges. This method is more interpretable because the authority and the hub scores are calculated by iterating over the scoring matrix multiple times.
2) We consider the importance of the apps to be different and the experience of the users to be different. By adding a weighting factor to the traditional Apriori algorithm, the importance of different item sets in the transaction database and the reliability of different transactions are set, and the calculation formulas of the support degree and the confidence degree are redefined, which improves the efficiency of the algorithm and effectively reduces the number of malicious user ratings.

The remainder of this paper is organized as follows. Section 2 introduces related work on mobile app recommendations. Section 3 describes the basic principle and problem definition of this article. Section 4 explains our model. Section 5 presents the experimental results and analysis. Conclusions appear in Section 6.

II. RELATED WORK

In this section, we will review related work on mobile app recommendation. Then, we discuss the advantages of these methods.

Mobile app recommendations have been explored by many researchers. To meet the needs of different users, mobile platforms provide different kinds of information to help users make decisions. This information includes the type of application, the user ratings, comments and number of downloads [33], [34]. How to select apps that users are interested in from the large number of candidate applications and give users personalized recommendations has become the focus of research on major platforms. A few studies have proposed extending the traditional recommendation algorithms and adapted them to the app domain. Yan *et al.* proposed AppJoy, which utilizes previous usage history to build a preference matrix and then performs item-based CF recommendations. However, due to the sparsity of the matrix, the recommendation accuracy is affected. Falaki *et al.* [35] used a tool to analyze the market share of applications. However, they

did not make recommendations by using the recorded data. Davidsson and Moritz [36] proposed an approach that integrated users' locations, as different users might be interested in the same applications when they are in the same geographic location. However, this approach does not take the users' preferences into account. Cui and Liang [37] used download recommendations based on the mobile application to analyze the download behaviors associated with the application and proposed a probabilistic recommendation. The above methods are based on the users' recommendations and the historical behavioral feedback on the applications and do not consider the potential connections between applications. It is impossible to remove bad users who provide malicious feedback. Liang *et al.* proposed a feature-oriented matrix factorization method, which transformed the user-app rating prediction into a user-feature rating prediction problem and combined it with user preference information. They also solved the problem of sparse data. Xu *et al.* [38] presented a recommendation method that can generate app recommendations at the functionality level. They tend to consider users' functional requirements and they also propose an effective approach for functionality extraction. But their method did not take into account the effect of users' ratings on the results. They just simply consider the user's preference for functions and ignore the user's experience of functions. Wu *et al.* proposed two hybrid models based on collaborative filtering to make mobile app recommendations. Our method not only considers the experience of users' historical behavior but also considers the importance and internal relevance of different applications, which can effectively and significantly improve the recommendation accuracy.

The association rules algorithm is a classic algorithm in the field of data mining. The academic community now applies association rules to personalized recommendations. Liu *et al.* [39] studied how to provide recommendations to users with common interests and preferences on Weibo. They proposed a recommendation algorithm based on data mining. By combining characteristics of user information, they integrated the forwarding and commenting functions into related users and used an improved Apriori algorithm to improve the accuracy of the recommendations. Zhu *et al.* proposed a book recommendation method based on association rules, which mined the association between readers' browsing their favorite books and book recommendations. In addition, link analysis methods have been widely used in the recommendation field in recent years [40], [41]. The HITS algorithm is used for location-based personalized path recommendations via social networks. Zheng *et al.* [40] utilized the principle of mutual gain between the authority and the hub scores and iteratively calculated the directed edge of the user to the location and recommended the location with a high authority to the user. In their model, a hub represents a user accessing many different locations, and an authority indicates that a location is accessed by many users. Therefore, users' visit experiences and the interests of locations have a mutual reinforcement relationship. Their method is to construct the user and location matrix through certain locations accessed by users. However, our method constructs a user-app rating matrix by ratings, and calculates the authority and hub scores through the user rating matrix. Furthermore, our approach combines personalized factors for user ratings, which is more in line with the app's recommended application scenarios.

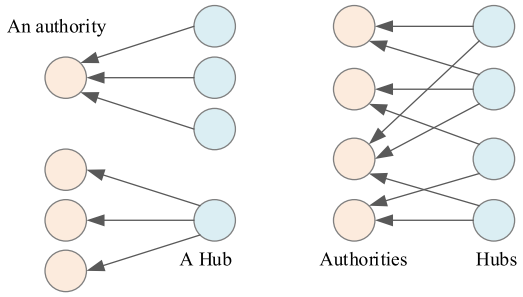


FIGURE 3. The basic concept of the HITS model used in our method.

III. PROBLEM FORMULATION

In this section, we present definitions of the key concepts in the HITS and association rules algorithms.

A. HITS ALGORITHM

HITS refers to Hypertext Induced Topic Search, which is a search query-dependent ranking algorithm for Internet information retrieval. Each page of the HITS algorithm is given two basic properties: the hub property and the authority property, as shown in Fig. 3. An authority page refers to a page that is pointed to by many page links. A hub page is a page that points to many other pages. The key to the HITS algorithm is that a good hub page points to many good authority pages and a good authority page is pointed by many hub pages. Thus, authorities and hubs have a mutual reinforcement relationship. The algorithm eventually returns a high-quality authority page and a high-quality hub page.

Suppose the current page p is accessed by the user, the total number of web pages is n, and the authority and hub are calculated by p. The link page pointing relationship is defined by:

$$\forall p, a(p) = \sum_{i=1}^{n} h(i) \tag{1}$$

$$\forall p, h(p) = \sum_{i=1}^{n} a(i) \tag{2}$$

Use the following formula to standardize:

$$\sum_{i=1}^{n} h(i)^2 = \sum_{i=1}^{n} a(i)^2 = 1 \tag{3}$$

We calculate the weight of the previous iteration and the current iteration. If there is no significant changes, it means that it is basically stable and can end the calculation. The authority and hub scores have converged.

B. ASSOCIATION RULES

The purpose of the association rules algorithm is to discover the associations and related relationships in a large number

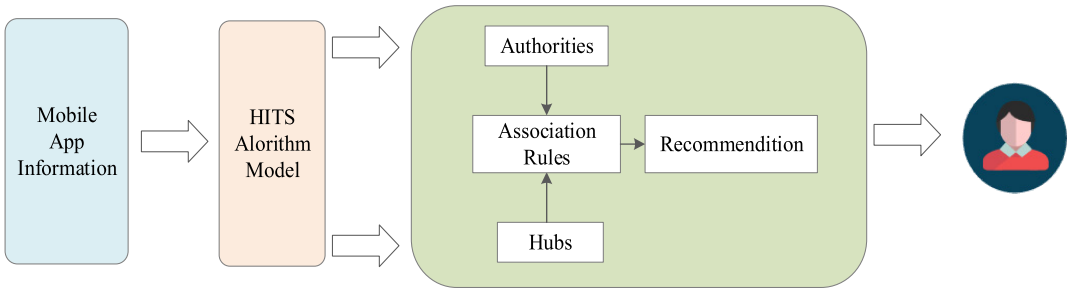


FIGURE 4. Framework of HITS-based association rules recommendation.

of item sets and to describe the probability of implicit associations between different items in the database. It can be formally defined as follows:

Definition 1 (Item Set): Given a transaction database DB, $I = \{i_1, i_2, \ldots, i_n\}$ is a set of n different items. The number of elements is called the length of the item set, and an item set of length k is called a k-item set.

Definition 2 (Transaction): T is a subset of item set I, a uniquely identified transaction number for each transaction. Recorded as TID, the transaction constitutes the transaction database D, and $|D|$ denotes the number of transactions in D.

Definition 3 (Association Rules): If $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \varnothing$, then there is association rule $X => Y$, which means that X has led to Y.

Definition 4 (Support): For an item set X, set $count(X \subseteq T)$ to the number of X item sets in the transaction set D. The support of item set X is recorded as:

$$Support(X) = \frac{count(X \subseteq T)}{|D|} \tag{4}$$

Definition 5 (Confidence): For association rules R, the confidence is the ratio of the number of transactions containing X and Y to the number of transactions containing X:

$$Confidence(X \Rightarrow Y) = \frac{Support(X \Rightarrow Y)}{Support(X)} \tag{5}$$

According to the above definitions, association rules obtain the support and confidence between item sets. The item sets with high frequency are screened out by setting the support threshold, and the item sets with strong correlation are also selected by setting the confidence threshold. Finally, the strongly related item sets in the transaction database are obtained.

IV. METHODS

In this section, we first describe the general framework of the HITS-based association rules recommendation model and then introduce the HITS model to join the user ratings matrix for bringing authority and hub into the next association rule calculation. Finally, the pseudocode of the algorithm model in this paper is presented. As shown in Fig. 4, the framework of our HITS-based recommendation model consists of the following main function modules.

1. User-app network construction and calculation: According to the historical behavior records of users of the app, a user-app network structure chart was constructed. The idea of the HITS model was used to designate users as hub nodes and apps as authority nodes. A user's rating of an app was expressed as the directed edge from the user to the app, using the relationship between the directed edges, the calculation is iteratively performed by the HITS algorithm, and the user scoring matrix is added to update the authority and hub scores after each iteration until the authority and the hub converge.

2. Authority and hub weight association rules: Authority and hub weighting factors will be for each user and app. The improved association rules were used to calculate the potential association between different apps, and new support degree and confidence degree calculation formulas were defined. By adding the association rules of authority and hub weights, the app item sets with strong relevance are calculated, which is more in line with the actual recommendation. The algorithm not only considers the importance of apps but also considers the experiences of users.

3. Recommend to users: After several iterations, the association rules that meet the threshold are obtained, and the association rules are arranged from largest to smallest in terms of confidence. Then, according to the principle of top-k, we select the top-k apps in the training set to recommend to each user. That means the recommendation lists of users are obtained.

A. HITS MODEL BASED ON THE USER RATING MATRIX

1) MODEL DESCRIPTION

Fig. 5 illustrates the main idea of the HITS model based on the user rating matrix. u denotes a user who has rated the application. Here, a circle is a scoring value, such as s_1, s_2, s_3, s_4, s_5. We regard the access of a single user to the application as a direct link of the user to s. s_1, s_2, s_3, s_4, s_5 contain the rating information of different applications by users u_1 to u_4. For instance, if a user u_1 downloads two different applications; two directed arrows are drawn from u_1 to the application, which is similar to the principle of the HITS algorithm. In our model, a hub is a user who has visited many apps, and an authority is a rated app that has been downloaded multiple times. Therefore, users' visit experiences (hub scores) and the

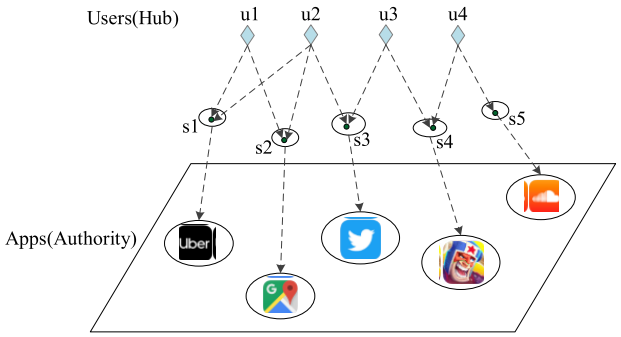


FIGURE 5. Model of user experience and app interest based on HITS.

amount of interest in an app (authority scores) have a mutual reinforcement relationship.

Definition 6 (App Interest): In our model, the amount of interest in the apps is represented by a set of authority scores $a = \{a_1, a_2, a_3, \ldots, a_n\}$, where a_j represents the amount of interest in the $j-th$ application, and a high score indicates that the application is downloaded and visited by multiple users.

Definition 7 (User Experience): In our model, users' visit experiences are represented by a set of hub scores $h = \{h_1, h_2, h_3, \ldots, h_m\}$, where h_i represents the experience of the $i-th$ user. A high score means that the user has selected multiple applications with high authority scores.

2) INFERENCE

Given a set of user ratings, we construct an adjacency matrix $M_{m\times n}$ based on the users' download histories and ratings for the application. In this matrix, r_{ij} represents users' ratings. All users and applications are built into a large matrix, and applications that are not rated are represented by zeros. For example, the following figure can be expressed as a rating matrix of users and applications, with rating information from five users for five applications.

$$M = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{array} \begin{array}{c} \begin{array}{ccccc} a_1 & a_2 & a_3 & a_4 & a_5 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \tag{6}$$

Using the principle that the users' hub h_i and the apps' authority a_j are mutually beneficial, we use the following formulas for iterative calculations.

$$a_j = \sum\nolimits_{u_i \in U} r_{ij} \times h_i \tag{7}$$

$$h_i = \sum\nolimits_{i_j \in I} r_{ij} \times a_j \tag{8}$$

where U represents a user set, u_i is a user in U, and i_j represents each application in an apps set I. The notation r_{ij} represents the preference for user u_i to app i_j. The larger r_{ij} is, the more user u_i likes app i_j. h_i is the experience of each user, and a_j the interest in each app. Through (7) and (8), we can obtain the application's authority and the users' hub scores.

For the sake of simplicity, we use a matrix to represent the product of vectors. a denotes the column vector with all the authority scores, and h denotes the column vector with all the hub scores. Thus, we can use the following vector products to simplify the above formulas.

$$a = M^T \cdot h \tag{9}$$

$$h = M \cdot a \tag{10}$$

We introduce the rating matrix M into the iterative process, where a_n and h_n represent the authority and the hub scores after n iterations, respectively. The iterative steps, calculation formulas and the final result products are shown in the following formulas:

$$a_n = M^T \cdot M \cdot a_{n-1} \tag{11}$$

$$h_n = M \cdot M^T \cdot h_{n-1} \tag{12}$$

In the first iteration, we set the initial value to $a_0 = h_0 = (1, 1, \ldots, 1)$, and then we iteratively calculate the authority score and the hub score to ensure that the result converges after the iteration. The end of each iteration requires unit normalization of the vector. We prove the convergence of the vector: using matrix M to represent the pointing relationship between authority and hub nodes, where 1 indicates a pointing relationship and 0 indicates no relationship. The authority and hub scores are both initially set to 1. Given $a_n = M^T \cdot h_{n-1}$ and $h_n = M^T \cdot a_{n-1}$, first calculate a_n in each round, then calculate h_n according to a_n, $a_n = (M^T M)^{n-1} M^T Z$, and $h_n = (MM^T)^n Z$. Through the above calculations, we can conclude that $M^T M$ and MM^T are symmetric matrices with n real eigenvalues. In the equation $h_n = (MM^T)^n Z$, the principal eigenvectors of Z and $M^T M$ are nonorthogonal, so the h vector will eventually converge to the main eigenvector of $M^T M$. To ensure that each round is a unit vector, it is normalized after each iteration.

3) ITERATIVE TERMINATION CONDITION

To make the obtained authority and hub more accurate and reliable, we need to perform multiple iterations and set the conditions for iteration termination. Let a_n denote the authority scores returned by the $n-th$ iteration, a_{n-1} represent the calculation result of the $(i-1)-th$ time. The conditions for iterative termination are as follows:

$$|a_n^2 - a_{n-1}^2| < \varepsilon \tag{13}$$

The above equation (13) indicates that the absolute value of the square difference of the authority calculated by two previous iterations is less than a given parameter threshold, and ε is a small tunable parameter.

B. AUTHORITY AND HUB WEIGHT ASSOCIATION RULES

Apriori is a commonly used algorithm for finding frequent item sets. The principle is that if an item is a frequent item set, all its subitems are also frequent items. Meanwhile, if an item is a nonfrequent set, all its supersets are also nonfrequent. According to the Apriori algorithm and the iterative method

of layer-by-layer searches, the candidate item set is searched. The Apriori algorithm uses a priori properties to improve the search efficiency. In the real world, the importance of different apps may be different, and the reference to users' ratings is also different. Therefore, weighting is introduced to the apps and users, and the returned authority and hub are added to the calculation formulas for support and confidence by using the HITS algorithm to include the scoring information, which not only take into account the scoring information but also introduce the weight information of app and users, making the recommendation result more consistent with the real recommendation scenario. The traditional Apriori algorithm set needs to scan the database to find each frequent item. When we have a large amount of data, the efficiency of this expensive algorithm is greatly reduced. Moreover, if we do not consider the weight of each item, a large number of rules may be unearthed, and the recommendation of app recommendations are meaningless. In the app download/rating information database DB, $I = \{i_1, i_2, \ldots, i_n\}$ is the set of items in the transaction database, which is a record of the different applications a user downloads and rates. Here, n denotes the number of items in the database, representing the total number of items in the applications, i_k represents the $k-th$ application. The probability that i_k appears in the database is $P(I_k)$, and the calculation formula is as follows (14). The weight of I_k is the weight $w(I_k)$ of each application, and the calculation depends on $P(I_k)$. The calculation formula (15) is as follows:

$$p(I_k) = l/m \tag{14}$$

$$w(I_k) = a_k/P(I_k) \tag{15}$$

where l is the number of times I_k appearing in the database and m is the total number of records,a_k the authority of the $k-th$ application. The transaction database T_x is represented by the $x-th$ transaction record, which means the download record for each app by each user, and the experience of each user is mapped to the hub score. We use the product of the average weight of the items included in the users' hub score; that is, the users' hub score is multiplied by the average of all users' $w(I_k)$ and is denoted by $wt(T_x)$, where $k = 1, 2, \ldots, n$, the formula is as follows:

$$wt(T_x) = h_x \cdot \sum_{k=1}^{n\&I_k \in T_x} w(I_k)/|T_x| \tag{16}$$

According to formula (16), $|T_x|$ denotes the number of items included in the transaction record, namely, the number of apps that the user has downloaded. The weight of each app is recorded as $w-support$, and the formula for calculating the weighted support of the app is the proportion of all user transaction records containing the app to all the user transaction weights. Based on the transaction database, we can easily include the transactions that are included in this app and then calculate the sum of the transaction set weights for all the apps. The formula is as follows:

$$w - support(S) = \sum_{x=1}^{m\&S \in T_x} wt(T_x) / \sum_{x=1}^{m} wt(T_x) \tag{17}$$

where m represents the number of transactions in the transaction database, that is, the number of user records. S represents the applications in the database. We calculate the weight support of the item set of apps through this formula and then calculate the confidence of the different item set of apps through the support. The formula is as follows:

$$w - conf(X, Y) = \sum_{x=1}^{m\&(X \cup Y) \subseteq T_x} wt(T_x) / \sum_{x=1}^{m\&X \subseteq T_x} wt(T_x) \tag{18}$$

where the confidence of X and Y represents the ratio of the support of item X and item Y appearing at the same time to the support of item X. Using the above formula (18), we can calculate the association rules between apps.

C. ALGORITHM DESCRIPTION

The first step of the algorithm is to obtain the application's authority scores and the users' hub scores. The initial setting is $a_0 = h_0 = (1, 1, \ldots, 1)$, and each iteration adds the score matrix M. When the result meets the iteration termination condition, the iteration is stopped, and then a and h vectors are used as the return values. The pseudocode of the algorithm is as follows:

The second part of the algorithm is described below. The authority and hub scores are added to the association, and the weight information is added to apps and users. The strong association rules that meet the support and confidence thresholds are calculated using the redefined support and confidence. Then, the confidence is ranked from largest to smallest, and the top-k apps according to confidence are recommended to users in the training set.

D. ALGORITHM TIME COMPLEXITY ANALYSIS

In order to analyze the efficiency of the algorithm proposed in this paper, the time complexity of H-S-Apriori method is analyzed in detail in this paper. The time complexity of H-S-Apriori algorithm mainly includes two stages. In the first stage, HITS algorithm with user rating was added to calculate the score of apps' authority and user hub iteratively, and in the second stage, Apriori algorithm with weight was added to analyze the association rules between different apps.

1) TIME COMPLEXITY ANALYSIS OF THE ALGORITHM IN THE FIRST STAGE

The HITS-Score Algorithm describes the time complexity of the algorithm in the first stage. Firstly, the authority and hub vectors are initialized. Two vectors are iteratively calculated through the user scoring matrix. Each iteration needs to normalize the two groups of vector. For details, refer to Algorithm 1. After *num_iter* iterations, the current apps' authority scores is compared with the previous iteration, until

Algorithm 1 HITS-Score Algorithm

Input: Collection of users' ratings
Output: Collection of users' hub scores, h, and the collection of app's authority scores, a

1: Initialize vector a and vector$h$, $a_0 = h_0 = (1, 1, \ldots, 1)$
2: M = Matrix Building (U, A, R)
3: $a = M^T \cdot h$
4: $h = M \cdot a$
5: **While** difference $(a_n, a_{n-1}) > \varepsilon$
6: $(h, a) = HIT(M)$
7: Update vectors a and h with matrix M
8: **Return** (h, a)

Algorithm 2 Weight Association Rules Algorithm

Input: Users' hub scores, h, and apps' authority scores, a
Output: The association rules L generated by the candidate item set

1: Scan the database to get the users' h and the apps' a
2: L_1 = Generate(D)
3: **For** (k = 2; $L_{k-1} \neq \varnothing$; k++) do begin
4: C_k = Apriori-gen(L_{k-1}, a, h)
5: **For** all transactions $t \in D$ do begin
6: C_t = subset(C_k, t)
7: L_k= setMinsup(c)
//all candidates $c \in C_t$
8: calculate the confidence between item sets in frequent item sets
9: rule = generareRules(L_k,minConf)
10: recommend List = sort(rule)
//Recommend the top-k items according to the confidence of users
11: **Return** recommend List

the difference between the authority scores is less than a certain threshold ε.The computational complexity of $a = M^T \cdot h$ is $O(N)$, the computational complexity of $h = M \cdot a$ is $O(N)$, and the time complexity to normalize the authority $a_n = M^T \cdot M \cdot a_{n-1}$ is $O(N)$.Similarly, the computational complexity of $h_n = M \cdot M^T \cdot h_{n-1}$ is $O(N)$. In summary, the time complexity of the iteration of the HITS-Score Algorithm is $O(4N * num\_iter)$.

2) TIME COMPLEXITY ANALYSIS OF THE ALGORITHM IN THE SECOND STAGE

The Weight Association Rules Algorithm describes the second-stage algorithm time complexity. We can obtain the authority and hub calculated by Algorithm 1 and integrate it into the association rule, and then use a priori principle to calculate the high-order frequent item sets using the low-order frequent item sets. The $k + 1$ candidate sets are generated by using the k frequent item sets connection. For the frequent item set L_m of length m and the number of items set n, the time complexity of comparing the connectable conditions is $O(m * n^2)$.

TABLE 1. Dataset information.

Data Field	Example	Explanation
App Name	WeChat	app name
User ID	Leo	identification of user
Rating	0-5	user preference
Comment Time	2016-12-20 09:36	time of users' comment
Comments	This is a good application	users' comments
Phone Model	Mata 8	user's phone model

By analyzing the time complexity of the two stages of H-S-Apriori in turn, the overall time complexity of the H-S-Apriori algorithm is $O(4N * num\_iter) + O(m * n^2) = O(n^2)$.

V. EVALUATION

In this section, we verify the generalizability of our method through a large number of experiments on the Huawei mobile application market and compare the results with the relatively newer methods and classic methods. Moreover, we also study the influence of different parameters on the accuracy of the experimental prediction.

Experimental Environment: JDK 1.7, Python 3.7, Windows 7, Intel i7-4790 CPU 3.60 GHz with 4 GB RAM

A. DATA PREPROCESSING

The dataset used in this experiment is from the download/score record for real apps. Each record represents a user's one-time use behavior. Table 1 shows the data-related information.

The dataset we used in the experiment was derived from real app download/score records from the Huawei application market. The dataset includes information such as the app name, the user ID, the rating, the comment time, comments and the phone model. In the real world, a user-app matrix is very sparse, and only a few apps have been used by users. To solve this problem, each user here has downloaded and commented on more than 20 mobile apps, and each app has been downloaded and commented on by more than 20 users, which is more consistent with the actual situation. After screening, the dataset contained 14,188 records, including 1,085 users and 700 apps.

B. AUTHORITY CORRELATION COMPARISON

Through the above calculations, We can analyze the correlation of the results of Algorithm 1 that scores are added to HITS algorithm. In this paper, Pearson Correlation Coefficient and Spearman Correlation Coefficient are used to measure the correlation between the ranking values of authority in algorithm 1 and the download volume and the average score of Huawei mobile application market. The following is a brief introduction to the difference between the two calculations. The Pearson correlation coefficient is a statistic used to reflect the degree of similarity between two variables. Calculating the similarity between features and categories, we can determine the relevance of the extracted features

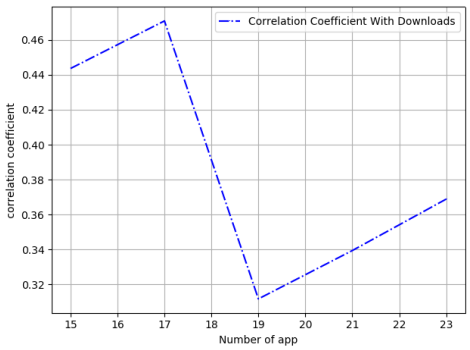


FIGURE 6. Correlation coefficient between authority and downloads.

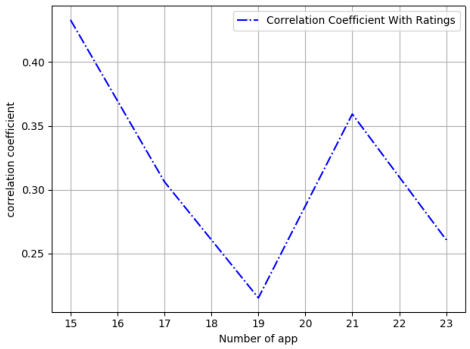


FIGURE 7. Correlation coefficient between authority and average ratings.

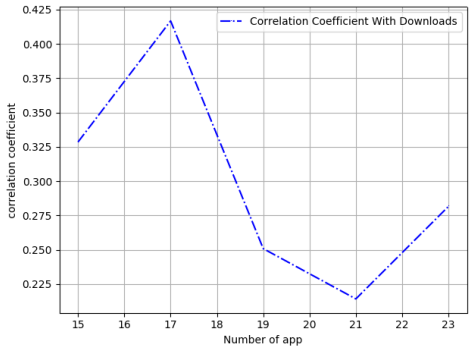


FIGURE 8. Correlation coefficient between authority and downloads.

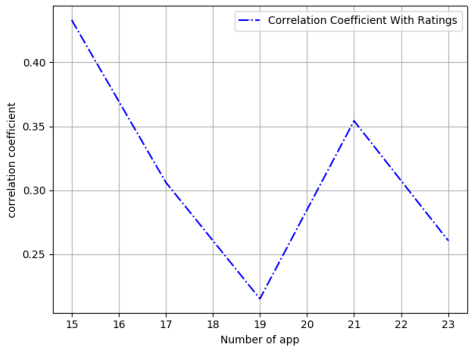


FIGURE 9. Correlation coefficient between authority and average ratings.

and categories. The Pearson correlation coefficient is explained by the following formula:

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{19}$$

where, x and y represent a given set of two consecutive vectors, $\bar{x}$ represents the average of vector x, and $\bar{y}$ represents the average of vector y, ρ stands for Pearson correlation coefficient, $\rho \in [-1, 1]$. Here, ρ close to 0 means there is no correlation between the two sets of vectors, to 1 means there is a positive correlation between the two sets of vectors, and to -1 means there is a negative correlation. Through the above formula, we discuss the correlation between the results of algorithm 1 and the downloads and average ratings of apps.

Fig. 6 and Fig. 7 reflect the Pearson correlation coefficient between the authorities of top-k apps and the apps downloads and the average ratings. As can be seen from Fig. 6, the Pearson correlation coefficient between authorities and downloads is about 0.3-0.49. It can be seen from Fig 7, the Pearson correlation coefficient between the authority and the average rating is about 0.21-0.42. In summary, the authorities calculated by algorithm 1 have a certain intensity correlation with the downloads and ratings of app. Therefore, it is meaningful to add user's ratings to the HITS algorithm, which has a certain impact on the experimental results and is more in line with the personalized recommendations in real life.

Similarly, the Spearman correlation coefficient is used to analyze the correlation between the apps authorities and the downloads and average ratings. The following is the calculation of Spearman correlation coefficient.

$$\rho_s = \frac{\sum_{i=1}^{n}(R_i - \bar{R})(S_i - \bar{S})}{\sqrt{\sum_{i=1}^{n}(R_i - \bar{R})^2 \sum_{i=1}^{n}(S_i - \bar{S})^2}} \tag{20}$$

The Spearman correlation coefficient is similar to the Pearson correlation coefficient. The difference is that the values are replaced by ranks. Here, R_i and S_i are the value levels of the observed value i, respectively. $\bar{R}$ and $\bar{S}$ are the average levels of the variables x and y, respectively, and n is the total number of observations.

The Fig. 8 and Fig. 9 reflect the Spearman correlation coefficient between the authorities of top-k apps and the apps downloads and the average ratings. It can be seen from Fig. 8 that the Spearman correlation coefficient between the authorities and the downloads is about 0.20-0.42. As can be seen from Fig. 9, the Spearman correlation coefficient between authorities and average ratings is about 0.20-0.44. To sum up, it can be seen that the authorities calculated by algorithm 1 in this paper has a certain correlation with the downloads and ratings, which has a certain impact on the experimental results.

C. RECOMMENDATION ACCURACY EVALUATION

To measure the accuracy of the recommendations, precision and recall were used. These evaluation indicators are defined by formula (21) and formula (22), respectively:

$$Precision = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |R(u)|} \tag{21}$$

$$Recall = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |T(u)|} \tag{22}$$

where U represents the set of users in the test set, u represents each user in U, $R(u)$ is the recommended list based on the users' behaviors on the training set, and $T(u)$ is the list of behaviors of the users in the test set. To prove the superiority of the proposed H-S-Apriori method, we choose the following four methods to test the performance of our experiment with respect to the precision and recall.

(1) User-based collaborative filtering (U-CF): Conduct similarity calculation on users, find the top-k user sets that are similar to the target users, and recommend the target users in this set who are preferred by users and who have not selected mobile Apps to the target users.
(2) Matrix factorization (MF): The user scoring matrix was used in matrix factorization to obtain the missing scores in the matrix, and the top-k predicted apps based on rankings were recommended to the target users.
(3) Probabilistic Matrix Factorization (PMF): This method is based on the matrix factorization, a probability model is introduced to further optimize. Assuming that the characteristic matrices of user U and user V obey Gaussian distribution, and get the characteristic matrix of U and V through the known value of the rating matrix, and use the characteristic matrix to predict the unknown value of the scoring matrix, the top-k apps can be recommended to the users.
(4) Hyperlink-Induced Topic Search (HITS): This method mainly constructs the directed graph of users and mobile Apps based on the historical behavior of users. Through multiple iterations, it returns multiple users pointing to many authorities and mobile Apps pointed to many hubs and returns the top-k recommended mobile Apps for the target users with respect to the authority.
(5) Apriori: This method is a data mining method. The transaction database is constructed according to the historical behavior of users, and mobile Apps with strong relevance are calculated. The association rules of the top-k apps with respect to confidence are taken as the basis of recommendation, and mobile Apps with strong correlations are recommended to target users.

Fig. 10 and Fig. 11 show that as the number of recommended applications increases, the overall precision decreases, while the recall of increases. The recommended precision and recall are interrelated and negatively correlated. As the number of recommended apps increases, the user behavior list in the test set remains unchanged, and the number of correctly recommended apps may increase, leading to an increase in the recall. However, the recommended list calculated by the user behaviors in the training set changes, and the number of correct recommended apps may not increase, leading to a decrease in precision.

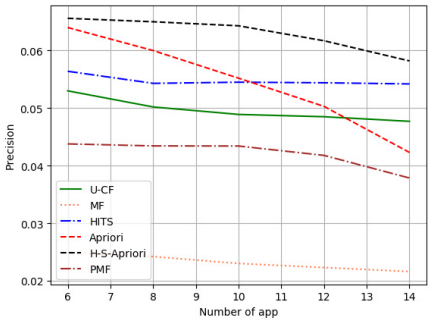


FIGURE 10. Comparison of the precision for different methods.

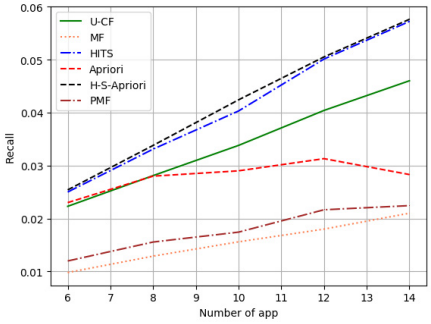


FIGURE 11. Comparison of the recall for different methods.

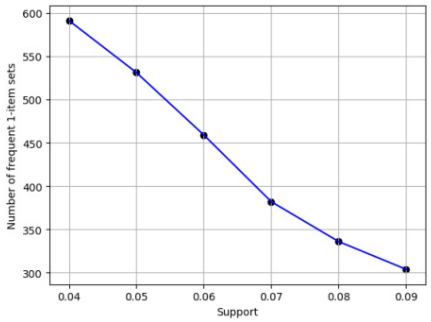


FIGURE 12. Influence of different supports on the frequent one-item sets.

The method in this paper is improved by association rules; thus, we study the influence of support on frequent item sets. To ensure that the mining association rules are more accurate, we should set a reasonable support threshold. Fig. 12, Fig. 13 and Fig. 14 list the impact of support on the apps of frequent one-item sets, frequent two-item sets, and association rules, respectively. It can be seen from the figure that the number of frequent two-item sets is far more than that of frequent one-item sets. With the increase of order, the number of frequent item sets mined by the algorithm also increases sharply. Therefore, the number of frequent item sets

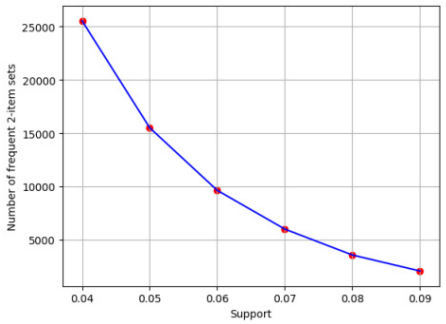


FIGURE 13. Influence of different supports on the frequent two-item sets.

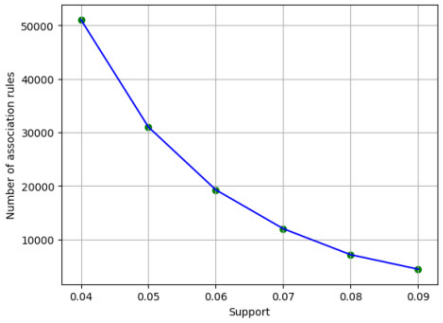


FIGURE 14. Influence of support on the number of association rules.

mined by the algorithm should be controlled to eliminate a large number of redundant association rules.

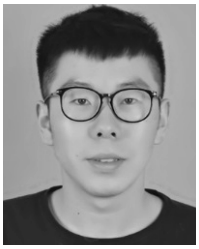
VI. CONCLUSION

The traditional application recommendation method reveals many problems. This paper proposes a method that combines the HITS algorithm and association rules. It integrates user ratings to iteratively calculate authority and hubs scores, then combines the result with the association rules, and redefines the calculation formulas of support and confidence. Finally, the top-k applications based on confidence are recommended to the users, which avoids the problem of large time consumption of the traditional association rules, improves the efficiency of the algorithm, and effectively solves the problem of malicious user ratings. Through the experimental verification of the user records of the Huawei mobile phone application market, the results show that the proposed method has excellent performance in terms of precision and recall. However, our solution also has some limitations. Firstly, the Apriori algorithm is improved in our method, so the time overhead is still large, and the efficiency of mining association rules is not high when facing large data. In the future work, we will improve the mining method of association rules, reducing the time overhead and redundant data, and improve the mining efficiency of association rules. Secondly, our method only considers user's ratings information, with low data dimension and mediocre recommendation effect. In the future, our work will further add more app feature information and user's personalized factors, such as users' functional requirements, location information and users' comments.

REFERENCES

[1] Z.-R. Fang, S.-W. Huang, and F. Yu, "AppReco: Behavior-aware recommendation for iOS mobile Apps," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun./Jul. 2016, pp. 492–499.
[2] A. Girardello and F. Michahelles, "AppAware: Which mobile applications are hot?" in *Proc. 12th Int. Conf. Hum. Comput. Interact. Mobile Devices Services*, 2010, pp. 431–434.
[3] H. Gao, W. Huang, X. Yang, Y. Duan, and Y. Yin, "Toward service selection for workflow reconfiguration: An interface-based computing solution," *Future Gener. Comput. Syst.*, vol. 87, pp. 298–311, Oct. 2018.
[4] H. Gao, D. Chu, Y. Duan, and Y. Yin, "Probabilistic model checking-based service selection method for business process modeling," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 27, no. 6, pp. 897–923, Feb. 2017.
[5] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, p. 13, 2016.
[6] Y. Xu, J. Yin, S. Deng, N. N. Xiong, and J. Huang, "Context-aware QoS prediction for Web service recommendation and selection," *Expert Syst. Appl.*, vol. 53, pp. 75–86, Jul. 2016.
[7] E. Ashley-Dejo, S. Ngwira, and T. Zuva, "A survey of context-aware recommender system and services," in *Proc. Int. Conf. Comput., Commun. Secur. (ICCCS)*, Dec. 2015, pp. 1–6.
[8] Y. Yin, S. Aihua, G. Min, X. Yueshen, and W. Shuoping, "QoS prediction for Web service recommendation with network location-aware neighbor selection," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 26, no. 4, pp. 611–632, 2016.
[9] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Comput. Commun.*, vol. 41, pp. 1–10, Mar. 2014.
[10] V. Agarwal and K. Bharadwaj, "A collaborative filtering framework for friends recommendation in social networks based on interaction intensity and adaptive user similarity," *Soc. Netw. Anal. Mining*, vol. 3, no. 3, pp. 359–379, 2013.
[11] Z. Zhou, B. Wang, J. Guo, and J. Pan, "QoS-aware Web service recommendation using collaborative filtering with PGraph," in *Proc. IEEE Int. Conf. Web Services*, Jun./Jul. 2015, pp. 392–399.
[12] A. Bellogin and J. Parapar, "Using graph partitioning techniques for neighbour selection in user-based collaborative filtering," in *Proc. 6th ACM Conf. Recommender Syst.*, 2012, pp. 213–216.
[13] J. Wang and J. Yin, "Enhancing accuracy of user-based collaborative filtering recommendation algorithm in social network," in *Proc. 3rd Int. Conf. Syst. Sci., Eng. Design Manuf. Inf.*, vol. 1, Oct. 2012, pp. 142–145.
[14] P. Pirasteh, J. J. Jung, and D. Hwang, "Item-based collaborative filtering with attribute correlation: A case study on movie recommendation," in *Proc. Asian Conf. Intell. Inf. Database Syst.* Cham, Switzerland: Springer, 2014, pp. 245–252.
[15] S. Wei, N. Ye, S. Zhang, X. Huang, and J. Zhu, "Item-based collaborative filtering recommendation algorithm combining item category with interestingness measure," in *Proc. Int. Conf. Comput. Sci. Service Syst.*, Aug. 2012, pp. 2038–2041.
[16] 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.
[17] J. Meng, Z. Zheng, G. Tao, and X. Liu, "User-specific rating prediction for mobile applications via weight-based matrix factorization," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun./Jul. 2016, pp. 728–731.
[18] H. Gao, H. Miao, L. Liu, J. Kai, and K. Zhao, "Automated quantitative verification for service-based system design: A visualization transform tool perspective," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 28, no. 10, pp. 1369–1397, 2018.
[19] H. Gao, S. Mao, W. Huang, and X. Yang, "Applying probabilistic model checking to financial production risk evaluation and control: A case study of Alibaba's Yu'e Bao," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 3, pp. 785–795, Sep. 2018.
[20] H. Gao, Y. Duan, H. Miao, and Y. Yin, "An approach to data consistency checking for the dynamic replacement of service process," *IEEE Access*, vol. 5, pp. 11700–11711, 2017.
[21] Y. Yin, L. Chen, Y. Xu, and J. Wan, "Location-aware service recommendation with enhanced probabilistic matrix factorization," *IEEE Access*, vol. 6, pp. 62815–62825, 2018.

[22] J. Chen, C. Miller, and G. G. Dagher, "Product recommendation system for small online retailers using association rules mining," in *Proc. Int. Conf. Innov. Design Manuf. (ICIDM)*, Aug. 2014, pp. 71–77.
[23] J. Wang, L. Hong, and B. D. Davison, "RSDC'09: Tag recommendation using keywords and association rules," *Expert Syst. Appl.*, vol. 32, no. 3, pp. 753–764, 2010.
[24] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "QoS prediction for service recommendations in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 134–144, May 2019.
[25] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven IoT service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54258–54269, 2018.
[26] C. Zhang, H. Zhao, and S. Deng, "A density-based offloading strategy for IoT devices in edge computing systems," *IEEE Access*, vol. 6, pp. 73520–73530, 2018.
[27] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," *Mobile Netw. Appl.*, Apr. 2019. doi: 10.1007/s11036-019-01241-7.
[28] Y. Yin, Y. Xu, W. Xu, M. Gao, L. Yu, and Y. Pei, "Collaborative service selection via ensemble learning in mixed mobile network environments," *Entropy*, vol. 19, no. 7, p. 358, 2017.
[29] Y. Yin, W. Xu, Y. Xu, H. Li, and L. Yu, "Collaborative QoS prediction for mobile service with data filtering and SlopeOne model," *Mobile Inf. Syst.*, vol. 2017, Jun. 2017, Art. no. 7356213.
[30] X. Xiao, W. Zheng, Y. Xia, X. Sun, Q. Peng, and Y. Guo, "A workload-aware VM consolidation method based on coalitional game for energy-saving in cloud," *IEEE Access*, vol. 7, pp. 80421–80430, 2019.
[31] W. Pan, N. Aharony, and A. S. Pentland, "Composite social network for predicting mobile apps installation," in *Proc. 25th AAAI Conf. Artif. Intell.*, Aug. 2011, pp. 821–827.
[32] H. Gao, W. Huang, Y. Duan, X. Yang, and Q. Zou, "Research on cost-driven services composition in an uncertain environment," *J. Internet Technol.*, vol. 20, no. 3, pp. 755–769, 2019.
[33] E. Ha and D. Wagner, "Do Android users write about electric sheep? Examining consumer reviews in Google Play," in *Proc. IEEE 10th Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2013, pp. 149–157.
[34] S. L. Lim and P. J. Bentley, "Investigating app store ranking algorithms using a simulation of mobile app ecosystems," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 2672–2679.
[35] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin, "Diversity in smartphone usage," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services*, 2010, pp. 179–194.
[36] C. Davidsson and S. Moritz, "Utilizing implicit feedback and context to recommend mobile applications from first use," in *Proc. Workshop Context-Awareness Retr. Recommendation*, 2011, pp. 19–22.
[37] Y. Cui and K. Liang, "A probabilistic Top-N algorithm for mobile applications recommendation," in *Proc. 5th IEEE Int. Conf. Broadband Netw. Multimedia Technol.*, Nov. 2013, pp. 129–133.
[38] 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, 2018.
[39] L. Liu, S. Yu, X. Wei, and Z. Ning, "An improved Apriori–based algorithm for friends recommendation in microblog," *Int. J. Commun. Syst.*, vol. 31, no. 2, 2018, Art. no. e3453.
[40] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 791–800.
[41] Y. Yin, F. Yu, Y. Xu, L. Yu, and J. Mu, "Network location-aware service recommendation with random walk in cyber-physical systems," *Sensors*, vol. 17, no. 9, p. 2059, 2017.



XIANGLIANG ZHONG received the bachelor's degree, in 2017. He is currently pursuing the master's degree with the School of Computer Science and Technology, Anhui University. His current research interest include service computing and big data.

YIWEN ZHANG received the Ph.D. degree in management science and engineering from the Hefei University of Technology, in 2013. He is currently a Professor with the School of Computer Science and Technology, Anhui University. His research interests include service computing, cloud computing, and big data analytics.

DENGCHENG YAN received the B.S. and Ph.D. degrees from the University of Science and Technology of China, in 2011 and 2017, respectively. From 2017 to 2018, he was a Core Technology Researcher and a Big Data Engineer with Research Institute of Big Data, iFlytek Company Ltd. He is currently a Lecturer with Anhui University, China. His research interests include complex networks, big data, and service computing.

QILIN WU received the Ph.D. degree in computer application technology from the Hefei University of Technology, in 2011. He did his Post-doctoral Research at Nanjing University, Nanjing, China. He is currently a Professor with the School of Information Engineering, Chaohu University. His research interests include resource allocation and optimization for wireless networks, edge computing, and service computing.

YUAN TING YAN received the Ph.D. degree in computer science from Anhui University, in 2016, where he is currently a Lecturer. His research interests include machine learning, granular computing, and bioinformatics.

WEI LI received the Ph.D. degree in computer science from Anhui University, in 2006, where she is currently a Professor with the School of Computer Science and Technology. Her research interests include software engineering and embedding systems.