A Hybrid Approach based on Collaborative Filtering to Recommending Mobile Apps

Xia Wu[‡], Yanmin Zhu [‡,‡,♯]
[†] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[‡] Shanghai Key Lab of Scalable Computing and Systems, Shanghai, China
[♯] Corresponding Author
{yishuyan, yzhu}@sjtu.edu.cn

***Abstract*—With the rapid emergence of mobile devices, smart phones have penetrated into every aspect of people's daily life. The explosive growth of mobile applications makes it difficult for mobile users to find suitable and interesting applications. Mobile app recommendation has been explored by many researchers and some industry solutions are proposed for mobile users. Collaborative filter (CF) is a popular technique in recommendation system, but it requires explicit feedback data like ratings, which are usually difficult to be collected. Many application markets provide keywords search functions and recommend applications with high download counts. However, downloading an application is a vague indicator of whether the user truly likes that application, as the user may probably uninstall that application immediately after it has been installed. Some other industry solutions involve users' personal data such as social networking, which may result in privacy leaks. In this paper, we propose two hybrid models based on collaborative filtering to make mobile app recommendations. We leverage RFD (Recency, Frequency, Duration) model to label the preference for users over applications from users' usage data. The first model, namely, improved item-oriented collaborative filtering (IIOCF), improves the performance of the item-oriented approach by leveraging the latent factor model to discover latent factors among applications. The second model (HLF) is a hybrid model of the latent factor and item-oriented approach. This model sums the predictions of the latent factor model and item-oriented approach, thereby capturing the advantages of both approaches. Our experiment results over 6,568 applications and 25,302 users clearly show that HLF model has better performance than both the item-oriented and latent factor approach.**

***Index Terms*—mobile app recommendation, hybrid models, latent factor model, item-oriented approach, RFD Model**

I. Introduction

With the rapid development of mobile devices, smart phones have played an important role in people's life. People can easily download their interested mobile applications from application markets such as App Store [1] or Google play [2]. The number of mobile apps has experienced explosive growth in recent years. For example, Apple announced at its annual Worldwide Developers Conference 2015 (WWDC) [3] that the App Store had over 1.5 million apps by 2015. Being faced with a huge number of applications of which the functions are highly varied, people can find it difficult to have enough time and energy to experience all applications. As a consequence, mobile apps recommenders are definitely needed by users for finding suitable applications. By now, many researchers have explored mobile app recommendation and some industry solutions have been proposed to address the problem.

Collaborative Filtering (CF) [4] is one of the most popular techniques for recommendation systems, as it relies only on user's previous behavior, e.g., their previous browsing history or product ratings. The two main categories of collaborative filtering are the neighborhood methods and the latent factor models. Latent factor models have better performance than neighborhood methods in general, but neighborhood methods have acceptable explanations of recommended products. Both methods are primarily based on high quality explicit feedback, which usually refers to the ratings by users with respect to their degree of preference over products. However, the rating datasets are not easy to be collected as they require extra efforts to manual label. The statistic that a popular application can only have 2 - 3% ratings provide better evidence [5]. Some application markets allow users to perform keyword search and provide ranked lists of applications across different categories, as shown in figure 1. Typically, those high-ranking applications on the list are most recommended. The order of rankings relies to a large extent on overall downloads

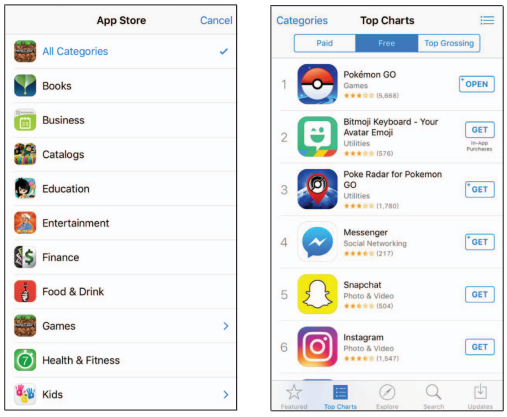

Figure 1. Two screenshots of App Store from iPhone. The left figure shows the categories of all applications. Mobile users can choose the category such as Books or Business to find suitable applications. Also, users can choose All Categories as shown in the left figure and the right figure shows the top six free apps regards of categories.

1521-9097/16 $31.00 © 2016 IEEE
DOI 10.1109/ICPADS.2016.9

within a short time. AppBrain [6] is such a recommendation software that can recommend popular applications with high volume of downloads. However, according to our experience of downloading applications, many applications are uninstalled immediately after being installed. It indicates that the behavior of downloading an application is a vague indicator of whether the user actually likes that application. What's more, the most popular applications may not be the most suitable ones for certain mobile users, as generally people have different preference applications from each other. Other industry solutions like AppFire [7] leverage social network to realize recommendation among friends, which may lead to potential privacy leaks for users.

In this paper, we present two hybrid approaches based on collaborative filtering to recommending mobile apps. We leverage RFD (Recency, Frequency, Duration) model to label the degree of user's preference over mobile apps from users' usage data, which can be easily collected from telecom operators. The first model, namely, improved item-oriented collaborative filtering (IIOCF), improves the performance of the item-oriented approach by leveraging the latent factor model to discover latent factors among applications. Since the preference matrix we construct is very sparse, the performance of item-oriented approach is not very satisfactory. Then instead of using sparse preference matrix to compute similarities among apps, which is a common method in neighborhood models, we leverage latent factor models to discover more latent relationships among apps, thereby improving the performance of item-oriented approach. The second model (HLF) sums the predictions of the latent factor model and item-oriented approach, thereby capturing the advantages of both latent factor and item-oriented approach. Our experiment results over 6,568 applications and 25,302 users clearly show that HLF model has better performance than both the latent factor and item-oriented approach. The key contributions of this paper are summarized as follows. We first propose a hybrid model combining the advantages of neighborhood models and latent factor models.

- Instead of using explicit feedback data, which is difficult to collect, we take advantage of users' implicit feedback, such as frequency and time spent on each application by users, to measure the degree of preference for a certain app. Inspired by the famous RFM (Recency, Frequency, Monetary) model in marking that measures the loyalty of customers [8], we use a modified version, namely, RFD (Recency, Frequency, Duration) model, to measure the degrees of preference for users over applications.
- We propose two hybrid models of the latent factor and item-oriented approach. The first model (IIOCF) can improve the performance of the item-oriented approach and the second model has better performance than both the latent factor and the item-oriented approach.

The rest of the paper is organized as follows: Section II enumerates the related works. In Section III we define our problem and give a description of our dataset. The details of our methodology and the results of our experiment are presented in Section IV and Section V respectively. Finally, we make a conclusion in Section VI.

II. Related Work

In this section, we introduce the related work of our study. The related work can be categorized into two parts: mobile app recommendation collaborative filtering.

A. Mobile App Recommendation

So far, mobile app recommendation has been explored by many researchers. Woerndl et al. proposed an approach integrating users' locations [9], as different users might be interested in the same application when they are at the same location. For example, if many people use a train timetable application at the train station, then the application can be recommended to someone who is around or in the same train station. This approach involves real-time locations, which may result in security risks. Falaki et al. have developed a tool to record the time a user spent on an application, and used the recorded to analyze the market share of applications [10]. However, they did not make the step of recommendations by using the recorded data. Our dataset is similar to what Falaki et al. recorded, and we can use our dataset to recommend suitable applications for mobile users. Some other researchers proposed content-oriented approaches. For example, Shi et al. proposed a model named Eigenapp for recommendation in Getjar, which is a platform second only to Apple Inc [11]. Yu et al. leveraged Latent Dirichlet Allocation topic model to deal with complex content [12]. Zhu et al. used content logs for app recommendation [13]. These content-oriented approaches all need extra content information, which is not needed in our approach. The most similar approach to our work is Appjoy proposed by Yan et al. [5]. They leveraged previous usage history of users to build a preference matrix, and then used slope one, a simple collaborative filtering approach, to realize the application. However, the preference matrix is actually very sparse, and the performance of recommendation by using slope one in our dataset is not very good. So we first use latent factor models to discover more latent factors among applications and then use neighborhood methods to make recommendations. The results show that our approach is more accurate either the latent factor models or the neighborhood models.

Also, many industry solutions are proposed to facilitate mobile users to find interesting applications. Some application markets allow users to use keywords search and recommend applications with a large number of downloads. Two industry softwares AppBrain and AppFire are respectively used for recommendation on Android platform and iOS platform. AppFire requests the Google accounts, and it will upload a list of applications which are already installed by users. AppFire leverages social network to allow sharing and recommendation of mobile applications among friends. AppBrain classifies mobile applications according to users' usage pattern, and recommends other applications in the same category. Also,

AppBrain can recommend popular applications with high volume of downloads. However, downloading an application is a weak indicator of whether the user are indeed interested in this app, many applications may be uninstalled immediately after being installed. What's more, the most popular applications may not be the most suitable ones for certain mobile users. This means that it is not suitable to recommend applications in accordance with a large download count. Our approach leverage collaborative filtering, which relies on users' historical usage data and do not require other extra information.

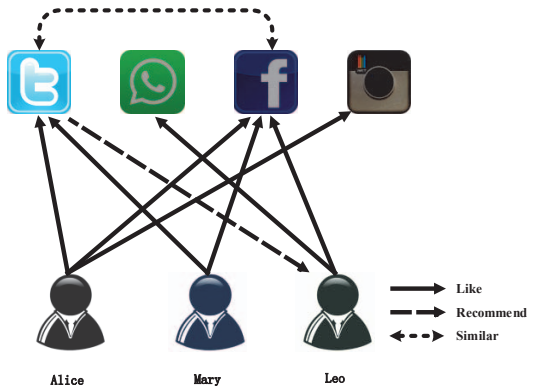

Figure 2. The item-oriented collaborative filtering. Alice likes application of twitter, Facebook, and Instagram. Mary likes twitter and Facebook. Leo likes Whatsapp and Facebook. After we consider twitter as the neighbor of Facebook, we can recommend twitter to Leo.

B. Collaborative Filtering

Collaborative Filtering (CF) is a popular technique for recommendation in E-Commerce. For example, the recommendation system of Amazon, which makes a contribution of 20%-30% its sales, is based on Item-to-Item Collaborative Filtering [14]. CF relies only on previous user behavior, e.g., their previous browsing history or product ratings. The two main categories of collaborative filtering are the neighborhood methods and the latent factor approaches. Neighborhood methods concentrate on analyzing the similarities among users or items. Top-N most similar users are called "neighbors", $N$is defined according to your datasets. The user-oriented approach predicts how a user likes an item based on the ratings by the user's neighbors. The item-oriented neighborhood models recommend the neighbors of products used by users. The neighbors of a product are the products that are likely to get similar ratings by the same user. Figure 2 shows an example of item-oriented collaborative filtering. In our study, we use r_{ui} to denote how interested user u is in an app i, and r_{ui} comprises the interest degree matrix $\boldsymbol{R}$. Neighborhood methods are easy to be understood and can receive better explanations, but are generally less accurate than latent factor models, especially when most elements of matrix $\boldsymbol{R}$ are unknown.

Latent factor models try to fill in the missing values in matrix $\boldsymbol{R}$ by uncovering latent factors of users and make users and applications comparable. Some latent relationships can be explainable, for example, belonging to the same category like social, while some latent relationships are difficult to interpret. The value p_{ij} which is located in the i-th line and j-th column of user-factor matrix $\boldsymbol{P}$, indicates the preference for user i over factor j. The element q_{jf} in app-factor matrix $\boldsymbol{Q}$ denotes the weights of app j in factor f. The higher q_{ij} is, the more representative for app j in factor i is. Matrix $\boldsymbol{R}$ is the inner product of matrix $\boldsymbol{P}$ and $\boldsymbol{Q}$, that is, $R = PQ$ (Figure 3). In general, the values in matrix $\boldsymbol{P}$ and $\boldsymbol{Q}$ can be obtained by minimizing a cost function (equation 5). In this work, we optimize the cost function by using stochastic gradient descent, which will be elaborated in Section IV-B.

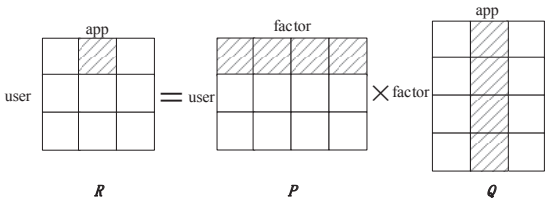

Figure 3. The latent factor model. The highlighted element in 1-th line and 2-th column is donated as r_{12}. r_{12} is the sum of products of the 1-th line in matrix $\boldsymbol{P}$ and 2-th column in matrix $\boldsymbol{P}$, that is, $r_{12} = \sum_{f=1}^{4} p_{1f} q_{f2}$.

III. Overview

In this section, we provide an overview of our approach. In the first instance, we define the mobile app recommendation problem, and then describe our dataset. Finally, we outline the framework of our methodology.

A. Problem Definition

The increasing prevalence of smart phones has led to an explosive increase of mobile applications. While the application markets allow users to perform fast keywords search or to browse top downloads of applications across different categories, it is still challenging for users to find applications that they actually like. For one reason, downloading an application is a poor indicator of whether the user likes the application, as the user may uninstall the application immediately after installing it. For another reason, the number of applications is large and applications are so varied that it is difficult for users to experience interesting applications that they would like. Although there are some industry softwares recommending applications for users, those softwares involves users' personal information more or less, which may result in privacy leaks. Considering the challenges, we present two hybrid approaches for mobile apps recommendation based on collaborative filtering in this paper. IIOCF model uses app-factor matrix got by latent factor approach instead of sparse preference matrix to compute the similarities among applications, thereby improving the performance of the item-oriented collaborative filtering. HLF model sums the results

TABLE I
KEY NOTATIONS

Notation	Description
M, N	The total number of users, the total number of apps
P, Q, R	The user-factor matrix, the app-factor matrix, the preference matrix
r_{ui}, $\hat{r}_{ui}$	The preference degree for user u over app i, the predicted preference degree for user u over app i
s_{ij}	The similarity between app i and app j
N(i, k)	The most similar k apps to app i
U(i, j)	The set of users that have used both app i and j

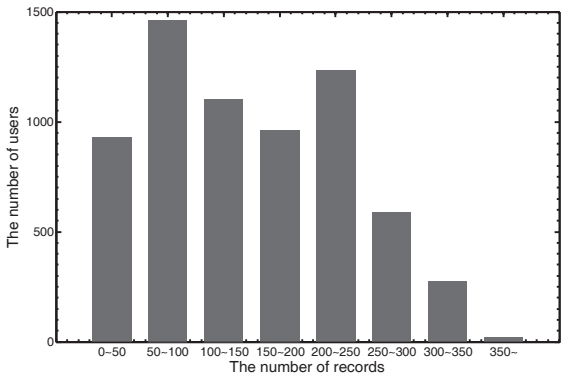

Figure 4. The number of records for individual mobile users. Most of the users have 50-250 records, while 17 users have over 350 records, indicating that they are heavy mobile users.

of the latent factor and item-oriented approach, and has better performance than both approaches.

Problem Definition. For user u and app i, the notation r_{ui} represents the preference for user u over app i. The higher r_{ui} is , the more user u likes app i. The set K stores the (u, i) pairs which are available from our dataset, and $K = \{(u, i)|r_{ui}$ is available$\}$. We use the notation $\hat{r}_{ui}$ to denote predicted preference for user u over app i, which can be obtained by our methodology. Given a set of *M* users and *N* apps, and usage records of apps, we aim to recommend each user the applications which the user has an interest in.

B. Data Description

Given the problem defined above, we then describe the dataset, including 25,302 users and 6,568 applications, we used in our experiments. We record usage behavior on applications for one week during November 2013, including the start time when a user opens an application, the end time when a user shuts down the application and the time during which the user is playing with the application. From figure 4 we can see most of the users have 50-250 records, while 17 users have over 350 records, indicating that they are heavy mobile users. The most frequently used apps, including QQ and wechat, belong to the category of social network service. Table I provides key notations and the corresponding descriptions throughout the paper.

C. The Framework of the Hybrid Approach

The whole framework consists of three stages: Preference Matrix Computation, Preference Matrix Factorization and Mobile App Recommendation.

- **Stage 1 (Preference Computation):** The preference computation is divided into two steps. Firstly, instead of using data of explicit feedback, which is difficult to collect, we take advantage of users' implicit feedback, such as usage frequency and time on each application by users, to measure how a user likes an application. We use the RFD (Recency, Frequency, Duration) model, a modified version of RFM (Recency, Frequency, Monetary) model, to label the preference for applications. Secondly, since the effects of users and applications are not independent, we add the deviations from the average preference of users and applications, known as biases, to the preference degree.
- **Stage 2 (Preference Matrix Factorization):** Given the matrix ***R*** with most elements are unknown, we leverage the latent factor model to extract user-factor matrix ***P*** and app-factor matrix ***Q***, elements in preference matrix ***R*** equals the inner product of ***P*** and ***Q***.
- **Stage 3 (Mobile App Recommendation):** We propose two hybrid models of the latent factor and item-oriented approach. Instead of using sparse matrix ***R***, IIOCF model realizes mobile app recommendation by leveraging the app-factor matrix ***Q*** we obtained in section IV-B to compute similarities among applications. HLF model sums the predictions of the latent factor model and item-oriented approach, thereby capturing the advantages of both approaches.

IV. Methodology

A. Preference Computation

1) Labeling Preference Degree: Collaborating filtering primarily takes explicit feedback, e.g., ratings as input. However, it is difficult to collect data of explicit feedback, as we can find that a popular application can only have 2 - 3% ratings. Hence, in this paper, we leverage implicit feedbacks data [15], such as usage frequency and time on applications by users to quantify the preference of users for applications. We use RFD (Recency, Frequency, Duration) model, to label the preference of users for applications. Recency means how recently the user has used the application, that is, the number of days that has passed since the last time the user played with the application. Frequency indicates how frequently the user used the application, that is, the number of times the user used the application. Duration indicates the total time the user spent on the application. Our work based on the assumption that the

more time a user spent on an application, the higher preference for the user over the application. So the higher RFD can be a strong indicator that the user likes the application.

We define recency a_{ui} as how recently user u interacts with app i, frequency f_{ui} as how frequently the user used the application, and duration d_{ui} as the total time the user spent on the application. The preference for user u over the application i is represented as follows.

$$r_{ui} = \alpha a_{ui} + \beta f_{ui} + \gamma d_{ui} \tag{1}$$

α, β, γ are the confidences based on their relative importance. We then define a preference matrix $\boldsymbol{R}$. If user u has usage history of app i, which represents a positive example, r_{ui} is computed using equation (1), otherwise, r_{ui} is temporarily labeled "?", as it represents an unknown example. Since the matrix has positive-only data, collaborating filter for learning from such matrix are referred as One-Class Collaborative Filtering (OCCF) [16].

2) Adding Biases: The preference for users over applications may be affected by other users and applications. The bias-approximation method takes the biased of users and applications into consideration, and adds biases b_{ui} to preference degree r_{ui}.

$$b_{ui} = u + b_u + b_i \tag{2}$$

u is the average preference degree of all applications, b_u and b_i denote the deviations from the average preference of the user u and app i respectively. For example, if we want to estimate the preference degree for Alice over 'Facebook', which is a popular social application got 0.21 higher than the average preference degree 0.46. Alice is very critical and her preference degree is 0.15 lower than the average. Hence, we approximate Alice's preference for 'Facebook' is 0.52 (0.46 + 0.21 - 0.15). The value of b_u and b_i can be obtained by optimizing the following least squares function:

$$\sum_{(u,i)\in K} (r_{ui} - u - b_u - b_i)^2 + \lambda_1(\|b_u\|^2 + \|b_i\|^2) \tag{3}$$

The term $\lambda_1({b_u}^2 + {b_i}^2)$ is used for avoiding overfitting, and λ_1 is the regularization parameter.

B. Preference Matrix Factorization

Given the matrix $\boldsymbol{R}$ with unknown data "?", our goal is to identify what "?" is from positive examples. Latent factor models aim to fill in the missing value in matrix $\boldsymbol{R}$ by uncovering latent factors of users and items. Some latent relationships can be explainable, for example, belonging to the same category like social, while some latent relationships are difficult to interpret. Taking three people Alice, Mary and Leo as an example, Alice may be more interested in applications in the category of social like Facebook, Mary prefers reading and Leo spends much time listening to music. If we recommend applications to the three people, we can give priority to those apps belonging to the same categories that they like. To do this, we need to extract the latent factors from applications

Algorithm 1: Minimization of the cost function

Input

The preference matrix $\boldsymbol{R}$;
Parameters: $\alpha, \beta, \gamma, \delta, \lambda_1, \lambda_2$;
The dimensions of latent factors f;

Output

Matrix $\boldsymbol{R}$,$\boldsymbol{P}$ and $\boldsymbol{Q}$;

1: Initialized the user-factor matrix $\boldsymbol{P}$ with i.i.d number
2: Initialized the app-factor matrix $\boldsymbol{Q}$ with i.i.d number
3: **repeat**
4: Modifying parameters,yielding:

$$\begin{aligned} e_{ui} &= r_{ui} - u - b_u - b_i - p_u q_i \\ b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda_2 b_u) \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda_2 b_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} q_i - \lambda_2 p_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} p_u - \lambda_2 q_i) \end{aligned}$$

5: **until** $C(P, Q)$ reaches convergence
6: **return** Matrix $\boldsymbol{P}$ and $\boldsymbol{Q}$

and then define user-factor matrix $\boldsymbol{P}$=$(P_{ij})_{m\times f}$ and app-factor matrix $\boldsymbol{Q}$=$(Q_{ij})_{f\times n}$. P_{ij} indicates the preference for user i over factor j, and Q_{ij} denotes the weights of app j in factor i, f is the dimensions of factors. The higher Q_{ij} is, the more representative for app j in factor i is. $\boldsymbol{PQ}$ captures the inner relationship between users and apps, and approximates the meaning of the preference matrix $\boldsymbol{R}$. Thus, the predicted preference for user u over app i can be denoted as follows:

$$\hat{r}_{ui} = u + b_u + b_i + p_u q_i \tag{4}$$

In light of the difficulty in extracting the latent factors, three steps are proposed to solve the problem. To begin with, we initialize the user-factor matrix $\boldsymbol{P}$ and app-factor matrix $\boldsymbol{Q}$ with independent identically distributed (i.i.d) numbers from a Gaussian distribution with zero mean and small standard deviation. After this, we establish the squared cost function $\boldsymbol{C}$ as follows.

$$\begin{aligned} C(P, Q) = \sum_{(u,i)\in K} &(r_{ui} - u - b_u - b_i - p_{u.} q_{.i})^2 + \lambda_2 \\ &(\|b_u\|^2 + \|b_i\|^2 + \|p_u\|^2 + \|q_i\|^2) \end{aligned} \tag{5}$$

K is the set of (u, i) pairs representing positive examples and the corresponding r_{ui} can be computed using equation 1 [17]. $\lambda_2(\|b_u\|^2 + \|b_i\|^2 + \|p_u\|^2 + \|q_i\|^2)$ is L2 norm regularization avoiding overfitting and the regularization term λ_2 is determined in general by cross-validation [18].

At last, we minimize $C(P, Q)$ in equation (5) to fill in the unknown data in the preference matrix $\boldsymbol{R}$. Stochastic gradient descent (SGD) [19] and alternating least squares (ALS) [20] are two popular approaches in matrix factorization. We prefer stochastic gradient descent because it has shorter running time than the other when the preference matrix is sparse. Taking

partial derivatives of $\boldsymbol{C}$, and we modify the parameters in the opposite direction of the gradient, which is the steepest descent direction, yielding:

$$e_{ui} = r_{ui} - u - b_u - b_i - p_u q_i \quad (6)$$
$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda_2 b_u) \quad (7)$$
$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda_2 b_i) \quad (8)$$
$$p_u \leftarrow p_u + \gamma(e_{ui} q_i - \lambda_2 p_u) \quad (9)$$
$$q_i \leftarrow q_i + \gamma(e_{ui} p_u - \lambda_2 q_i) \quad (10)$$

The notation γ is the learning rate, and has direct effect on the rate of descent. The details are described in **Algorithm 1**.

C. Mobile App Recommendation

After the above stage, we can obtain three matrix: preference matrix $\boldsymbol{R}$, app-factor matrix $\boldsymbol{Q}$, user factor matrix $\boldsymbol{P}$. Then we propose two hybrid models of the latent factor and item-oriented approach.

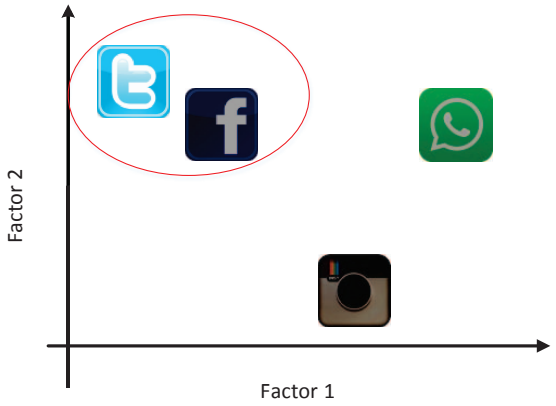

Figure 5. A simplified example of the distribution of applications in two-dimensional space. Facebook and twitter are very close and are considered as 'neighbors'.

1) IIOCF Model: The Item-oriented neighborhood models directly use the preference matrix $\boldsymbol{R}$ to compute the similarities among items, and recommend the most similar applications. However, the preference degree in the preference matrix $\boldsymbol{R}$ is sparse, and it is not convincing to merely use the preference degree to measure the similarities among applications. To handle the issue, we find that the latent factor model can discover the latent relationships between users and applications. Some latent relationships can be explainable, for example, belonging to the same category like social, while some latent relationships are difficult to interpret. Figure 5 illustrates a simplified example of the distribution of applications in two-dimensional space. The three apps: twitter, Facebook and Whatsapp are very close when merely considering factor 1. However, taking both factor 1 and factor 2 into consideration, only twitter and Facebook are similar apps. Hence, with the more factor been discovered by the latent factor model, the similarities among applications can be more accurate. Given the app-factor matrix $\boldsymbol{Q}$ we obtained in section IV-B, we smoothly integrate latent factor models with neighborhood models. We use the app-factor matrix $\boldsymbol{Q}$ to compute similarities among applications, thereby realizing the combination of latent factor models and neighborhood models. The i-th column in matrix $\boldsymbol{Q}$ can be put into a feature vector ${q_i}^T =< q_{i1}, q_{i2}, \ldots, q_{if} >$, which represents the latent factors of app i. Also, the feature vector of app j can be denoted as ${q_j}^T =< q_{j1}, q_{j2}, \ldots, q_{jf} >$. Then we can use Pearson correlation coefficient to measure the similarity between app i and app j:

$$s_{ij} = \frac{\Sigma_f(q_{if} - \overline{q_i}) * (q_{jf} - \overline{q_j})}{\sqrt{\Sigma_f(q_{if} - \overline{q_i})^2} * \sqrt{\Sigma_f(q_{jf} - \overline{q_j})^2}} \quad (11)$$

$\overline{q_i}$ denotes the mean value of the feature vector ${q_i}^T$, and $\overline{q_j}$ denote the mean value of ${q_j}^T$ correspondingly. Compared with using preference degree in matrix $\boldsymbol{R}$ as the signal indicator to measure the similarities among applications, the app-factor matrix $\boldsymbol{Q}$ can discover more latent relationships among applications.

After computing the similarities between any two applications, we rank the similarities for each application, the k ranked highest are called the k neighbors of the applications. For example, $N(i,k)$ denotes the K neighbors of app i, that is, the most similar k apps to app i. For each user, we predict the preference degree of those applications that users have not interacted with in our dataset. The predicted preference $\hat{r}_{ui}$ of user u for app i can be computed by the following equation:

$$\hat{r}_{ui} = b_{ui} + \frac{\Sigma_{j \in N(i,k)} s_{ij}(r_{uj} - b_{uj})}{\Sigma_{j \in N(i,k)} s_{ij}} \quad (12)$$

After this stage, we have obtained the predicted preference degree of users for any applications. So in the recommendation stage, we rank the preference degree and recommend the top N applications that users have never interacted with. The results show that the IIOCF model has better performance than the item-oriented approach.

2) HLF Model: The popular computation of the predicted value of r_{ui} is illustrated in equation 12. However, the interpolation weights [21] in equation 12 are fully depend on the neighbors of applications, and can result in problems in the extreme situation where some user merely have one recorded app i. In this situation, app i does not have neighbors, and equation 12 becomes not very precise. Hence, we use the more accurate equation in [18] as follows.

$$\hat{r}_{ui} = b_{ui} + \Sigma_{j \in N(i,k)} w_{ij}^u (r_{uj} - b_{uj}) \quad (13)$$

$\{w_{ij}^u | j \in N(i,k)\}$ is the interpolation weights. The detailed computation of w_{ij}^u refers to [21]. Equation 13 does not sum to one and overcomes the difficulty that relying fully on neighbors of apps. HLF model sums equation 4 and equation 13, thus the predicted preference for user u over app i is denoted as follows:

$$\hat{r}_{ui} = u + b_u + b_i + p_u q_i + \Sigma_{j \in N(i,k)} w_{ij}^u (r_{uj} - b_{uj}) \quad (14)$$

HLF model makes a combination of the item-oriented and latent factor approach, thereby capturing the advantages of both approaches. Latent factor approach is a learning method and has a very good theoretical foundation. Item-oriented approach does not have learning process, but its results are easy to be explained. The experiment results of HLF model is better than either item-oriented or latent factor approach.

V. Experiments

A. Data Pre-processing

As simply described in section III-B, each record has four core attributes including *UserID*, *AppName*, *RecTime*, *OnlineTime*. *UserID* is the unique identifier of each user and *AppName* is the name of application. *RecTime* is the time at which users opened applications. *OnlineTime* represents how much time the user spends on the application. The original dataset has 2,482,168 records. Firstly, we do some statistical analysis to get recency, frequency and duration that we need in RFD model. After the statistics work, the dataset has 352,172 records. And we find the number of recorded applications of some users is less than five. We consider these users as light mobile users, and remove them at the pre-processing stage since it is not beneficial for cross validation. After pre-processing, the number of records reduces to 264,191, and the number of users and applications are respectively 25,302 and 6,568. To simplify our calculation, we use a simple method, dividing the preference degree of each user is to the maximum degree of the user, to normalize the preference degree of each user to the interval between 0 and 1. Then we use 5-fold cross validation to divide the dataset into training set and testing set. The proportion of the training set and the testing set is 7/3.

B. Baseline Algorithms

Item-oriented Collaborative Filtering. Instead of using app-factor matrix $\boldsymbol{Q}$, which we got in section IV-B, the baseline approach directly use the preference matrix $\boldsymbol{R}$ to compute the similarities ρ_{ij} (equation 15) among items.

$$\rho_{ij} = \frac{\Sigma_{u \in U(i,j)}(r_{ui} - b_i) * (r_{uj} - b_j)}{\sqrt{\Sigma_{u \in U(i,j)}(r_{ui} - b_i)^2} * \sqrt{\Sigma_{u \in U(i,j)}(r_{uj} - b_j)^2}} \tag{15}$$

U(i, j) denotes the set of users that have used both app i and j. b_i is the average preference degree of app i, and correspondingly, b_j represents the average preference degree of app j. Since the ratio of unknown elements in the preference matrix $\boldsymbol{R}$ is as high as 99.2%, we borrow a shrunk correlation coefficient s_{ij} from [18] to measure the similarity:

$$s_{ij} = \frac{n_{ij}}{n_{ij} + \lambda_2} \rho_{ij} \tag{16}$$

n_{ij} denotes the number of users that have interacted with both app i and app j. λ_2 is a constant and can be modified according to different datasets. In our experiment, λ_2 is set as 30. The predicted preference of user u for app i, $\hat{r}_{ui}$, can be obtained by equation (12).

Latent Factor Model. Latent factor model is a pure learning method and in general has better performance than the neighborhood method. Latent factor can uncover latent relationships among users and applications and make users and applications comparable. In **Algorithm 1**, we obtain the user-factor matrix $\boldsymbol{Q}$ and app-factor matrix $\boldsymbol{Q}$. According to the equation $R = PQ$, we can obtain the preference matrix $\boldsymbol{R}$ with most elements are positive. Then for each user, we rank the predicted preference degree and recommend the k apps that rank highest. When learning rate γ is small, the rate of convergence becomes slower and the number of iterations gets larger. Hence, we set the learning rate γ 0.02 at first, then we use the following equation to update γ:

$$\gamma = \gamma \times 0.9 \tag{17}$$

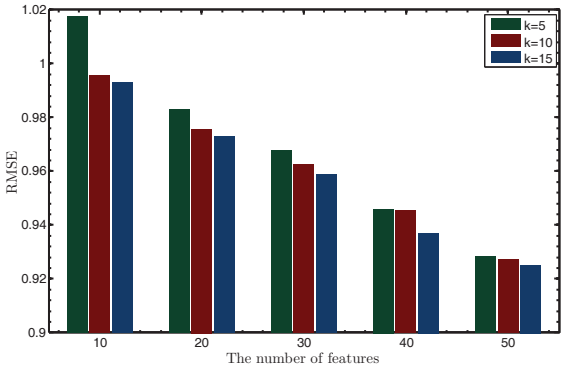

Figure 6. The influence of k and factors on RMSE. Within a certain range, the increase of k and f improves the performance.

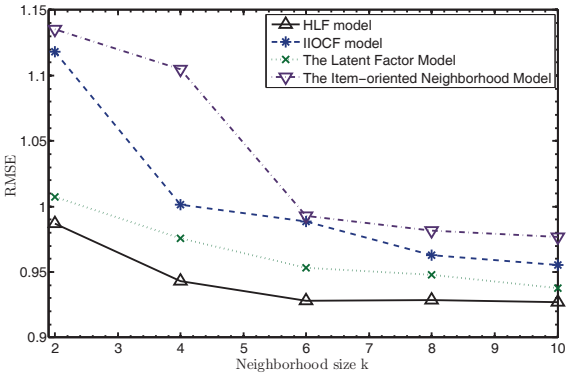

Figure 7. Comparison among the item-oriented collaborative filtering, the latent factor model, IIOCF model and HLF model. IIOCF model improves the performance of the item-oriented CF and HLF model has the best performance.

C. Evaluation Metrics

We use **RMSE** (Root Mean Square Error), a popular evaluation metric in recommendation system, as our evaluation metrics. For a pair (u, i) in testing Set, the real preference for

user u over app i is r_{ui}, and the predicted preference is $\hat{r}_{ui}$. Hence, RMSE is defined as follows:

$$RMSE = \frac{\sqrt{\Sigma_{(u,i)\in T}(\hat{r}_{ui} - r_{ui})^2}}{|T|} \tag{18}$$

T denotes the testing set and $|T|$ is the number of records in testing set.

D. Experiment Results

Figure 6 shows the influence of the neighborhood size k and factor f on RMSE. Our experiment shows: within a certain range, the increase of k and f improves the performance. What needs to be paid attention is that the increasing factors will increase the running time. Figure 7 shows the comparison among the item-oriented collaborative filtering, the latent factor model, IIOCF model and HLF model. The latent factor is set as 50. In this figure, we can see IIOCF model improves the performance of the item-oriented collaborative filtering. It is because IIOCF model leverages the latent factor approach to discover the latent relationships such as the categories among applications, while the item-oriented collaborative filtering leverages the preference degree as the signal indicator to measure the similarities among applications. Also, we can find that HLF model has better performance than both the item-oriented collaborative filtering and the latent factor model.

VI. Conclusion

The purpose of this paper is to realize mobile app recommendation for mobile users. In this paper, we leverage users' usage data, which can be easily collected from telecom operators, to help users find suitable applications. Since we only have implicit feedback of users, e.g., frequency and time spent on applications by users, we first use RFD (Recency, Frequency, Duration) model to label the preference for users over applications. Thus we turn the implicit data into explicit data, which can be taken as input in collaborative filtering. We propose two hybrid models combining the latent factor and item-oriented approach: IIOCF model and HLF model. IIOCF model leverages the latent factor models to find the latent relationships among applications, and after this we compute the similarities among applications using the app-factor matrix obtained in the latent factor models stage instead of using preference degree as the signal indicator to measure the similarities. IIOCF model improves the performance of the item-oriented approach. HLF model sums the predictions of the latent factor model and item-oriented approach, thereby capturing the advantages of both approaches. Our experiment results over 6,568 applications and 25,302 users clearly show that HLF model has better performance than both the item-oriented and latent factor approach.

Acknowledgements

This research is supported in part by 973 Program (2014CB340303), 863 Program (No. 2015AA015303), NSFC (No. 61472254, 61170238 and 61420106010), STCSM (Grant No.14511107500 and 15DZ1100305), Research Grant for Young Faculty in Shenzhen Polytechnic (No. 601522K30015), SZSTI (No. JCYJ20160407160609492) and Singapore NRF (CREATE E2S2). This work is also supported by the Program for New Century Excellent Talents in University of China, the Program for Changjiang Young Scholars in University of China, and the Program for Shanghai Top Young Talents.

References

[1] "App Store", https://itunes.apple.com/us/genre/ios/id36?mt=8.
[2] "Google Play", https://play.google.com/store.
[3] "WWDC", https://developer.apple.com/wwdc/.
[4] Yehuda Koren, Robert Bell, and Chris Volinsky, "Matrix factorization techniques for recommender systems", *Computer*, , no. 8, pp. 30–37, 2009.
[5] Bo Yan and Guanling Chen, "Appjoy: personalized mobile application discovery", in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 113–126.
[6] "AppBrain", https://www.appbrain.com/.
[7] "AppFire", https://www.appfire.com/.
[8] Jan Roelf Bult and Tom Wansbeek, "Optimal selection for direct mail", *Marketing Science*, vol. 14, no. 4, pp. 378–394, 1995.
[9] Wolfgang Woerndl, Christian Schueller, and Rolf Wojtech, "A hybrid recommender system for context-aware recommendations of mobile applications", in *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*. IEEE, 2007, pp. 871–878.
[10] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin, "Diversity in smartphone usage", in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 179–194.
[11] Kent Shi and Kamal Ali, "Getjar mobile application recommendations with very sparse datasets", in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 204–212.
[12] Kuifei Yu, Baoxian Zhang, Hengshu Zhu, Huanhuan Cao, and Jilei Tian, "Towards personalized context-aware recommendation by mining context logs through topic models", in *Advances in Knowledge Discovery and Data Mining*, pp. 431–443. Springer, 2012.
[13] Hengshu Zhu, Enhong Chen, Kuifei Yu, Huanhuan Cao, Hui Xiong, and Jilei Tian, "Mining personal context-aware preferences for mobile users", in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 1212–1217.
[14] Greg Linden, Brent Smith, and Jeremy York, "Amazon. com recommendations: Item-to-item collaborative filtering", *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76–80, 2003.
[15] Yifan Hu, Yehuda Koren, and Chris Volinsky, "Collaborative filtering for implicit feedback datasets", in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 263–272.
[16] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang, "One-class collaborative filtering", in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 502–511.
[17] Arkadiusz Paterek, "Improving regularized singular value decomposition for collaborative filtering", in *Proceedings of KDD cup and workshop*, 2007, vol. 2007, pp. 5–8.
[18] Yehuda Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model", in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.
[19] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent", in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 69–77.
[20] K Ruben Gabriel and Shmuel Zamir, "Lower rank approximation of matrices by least squares with any choice of weights", *Technometrics*, vol. 21, no. 4, pp. 489–498, 1979.
[21] Robert M Bell and Yehuda Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights", in *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. IEEE, 2007, pp. 43–52.