World Wide Web (2018) 21:89–104
https://doi.org/10.1007/s11280-017-0456-y

Personalized app recommendation based on app permissions

Min Peng[1] · Guanyin Zeng[1] · Zhaoyu Sun[1] · Jiajia Huang[2] · Hua Wang[3] · Gang Tian[1]

Received: 14 December 2016 / Revised: 17 February 2017 / Accepted: 27 March 2017 / Published online: 17 April 2017

Abstract With the development of science and technology, the popularity of smart phones has made exponential growth in mobile phone application market. How to help users to select applications they prefer has become a hot topic in recommendation algorithm. As traditional recommendation algorithms are based on popularity and download, they inadvertently fail to recommend the desirable applications. At the same time, many users tend to pay more attention to permissions of those applications, because of some privacy and security reasons. There are few recommendation algorithms which take account of apps' permissions, functionalities and users' interests altogether. Some of them only consider permissions while neglecting the users' interests, others just perform linear combination of apps' permissions, functionalities and users' interests to implement top-N recommendation.

This article belongs to the Topical Collection: *Special Issue on Security and Privacy of IoT*
Guest Editors: Tarik Taleb, Zonghua Zhang, and Hua Wang

✉ Gang Tian
tiang2008@whu.edu.cn

Min Peng
pengm@whu.edu.cn

Guanyin Zeng
gunnyzeng@whu.edu.cn

Zhaoyu Sun
sunzhaoyu@whu.edu.cn

Jiajia Huang
whujenny@qq.com

Hua Wang
Hua.Wang@vu.edu.au

[1] School of Computer, Wuhan University, Wuhan, China

[2] School of Technology, Nanjing Audit University, Nanjing 211815, People's Republic of China

[3] Centre for Applied Informatics, Victoria University, Melbourne, Australia

In this paper, we devise a recommendation method based on both permissions and functionalities. After demonstrating the correlation of apps' permissions and users' interests, we design an app risk score calculating method ARSM based on app-permission bipartite graph model. Furthermore, we propose a novel matrix factorization algorithm MFPF based on users' interests, apps' permissions and functionalities to handle personalized app recommendation. We compare our work with some of the state-of-the-art recommendation algorithms, and the results indicate that our work can improve the recommendation accuracy remarkably.

Keywords App permissions and privacy · App recommendation algorithms · App functionalities · User interests

1 Introduction

Recent years have witnessed the rapid development in smart phone industry, and its popularity prospers mobile phone application software market. According to statistics, by the July of 2013, there are more than one million apps on the Google Play, with fifteen billion times of download. And by the end of 2014, the amount of download adds up to seventy-five billion times in just one year. What is more astonishing is that the amount of download increases to 156 billion all around the world in 2015. For users, there is an urgent need for selecting interested apps from massive amounts of apps. Since 2012, many recommendation algorithms have been proposed, such as mining personal context-aware preferences for mobile users [27], personalized context-aware recommendation by mining context logs through topic models [23], and mobile application recommendation with very sparse datasets [20, 24]. All these recommendation algorithms are designed for mobile phone users.

Generally, users know very little about apps, especially in the aspect of apps' permissions, as many apps require some permissions to some authorities [22] in order to provide better user experience. Therefore, many malicious apps may disclose users' privacy information by furtively applying permissions [25]. Even worse, with the purpose of occupying the app market, some apps apply for unnecessary permissions to obtain users' personal information [15]. In case of this undesirable phenomenon, some recommendation algorithms are devised to solve these problems, such as using probabilistic generative models to rank risks of apps [23], mobile app recommendation with security and privacy [26, 28], etc. But all these recommendation algorithms just quantify permissions and functionalities, or make a simple linear combination of them [9]. They potentially ignore the relationship between the apps' permissions and users' interests in functionalities. However, according to empirical analysis, when two apps have similar functionalities and their risk scores are known to users, they tend to choose the one that is more security. In this case, malicious apps usually have lower users' ratings. In other words, if two apps have the same level of permission security, then their ratings are similar, too. As users' ratings are reflections of their preferences on apps' functionalities, we hence assume that apps' permissions are internally related to users' interests.

In this paper, we propose a novel matrix factorization algorithm based on permissions and functionalities (MFPF). Our algorithm exploits the relationship between the permissions and functionalities to achieve personalized recommendation for the first time. The main contributions of this paper are as follows: (1) Based on the experimental dataset, the distribution of the apps' permissions is analyzed and we also elaborate specific conditions of apps' permissions. (2) We analyze the relationship between the apps' permissions and users' ratings, and then verify this relationship in detail. (3) We construct a matrix factorization algorithm to perform recommendation. This algorithm recommends apps by integrating users' interests on apps' functionalities, as well as the apps' permissions. (4) In experiments, we evaluate the proportion and influence of the permissions in the users' ratings. Then we make further analysis on the relationship between users' ratings and apps' permissions. We also compare our algorithm with other mainstream algorithms, and find that our algorithm is more accurate.

2 Related work

Most current recommendation methods are based on collaborative filtering, which can be classified in two types. One is the collaborative filtering based on users or items [12, 19], which performs personalized recommendation according to the similarity between users or items. The other is based on latent factor model (e.g., matrix factorization) [3, 5, 6, 10].

Meanwhile, the popularity of smart phone has raised the related researches on personalized recommendation system. Among them is recommendation system about users' preferences, such as mining personal context-aware preferences [27] and personalized context-aware recommendation by mining context logs through topic models [11, 23]. Although these methods can do pretty well in preference recommendation, they still require a large number of information about users' mobile phones in order to achieve better results. These methods on one hand have problems of disclosing users' personal information, on the other hand they have trouble in obtaining enough information, so there exists a cold-start problem in these recommendation systems.

Since 2012, mobile app recommendation system has received wide attraction from researchers, among them are a sort of recommendation systems that can predict users' usage patterns on mobile app management, such as predicting the mobile application patterns [21] and the most probable app to be used [1]. This kind of recommendation system, which is one of the main topics in research about app recommendation system, is a combination of users' behaviors and apps' functionalities. There are also some recommendation systems which focus on solving sparseness and cold-start problem, such as recommending mobile app with very sparse datasets [20], which aims at solving the problem of data sparseness, and addressing cold-start problem by using social network [8, 14]. The above mentioned recommendation systems are a part of obtained fruit of researches in app recommendation system with the increasing popularity of smart phone and in-depth studies.

In recent years, there has emerged a new realm in recommendation system, which focuses on security of users' privacy [2, 7]. In order to improve user experience, mobile app developers need to apply some permissions which are usually related to users' personal information. Due to users' ignorance about apps' permissions, offenders can also utilize these permissions to develop malwares, so as to disclose users' privacy. In this paper, we conduct code detection for each app in our experiments to analyze apps' permissions in detail. From our analysis, we find that there are about 120 kinds of app permissions, and we also notice that among the frequently used permissions, almost 80% of them concentrate on certain kinds of permissions like "WRITE_EXTERNAL_STORAGE", "ACCESS_NETWORK_STATE", and so on. But there remains some kinds of permissions which are used relatively infrequently, and these permissions are usually more dangerous, like "INSTALL_PACKAGES", "SEND_SMS", and so on. Peng, et al. [23] point out that these kinds of permissions are usually used in malwares. Obviously, we can preliminarily determine whether an app is malware or not through analysis of its permissions. Therefore, it is necessary and effective to consider aspects of apps' permissions in personalized recommendation system.

Despite of the fact that app's permission is an important factor in personalized recommendation algorithm, there are few mature researches in this field. Although there are some excellent achievements, their works are still in the early stage. The method proposed by Peng, et al. ranks app risk by incorporating probabilistic generative models [23], and recommends apps with respect to this risk score. It does obtain good results, but it only analyzes security aspect from the code, the apps' permissions and users' interests in apps' functionalities are not considered. Liu, et al. make improvements by conducting top-N recommendation with integrating apps' functionalities, apps' permissions and users' interests [9]. However, there still remain some flaws in their method. They just make a simple linear combination of all the components to perform top-N recommendation, while ignoring the relationship between apps' permissions and users' interests.

In this paper, we first elaborate and verify the relationship between users' interests and apps' permissions. Then we propose a new recommendation algorithm based on matrix factorization (MFPF), which combines apps' permissions, apps' functionalities as well as users' interests. Finally we compare our algorithm with other recommendation algorithms such as SVD++ [5], BiasedMF [6], ItemKNN [19], as well as Privacy_Res and Sensitive_Perm, which are mentioned in [9].

3 Preliminary analysis of relationship between apps' permissions and users' interests

Most of the security apps need fixed kinds of permissions, only a handful of apps need relatively uncommon permissions. However, the malwares are just the opposite [23]. Figure 1 is the distribution map of apps' permissions. From Figure 1, we can find that apps' permissions concentrate on dozens of specific kinds, but there still remain some seldom-used permissions, and these permissions are more dangerous. So if we analyze apps'

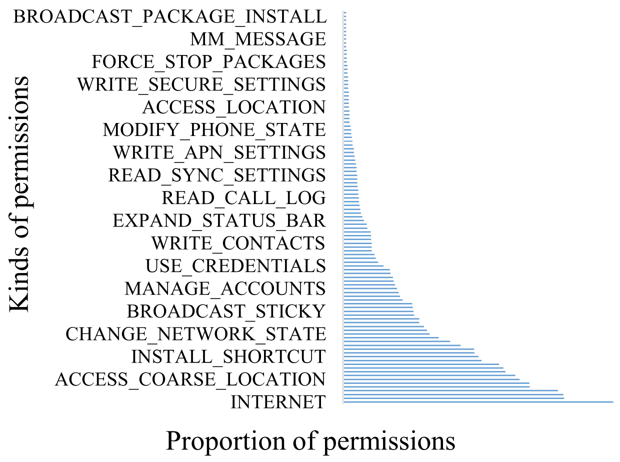

Figure 1 Distribution of app permissions

permissions, we can determine whether an app is dangerous and malicious subjectively. In order to achieve a more accurate and objective effect, Zhu, et al. propose a method which can judge app security by introducing the concept of risk score [28]. App risk score reflects its privacy security, the higher the risk score, the lower the privacy security, and vice versa.

Based on [28], we propose an app risk score calculating method (ARSM) to ingeniously quantify the degree of app potential risk. ARSM is inversely proportional to app reliability, and its construction refers to the app-permission bipartite graph model [28], as shown in Figure 2. Let *Edge* be the set of edges ($e_{ij} \in Edge$ only when app a_i requests permission p_j), and $pro_{ij} \in Pro$ be the weight of edge e_{ij}, which is the probability of a_i needing p_j. pro_{ij} is obtained by calculating permissions of apps which are in the same category with a_i, that is:

$$pro_{ij} = \frac{c_{ij}}{\sum_{e_{ik} \in Edge} c_{ik}}, \tag{1}$$

where c_{ij} is the total number of apps in category $t(a_i \in t)$ requesting permission p_j. We express a_i and p_j in vectors, i.e., $\mathbf{a}_i = \{pro_{i1}, pro_{i2}, ..., pro_{iN}\}$, $\mathbf{p}_j = \{pro_{1j}, pro_{2j}, ..., pro_{Mj}\}$, and formulate app similarity sim^a_{ij} as follows:

$$sim^a_{ij} = \cos\left(\mathbf{a}_i, \mathbf{a}_j\right) = \frac{\mathbf{a}_i \cdot \mathbf{a}_j}{\|\mathbf{a}_i\| \cdot \|\mathbf{a}_j\|}. \tag{2}$$

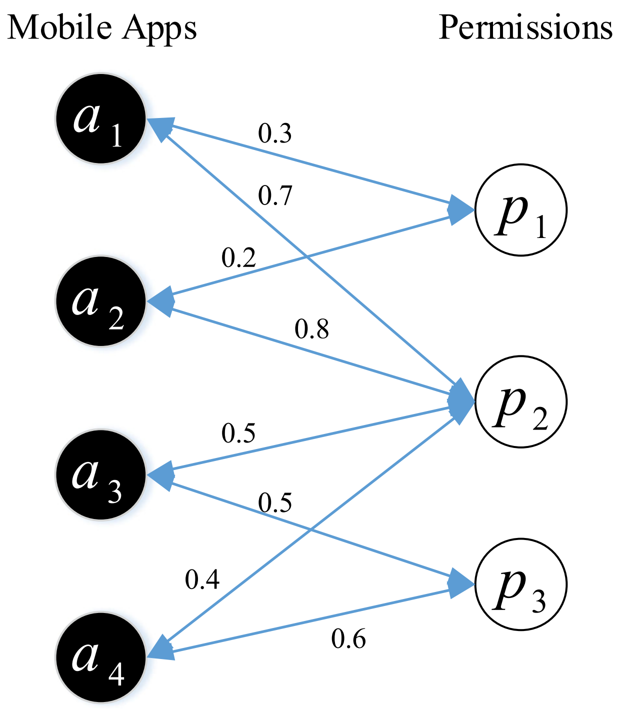

Figure 2 An example of the bipartite graph between apps and permissions

Similarly, permission similarity sim^p_{ij} can also be calculated by $sim^p_{ij} = \cos(\mathbf{p}_i, \mathbf{p}_j)$. Therefore, we present the calculation of app risk score loss function in our ARSM as follows:

$$\begin{aligned} f(a,p) = &\frac{\lambda}{2}\left\{\sum_i \left\| Rs^a_i - \widetilde{Rs^a_i}\right\|^2 + \sum_j \left\| Rs^p_j - \widetilde{Rs^p_j}\right\|^2\right\} \\ &+\frac{\mu}{2}\left\{\sum_{i,j} sim^a_{ij}\left\| Rs^a_i - Rs^a_j\right\|^2 + \sum_{i,j} sim^p_{ij}\left\| Rs^p_i - Rs^p_j\right\|^2\right\} \\ &+\frac{1}{2}\sum_{i,j} pro_{ij}\left\| Rs^a_i - Rs^p_j\right\|^2, \end{aligned} \tag{3}$$

where Rs^a_i, Rs^p_j are the risk scores of app a_i and permission p_j derived from prior knowledge. The calculation of app risk score loss function (3) is composed of three parts. The first part controlled by λ defines the constraint that the apps' and permissions' risk scores should fit prior knowledge. The second part is the global constraint controlled by parameter μ, which is the balance of risk scores, thus needs to satisfy the hypothesis that if two apps (or two permissions) are highly similar, then their risk scores should also be similar. The third part is the smoothness constraint between apps and permissions, which guarantees that if two apps have high probabilities to request a specific permission, then their risk scores should be similar. Finally, we only need to calculate values of Rs^a_i, Rs^p_j with iterative minimization of $f(a,p)$ by (4) and (5). Notice that in this paper, users' ratings refer to users' interests in apps' functionalities, and in our experiments, users' ratings are in the interval [1-5], so we use sigmoid function $\text{Sig}(x) = 5/(1+e^{-x})$ to map risk scores Rs^a_i, Rs^p_j to [1-5].

$$\begin{aligned} \frac{\partial f(a,p)}{\partial a_i} &= \lambda(Rs^a_i - \widetilde{Rs}^a_i) + \mu\sum_j sim^a_{ij}(Rs^a_i - Rs^a_j) + \sum_j pro_{ij}(Rs^a_i - Rs^p_j), \\ Rs^a_i &= Sig\left(\frac{\lambda\widetilde{Rs^a_i} + \mu\sum_j sim^a_{ij}Rs^a_j + \sum_j pro_{ij}Rs^p_j}{\lambda + \mu\sum_j sim^a_{ij} + \sum_j pro_{ij}}\right). \end{aligned} \tag{4}$$

$$\begin{aligned} \frac{\partial f(a,p)}{\partial p_j} &= \lambda(Rs^p_j - \widetilde{Rs}^p_j) + \mu\sum_i sim^p_{ij}(Rs^p_j - Rs^p_i) + \sum_i pro_{ij}(Rs^p_j - Rs^a_i), \\ Rs^p_j &= Sig\left(\frac{\lambda\widetilde{Rs^p_j} + \mu\sum_i sim^p_{ij}Rs^p_i + \sum_i pro_{ij}Rs^a_i}{\lambda + \mu\sum_i sim^p_{ij} + \sum_i pro_{ij}}\right). \end{aligned} \tag{5}$$

Firstly we initialize Rs^a_i to $1/M$, Rs^p_j to $1/N$. M and N is the number of apps and permissions respectively. Then according to PNB (Naive Bayes with information Priors) [4, 13], we can derive $\widetilde{Rs^a_i} = -lnP(p_1, ..., p_k|\theta)$ and $\widetilde{Rs^p_j} = -lnP(p_j|\theta)$, where $P(p_j|\theta)$ can be estimated by (6), and parameter θ follows beta prior distribution $Beta(\theta; \alpha_0, \beta_0)$.

x_{ij} is the binary function, when a_i requests p_j, its value is 1, or its value is 0.

$$P(p_j|\theta) = \frac{\sum_{i}^{M} x_{i,j} + \alpha_0}{M + \alpha_0 + \beta_0}. \tag{6}$$

Therefore, we use gradient descent method to derive the risk scores according to iterative update, as shown in (4) and (5). Meanwhile, we can calculate app risk score Rs_i^a of every app, and make comparison between Rs_i^a and users' average ratings. Figure 3 is the line chart of app risk scores and average users' ratings. By analyzing Figure 3, we can find that the higher the app risk scores, the lower the users' ratings (significantly exhibits in the black box). This means users' ratings can not only figure out users' interests in apps' functionalities, but also reflect security of apps' permissions, even though the permissions are not intuitive enough for users directly. In other words, there exists a relationship between users' interests and apps' permissions. But this relationship is not explicit enough, as almost all app risk scores are in the interval [2.5-3], making them too crude to detect malware. In the following part, we will propose a new method which can formulate the relationship between users' interests and apps' permissions more intuitively. Furthermore, we conduct personalized recommendation based on this relationship.

4 Matrix factorization algorithm MFPF

In last section, we have proved that there is a relationship between users' interests in apps' functionalities and apps' permissions preliminarily. In this section, we will introduce our algorithm more precisely. Specifically, we propose a new matrix factorization algorithm based on users' interests and apps' permissions (MFPF). By integrating users' interests, apps' functionalities, as well as apps' permissions, we can provide a more precise and effective personalized recommendation algorithm.

4.1 Definition description

In our recommendation algorithm, the set of users is denoted as $U = \{u_1, u_2, ..., u_M\}$, the set of apps is denoted as $A = \{a_1, a_2, ..., a_I\}$, and the set of permissions is denoted as $P = \{p_1, p_2, ..., p_N\}$. Table 1 shows all of the important symbols used in this paper.

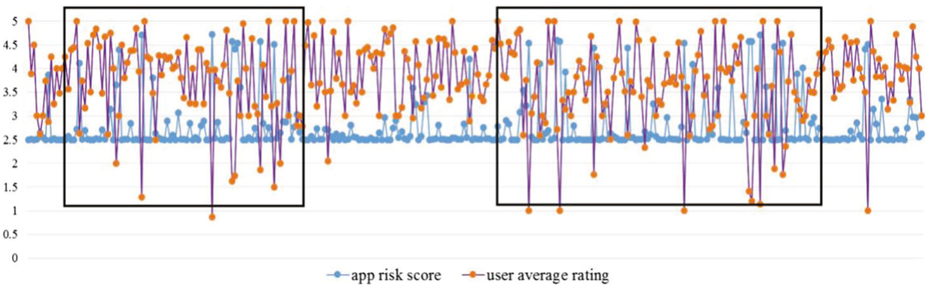

Figure 3 Risk score of app and average of user rating. The higher the app risk scores are, the *lower* the users' ratings (significantly exhibits in the *black* box)

Table 1 Description of symbols

Symbol	Denotation
U	Set of users
A	Set of apps
P	Set of permissions
M	Number of users
I	Number of apps
N	Number of permissions
R	User-app ratings matrix
$\widetilde{R}$	Predicted value of R
L	Permission matrix
Q	User's interest latent matrix
V	App functionality latent matrix
K	Dimension of latent factor space

4.2 Model construction

In recent years, collaborative filtering (CF) has been widely used in recommendation system. There are two primary approaches, the neighborhood approach [6], and latent factor models [17, 18]. Neighborhood methods focus on relationships between items or, alternatively, between users. For example, an item-item approach models the preference of a user to an item based on ratings of similar items by the same user. Latent factor models, such as matrix factorization, comprise an alternative approach by transforming both items and users to the same latent factor space. The former algorithms execute quickly, but due to the lack of knowledge learning, their accuracy is pretty low. On the contrary, due to the incorporation of user profiles and item profiles, the latter ones have a higher accuracy, but a relatively lower speed. In order to achieve high precision, our recommendation algorithm is based on matrix factorization, and then construct MFPF according to ARSM.

The predicted score of basic matrix factorization recommendation is as follows:

$$\widetilde{R_{ui}} = Q_u^T V_i, \tag{7}$$

where $Q^{M\times K}$ is latent matrix of user's interests, $V^{K\times I}$ is latent matrix of items profiles, and K is the dimension of the latent space (latent factor).

In the hypothesis that users' ratings are related to permissions, we combine users' interests, functionalities, as well as permissions, and calculate the predicted rating as follows:

$$\widetilde{R}_{ui} = \alpha Q_u^T V_i + (1-\alpha)\frac{Q_u^T \sum_{j\neq i}^{I} (s_{ij} V_j)}{\sum_{j\neq i}^{I} s_{ij}}, \tag{8}$$

where $Q^{M\times K}$ is latent matrix of users' interests, $V^{K\times I}$ is latent matrix of apps' functionalities, s_{ij} is the cosine similarity between permissions of app a_i and permissions of app a_j, namely, $s_{ij} = \cos(L_i, L_j)$, α is the permission weight control factor. In (8), the calculation of user's predicted ratings for apps contains two parts. The first part is the basic matrix factorization, by incorporating the users' interests and apps' latent functionalities. The second part is the rating prediction, which represents user u's interests on app a_j, by considering the permission similarity between app a_i and app a_j. Note that this part is on the basis of the hypothesis of the relationship between user ratings and permissions, i.e., user ratings to apps which have similar permissions are similar, too. This has been preliminarily verified in ARSM (see details in Section 3).

Figure 4 depicts the MFPF model, which is composed of two parts with accordance to (8). The first part is the combination of users' interests and apps' functionalities controlled by parameter α. This part is based on basic matrix factorization recommendation algorithm. The second part is the combination of users' interests, apps' functionalities and permissions controlled by $1 - \alpha$. The whole model is based on the hypothesis that users' ratings are related to apps' permissions, and the proportion of two parts is controlled by α. When $\alpha = 1$, the model is the basic matrix factorization recommendation algorithm, namely, ignoring apps' permissions. When $\alpha = 0$, apps' permissions are the key factor of the model. When α is in (0, 1), our model runs personalized recommendation according to users' interests, apps' functionalities and permissions simultaneously.

In order to calculate matrix $Q^{M\times K}$ and $V^{K\times I}$ in (8), we employ minimize prediction error method:

$$\min_{Q,V} \sum_{u,i} (R_{ui} - \widetilde{R_{ui}})^2. \tag{9}$$

Meanwhile, we avoid over-fitting through regularization, then obtain the optimization function as follows:

$$\min_{Q,V} \sum_{u,i} (R_{ui} - \widetilde{R_{ui}})^2 + \lambda(\|Q\|^2 + \|V\|^2). \tag{10}$$

In this paper, we exploit stochastic gradient descent [16] to calculate (10). In order to minimize prediction error $J_{ui} = R_{ui} - \widetilde{R_{ui}}$, we use (11) and (12) to update iteratively:

$$Q_u = Q_u + \beta_1 \left(\alpha V_i + (1-\alpha) \frac{\sum_{j\neq i} (s_{ij} V_j)}{\sum_{j\neq i} s_{ij}} \right) J_{ui} - \lambda Q_u, \tag{11}$$

$$V_i = V_i + \beta_2 \alpha Q_u J_{ui} - \lambda V_i, \tag{12}$$

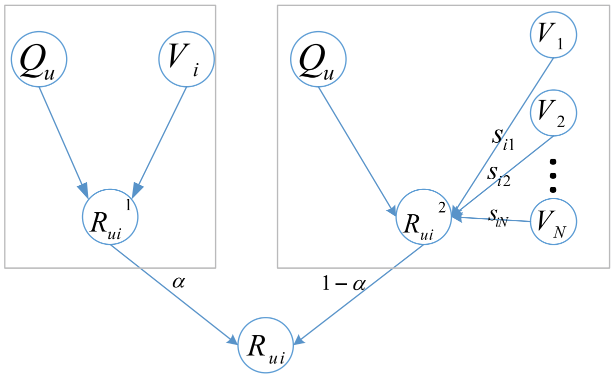

Figure 4 The model of MFPF

where β_1, β_2 are the learning rate, which need to be specified manually. When the prediction error reaches the stable value or the maximum number of iterations is satisfied, the computation ends.

5 Experimental analysis

5.1 Dataset

All the experimental data comes from the AnZhi market (http://360.anzhi.com), including 5534 apps, 2762 users, and 1075401 comments. We further exclude apps and users whose comments are less than 10. Finally we obtain 1287 apps, 975 users, and 98621 comments.

5.2 Experimental description

In this paper, the experimental section is mainly to answer the following four questions:

- What is the optimum of permission control factor?
- Are apps' permissions related to users' ratings?
- How does the latent spatial dimension (latent factor) affect the experimental results?
- Whether our recommendation algorithm MFPF is superior to other algorithms or not?

In order to solve these problems, we conduct a lot of experiments, and compare our results with some excellent recommendation algorithms. We use 5-fold cross validation, the training data set accounts for 80%, and the testing data set accounts for 20%. There are two types of inspection criteria in our experiment, the first is root mean square error (RMSE), as shown in (13), where R_{test} is testing user-app rating matrix. The second is precision, as shown in (14), where N is the number of apps which get the highest predicted rating in testing data set, $C_{N,rec}$ is the apps that top-N recommendation algorithm recommends to users, and $C_{adopted}$ is the apps which are adopted in testing data set.

$$RMSE = \sqrt{\frac{\sum_{(u,i)|R_{test}} (R_{u,i} - \widetilde{R_{u,i}})}{|R_{test}|}}, \tag{13}$$

$$Precision@N = \frac{|C_{N,rec} \cap C_{adopted}|}{N}. \tag{14}$$

5.3 Analysis of permission control factor and relationship

In the algorithm proposed in our paper, the permission weight control factor plays a pivotal role, it is necessary to cover both users and items profiles (i.e., users' interests and apps' functionalities in our recommendation algorithm) in basic recommendation algorithm, and at the same time permissions should also be taken into account. So the value of α not only determines relevance between apps' permissions and users' ratings, but also plays an important role in evaluating results of our recommendation algorithm MFPF.

Figure 5 shows different RMSE values under different α. The experimental parameters are set as follows: regularization parameter λ is 0.1, the dimension of latent space (factor of matrix factorization) K is 30, learning rate $\beta_1 = \beta_2 = 0.01$, and the control factor α is between [0,1]. From Figure 5, we can find that different α makes different RMSE values. When α=0, the RMSE reaches its maximum value, which means the worst effect. Then with

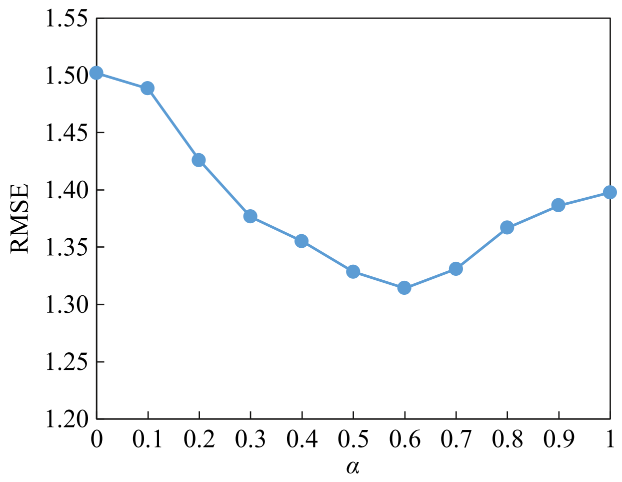

Figure 5 The RMSE of different α

the increasing of α, the value of RMSE becomes lower, and when α is about 0.6, RMSE value is the lowest, which means under this circumstance we have the best results, then with the increasing of α, the value of RMSE grows higher again. Therefore, we can conclude that although users' rating is essentially a subjective evaluation of apps' functionalities, it can reflect the security of apps' permissions. This indicates that there exists some kind of relationship between users' ratings and apps' permissions. And we notice a fact that when α is in [0.5, 1], our recommendation algorithm performs better than the performance when α is in [0, 0.5]. This means that although the aspect of apps' permissions plays an important role in our recommendation algorithm, the users' interests and apps' functionalities are more intuitive and valuable in predicting users' ratings.

Figure 6 presents the influence of different dimension K of latent space (latent factor) on the accuracy of our algorithm MFPF. The abscissa is the value of K, and the ordinate is the value of RMSE. Other experimental parameter settings are: regularization parameter λ is 0.1, learning rate $\beta_1 = \beta_2 = 0.01$, permission control factor α is 0.6. From Figure 6, we can see that when K is in [0, 50], the value of RMSE begins to decrease, when K is about 50, RMSE reaches its minimum values, and then with the increasing of K, the value of RMSE begins to increase and finally tends to converge. The main reason is that our algorithm takes permissions into account, and we can obtain apps' permissions through analyzing the code, by which the problem of cold-start could also be solved.

5.4 Experimental comparisons

In this section we will compare our algorithm MFPF with two groups of other recommendation algorithms. The first group includes some excellent classic recommendation algorithms: (1) SVD++ [5], which is an improvement of SVD. (2) BiasedMF [6], which is a matrix factorization similar to SVD, but takes into deviation data. (3) ItemKNN [19], which is a KNN algorithm based on items. The second group includes algorithms which also focus on apps' functionalities and permissions, such as Privacy_Res and Sensitive_Perm [9]. Privacy_Res and Sensitive_Perm both take apps' permissions into account. The difference between them is that Privacy_Res takes 10 types permissions into account, but Sensitive_Perm takes 23, leading to a more precise result.

5.4.1 Analysis of RMSE

Figure 7 shows different RMSE values under different dimensions K. Here note that K in ItemKNN is the number of nearest neighbors, not the dimension. The settings of experimental parameters are as follows: (1) In SVD++, regularization parameter λ is 0.1, learning rate is 0.1. (2) In BiasedMF, regularization parameter λ is 0.1, learning rate is 0.01. (3) In the experiment of ItemKNN, there is no extra setting of parameters, only the nearest neighbor number K needs to be fine-tuned. (4) In our MFPF, regularization parameter λ is 0.1, learning rate $\beta_1 = \beta_2 = 0.01$, permission control factor α is 0.6.

From Figure 7 we can find that with the increasing of K, ItemKNN tends to be stable when $K \geq 10$. The change of ItemKNN is not so obvious as other algorithms, which is mainly because the sparsity of experimental data has a great influence on the effect of ItemKNN. When experimental data reaches a certain degree, nearest neighbor number tends to be stable, so the effect also tends to be stable. Except for ItemKNN, RMSE values of other algorithms decrease when K changes from 5 to 50. However, when K turns to be around 30, RMSE value becomes stable. Meanwhile, we compare the effect of different algorithms, we find that SVD++ is better than BiasedMF and ItemKNN, because SVD++ and BiasedMF both make use of historical data and learn implicit feedback information. And we also find that our algorithm is the best among all the algorithms involved in experiments. On one hand, apps' permissions are important in app recommendation algorithm. On the other hand, although classical recommendation algorithms employ latent factor model such as matrix factorization, they cannot learn enough implicit information, because many latent feedback information is not explicitly expressed.

Figure 8 shows different RMSE values under different proportions of training data. The settings of experimental parameters are as follows: (1) In SVD++, regularization parameter λ is 0.1, learning rate is 0.1, dimensions K is 30. (2) In BiasedMF, regularization parameter λ is 0.1, learning rate is 0.01, dimensions K is 30. (3) In the experiment of ItemKNN, the number of nearest neighbor K is 30. (4) In our MFPF, regularization parameter λ is 0.1, learning rate $\beta_1 = \beta_2 = 0.01$, permission control factor α is 0.6, dimensions K is 30. From Figure 8 we can find that the change of ItemKNN is obvious with the increasing of proportion of training data, while other algorithms are relatively stable. We can also find

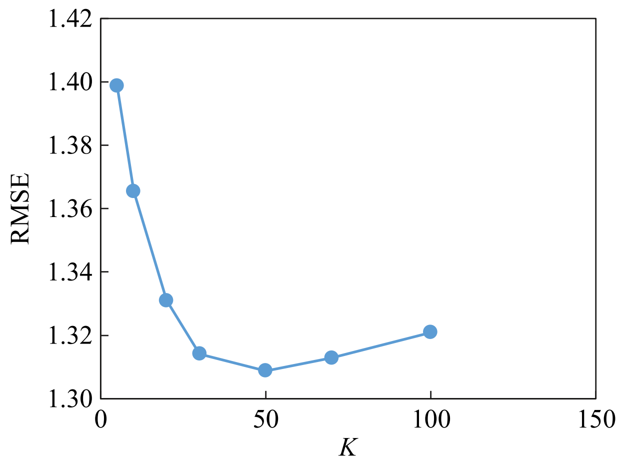

Figure 6 The RMSE of different K

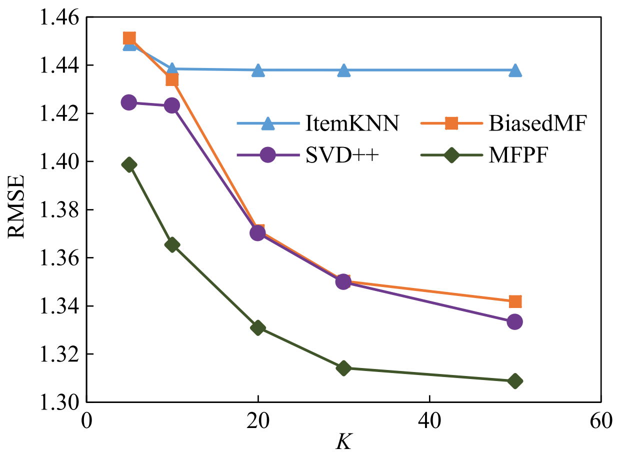

Figure 7 RMSE of different algorithms under different dimensions K

that the RMSE is satisfactory when proportion of training data is between 70 and 90%. But totally, our algorithm is better than the others, no matter what the proportion is.

5.4.2 Analysis of precision

Figure 9 is the bar graph about precision of our algorithm MFPF and Sensitive_Perm as well as Privacy_Res under different cases top-N recommendation, where latent dimension K is 30 (because when $K = 30$, the experimental results are more accurate). Other parameters of Privacy_Res and Sensitive_Perm are set as: $\alpha_U = \alpha_V = \alpha_P = 20$, $\beta_U = \beta_V = \beta_P = 0.5$, $\eta = 0.00001$, λ=1.

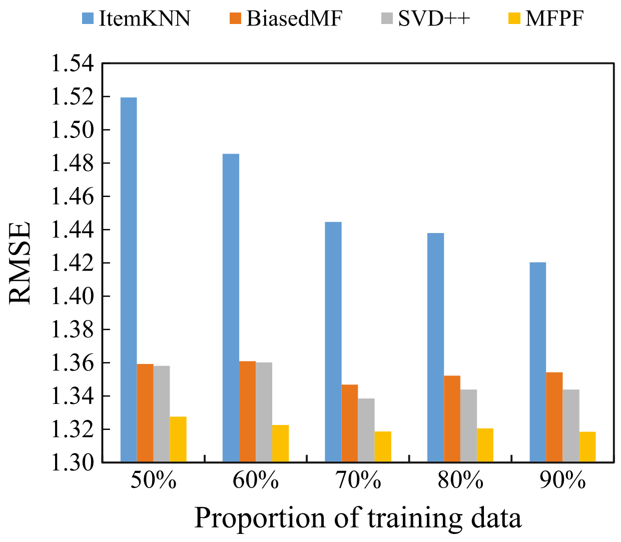

Figure 8 RMSE of different algorithms under different proportions of training data

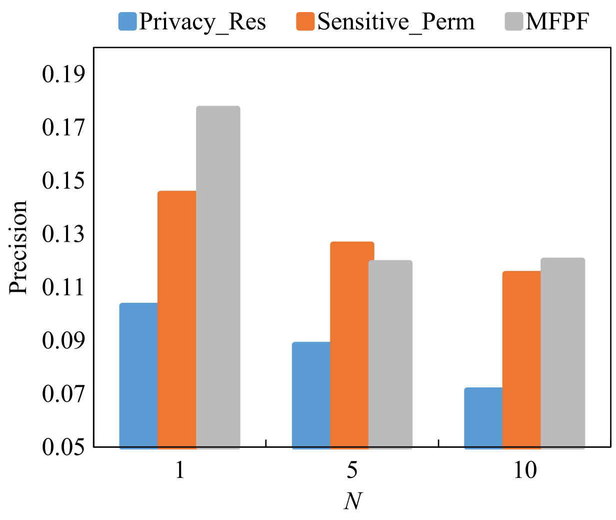

Figure 9 Precision @N

From Figure 9, we can find that our MFPF is the most accurate algorithm among all the algorithms. Since Privacy_Res takes less permissions into account, its result is unsatisfactory, which proves the importance of apps' permissions in recommendation algorithm from the opposite side. As for Sensitive_Perm, although its performance is better when $N = 30$, our algorithm is clearly better in a global perspective. This is mainly because that our algorithm takes the relationship of apps' permissions and users' interests into account.

6 Conclusion

In this paper, we analyze apps' permissions and verify our assumption that there exists a relationship between apps' permissions and users' rating (users' interest). According to this, we propose a matrix factorization based on apps' permissions and apps' functionalities called MFPF, which integrates users' interests, apps' functionalities, as well as apps' permissions, and regulate permission control factors to conduct personalized mobile app recommendation. Finally, by comparing with other state-of-the-art algorithms, we prove that our algorithm is more effective in terms of accuracy. The future work contains two parts. The first part is to further demonstrate the influence of sparsity in recommendation algorithm. The second part is to introduce social network into our algorithm, which can solve the cold-start problem and further improve the recommendation accuracy.

Acknowledgments This work is supported by the National Science Foundation of China (NSFC, No.61472291 and No.41472288).

References

1. Baeza-Yates, R., Jiang, D., Silvestri, F., Harrison, B.: Predicting the next app that you are going to use. In: Proceedings of the 8th ACM International Conference on Web Search and Data Mining, pp. 285–294. ACM (2015)

2. Chia, P.H., Yamamoto, Y., Asokan, N.: Is this app safe?: A large scale study on application permissions and risk signals. In: Proceedings of the 21st International Conference on World Wide Web, pp. 311–320. ACM (2012)
3. Guo, G., Zhang, J., Yorke-Smith, N.: Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In: AAAI, pp. 123–129 (2015)
4. Huang, J., Peng, M., Wang, H., Cao, J., Gao, W., Zhang, X.: A probabilistic method for emerging topic tracking in microblog stream. World Wide Web, pp. 1–26 (2016)
5. Koren, Y.: Factorization meets the neighborhood: A multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434. ACM (2008)
6. Koren, Y., Bell, R., Volinsky, C., et al.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
7. Li, M., Sun, X., Wang, H., Zhang, Y., Zhang, J.: Privacy-aware access control with trust management in Web service. World Wide Web **14**(4), 407–430 (2011)
8. Lin, J., Sugiyama, K., Kan, M.Y., Chua, T.S.: Addressing cold-start in app recommendation: Latent user models constructed from twitter followers. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 283–292. ACM (2013)
9. Liu, B., Kong, D., Cen, L., Gong, N.Z., Jin, H., Xiong, H.: Personalized mobile app recommendation: Reconciling app functionality and user privacy preference. In: Proceedings of the 8th ACM International Conference on Web Search and Data Mining, pp. 315–324. ACM (2015)
10. Ma, H., Yang, H., Lyu, M.R., King, I.: Sorec: Social recommendation using probabilistic matrix factorization. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 931–940. ACM (2008)
11. Ma, J., Sun, L., Wang, H., Zhang, Y., Aickelin, U.: Supervised anomaly detection in uncertain pseudoperiodic data streams. ACM Trans. Internet Technol. **16**(1), 1–20 (2016)
12. Mooney, R.J., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the 5th ACM Conference on Digital Libraries, pp. 195–204. ACM (2000)
13. Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Potharaju, R., Nita-Rotaru, C., Molloy, I.: Using probabilistic generative models for ranking risks of android apps. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 241–252. ACM (2012)
14. Peng, M., Gao, B., Zhu, J., Huang, J., Yuan, M., Li, F.: High quality information extraction and query-oriented summarization for automatic query-reply in social network. Expert Syst. Appl. **44**, 92–101 (2016)
15. Peng, M., Huang, J.J., Ghani, N., Sun, S.T., Wu, B., He, Y.X., Wen, W.D.: Micro-blogger influence analysis based on user features. J. Internet Technol. **14**(2), 307–314 (2013)
16. Saad, D.: Online algorithms and stochastic approximations. Online Learning
17. Salakhutdinov, R., Mnih, A.: Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: Proceedings of the 25th International Conference on Machine Learning, pp. 880–887. ACM (2008)
18. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: NIPS, vol. 20, pp. 1–8 (2011)
19. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285–295. ACM (2001)
20. Shi, K., Ali, K.: Getjar mobile application recommendations with very sparse datasets. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 204–212. ACM (2012)
21. Tan, C., Liu, Q., Chen, E., Xiong, H.: Prediction for mobile application usage patterns. In: Nokia MDC Workshop, vol. 12 (2012)
22. Wang, H., Cao, J., Zhang, Y.: A flexible payment scheme and its role-based access control. IEEE Trans. Knowl. Data Eng. **17**(3), 425–436 (2005)
23. Yu, K., Zhang, B., Zhu, H., Cao, H., Tian, J.: Towards personalized context-aware recommendation by mining context logs through topic models. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 431–443. Springer (2012)
24. Zhang, J., Tao, X., Wang, H.: Outlier detection from large distributed databases. World Wide Web **17**(4), 539–568 (2014)
25. Zhang, Y., Shen, Y., Wang, H., Yong, J.: On secure wireless communications for iot under eavesdropper collusion. IEEE Trans. Autom. Sci. Eng. **13**(3), 1281–1293 (2016)

26. Zhang, Y., Shen, Y., Wang, H., Zhang, Y., Jiang, X.: On secure wireless communications for service oriented computing. IEEE Transactions on Services Computing (2015)
27. Zhu, H., Chen, E., Yu, K., Cao, H., Xiong, H., Tian, J.: Mining personal context-aware preferences for mobile users. In: 2012 IEEE 12th International Conference on Data Mining, pp. 1212–1217. IEEE (2012)
28. Zhu, H., Xiong, H., Ge, Y., Chen, E.: Mobile app recommendations with security and privacy awareness. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 951–960. ACM (2014)