Personalized Mobile Application Discovery

Cheng Yang, Tao Wang, Gang Yin, Huaimin Wang, Ming Wu, Ming Xiao
Key Lab. of Parallel and Distributed Computing
Colledge of Computer, National University of Defence Technology, ChangSha, Hunan, China
delpiero710@126.com, {taowang2005, jack.nudt,hmwang}@nudt.edu.cn

ABSTRACT

With the dramatic growing of mobile application markets, users can find apps with any function they desire in these markets. However, the huge amounts of apps make it quite a challenge for users to discover good applications efficiently. Previous studies recommend applications based on the download history, user ratings or app usage records. Most of these studies fail to capture users' personal interests in mobile applications precisely.

In this paper, we leverage apps as features for describing user's personal interests and propose a novel approach to do personalized recommendation. We introduce a Small-Crowd model to distinguish apps at reflecting users' personal interests, and design a weighting method to rank the installed apps for users by combining the global download information with fine-grained app usage records. The extensive experiments validate the effectiveness of our approach which outperforms state-of-the-art method.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Performance

Keywords

Mobile App, Small-Crowd, Personalized Recommendation

1. INTRODUCTION

With the development of smart phones, the mobile app markets have experienced an explosive growth. Various app markets like Apple App Store, Google Play and so on emerge. In these markets, huge amounts of apps are available. For example, there are more than 1,000,000 apps for iOS available in Apple App Store until Oct. 2013, and also more than 1,000,000 ones for Android in Google Play until Jul. 2013 [1]. The massive amounts of apps cover almost every aspects of everyday life. On one hand, it brings great convenience to users. On the other hand, users will be overwhelmed by the quantity, and it becomes quite a challenge to find good apps efficiently.

Most of the app markets provide search function to help users find desired apps based on keywords matching. The related apps are returned and ranked based on the closeness to the query. Many studies have been conducted in which global popularity information like download times and user ratings are often used to recommend hot apps. However, such information can not reflect the user's personal interests.

Recently, researchers try to mine the app usage information to do recommendation. Bòhmer et al. [3, 2] conducted an extensive study on mobile application usage behaviours, and propose a framework for evaluating usage-centric evaluation. Bo Yan et al. [11] proposed to quantize users' personal interests and do recommendation based on their app usage records. Yin et al. [12] focused on analyzing users' view/install behaviours to recommend new apps to replace old ones that users have installed. Differently, Andrea et al. [5] employed a social approach that allows its users to be aware of what applications are installed, updated or removed in real-time around them.

It is obvious that the more time a user spending on an app, the more attractive the app is to the user. However, not all attractive apps reflect the user's personal interests equally. For example, a chat app and a swimming app may be used with similar frequency, but obviously the swimming app reflects user's personal interests better than the chat app. The previous works do recommendation based on analysing user viscosity over apps which fail to capture such difference.

In this paper, we solve this problem from a novel perspective. We view each installed app as a feature for describing the users' personal interests, and do personalized recommendations by predicting their personal interests. We propose a model named "Small-Crowd" to model apps' weights by combing the global download information with fine-grained usage behaviours. Apps which are better indicators than common apps for reflecting user interests will be assigned with higher weights. Based on the "Small-Crowd" model we design a new weighting method to quantify users' personalized interests with these apps, and employ Slope One algorithm [6] to do personalized recommendations. The main contributions made in this paper are as follows.

[1]http://en.wikipedia.org/wiki/List_of_digital_distribution_platforms_for_mobile_devices

CrowdSoft'14, November 17, 2014, Hong Kong, China
Copyright 2014 ACM 978-1-4503-3224-8/14/11...$15.00
http://dx.doi.org/10.1145/2666539.2666574

- A Small-Crowd model is proposed to model users' personal interests with apps they installed. This provides a brand new perspective for app recommendation.
- A novel weighting approach is designed to quantify user's personalized interests by apps. We learn from the TFIDF approach in Text Mining to measure how important an app is at reflecting a user's personal interests.
- Extensive experiments are conducted on large scale datasets. It proves the effectiveness of our approach, which is significantly higher than state-of-the-art method.

The remainder of this paper is organized as follows. Section 2 reviews the previous work in mobile app recommendation. Section 3 presents the Small-Crowd model and the recommendation algorithm. Section 4 illustrates the experiment dataset and the evaluation metrics used in this paper. Section 5 evaluates our approach with extensively experiments. In Section 6 we discuss the possible threats and summarize this paper in Section 7.

2. RELATED WORK

The explosive growth of mobile apps requires better mechanism for searching and locating good apps, and makes it a hot topic for app recommendation. In industry and previous researches, different types of information are employed for recommendation.

In industry, many app markets like Google Play and App Stores host huge amounts of mobile apps. These markets provide comment and rating functions to users, where users can rate the apps based on their own experience. Besides, the markets will record the download times of every app in them, which illustrate the popularity of these software. Such rating and download information reflects the global popularity of the apps, and are valuable data for recommendation. In many app markets like AppBrain, the apps are ranked according to such factors and the most prevalent apps are presented to users.

Download information and user ratings directly reflect the global popularity of a given app. However, such information only presents the global trends of an app. It can not reflect the user's personal interests in apps. Thus, such approach fails to recommend proper apps to users which just fit user's needs. Besides, the download information and the user ratings are often insufficient [7], which hinders the use of such approach for recommendation.

Instead of using the global popularity information, app usage information are explored to do recommendation. Bòhmer et al. [3] conducted an extensive study on mobile application usage behaviour. The authors explore the application usage over time and location, and their findings can be exploited for context-aware recommendation. Yan et al. [11] employ app usage records to explore users' personalized interests in different apps. They design a RFD model including Recency, Frequency and Duration to measure user's interest in a given app. Such model avoids the dependence on manual user ratings which are often insufficient, and only relies on analysing how users use these apps. Differently, Yin et al. [12] focus on exploring users' view/download behaviour sequences to recommend new apps to replace old ones users have installed. They propose an Actual-Tempting model to capture factors that may motivate a user to replace one app with a new one and employ it to recommend app recommendation. These studies emphasize on how users use apps. However, not all apps can reflect user's personal interests equally even they are used similarly. Differently, our approach distinguishes such apps as "Small-Crowd" ones which reflect personal interests from the common apps when doing recommendation.

In addition, as the prevalence of social media, rich information about apps are available in social network, which are employed to do recommendation. AppAware [5] employs a social approach which allows its users to be aware of what applications are installed, updated or removed in real-time around them. This provides a new way for users to discover new apps in a serendipitous manner. Lin et al. [7] introduce the Twitter information to overcome the cold-start problem. They leverage the nascent information about apps and their followers in their Twitter accounts to analyse users' interests and do recommendation. Different from these works, our approach only relies on the global download information and app usage records.

3. OUR APPROACH

In this section, we describe our Small-Crowd model and recommendation algorithm in detail. Firstly, we discuss the definition of Small-Crowd model and how to identify small-crowd apps quantitatively. Then we present the personalized recommendation algorithm based on Small-Crowd model.

3.1 Small-Crowd Model

3.1.1 Basic Intuition

Nowadays, mobile applications almost cover all aspects of everyday life. Users search and install apps which fit their needs in daily life or occupations. Thus, what apps a user install and use can reflect the user's occupation or personal interests. For example, if a user installs a skiing app and use it steadily, we can deduce that he is interested in skiing. Based on this intuition, we leverage the installed apps as features to describe the users and then do recommendation based on these features.

However, not all apps reflect user's personal interests equally, even they are used with similar manner. Taking a chat app as an example. Even the chat app is used more frequently than the skiing app by the user, it is not an effective indicator of the user's personal interests as the swimming app, because it is commonly used by people with different ages or hobbies.

3.1.2 Model Definition

Observing the applications installed by users, we find that the apps which reflect users' personal interests often have a specific user group. For example, the skiing app SkiApp can receive GPS signals, and allow its users to track and analyse their skiing actions. Such a app is targeted at these ski enthusiasts, and the others who are not interested in ski will not install this app. Furthermore, in the target user group, the users are often enthusiastic in the app and use it steadily. In contrary, some other apps like WebChat which provides free voice and message chat functions to contact friends also used very frequently. However, such apps often provide general functions and enjoy huge number of users. Such apps are little related to personal interests.

In this paper, we define the "Small-Crowd" model. In this model, each app is viewed as a feature for reflecting user's personal interests. Each user can be depicted by the apps he used. For a given user, different apps are of different importance at reflecting his personal interests. For a given app, it is also of different importance for reflecting different users' personal interests. If one app is used frequently and constantly, it is obvious that this app is enjoyed by the user and can reflect the user's interest.

Based on the observation discussed in the beginning of Section 3.1.2, we focus on two factors to quantify the weights of apps for users: the download rate and user viscosity. The download rate reflects the global popularity of the app, and the user viscosity presents the degree of interest for the user over the app. We borrow the idea from Term frequency-inverse document frequency (TF-IDF) in text mining to combine these two factors, and propose a novel approach to specify the weights of installed apps for users.

We view each user as an document, and the apps he installed are the items appear in the document. Then, all the users in the dataset represent the document set and all apps are the words in the whole corpora. In text mining, each document can be represented as a weighted vector. Each element in the vector is a term, and the value stands for its importance to the document. The importance is often measured by the times it appears in the document and the number of documents it appears. That is, for a given word, the more times it appears in a document, the more important it is for that document. On the contrary, the more documents it appears in, the less important it is for distinguishing these documents. Such method for measurement is TF-IDF, in which term frequency (TF) is the frequency a word appears in a document, and document frequency is the number of documents in which the word appears. IDF is the reciprocal of document frequency. TF-IDF synthesizes these two parts together to reflect the importance of a word.

Learning from the basic idea of TF-IDF, we measure the weights of apps for users, which combines download information and user viscosity together. The specific definition is as follow.

$$W_{i,j} = v_{i,j} * log(\frac{|U|}{|t : a_i \in u_j|}) \tag{1}$$

where $W_{i,j}$ means the weight of app a_i for user u_j, which is measured by multiplying user viscosity $v_{i,j}$ with the reverse app download rate. The reverse download rate is calculated at the second part where numerator $|U|$ is the number of all users, and denominator is the number of users who installed app a_i.

In Equation 1, we view user viscosity as the term frequency in TF-IDF, and view the download times as the document frequency. Based on the Small-Crowd model, the more a user enjoys an app, the more this app reflecting this user's interest. Meanwhile, The more an app was installed, the more common the app is, and thus the less importance it is for distinguishing this user.

However, the download rate is not completely similar to document frequency in text mining. According to the idea of Small-Crowd model, both apps with too little and too much download rate will not be small-crowd app. The weights for apps with too little download rate should also be reduced as these for apps with too much download rates.

To solve this problem, we adopt an "double-back" strategy. For all apps, we first count their download rates and sort them ascendingly. Then we double-back the download rate at the center of this sequence, and map these smaller download rates to the bigger ones at the opposite positions. The particular mapping function is defined as Equation 2.

$$d(i) = d(|A| - 1 - i) \quad if \quad i < \frac{(|A| - 1)}{2} \tag{2}$$

where i is the position of the app in the sorted download rate sequence, $d(i)$ is the download rate for app ranked at the i_{th} position, and $|A|$ is the total number of apps. This will map the download rates of these apps who are ranked at the lower half sequence to the upper half. After such processing, the apps with median download rate will be assigned with largest reverse user frequency, and those with least or largest download rates will be assigned with least reverse user frequency.

To measure the viscosity of a user over an app, we focus on the user's usage behaviour over the app. We employ the idea in AppJoy [11] to quantitatively analyse the user viscosity. Three metrics including Recency, Frequency and Duration are taken into consideration. Recency measures how recently a user used the app, Frequency means how frequently a user interacts with the app in a given period of time, and Duration measures how long the user interact with the app. These three metrics reflect how much a user is interested in the app from different perspectives. The combination of it can be a good predictor. Specially, we set the combination weights of the three metrics with value 0.5, 0.3 and 0.2.

3.2 Recommendation Algorithm based on Small-Crowd Model

Small-Crowd model describes users by the apps they used. If two users install similar apps and use them with similar manner, we can conclude that these two users have similar interests and do personalized recommendation based on the apps they installed.

Collaborative Filtering (CF) algorithm is a widely used algorithm in current recommender systems [8, 10, 4]. The basic idea for CF is that users with similar interests often install similar apps. Thus, recommendation can be done based on the the similar users which is classified as user-based CF [9]. Besides, recommendation can also be made based on similar apps, which is known as item-based CF [8].

In this paper, we employ an modified CF approach named Slope One algorithm to do app recommendation [11, 6, 1]. For an given user u and an app i he has not installed, Slop One algorithm predicts the probability for the user to install this app as follow.

$$P(u)_i = \overline{u} + \frac{\sum_{v \in S_{u,i}} (v_i - \overline{v})}{Card(S_{u,i})} \tag{3}$$

where $S_{u,i}$ is the set of apps each of which has been installed by user u and meanwhile has more than one users who installed both it and app i, $Card(S_{u,i})$ is the number of apps in this set. $\overline{u}$ is the average rating for user u over all apps in the set $S_{u,i}$, $\overline{v}$ represents the average ratings for all users over apps in $S_{u,i}$, and v_i is that for app i. This equation combines the user's average rating with the average deviation between other apps and the predicting app for all the users in the training dataset who used both of them.

4. EXPERIMENT SETTING

In this section, we first present the experiment dataset we used in this paper. Then we discuss the evaluation metrics.

4.1 Experiment Dataset

To prove the effectiveness of our approach, we employ the dataset used in AppJoy [11], which are collected by the AppJoy. AppJoy records the users' behaviours including the interaction time between the user and the app, the interaction count of the application, the location where the user used the app and so on. These information are recorded and uploaded to the server each hour.

Initially, AppJoy published three versions and collected data from Feb. 2010 to Mar. 2011. The whole dataset contains more than 1,329 users, 1,590,455 records. However, in the dataset there are some records with record_time as 2005, which are obviously incorrect. We delete such records from the dataset. Besides, a large proportion of applications are only installed by very few users. We delete such apps which have less than 5 users. Then, we delete users who have no records in the dataset. By this, the number of records decreases to 1,374,620, the number of users decreases to 1,325, and apps decrease from 11,532 to 1,489. The detailed information of the processed dataset is shown in Table 1.

4.2 Evaluation Metrics

To evaluate the performance of our approach, we divide the whole dataset into two parts at a specific time point: the first part is used as training data; the second part is used as the testing set. The newly installed apps at the time period of the second part will be viewed as the ground truth, and recommendation will be done to these users who installed new apps. For a given user, if the new apps he installed are contained in the recommendation list, the recommendation for him will be viewed as correct.

In this paper we use Accuracy, Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP) as the metrics for evaluating recommendation performance. For a given user, if there is atleast one app in the top-k recommendation installed by him in the testing period, then we will view the recommendation for this user as correct, otherwise incorrect. We calculate the performance over all the testing users and get the Average Accuracy for recommending K(**A@K**).

In addition, the rank of the correct answers in the top-k results is another important metric. Here we use MRR and MAP over K(**MRR@K** and **MAP@K**) to evaluate. MRR measures the average rank of the first correct answer in the recommendations over all testing users, while MAP measures the average rank of all the correct answers over all the testing users. The detailed definition of these two metrics are as follows.

$$A@K = \frac{1}{|U|}\sum_{i=1}^{|U|} result_i \tag{4}$$

$$MRR@K = \frac{1}{|U|}\sum_{i=1}^{|U|} \frac{1}{Rank_{K_i}} \tag{5}$$

$$MAP@K = \frac{1}{|U|}\sum_{i=1}^{|U|} \frac{1}{\sum_{j=1}^{C_i} Rank_{K_j}} \tag{6}$$

In the Equations, U means the set of testing users, and $result_i$ stands for the value of recommendation for user i. If the correct answer appear in the top-k results, then the value is 1, elsewhere 0. $Rank_{K_i}$ in Equation 5 represents the rank of the correct answer in the top-k recommendations. For user i, if the correct answer appear in top-k results, then $Rank_{K_i}$ is its rank. Otherwise, its rank will be viewed as infinite, which make the reciprocal as zero. In Equation 6, C_i is the number of new installed apps for user i, and $Rank_{K_j}$ is the rank of *jth* new installed app for user i.

5. RESULTS AND DISCUSSION

In this section, we firstly compare the effectiveness of our recommendation algorithm with the previous work AppJoy. Then we change the departing time point to see how long a period of usage record is efficient enough to achieve acceptable recommendation performance.

5.1 Comparison AppJoy and Small-Crowd

Both AppJoy and our Small-Crowd model employ the app usage behaviours to predict users' probable interests in new apps. The difference is that we take the global download information into consideration, and design a weighting method which combines the global download information with fine-grained usage information to calculate the weights of apps at reflecting users' interests.

In this section, we reimplement Yan's work of AppJoy and compare our Small-Crowd model with it. The whole dataset contains data from Oct. 30th, 2010 to Nov. 28th 2011. In this experiment, we set the time Aug. 1th, 2011 as the breakpoint which separates the dataset into two parts: the first part takes up about 2/3 of the whole dataset which will be used as the training data, and the second part takes up about 1/3 which will be used as the testing data. In the second part, there are 103 users who have installed new apps. We will recommend apps for these users and see how many users installed the recommended apps. The detailed results are shown in Figure 1.

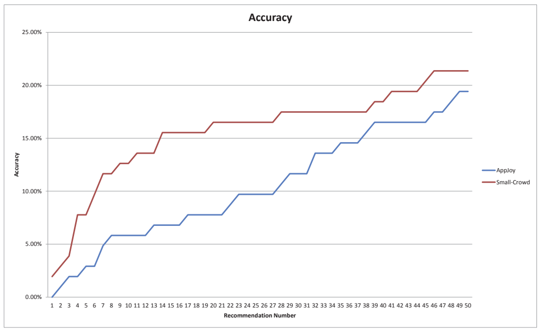
Figure 1 compares the recommendation performance of Accuracy, MRR and MAP between AppJoy and Small-Crowd. From the figure we can see that Small-Crowd outperforms AppJoy significantly for all three metrics. Taking recommending 10 candidates as an example. Our approach achieves accuracy of 12.62%, while AppJoy only achieves 5.82%. For MRR and MAP, our approach get 4.43% and 0.45%, which are significantly higher than AppJoy. Similar trends can be observed steadily with the increase of recommendation number. This suggests that Small-Crowd model captures the users' personal interests precisely.

Focusing on each single approach, we can see that the Accuracy and MRR raises with the increasing of the recommendation number, while the MAP decreases. Taking Small-Crowd as an example. When recommending 5 candidates, the accuracy is about 7.76%. While for recommending 50, the precision reaches to 21.36%. This is feasible as more candidates will increase the probability for being installed by some users.

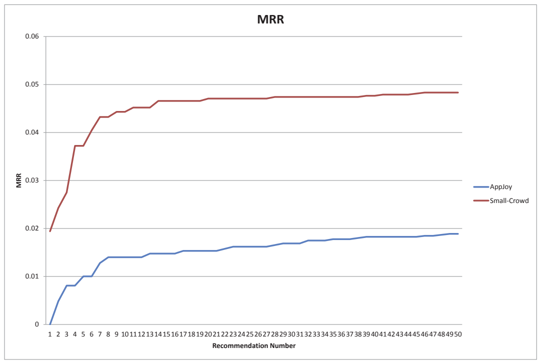
Overall, the recommendation performance seems very low to be practical. However, we need to note that such performances are achieved under the situation that no users actually viewed the recommendation list. In previous work like Appjoy, the authors have developed and deployed the AppJoy. The recommendation results are presented to users,

Table 1: Statistics of Experiment Dataset

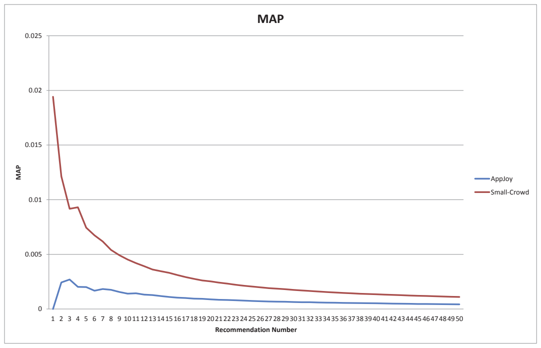
	# Applications	#Users	#Usage records	Time period
Before processing	11,532	1,329	1,590,455	2010-12-30 ∼ 2011-11-28
After processing	1,489	1,325	1,374,620	2010-12-30 ∼ 2011-11-28



(a) Accuracy



(b) MRR



(c) MAP

Figure 1: Comparison of recommendation performance between AppJoy and Small-Crowd

and the users will make their choices after viewing the recommendations. However, in this paper, we use the collected data by AppJoy instead of developing our own app to validate our approach. It just looks like that users are making their choices based on their own searching or other ways instead of viewing our recommendations. This will greatly affect the final evaluation results. So, although the overall recommendation accuracy seems very low in this paper, the performance in practice should be much higher if we present the recommendations to users and ask them to choose.

5.2 Influence of Training Scale

Using the user behaviours on apps to do recommendation, the number of usage records used for training will have great influence over the performance. In the first experiment, we divide the dataset at the time point of about 2/3, for which the whole training data takes up about 2/3 of the whole dataset, and testing dataset takes up the last 1/3 part. In this section, we fix the testing dataset as the last 1/3 part, and vary the scale of training dataset to analyse such influence.

We divide the whole training dataset into 3 equal periods of time, and then training the Small-Crowd model based on the last 1/3, 2/3 parts of the training dataset and the whole respectively. Then we recommend for users in the testing dataset. Table 2 and 3 present the detailed results for recommending ten and twenty candidates respectively.

Table 2: Recommendation performance of Top-10 with different training data scale

Training Set	Accuracy	MRR	MAP
1/3	6.67%	1.07%	0.11%
2/3	6.73%	2.90%	0.30%
3/3	**12.62%**	**4.43%**	**0.45%**

Table 3: Recommendation performance of Top-20 with different training data scale

Training Set	Accuracy	MRR	MAP
1/3	10.48%	1.31%	0.07%
2/3	11.54%	3.25%	0.17%
3/3	**16.51%**	**4.71%**	**0.25%**

From Table 2 and 3 we can see the growing trend with the increase of training scale in both tables for recommending ten and twenty candidates. For example, the accuracy will increase from 6.67% to 12.62% for recommending top-10 when the training data increases from 1/3 to 3/3. This is because the more training dataset, the more usage records will be included, and thus the more precise for reflecting the users' personal interests.

6. THREATS TO VALIDITY

There are some threats that may affect the validation of our approach. Firstly, the experiments are carried out based on the dataset which only contains about 1,000 apps and 1,000 users. The download information and usage records may be not consistent with the practical trends in the app markets. Further experiments will be conducted on other dataset.

In this paper we evaluate our approach based on the dataset used in AppJoy. The tested users choose and install apps without viewing our recommendations. Thus, the performance results presented in this paper may not reflect the practical performance of our approach precisely. We will publish our own app to get practical feedbacks and do further evaluation in the future.

7. CONCLUSION

The rapid growth of mobile apps exhibits great challenge for users to discover suitable apps. In this paper, we propose a Small-Crowd model to use apps as features for describing users' personal interests. Based on the idea of Small-Crowd we design a weighting method. Such method combines the global download information with fine-grained usage records, which distinguishes those apps which are better indicators of users' interests with higher weights. We conduct extensive experiments on more than 1,000 users and apps. The results proves the validity of our approach which outperforms state-of-the-art methods significantly.

Currently, we do experiments based on the dataset used in AppJoy. In the future work, we will implement an app of our approach and publish it to collect our own data to do validation more precisely. Besides, because of the whole dataset in the experiment is not large enough, the download information for apps are not well-distributed.ãĂĂThus, the "double-back" function used in weighting approach may introduce some deviations. We will revise this function to improve the performance further.

8. ACKNOWLEDGMENT

This research is supported by the National High Technology Research and Development Program of China under Grant No. 2012AA011201, and the National Natural Science Foundation of China (NSFC) under grant No. 61432020 and 61472430.

9. REFERENCES

[1] A. Basu, H. Kikuchi, and J. Vaidya. Privacy-preserving weighted slope one predictor for item-based collaborative filtering. In *Proceedings of the international workshop on Trust and Privacy in Distributed Information Processing (workshop at the IFIPTM 2011), Copenhagen, Denmark*, 2011.

[2] M. Böhmer, L. Ganev, and A. Krüger. Appfunnel: A framework for usage-centric evaluation of recommender systems that suggest mobile applications. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 267–276. ACM, 2013.

[3] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer. Falling asleep with angry birds, facebook and kindle: a large scale study on mobile application usage. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 47–56. ACM, 2011.

[4] K. Chen, Z. Peng, and W. Ke. Study on collaborative filtering recommendation algorithm based on web user clustering. *Int. J. Wire. Mob. Comput.*, 5(4):401–408, Jan. 2012.

[5] A. Girardello and F. Michahelles. Appaware: Which mobile applications are hot? In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 431–434. ACM, 2010.

[6] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SDM*, volume 5, pages 1–5. SIAM, 2005.

[7] J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua. 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*, pages 283–292. ACM, 2013.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[9] Y. Shi, M. Larson, and A. Hanjalic. Exploiting user similarity based on rated-item pools for improved user-based collaborative filtering. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 125–132, New York, NY, USA, 2009. ACM.

[10] P. Su and H. Ye. An item based collaborative filtering recommendation algorithm using rough set prediction. In *Proceedings of the 2009 International Joint Conference on Artificial Intelligence*, JCAI '09, pages 308–311, Washington, DC, USA, 2009. IEEE Computer Society.

[11] B. Yan and G. Chen. Appjoy: personalized mobile application discovery. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 113–126. ACM, 2011.

[12] P. Yin, P. Luo, W.-C. Lee, and M. Wang. App recommendation: a contest between satisfaction and temptation. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 395–404. ACM, 2013.