Using Mobile Services Based on SNS to Recommend Who, How and When to Collaborate

Yanchun Sun, Xiwei Zhuang, Kui Wei, Xudong Shan, Tianyuan Jiang
Institute of Software, School of Electronics Engineering & Computer Science, Peking University,
Key laboratory of High Confidence Software Technologies, Ministry of Education,
Beijing 100871, P.R.China
Email: sunyc@pku.edu.cn, {zhuangxw12, weikui13}@sei.pku.edu.cn, {shanxd, jtyuan}@pku.edu.cn

***Abstract*—The rising popularity of smart phones and social networking services (SNS) is changing many aspects of people's collaboration. With the wide use of smart phones, collaborative work based on mobile web becomes loose and flexible. Collaborators have more chances to collaborate with other people anywhere at anytime. But most supporting tools for traditional computer supported cooperative work just support defined collaborative process for certain collaborators. They can't satisfy the new requirements for loose collaboration where collaborators, collaborative process and time are unknown.**

In this paper, we present an approach to using three mobile services based on SNS and mobile sensor data to recommend who, how and when to collaborate. This collaborative approach based on mobile services solves three basic key problems of modern collaboration. Firstly, we collect abundant data from SNS, do the semantic analysis, and dig out the suitable collaborators. Secondly, by analyzing the data from calendars and smart phones, we figure out the situations which collaborators are in, then reason the suitable contacts by our novel rules and finally recommend whether we can call or not, as well as the ranked text contacts. Thirdly, we use the calendar information to recommend the common free time for collaborators to work together. To verify the effectiveness of the approach and accuracy of collaborative recommendations, we have implemented an app including the services on android platform and designed two independent experiments. The case studies show the approach provides an effective and accurate means for collaborative recommendations.

***Keywords*—mobile services, recommendation, social networking service, mobile applications**

I. INTRODUCTION

The popularity of smart phones is changing many aspects of people's collaboration. In the past, collaborative work is mostly accomplished under certain process by certain collaborators. Not only collaborative tasks but also collaborators and time are strictly defined. However, with the wide use of smart phones, the collaborative work based on mobile web becomes loose and flexible. People have more chances to collaborate with other people anywhere at anytime. The traditional computer supported cooperative work supporting tools can't satisfy the new requirements for loose collaboration where collaborators, collaborative process and time are unknown.

Although the collaborative work based on mobile web becomes loose and flexible, several basic key problems are common. First, it is to find collaborators. As modern collaboration becomes user-centered, any mobile user may ask for collaborations without preparation. Usually he or she doesn't know who will be the most suitable collaborators. So the first problem is to find whom to collaborate with.

Assuming we have found the collaborators, the second problem is how to contact them. Not only calls and SMS, but also SNS like Wechat and Skype are common contacts in smart phones. Various SNS bring users a big problem, that is, which contact is the most appropriate way to communicate with specific collaborator currently.

At last, in most cases we should figure out the common free time as potential synchronous collaboration time for all the collaborators. So the third problem is to find when to collaborate as soon as possible for the mobile users.

Who, how and when to collaborate are three basic collaborative problems. If they can be solved well, the loose mobile collaborations will be supported well.

As mentioned, SNS is a platform to build social networks or social relations among people who share interests, activities, backgrounds or real-life connections. By using their mobile phones and SNS, mobile phone users can create their own profiles, make friends, share photos and videos, and share blogs. By September 30, 2014, Renren had approximately 219 million active users with over 80% of user time accessing services through mobile devices [1]. SNS users create lots of information every day, which provides us enough data to infer the information related to people, such as what they do, or what they are good at etc.

In this paper, we put forward a mobile collaborative approach using three mobile services to recommend who, how and when to collaborate, based on the former work [4].

The approach has the following key contributions:

- It provides an effective algorithm for the collaborator recommendation.
- It's the first time to propose and solve the collaborative contact problem.
- It uses data from SNS, calendars and smart phones. It's convenient to use and mobile users don't need to do extra work.

The three mobile services on who, how and when to collaborate, can not only be used together to improve the collaboration efficiency among mobile users, but also be

978-1-4673-7284-8/15 $31.00 © 2015 IEEE
DOI 10.1109/MS.2015.13

used independently. We have implemented an app including the services on android platform.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 describes the collaborative approach. Section 4 demonstrates the implementation of the approach. Section 5 introduces the case studies. Section 6 presents concluding remarks and future work.

II. RELATED WORK

Recommender systems have been studied widely due to the incredible increasing information in the world, especially on the Web. These systems apply knowledge discovery techniques to make personalized recommendations that help people to sift through huge amount of available data [5, 6, 13].

Social Recommender systems are often used to recommend the suitable persons for users based on their preferences and activities, and they have been widely used to assist users in finding relative collaborators in various fields, such as movies (Netflix), SNS (Facebook) etc. Researchers have made a lot of research on the methods or tools of social recommendation systems [2, 5, 7, 8, 12].

Elnaz 2012 et al. have presented an expert recommendation method that integrates content-based recommendation algorithms into a social network-based collaborative filtering [7]. Xu et al. have proposed a unified framework to extend the traditional CF (Collaborative Filtering) algorithms by utilizing the subgroups information for improving their top-N recommendation performance [5]. Liu et al. propose SoCo, which systematically combines contextual information with social network information to improve the quality of recommendations [8]. However, these methods above don't concern the mobile sensor data that is very important to predict the user's situation.

The most straightforward method for keyword extraction is using TFIDF [14] to rank candidate keywords and select the top-M as keywords. After PageRank becomes popular, graph-based ranking methods like TextRank [15], are becoming the state-of-the-art methods for keyword extraction. Both TFIDF and TextRank couldn't solve the problem of vocabulary gap. To solve the vocabulary gap, a potential method that is called topic model has been proposed and the latent Dirichlet allocation (LDA)[11] is the most popular topic model. Based on these methods, we propose an efficient and effective method for collaborator recommender service, which is mainly based on TFIDF to extract keywords from user data and calculate the relevance between keywords and users.

There is also some related work on collaborative contact recommenders. Min et al. have implemented a contact recommendation application on mobile device that recommends phone numbers in a phonebook according to the user's situation based on logs stored in mobile devices [9]. But it recommends people suitable to contact, rather than suitable ways to contact specific friends. Gomes et al. design the Mobile Activity Recognition System (MARS) where the classifier is built on-board mobile device through ubiquitous data stream mining in an incremental manner [10]. However, MARS doesn't use calendar information, which is effective to find whether people are having meetings or doing other things that should not be disturbed. Yu et al. propose to mine common context-aware preferences from many users' context logs and represent each user's personal context-aware preferences as a distribution of the mined common context-aware preferences [3].

Analyzing the related work above, they only concern one issue of collaborations, and seldom consider mobile data. Considering who, how and when to collaborate as three key issues for collaboration, we put forward an approach which uses data from SNSs, calendars and smart phones to solve the key issues on collaboration together.

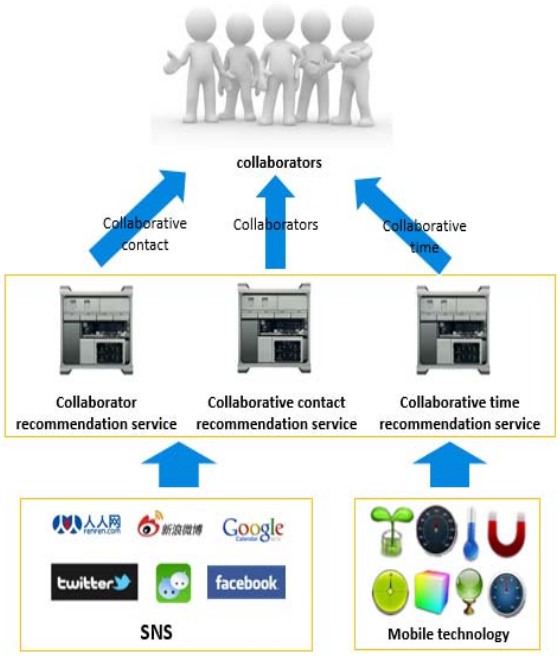

Fig. 1. A collaborative approach based on mobile services

III. MOBILE COLLABORATIVE APPROACH

Our mobile collaborative approach is described as figure 1.

The approach consists of three mobile services, including collaborator recommendation service, collaborative contact recommendation service and collaborative time recommendation service. The collaborator recommendation service collects and deals with abundant data from SNS. It can recommend suitable collaborators according to queries, which solves the problem of whom to collaborate with.

Collaborative contact recommendation service analyzes the data from calendars and smart phones, figures out the situations which collaborators are in, reasons the suitable contacts by some novel rules and finally recommends the ranked suitable text contacts and whether we can call. It aims to recommend how to collaborate. The potential collaborators can be grouped and collaborative time recommendation service utilizes the data from bound calendars and recommends common free time, which solves the problem of when to collaborate.

A. Collaborator Recommendation service

Collaborator recommender pretreats data from SNS and recommends the collaborators according to input queries. It helps to find the collaborators satisfying specific requirements. In figure 2, we show the process.

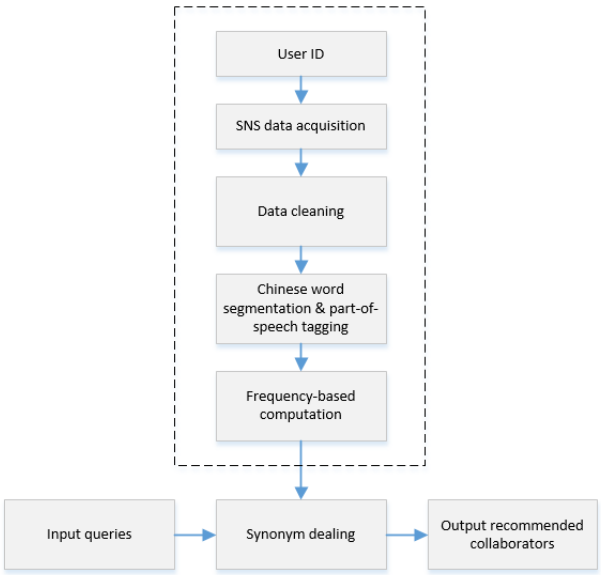

Fig. 2. Collaborator recommender process

As mentioned, renren is one of the most popular SNS in China. Statuses are short messages limited in 240 words and blogs are usually long articles. Both of them are the records of users' daily life, which can reflect the interests and expertise of users. Besides, statuses and blogs are the most popular services, which have relatively large contents for data mining. We can get blogs and statuses posted by user's friends via API provided by Renren Platform in order to dig out the interests and expertise of users. Firstly we clean the useless data from statuses and blogs, such like emotions icons and mentioned users. Secondly we choose "jieba" Chinese words segmentation tools to segment the statuses and blogs, and leave nominal words and English words to analyze.

Next we introduce the two key steps in detail.

1) Frequency-based computation

Term frequency and inverse document-frequency (TFIDF) is a numerical statistic that is intended to reflect how important a word is to a document. But it's designed for a single formal document such as a news article or a scientific paper. However, a Renren user will post hundreds of statuses and blogs. Our collaborator recommender should rank all the users based on all the statuses and blogs according to the queries. That requires us to figure out a numerical statistic of the user with lots of documents rather than the TFIDF of every single status and blog. We bring in status frequency and our improved algorithms with statuses and blogs are different from each other. We will introduce them separately.

a) Statuses

A user has hundreds of statuses that are limited in 240 words each. A status is too short to be considered as a document so that we can't calculate the TFIDF for every status. It's a straightforward solution to consider all the statuses of one user as one document. However it will lose some information. We adopt status frequency (SF) to solve the problem.

Given a user u and his/her status $s \in d_u$, the term frequency TF(q) of the candidate query q is the number of the occurrences of q in du. We get IDF(q) via dividing the total number of documents by the number of documents containing the term q. Status frequency (SF) is measured as follows:

$$\mathrm{SF(q)} = \frac{|\{s: q \in s\}|}{|d_u|}$$

Where $|\{s: q \in s\}|$ is the number of statuses containing q, and |du| is the total number of statuses for user u.

Based on TF(q), IDF(q) and SF(q), we define the value VS(q) as follows:

$$VS(q) = TF(q) * IDF(q) * \log_2(SF(\mathrm{q}) + 1)$$

TF(q) weighs how important the query q is in the document. IDF(q) is a measure on how much information the query q provides, that is, whether the term is common or rare across all documents. SF(q) considers the amount of statuses, that is, the habit of users, because if we only use TFIDF, users who post more will have a higher score, which is not fair.

b) Blogs

Blogs are usually much longer than statuses, so they contain enough information for TFIDF calculation. We calculate TFIDF for every blog of users.

Given a user u and his/her blogs $b \in d_u$, the term frequency $TF_b(q)$ of the candidate query q is the number of the occurrences of q in blogs b. We get $IDF_b(q)$ via dividing the total number of blogs by the number of blogs containing the term q. We define the value VB(q) as follows:

$$\mathrm{VB(q)} = \frac{\sum_{b \in d_u} (TF_b(q) * IDF_b(q))}{|d_u|}$$

Where $|d_u|$ is the total number of blogs for user u. The reason why it's divided by $|d_u|$ is to remove the effects of habits of users. Unless, users who like to share interests in blogs will have a higher score than those who have the same interests.

The final value of the user according to query q is defined as follows:

$$\mathrm{V(q)} = \mathrm{VB(q)} + \alpha * \mathrm{VS(q)}$$

Constant α is used to weigh the importance of blogs and statuses. In our experiments, we set $\alpha = 0.4$ which achieves the best performance.

2) Synonym dealing

Our frequency-based computation is based on an assumption that the query words are all exactly in the statuses and blogs. However, in many conditions there is a gap between the queries and real needs. For example, if a user wants to find someone who is interested in programs, probably he/she will search by the query "program". In fact, people interested in "code" also satisfy the requirements. We use synonym set to overcome the gap.

In our real system, we utilize a synonym set based on HowNet lexicon [16], which consists of about 60 thousands of Chinese words. Every word in the synonym set has a collection of similar words with similarity degree. Given a query q, the synonym value of p is defined as follows:

$$VS(q) = \sum_{S(q,w(q))>k} V(w(q)) * S(q, w(q))$$

Where w(q) is the similar word of q and s(q, w(q)) is the similarity degree between q and w(q), which is a value between 0 and 1. k is the min threshold value.

Then we define the final value of q as follows.

$$VF(q) = V(q) + \beta * VS(q)$$

Constant β is to deduce the bias of synonym. Since users use q to search, q should be the most important query word.

If the query consists more than one word, we add all the VF(q). When q is inputted by a user, we calculate VF(q) of all his friends. We rank the friends by the value VF(q) and return the most relevant friends as potential collaborators.

B. Collaborative Contact Recommendation Service

We have proposed contact recommender in [4]. So here we just give a general introduction of the approach to make the paper self-complete.

Figure 3 shows the whole process of collaborative contact recommendation service. First of all, users should register in the service, then they can bind SNS and calendar accounts on the service, and add friends through their phone number, so that the data can be calculated and people can get the recommended contacts of friends. The service can be useful only when both sides use it.

Besides the calendar data, the service collects lots of information from sensors in smart phones, such as wifi signal, carrier signal, time when phones are last used, alarm clock information in the phone, microphone using condition, recently answered calls, recently rejected calls, running states of SNS, location, speed, accelerated speed and so on. The process consists of two aspects, call recommendation and text contact recommendation.

1) Call recommendation

Calls are the most direct way to reach people, but at the same time calls disturb people most in some specific conditions. It disturbs in two ways. Firstly, calls always come with long time rings and vibrations that will disturb people. Secondly, call is a synchronous communication way that needs people to answer immediately. As a result, people should stop their work in hand.

When calls should not be recommended, conditions can be divided into three categories. We use two ways to figure out the conditions. The first way is that, through activity recognition we recognize the specific conditions of the user we want to contact. Based on the recognized conditions we recommend whether calls are suitable. We introduce some conditions in details as follows.

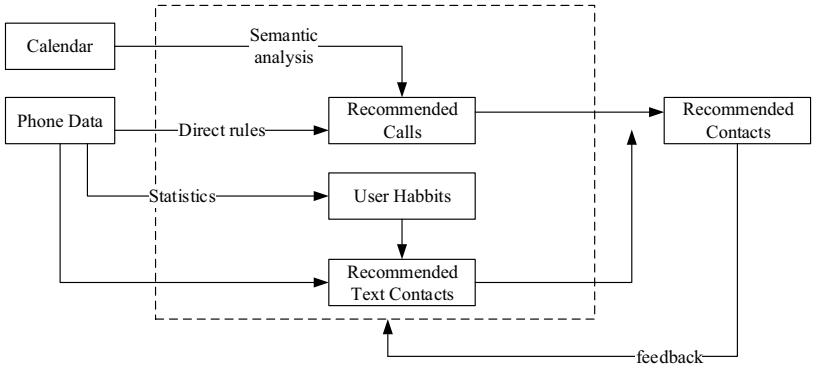

Fig. 3. Collaborative contact recommendation service process

a) **Meeting.** The server gets the bound calendar data by google calendar API. Through semantic analysis of current event, we will know whether there are any meetings right now. The same applies to having classes. The recommender recommends no calls if the users we want to contact are in periodic activities or meetings.

b) **Driving.** It's relatively complex to figure out driving because we should consider several elements. First, the speed is relatively high, like 100 km/h. Second, the current location should be a road rather than a room. Here comes the third factor. If phones have not been used for a long time, like 30 minutes, probably people are driving and should not be recommended to call.

c) **Sleeping.** We find that alarms have large potential relationships with sleep. If there is a waking up alarm, calls should not be recommended in a short period of time before it alarms, like 20 minutes. We also find the periodic alarms in the morning or at noon will enlarge the possibility of being a waking up alarm. Besides, if the locations of phones don't change, phones are not used for a long time, the locations are at home, and the time is night, there is a large possibility that people are sleeping, or people don't take the phone by themselves.

The second way is that we set some novel general rules to reason the call recommendation. We take several rules for example.

a) **Carrier signals.** If there are no carrier signals, calls can't be made successfully. The recommender will recommend not to call people.

b) **Temporal locality.** Generally the situations are continuous because doing anything will take a period of time. If there are any rejected calls recently, probably it's better not to call. Vice versa. Though we don't know what specific situations people are in, we can still recommend the right result.

c) **Microphone.** If the microphone is being used, it implies that people are available to listen to the sound information. If calls are not being made currently, we can directly recommend calls.

d) **Sound.** If the sound sensor finds it's a noisy environment, the calls quality will be affected. We need not figure out whether it's in a KTV or a market. We intend not to recommend calls.

When different conditions are satisfied at the same time, we have priorities. The general rules, such as temporal locality and microphone have the highest priority.

2) Text Contact recommendation

In the past, SMS is the most usual way to send text among mobile phones. However, with the development of the mobile data, people use SNS to send text messages much more often than before because SNS are less expensive ways, and also very convenient.

We choose the most popular SNS that are used as text communication platforms, such as Wechat and QQ. The recommender collects SNS and SMS data, and calculates the using frequency of them, which indicates people's habits of using text messages. It ranks the text message contacts based on two elements. 1) The last time the SNS was used on desk. 2) The frequency of using the SNS. We don't consider the last time the SNS run, because in smart phones if you don't stop the program, the program will not stop by itself.

Now we introduce how to rank the contacts list. We take QQ and Wechat as an example. Assume Tw and Tq are respectively the last used time of Wechat and QQ on desk. Fw and Fq are respectively the use frequency of Wechat and QQ. And we set a constant k, which means a time interval. The current time is t. The Order rule:

if $t > T_w > T_q > t-k$ or $t > T_w > t-k > T_q$ then
    Wechat is ranked before QQ
else if $t > T_q > T_w > t-k$ or $t > T_q > t-k > T_w$ then
    QQ is ranked before Wechat
else if $T_w < t-k$ and $T_q < t-k$ then
        if $F_w > F_q$ then
            Wechat is ranked before QQ
        Else QQ is ranked before Wechat

Besides, if there is a wifi but no carrier signals, SMS will not be recommended even people use it very often because SMS can't be sent to friends' phones without carrier signals. Finally, the recommender provides a list of ranked text message contacts.

To make the recommendation algorithm smarter when more people use the recommender, people are encouraged to return feedbacks, which contain the real best contact used in every communication.

C. Collaborative Time Recommendation Service

After users get the recommended collaborators and contact, the user as an organizer can organize all the collaborators for one specific task into one group. Collaborative time recommender aims to recommend a suitable time for all the collaborators in one group.

The organizer can set some requirements for time. For example, the time should be at night and the length of the time should be 2 hours. Then the recommender will get calendar information of all the collaborators through API provided by google and calculate the common free time satisfying the time requirements. If there is no satisfied time, the recommender will recommend the time when most collaborators are free. After the organizer chooses one from the list of free time, the recommender will inform all the collaborators in the group about the chosen time. If

collaborators accept the time, the new event will be added to the google calendar of collaborators.

Collaborative time recommender reduces the costs of finding a common time and helps the arrangement of schedule.

IV. IMPLEMENTATION OF THE APPROACH

We implement the approach as an application including three mobile services on Android smart phones. A user should register an account with a mobile phone number and an Email address. The application will get contacts of the phone, and the user can add friends through the contacts. Besides, the user can bind his SNS accounts, such as google calendar account and Renren account to offer more information. The application will collect all the related data, preprocess and send intermediate result to the server in time. The server will get the information from bound SNS accounts, calculate and analyze the data both from mobile phones and SNS.

When users want to find some specific people to collaborate with, they just type the query in the search box. The application will return a list of friends with the related statuses and blogs. Users can check all the information and choose most suitable collaborators. If it's a complex task that needs more than one collaborator, users can add the collaborators in one group.

To make sure the collaborations, users have many situations to contact friends. Users can get the real time recommendation of friends through the server so that they can contact friends in the best way. What users need to do is only to click the friend icon in the list and it will show whether the friend is available to be called and list the ranked text contact based on the user habits.

When users have a group of collaborators, users usually need a common free time for synchronous collaboration. Users enter the group and choose the time limitations. For example, they can choose the length of the time and the preference for morning or afternoon. The application will get schedules from the bound calendars and calculate the satisfied time. Users can choose the best one from the list and the application will inform the rest collaborators in the group. Collaborators can confirm the collaborative time and the time will be allocated in the calendar.

V. EXPERIMENTS

In this section, to verify the effectiveness of the approach and accuracy for collaborative recommendations, we design two independent experiments. The first experiment is to verify whether we can dig out the suitable collaborators when the users search specific topics related to the suitable collaborators. The second experiment is to evaluate the service accuracy of how to contact.

A. Experiment results of collaborator recommendation service

The maximum number of test users for Renren Open Platform is ten, so we select 10 students from PKU, consisting of 5 undergraduate students and 5 graduate students, to use the collaborator recommender service. All of them have never heard about the application and we install the application in their own mobile phones and they will feedback whether the recommended collaborators are related to the specific topics or not.

Before the students use the application, we need them to bind their SNS accounts on the application and we will get their friends list, the blogs and statuses of their friends for collaborator recommendation.

Figure 4 shows the number of the friends, friends' blogs and friends' statuses for each student. To make the figure clear, we just show the data of 3 typical users whose id are 15, 17, 18. We can see that every student has about 500 friends and the number of statuses is much more than the number of blogs. What's more, for every student, the data needing to be processed is large.

After binding the SNS accounts, all the students will use the application when they want to find the collaborators.

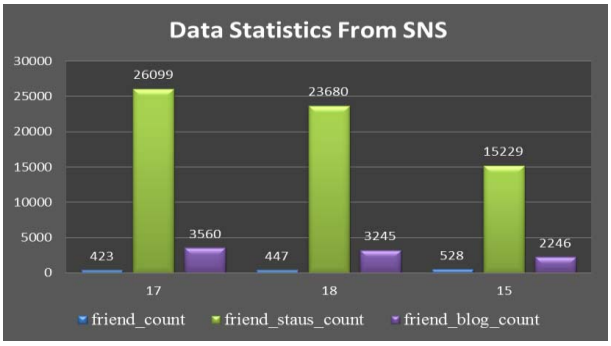

Fig. 4. Statistics data for each user

The application will recommend a sorted list of 20 relevant collaborators after a student searches one topic related to the collaborators. The student can click the recommended collaborator, read the related statuses and blogs that are the reasons why this friend is recommended, judge whether the recommended collaborators are suitable for collaboration and return the feedback. After the experiment, every student has searched about 80 topics to get recommended collaborators and we got 713 feedbacks.

Because there are many different collaborative situations in which users will need to find different numbers of collaborators. For Example, when users have a question about Python programming, they just need to find one suitable collaborator while they need about 5 suitable collaborators when they want to develop an android application. What's more, they often need to find about 10 suitable collaborators to create a small discussion group and

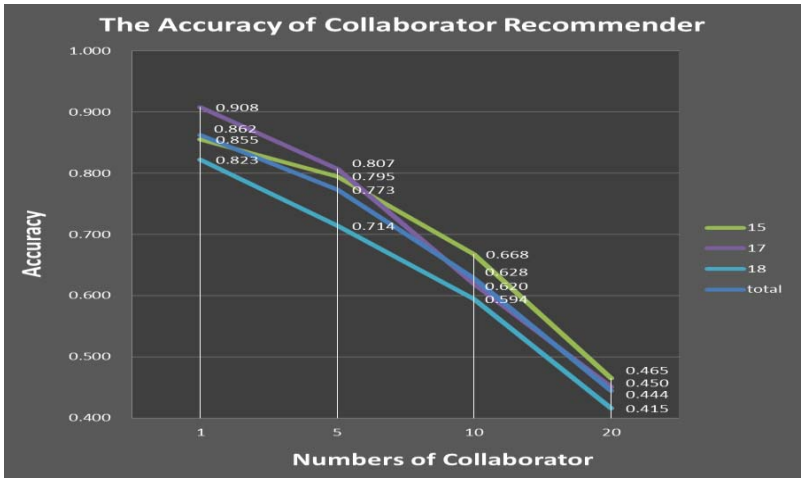

Fig. 5. Accuracy for different number of collaborators

about 20 suitable collaborators to create a big discussion group. These are the main situations when the application will be used. So we need to calculate the accuracy for the different situations.

Figure 5 shows the experiment results. To make the figure clear, we just show the feedback accuracy of 3 typical users whose id are 15, 17, 18 and the total feedback accuracy in different situations. The results show that the accuracy for one suitable collaborator recommendation is relatively high, which is 86.2%. With more collaborators users want to find, the accuracy decreases. The worst average accuracy is 44.4% when users want to find 20 suitable collaborators. That's because the recommended collaborator list is sorted by relevance and the later recommended collaborators are less relative to the specific topic, compared with the top ones.

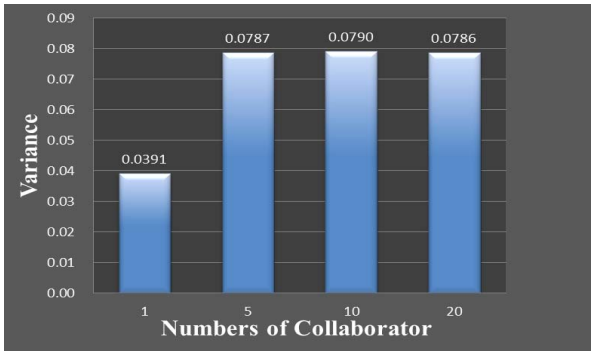

Fig. 6. Variance of accuracy

Figure 6 shows the variance of accuracy. It implies that the accuracy is stable because variances of all the accuracy are small. Besides, variance of one collaborator is smallest which means the first recommended collaborator is most stable to be the suitable one.

The result implies that the application is effective when users need to find a small group of collaborators who are related to the specific topics. It has the limitation when users want to find more than 10 collaborators because there are not enough friends on the SNS who are relative to the specific topics.

B. Experiment results of collaborative contact recommendation service

We find 50 students from PKU, consisting of 21 undergraduate students and 29 graduate students, to use the collaborative contact recommender service. All of them have never heard about the application, and we install the application in their own mobile phones and train them for 10 minutes. They can see which recommendation is made to their friends from the application. All the students do what they will as usual. They are asked to take the phones with them and check the application as often as possible and return the feedback that contains their genuine best contacts. If the recommendation is the same as the feedback, we define it is right. That means users can feed back whether the call recommendation and text contact recommendation are right. We got 1021 pieces of feedbacks totally.

TABLE 1. CONTACT RECOMMENDATION RESULTS

Cases	Ratio
Calls recommended, right	97.1%
Calls not recommended, right	82.4%
Total calls, right	92.2%
Ranked text messages contacts ,right	98.0%

Table 1 shows the experiment results. The right ratio is relatively high. For example, when the application recommends calls, 97.1% people are suitable to be called. However, when the application doesn't recommend calls, 17.6% people can be called in fact. One reason is that we use strict principles. For example, though a man rejected a call minutes ago, the situation may change right now, but we still don't recommend calling. 98.0% of the ranked text message contacts are right, probably although many SNS may be installed on the phones, people use one much more than the others.

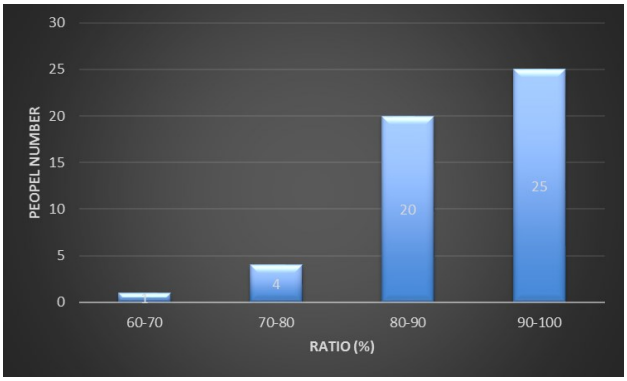

Fig. 7. People distribution in ratio intervals

Figure 7 shows people distribution in ratio intervals. We calculate people number in different ratio intervals. Most people have the right recommendation ratio between 80% and 100%. It implies that the differences of recommendation ratio among people are not that obvious. That is to say, the application is suitable for all kinds of people to use. The reason may be that conditions are basically similar when people are available to answer calls.

VI. CONCLUSION AND FUTURE WORK

We design and implement a collaborative approach using three mobile services based on SNS, calendar data, and sensor data from smart phones. The services can help users to find the most suitable collaborators according to specific topics, the most appropriate way to contact collaborators, which makes the information accessible to friends and gets the replies from collaborators as soon as possible with least bother, and the common free time for synchronous collaboration. The three services can be used either together or independently.

In the collaborator recommendation service, two different but effective algorithms for statuses and blogs are designed and used respectively. In the collaborative contact recommendation service, calendar data is used for the first time to figure out the current activity of a collaborator, which makes the inference more sufficient. Furthermore, smart phone data are effectively used by novel rules and mathematical statistics to recommend the best way to contact the collaborator. The experiments show that the precision of collaborator recommendation and contact recommendation is high respectively.

However, there are still several open problems that need to be further studied: 1) The testers are all students and the amount of feedbacks is relatively small; 2) Our synonym dealing approach uses constant dictionary that can't deal with new words created every day.

ACKNOWLEDGMENTS

This effort is sponsored by the National Basic Research Program of China (973) under Grant No. 2011CB302604, the National Natural Science Foundation of China under Grant No.61421091, No.U1201252, the Seeding Grant for Medicine and Information Sciences of Peking University (2014-MI-23).

REFERENCES

[1] http://www.renren-inc.com/en/

[2] Weilong Yao, Jing He, Guangyan Huang, Yanchun Zhang. SoRank: Incorporating Social Information into Learning to Rank Models for Recommendation, In Proceedings of 2014 World Wide Web Conference (WWW'14), Seoul, Korea, April 7-11, 2014, 409-410.

[3] Kuifei Yu, Baoxian Zhang, Hengshu Zhu, Huanhuan Cao, Jilei Tian. Towards personalized context-aware recommendation by mining context logs through topic models. In Proceedings of the 16th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining (PAKDD 2012), Part I, LNAI 7301, pp. 431–443, 2012.

[4] Xiwei Zhuang, Yanchun Sun, Kuiwei. A Smart Mobile Contact Recommender Based on Smart Phone Data. In Proceedings of ACM Internetware 2014 (Internetware'14), Hongkong, Nov. 16-21, 2014.

[5] Bin Xu, Jiajun Bu, Chun Chen, Deng Cai. An Exploration of improving Collaborative Recommender Systems via User-Item Subgroups. In Proceedings of 2012 World Wide Web Conference (WWW'12), Lyon, France, April 16-20, 2012, 21-30.

[6] Ido Guy, Uri Avraham, David Carmel, Sigalit Ur, Michal Jacovi, and Inbal Ronen. Mining Expertise and Interests from Social Media. In Proceedings of 2013 World Wide Web Conference (WWW 2013), Rio de Janeiro, Brazil, May 13-17, 2013, 515-526.

[7] Elnaz Davoodi, Mohsen Afsharchi, Keivan Kianmehr. A Social Network-Based Approach to Expert Recommendation System. In Proceedings of Hybrid Artificial Intelligent Systems (HAIS 2012), Lecture Notes in Computer Science Volume 7208, 2012, 91-102.

[8] Xin Liu, Karl Aberer, SoCo: A Social Network Aided Context-Aware Recommender System. In Proceedings of 2013 World Wide Web Conference (WWW 2013), Rio de Janeiro, Brazil, May 13-17, 2013, 781-791.

[9] Min J K, Kim H T, Cho S B. Social and Personal Context Modeling for Contact List Recommendation on Mobile Device[C]. In Proceedings of Web Intelligence and Intelligent Agent Technology (WI-IAT'08), 2008, 381-384.

[10] Gomes J B, Krishnaswamy S, Gaber M M, et al. Mobile activity recognition using ubiquitous data stream mining[M]. In Proceedings of 2012 Data Warehousing and Knowledge Discovery. Springer Berlin Heidelberg, 2012, 130-141.

[11] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet allocation. Journal of Machine Learning Research, 2003, 3: 993–1022

[12] Quan Yuan, Gao Cong, Chin-Yew Lin. COM: a Generative Model for Group Recommendation. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'14), New York, NY, USA, August 24-27, 164-172.

[13] Ge Gao, Pamela Hinds, Chen Zhao. Closure VS. Structural Holes: How Social Network Information and Culture Affect Choice of Collaborators. In Proceedings of 2013 ACM's conference on Computer Supported Cooperative Work (CSCW'13), San Antonio, Texas, USA, Feb. 23-27, 2013, 5-17.

[14] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information Processing and Management, 1988, 24(5): 513–523.

[15] Mihalcea R, Tarau P. Textrank: bringing order into texts. In Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing, 2004, 404–411.

[16] http://www.keenage.com/