Using Mobile Services Based on SNS to Support On-demand Collaborations

Yanchun Sun, *Member, IEEE*, Kui Wei, Xiwei Zhuang, Tianyuan Jiang and Wenpin Jiao

Abstract—With the wide use of smart phones, mobile users' collaborative work based on the mobile web becomes flexible, and users (i.e., collaborators) have more chances to collaborate with others anywhere and anytime. In this kind of collaborations (named as on-demand collaborations), collaborators, collaboration contacts and collaboration time are often dynamically determined. However, most existing approaches and supporting tools for traditional computer supported cooperative work just support defined collaborative process for specific collaborators. They cannot satisfy the new requirements for mobile users' on-demand collaborations where collaborators, collaborative processes and time are unknown. Furthermore, they aim at solving certain cooperative work, and cannot be extended for on-demand collaborations. To address the aforementioned issues, this paper presents an on-demand collaboration support framework that utilizes: 1) extendible mobile collaboration support services, 2) mobile users' hybrid data including friend relationships, SNS (Social Network Service) data, mobile sensor data and calendar events etc. The framework is cloud-based, that is, mobile services and mobile users' hybrid data are on cloud while mobile users request on-demand collaboration services from their mobile clients. This cloud-based architecture makes the framework scalable. Our approach uses or extends existing mobile services based on SNS and mobile sensor data to recommend on-demand collaboration support services, such as who, how, when or where to collaborate, etc., which can be used either together or independently. To verify the effectiveness of the approach and the accuracy of collaboration recommendations, we implemented three types of basic key collaboration recommendation services, i.e., collaborator recommendation service, collaboration contact recommendation service and collaboration time recommendation service on cloud, and also implemented an App on the Android platform as a client. Moreover, we designed three independent experiments. Firstly, we collected abundant data from SNS and dug out the suitable collaborators by analyzing the semantics of the collected data. Secondly, we figured out the situations which collaborators were in by analyzing the data from calendars and smart phones, then reasoned about the suitable contacts by using our novel rules, and finally recommended whether the collaboration contacts, such as call and text contacts, were suitable or not. Thirdly, we used the calendar information to recommend the common free time for collaborators to work together. The case studies show that the approach provides an effective and accurate means for on-demand collaboration recommendations.

Index Terms—mobile services, recommendation, social networking service, on-demand collaboration

1 Introduction

THE popularity of smart phones is changing many aspects of people's collaboration [1, 2]. In the past, collaborative work was mostly accomplished under specific processes by certain collaborators. Not only collaborative tasks but also collaborators and time were strictly defined. With the wide use of smart phones, mobile users' collaborative work based on the mobile web becomes flexible. Mobile users have more chances to collaborate with one another anywhere and anytime [3]. However, while collaborating on the mobile web, people have to be confronted with several basic key problems. First, it is to find collaborators. As mobile on-demand collaborations become user-centered, any mobile user may ask for collaborations without preparations. 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 (Short Message Service), but also SNS (Social Networking Services) 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 a specific collaborator at that time. 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 problems. If they can be solved well, the flexible mobile collaborations will be supported basically. However, traditional computer supported cooperative work supporting approaches and tools can't satisfy the new requirements for mobile users' on-demand collaborations where collaborators, collaboration processes and time are unknown.

To support mobile users' on-demand collaboration, it is necessary to collect and analyze various data related to mobile users. As mentioned, SNS is a platform to build social networks or social relationships among people who share interests, activities, backgrounds or real-life connections. By using their mobile phones and SNS, mobile users can create their own profiles, make friends, share photos

• *Y. Sun, K. Wei, X. Zhuang, T. Jiang and W. Jiao are with the School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China, as well as Key laboratory of High Confidence Software Technologies, Ministry of Education , China. Email: {sunyc, ccweikui, jtyuan and jwp}@pku.edu.cn, zhuangxw12@sei.pku.edu.cn.*
Please provide a complete mailing address for each author, as this is the address the 10 complimentary reprints of your paper will be sent

*Please note that all acknowledgments should be placed at the end of the paper, before the bibliography (**note that corresponding authorship is not noted in affiliation box, but in acknowledgment section**).*

and videos, and share blogs [4]. By June 30, 2015, Renren had approximately 227 million active users with over 80% of user time accessing services through mobile devices [5]. 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.

We put forward an approach [6, 7] using three mobile services to recommend who, how and when to collaborate for supporting mobile collaborations.

With further researches and experiments, we find that our previous work [6, 7] has one main shortage, that is, the approach is not scalable so that it could not satisfy additional on-demand collaboration support services of mobile users, such as where to collaborate etc. To solve the issue, this paper further improves the approach and presents a scalable on-demand collaboration support framework that utilizes: 1) extendible mobile collaboration support services, 2) mobile users' hybrid data including friend relationships, SNS data, mobile sensor data and calendar events etc. The framework is cloud-based, that is, mobile services and mobile users' hybrid data are on cloud while mobile users request on-demand collaboration services from their mobile clients. This cloud-based architecture makes the framework scalable to support mobile users' on-demand collaborations. To verify the effectiveness of the approach and the accuracy of collaboration recommendations, we implemented three basic key collaboration recommendation services for recommending collaborators, collaboration contacts, and collaboration time, respectively, on cloud, and implemented an App as a client on the Android platform. Moreover, we designed three independent experiments in the case studies to show that the approach offered an effective and accurate means for on-demand collaboration recommendations of mobile users.

Our approach has the following key contributions:

- The proposed framework is scalable and easy to extend new services to support on-demand collaborations.
- It provides an effective algorithm for the collaborator recommendation.
- It's the first time to propose and solve the collaboration contact problem.
- It presents a mechanism to maintain and filter multi-dimensional and hybrid data from SNS, calendars and smart phones to improve the effectiveness of collaboration recommendation services.

The rest of the paper is organized as follows. Section 2 describes the system architecture of the on-demand collaboration support framework. Section 3 introduces mobile services in detail. Section 4 gives the implementation of the approach. Section 5 introduces the case studies. Related work is discussed in Section 6. Section 7 presents concluding remarks and future work.

2. System Architecture

Most of the existing collaboration support systems are based on centralized architecture [8] and they mainly focus on providing specific collaboration support services [9], such as collaboration location or activity recommendation. However, to support mobile users' on-demand collaboration, the collaboration support framework needs to provide many services, such as the management of collaborations, the recommendation for collaborators, collaboration time, and collaboration location and so on. What's more, different services need the support of a single or mixed data sources and different users may require a single or mixed support services. Therefore, to address these issues, we propose an on-demand collaboration support framework which is scalable enough as reflected in Fig.1.

The proposed framework has the following advantages:

- It's easy to extend new services according to mobile users' on-demand collaborations.
- The collaboration support services not only can be used together as a whole, but also can be used independently.
- The users' hybrid data can be maintained in the same module and be shared by different services.

The following are the major components of the proposed on-demand collaboration support framework.

2.1 Data Sources

To support the services, the framework needs to get mobile users' data from various data sources. This module mainly considers two types of data sources.

- *Data from Open Platforms*. This module can retrieve uses' data through the open platform API with users' authorization. For example, in Fig.1, as users authorize this module to use their information on Google Calendar, this module can get the calendar events of the users through the Open Platform API. With the same process, this module can get the SNS data through SNS Open Platform. The SNS data record has the following fields: (a) user identification, (b) user's friend relations, (c) user's blogs, and (d) user's status and so on.
- *Data from Mobile Sensor*. To support services based on the analysis of the mobile sensor data, this module will get the mobile sensor data such as Wi-Fi signal, carrier signal, time when phones are last used, alarm clock information in the phone, microphone using conditions, recently answered calls, recently rejected calls, running states of SNS, location, speed, accelerated speed and so on.

2.2 User Hybrid Data Maintain

Because the data are retrieved from different sources and the data will be shared by different services, the proposed framework provides a uniform module to maintain the users' hybrid data. This module mainly maintains the following data:

- *Raw data.* This type of data mainly contains the data retrieved through the data sources in 2.1. And these data are not processed by any services.
- *Generated data by offline preprocess.* These data are mainly generated in the offline preprocess of the collaboration support services. For example, to support the collaborator recommendation service, this module will preprocess the users' blogs to get the relevance between topics and blogs and the data will be generated periodically with new blogs.

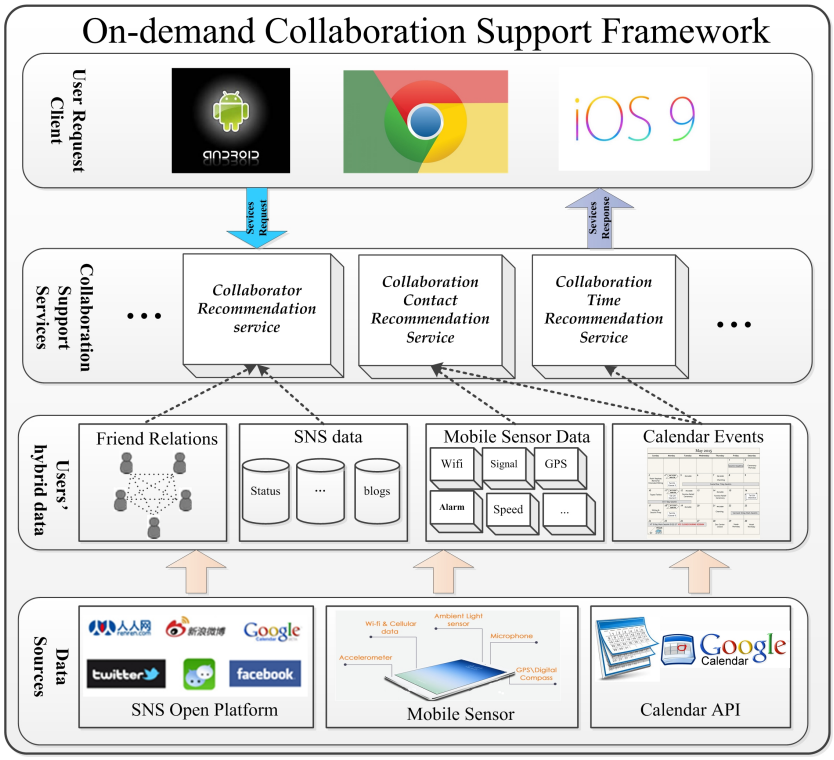

Fig. 1. Top Level architecture of On-demand Collaboration Support Framework

- *Static data.* This type of data mainly contains the configuration data and profile data used in different services. Taking the synonym problem in collaborator recommendation service as an example, the service needs the relevance between different topics. Whereas the relevance needn't be generated dynamically.

2.3 Collaboration Support Services

The collaboration support services are the core of the framework. The purpose of this module is to satisfy the requirements for mobile users' on-demand collaborations. Because users may have different requirements for the services for different collaboration tasks, this module needs to allow users to use a single service for specific collaboration task or use these services as a whole to support complete collaboration workflow. For example, not only can a user just use collaborator recommendation service to find a friend related to "Android" or use collaboration contact recommendation service to determine the best way to contact his friend right now, but also can the user use these services to support the collaboration in holding an interest group about "Android".

What's more, this module is easy to extend services because the data in the User Hybrid Data module can be shared by all services in the module and all the services are implemented independently. For example, if a mobile user wants to find a friend who has the most similar interest as him. It's easy to deploy a new collaborator similarity recommendation service based on the shared SNS and friend relations data.

Because the collaboration support services in this module are loosely coupled, the implementations of the services are independent. We will give the details of the implementations of the services in the Section 4.

2.4 User Request Client Module

As reflected in Fig.1, the proposed framework follows Software as a Service (SaaS) through a cloud based architecture. One of the major advantages of the approach is that the proposed framework can scale on demand as additional virtual machines are created and deployed. The module of collaboration support services offers real-time on-demand service to collaborators, while abstracting underlying implementation details. Users can access the services through clients such as Android, iPhone or Web Browser without being aware of the physical locations of the hosted services.

3 Extendible Collaboration Support Services

As mentioned, who, how and when to collaborate are three basic collaboration problems. If they can be solved well, the flexible mobile collaborations will be supported effectively. Therefore, we have implemented three basic key collaboration recommendation services including collaborator recommendation service, collaboration contact recommendation service and collaboration time recommendation services on cloud, as well as their clients as an app on Android platform.

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. Collaboration contact recommendation service analyzes the data from calendars and smart phones, figures out the situations

which collaborators are in, reasons about the suitable contacts with some novel rules and finally recommends the ranked suitable text contacts whom we can call. The potential collaborators can be grouped and collaboration time recommendation service utilizes the data from bound calendars and recommends common free time, which solves the problem of when to collaborate.

3.1 Collaborator Recommendation Service

Collaborator recommendation service helps to find the collaborators related to specific topics which are based on users' SNS data. In China, Renren is one of the most popular SNS platforms. In Renren platform, 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 data. 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. Then we will store the raw data in Users Hybrid Data module.

In this section, we discuss the design of the collaborator recommendation service in detail. In terms of functionality, the service consists of two main modules: 1) an offline preprocessing module and 2) an online recommendation module. The offline-processing module runs periodic jobs to pre-processing users' hybrid data. Data pre-processing involves three sections: 1) topic relevance of users, 2) semantic similarity between words, and 3) similarity between collaborators. The online recommendation process is responsible for generating the collaborators who are related to the specific topic. The detailed functionality of the above-mentioned modules is discussed in the following subsections.

3.1.1 Periodic Offline Preprocessing

1) Frequency Based Relevance

Term frequency and inverse document-frequency (TFIDF) is a numerical statistic that is intended to reflect how important a word is to a document [10]. 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 recommendation service 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 users with lots of documents rather than the TFIDF of every single status and blog. We propose status frequency and the 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.

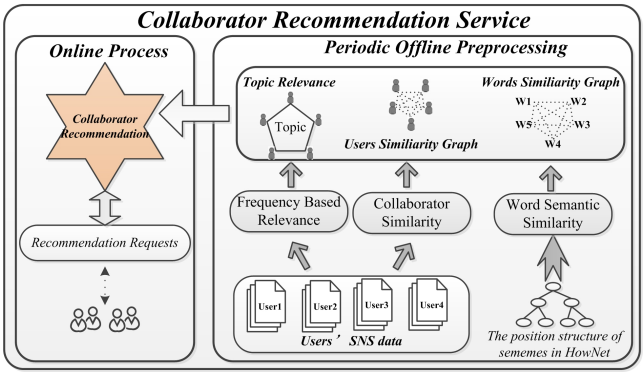

Fig. 2. Architecture of Collaborator Recommendation Service

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:

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

Where $|\{s: q \in s\}|$ is the number of statuses containing q, and $|du|$ is 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:

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

TF(q) weighs how important the query q is in the document. IDF(q) is a measurement of 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 sufficient 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:

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

Where $|du|$ is the total number of blogs of user u. The reason why it's divided by $|du|$ 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 related to query q is defined as follows:

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

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) Words Semantic Similarity

The frequency-based relevance calculation 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 requirement. We use a synonym set to overcome the gap.

In this section, we utilize the proposed method [11] to measure semantic similarity between topics based on *HowNet* lexicon [12], which consists of about 60 thousands of Chinese words. The proposed method not only refines the distance-based methodology, but also explores *HowNet* as knowledge base. *HowNet* organizes all the sememes into several trees and each sememe is considered as a node of a tree as shown in Fig.2. We define the distance between sememes as the number of edges of the shortest path between them. Then the similarity between sememes as:

$$\mathrm{Sim}(S1, S2) = \alpha/(d + \alpha)$$

α is an adjustable parameter and represents the path length when the similarity is 0.5. d means the path length of two sememes S_1, S_2 in the sememe level system. With the sememes similarity, we can compute the similarity between concepts in a natural language.

Similarity between words actually means the similarity between concepts associated with them. But in a natural language, a word can represent one or more concepts. Under this circumstance, we take the maximum similarity between the concepts as the similarity between the two words. Formally it is defined as:

$$\mathrm{Sim}(W_1, W_2) = \max Sim(C_{1i}, C_{2j})$$

C_{1i}, C_{2j} are i-th and j-th concepts associated with W_1 and W_2 respectively.

3) Collaborator Similarity

The frequency-based relevance calculation just considers the relevance between query topics and users. However, in real-world situation, users not only consider topic relevance, but also consider the similarity between collaborators. For example, if a user in Beijing wants to find some collaborators who are interested in programming to participate in a programming contest, the service will first recommend collaborators in Beijing with high topic relevance because most of the friends live in the same place. However, if the service just considers the topic relevance, the service will also recommend collaborators with high topic relevance though they live in Shanghai instead of those with less topic relevance even though they live in Beijing.

Therefore, the service not only needs to consider the topic relevance, but also needs to consider the collaborator similarity.

In this subsection, we introduce the method to calculate collaborator similarity based on users' SNS data. For each user, the method utilizes the result by Frequency-based Relevance module and represents the result as shown in Table 1.

TABLE 1
USER-TOPIC RELEVANCE TABLE

	Topic1	Topic2	Topic3	…
User1	r_{11}	r_{12}	r_{13}	…
User2	r_{21}	r_{22}	r_{23}	…
User3	r_{31}	r_{32}	r_{33}	…
…	…	…	…	…

With the user-topic relevance matrix, the method will calculate the collaborator similarity based on Cosine Similarity.

3.1.2 Online Recommendation Process

In this subsection, we present the online collaborator recommendation process which is based on the generated data of offline preprocess.

Algorithm 1. Online Collaborator Recommendation

Input: Active user: u, search topic: q.
Output: A set of recommended collaborators.
Definitions: Users' topic relevance V, topics semantic similarity S, Users' group similarity G.

```
1: Tr[] <— {0}      // users relevance related to topic q
2: STS <— getSimTopics(q)
3: F <— getFriendList(u)
4: for each f ∈ F do
5:     Tr[f] <— V(q)
6:     for st in STS :
7:         Tr[f] <— Tr[f] + V(st) * S(q, st) * β
8: candidates <— getTopKCandidate(STS)
9: for each cand in candidates do
10:    for each sim_cand in candidates do
11:        Tr[cand] <— Tr[cand] + G(cand, sim_cand) *
                       Tr[sim_cand]
12: results <— rank(candidates, Tr)
13: return results
```

Algorithm 1 illustrates the procedure of the online recommendation process.

Every search topic has a collection of similar words with similarity degree. Line 2 gets the similar topics of query q. Lines 3-7 calculate the relevance between the query topic q and the user's friends. 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 relevance of q as follows.

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

Constant β is to deduce the bias of synonyms. Since users use q to search, q should be the most important query word. If the query consists of more than one word, we add all the VF(q).

Line 8 gets the top K candidate collaborators by the relevance related to q. Lines 9-11 recalculate the candidates' weight based on the relevance to the topic q and the similarity between candidate users. The calculation is based on the idea of PageRank. Line 12 will rank the candidate based

on the weight and return the most relevant friends as potential collaborators.

3.2 Collaboration Contact Recommendation Service

We have proposed contact recommender in [7]. So here we just give a general introduction of the approach to make the paper self-contained.

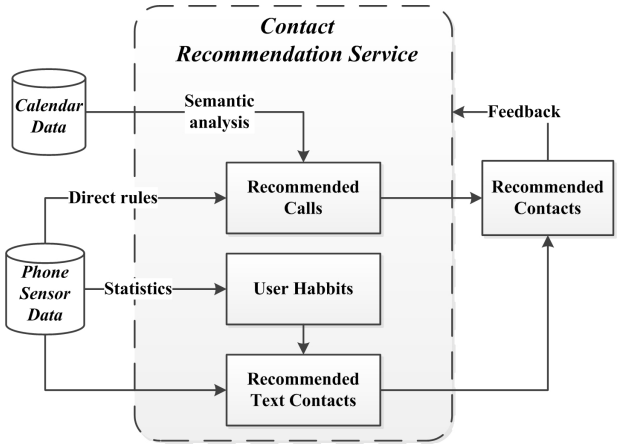

Fig.3. Collaboration contact recommendation service process

Figure 3 shows the whole process of collaboration 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 numbers, so that the data can be calculated and people can get the recommended contacts of friends. The service is useful only when both sides use it.

Besides the calendar data, the service collects lots of information from sensors in smart phones, such as Wi-Fi signals, carrier signals, time when phones are last used, alarm clock information in phones, microphone using conditions, recently answered calls, recently rejected calls, running states of SNS, locations, speeds, accelerated speeds and so on. The process consists of two aspects, call recommendation and text contact recommendation.

3.2.1 Call recommendation

Calls are the most direct way to reach people, but at the same time calls disturb people most in some specific conditions. They disturb in two ways. Firstly, calls always come with long time rings and vibrations that will disturb people. Secondly, calls are synchronous communication ways that need 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.

1. **Meeting.** The server gets the bound calendar data by Google Calendar API. Through semantic analysis of the 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.
2. **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.
3. **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 about the call recommendation. We take several rules for example.

1. **Carrier signals.** If there are no carrier signals, calls can't be made successfully. The recommender will recommend not to call people.
2. **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.
3. **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.
4. **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.

3.2.2 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 runs, 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 >Tw> Tq >t-k or t>Tw>t-k> Tq then
      Wechat is ranked before QQ
else if  t> Tq>Tw>t-k or  t> Tq>t-k> Tw then
      QQ is ranked  before Wechat
else if  Tw <t-k and Tq<t-k then
          if Fw > Fq 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.

3.3 Collaboration 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. Collaboration 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 collaboration time recommendation service will get calendar information of all the collaborators through the API provided by Google and calculate the common free time satisfying the time requirements. If there is no satisfactory time, the collaboration time recommendation service will recommend the time when most collaborators are free. After the organizer chooses one from the list of free time, the collaboration time recommendation service will inform all the collaborators in the group of the chosen time. If collaborators accept the time, the new event will be added to Google Calendar of the collaborators.

Collaboration time recommendation service reduces the costs of finding a common time and helps the arrangement of schedule.

In this section, we provide in detail the design and implementation of the collaboration time recommendation service as shown in Figure 4. This service also comprises two main modules: 1) an offline preprocessing module and 2) an online collaboration time recommendation module. The offline preprocessing module runs periodically on the server to collect and preprocess users' calendar data as well as to establish a future event prediction model for each user based on Support Vector Machine (SVM). The online module is invoked when an organizer requests a collaboration time recommendation. The following subsections serve to discuss the detailed functionalities of the two aforementioned modules.

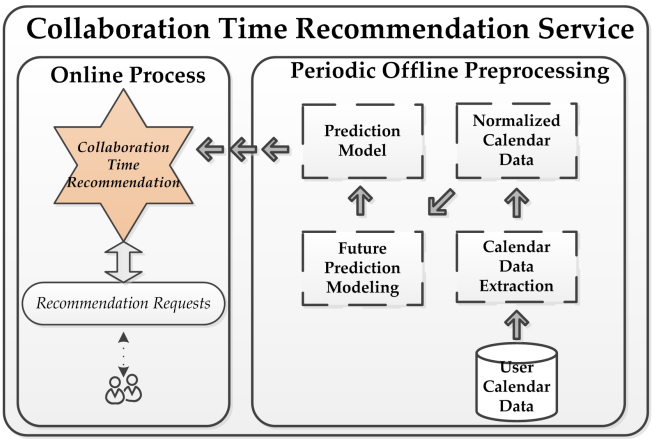

Fig. 4. Architecture of Collaboration Time Recommendation Service

3.3.1 Periodic Offline Preprocessing

1) Calendar Data Extraction

User calendar data from Google Calendar or other sources are stored in varied structures. It is essential to preprocess the raw data, extract key information and re-store the structured and normalized data on the server database before further use. Fortunately, for online calendar services like Google Calendar, it is convenient to call the official APIs to retrieve information on demand.

Three fields are selected for their importance: a) date, b) occurrence time, and c) event description. The first two fields together reflect the exact time when an event happens. The last field helps to determine whether that time period is available or occupied.

The server periodically fetches users' data from Google Calendar or mobile calendar and preprocesses the data in the aforementioned method every day.

2) Future Prediction Modeling

A crucial problem in the intelligent collaboration time recommender is that there is not sufficient information for future events in the calendar, for the reason that most users tend to fill in their calendar only one or two weeks ahead. The sparsity of data would result in significant inconvenience for collaborators who thus have to determine the collaboration time manually. Considering both the response time and accuracy of the collaboration time recommendation service, the ideal solution would be firstly establishing an offline model, with which the service could provide online results in linear time complexity later.

In the light of the ideal solution above, we improve the recommender with a sub-module to predict forthcoming events based on users' previous calendar data. We extracted the essence of the prediction problem and abstracted it as the classification of a future time slot to be whether available or occupied. Compared to previous methods, like Neural Network and Genetic Algorithms, SVM stands out as an exceptional one that fits our purpose

and the small scale of user calendar data best. SVMs' robust performance with even sparse and noisy datasets has made it the perfect choice for our approach.

Support Vector Machine. Based on the structural risk minimization principle of SVM [13] [14], the quality of our approach's results is independent of the dimensionality of input space. Moreover, since we introduce SVM into the system, the original classification problem can be expressed as a quadratic programming optimization problem [13]:

$$\min \mathcal{P}(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^2$$
$$s.t.\ \forall i \quad y_i(\mathbf{w}^T \mathbf{\Phi}(\mathbf{x}_i) + b) \geq 1$$

where **w** is a slope vector, *b* is the intercept of the hyperplane that SVM learns to separate, i.e. classify, the data in the feature space; And **Φ** is the feature vector chosen by hand. This problem is still difficult to solve directly, but can be simplified to a dual problem with the Lagrangian Duality Theory [15]:

$$\max \mathcal{D}(\boldsymbol{\alpha}) \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i \alpha_i y_j \alpha_j \mathbf{\Phi}(\mathbf{x}_i)^T \mathbf{\Phi}(\mathbf{x}_j)$$
$$s.t. \quad \forall i \quad \alpha_i \geq 0$$
$$\textstyle\sum_i y_i \alpha_i = 0$$

where the dot product of $\mathbf{\Phi}(\mathbf{x}_i)^T \mathbf{\Phi}(\mathbf{x}_j)$ can be replaced by the kernel function $K(\mathbf{x}, \mathbf{x}')$. The kernel function used in our approach is a positive semi-definite matrix, leading to the convexity of the optimization problem, which assures our solution to be globally optimal.

Feature Definition. There are three key features selected to be taken into consideration by our future prediction modeling process.

1. **Date in a month.** Considering there are bi-weekly events and they are with longer intervals, the date in a month is an important factor.
2. **Weekday.** Most users have a weekly schedule, for example, courses in college for a student, or weekly meetings for faculties.
3. **Hour in a day.** Every hour in a day could have varied events. It is necessary to consider each hour separately.

All user calendar data on the server are preprocessed to extract the three values to construct the feature space, and label the corresponding items as available or occupied, according to the state on the calendar.

The server runs SVM to train a model when a new mobile user enters the system, and will do re-modeling once a week, to update the model with newly fetched user calendar data. Since mobile users' periodical schedules tend to change though infrequently, we only use data in the last six months during the modeling process.

3.3.2 Online Recommendation Process

In this subsection, we present the online collaboration time recommendation process which is based on users' normalized calendar data and pre-trained models of offline preprocess.

Algorithm 2 demonstrates the process of the online collaboration time recommendation module.

Algorithm 2. Online Time Recommendation

Input: group user list: U, preferred time period Pt.
Output: A set of recommended collaboration time sorted by date and the number of available group members.

```
1: Au[] ← {∅}      // available user list in each period
2: CPt ← ∅         // time period candidate set
3: for each u ∈ U do
4:     m ← getUserModel(u)
5:     for each pt ∈ Pt do
6:         us ← getUserSchedule(u, pt)
7:         if us is null then
8:             us ← predictSchedule(m, pt)
9:         if isUserAvailable(us) is true then
10:            CPt.add(pt)
11:            Au[pt].add(u)
13: results ← sort(CPt, AuNa[])
14: return results
```

Lines 3-11 calculate the available time periods for each user in the given group recursively.

Line 4 retrieves the user's prediction model pre-trained in the offline process.

Lines 6-11 first get the user's schedule in the period **pt**. If there is not schedule data for this user, line 8 will try to predict the schedule based on the pre-trained model **m**. For either situation, when a user is judged as available for that period, lines 10-11 will add **pt** to the candidate time period set and push the user into the **Au** array that maintains an available user list for each time period. Line 13 sorts the candidate time periods according to the number of available users and periods with the same user number are ranked in time order.

4 Implementation of The Approach

We implement the approach as an application including three mobile services on Android smart phones and a cloud server.

The server is mainly responsible for a) periodical data retrieval and preprocessing, b) providing the aforementioned three major collaboration recommendation services and any other extensions to clients on all possible platforms, and c) collect users' feedback information for further analysis and improvement.

The Android client mainly comprises three modules in correspondence with the three major services in the server: a) a built-in intelligent search engine that will give collaborator recommendation results according to users' query; and a group management module, in which b) the organizer is able to appoint a time for collaboration with the assistance of the collaboration time recommendation service, and c) group members will be informed in the contact methods given by contacts recommendation service. There are several peripheral modules help with client local data storage and SNS account binding, etc.

To use the client, a mobile user should first register an account with a mobile phone number and an Email address. The client will get contacts from the phone, and the user can add friends from the contacts. Besides, the mobile user can bind his SNS accounts, such as Google account and Renren account to offer more information that will

help improve the recommendation services. The application will collect all the related data, preprocess and send intermediate result to the server in time. The server will retrieve the information from bound SNS accounts, calculate and analyze the data from both mobile phones and SNS.

When a mobile user wants to find the specific persons to collaborate with, they can simply type the query in the search box (Fig. 5. left). The application will return a list of friends with the related statuses and blogs. Mobile users can check all the information and choose the most suitable collaborators. If it's a complex task that needs more than one collaborator, mobile users can add the collaborators in one group.

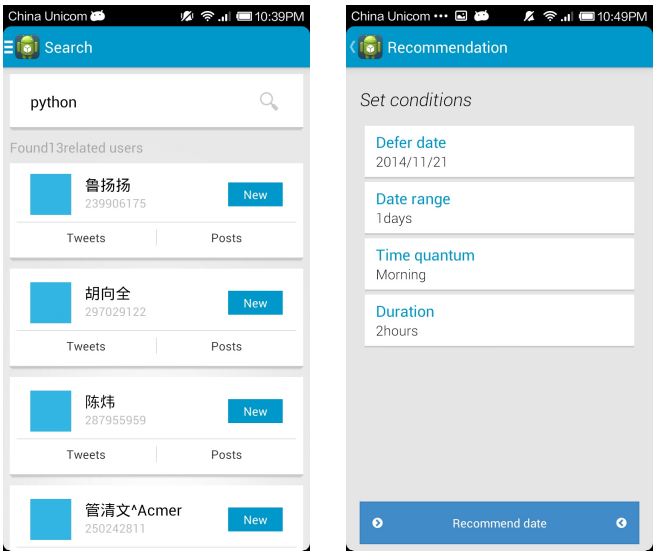

Fig. 5. Example screenshots of the android client. Left is from the collaborator recommendation module. Right is from the collaboration time recommendation module.

To make sure the collaborations, mobile users have many situations to contact friends. Mobile users can get the real time recommendation of friends through the server so that they can contact friends in the best way. What mobile 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 mobile users have a group of collaborators, mobile users usually need a common free time for synchronous collaboration. They enter the group and choose the time limitations (Fig. 5. right). 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. Mobile users can choose the best one from the list and the application will inform the rest collaborators in the group. Collaborators can confirm the collaboration time and the time will be allocated in the calendar.

5 Experiments

In this section, to verify the effectiveness of the approach and the accuracy for collaboration recommendations, we design three independent experiments. The first experiment is to verify whether the collaborator recommendation service can dig out the suitable collaborators when the mobile users search specific topics related to the suitable collaborators. The second experiment is to evaluate the service accuracy of the contact recommendation. And the last experiment is to assess the accuracy of the future event prediction models trained in the offline process of collaboration time recommendation service.

5.1 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.

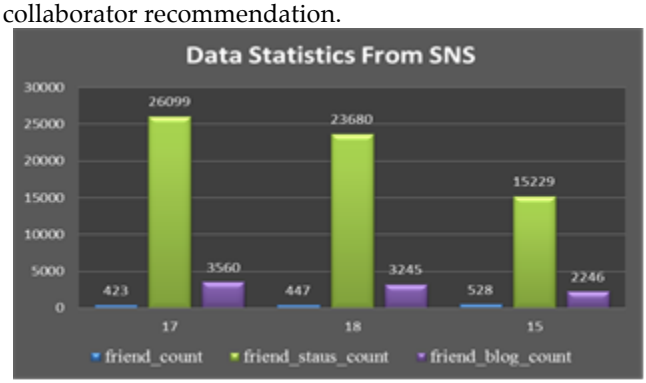

Fig. 6. Statistics data for each user

Figure 6 shows the number of the friends, friends' blogs and friends' statuses for each student. To make the figure clear, we only show the data of 3 typical mobile users whose id are 15, 17, 18. We can see that every student user has about 500 friends and the number of statuses is much more than the number of blogs. What's more, for every student user, the data needing to be processed is large.

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

The application will recommend a sorted list of 20 relevant collaborators after a student user searches one topic related to the collaborators. The student user 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 user has searched about 80 topics to get recommended collaborators and we got 713 feedbacks.

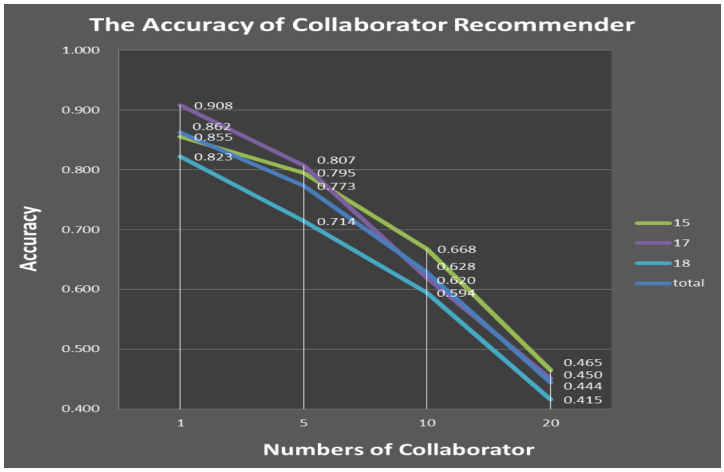

Fig. 7. Accuracy for different number of collaborators

There are many different collaborative situations in which mobile users will need to find different numbers of collaborators. For example, when mobile 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 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 7 shows the experiment results. To make the figure clear, we only 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 mobile 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.

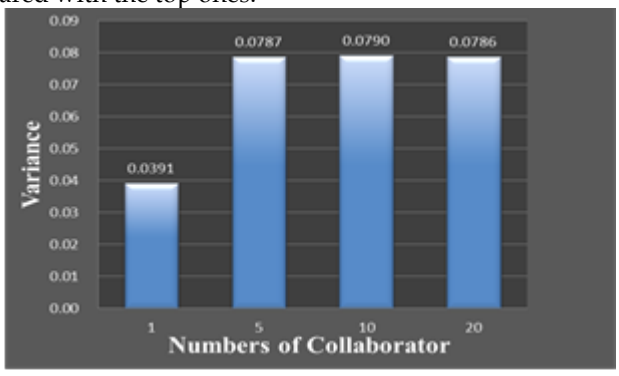

Fig. 8. Variance of accuracy

Figure 8 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 mobile users need to find a small group of collaborators who are related to the specific topics. It has the limitation when mobile users want to find more than 10 collaborators because there are not enough friends on the SNS who are related to the specific topics.

5.2 Experiment Results of Collaboration Contact Recommendation Service

We find 50 students from PKU, consisting of 21 undergraduate students and 29 graduate students, to use the collaboration 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 mobile users can feed back whether the call recommendation and text contact recommendation are right. We got 1021 pieces of feedbacks totally.

TABLE 2
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 2 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.

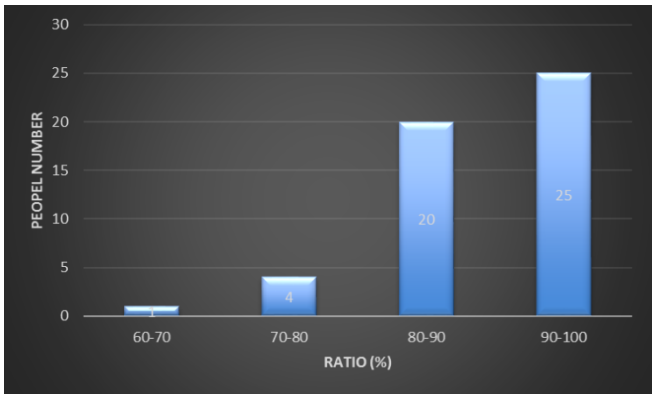

Fig. 9. People distribution in ratio intervals

Figure 9 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.

5.3 Experiment Results of Collaboration Time Recommendation Service

The experiment mainly focuses on quality of the future event prediction module. Due to some Internet restrictions, the surveyed group is selected from a small but typical group. The surveyed group is composed of 8 undergraduate students, 8 graduate students and 4 doctoral students from Peking University, who have installed and used the application for a long period of at least six months and who regularly update their schedule with Google Calendar. The users' calendar data are collected upon surveyed group members' agreement. The time spans of collected are from eight months at least to one year for each user.

In order to avoid overfitting, the experiments are conducted with the 10-fold cross validation method by randomly partitioning each user's data into 10 sets, and using 9 sets as training set while the remaining set left as test set each time until every one of the partitioned sets has been selected as test set for once. This method evaluates the prediction models independently of the data that are used to train the model.

We used three values to evaluate the collaboration time recommender:

a) **Precision (*P*).** The precision is defined as the number of the true positives over the number of the true positives (T_p) plus the number of the false positives (F_p).

$$P = \frac{T_p}{T_p + F_p}$$

b) **Recall (*R*).** The recall is defined as the number of the true positives over the number of the true positives plus the number of the false negatives (F_n).

$$R = \frac{T_p}{T_p + F_n}$$

c) **F1-Score.** The F1-score is defined as the harmonic mean of precision and recall. This parameter is an overall measure of the results.

$$F1 = 2 \times \frac{P \times R}{P + R}$$

Figure 10 shows the experimental results sorted in ascending order of ***P***. From the results, the precision varies in a relatively large range, while recall maintains a very high level.

The variety of precision is from the difference in calendar data. For low precision cases, the corresponding data have many irregular events, so it is difficult to be accurate to learn the pattern of uncertain events. While in high precision cases, the mobile users are more likely to live in regular routines.

The high recall means that our approach seldom misses events that will truly happen. It stands for the strictness of the standard of being "available". Because of the unpredictability of random incidences in people's life, this strictness is very practical in this specific situation – the recommended time is almost sure to be available for all group members. Only in very rare cases does the organizer need choose another time for the collaboration.

The F1-scores are over 0.9 in most cases, indicating that the overall quality of prediction results is at a very high level. This is partially because the surveyed group is mainly composed of college students, whose schedule data has strong periodical patterns. In the future, we plan to extend the experiment by widening the survey range to get more general results.

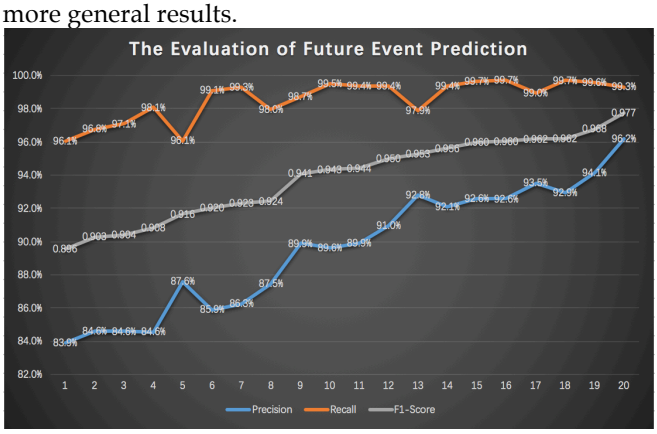

Fig. 10. The evaluation of future event prediction module, ranked in ascending order of F1-Score.

6 Related Work

Since our work focuses on using a scalable collaboration support framework, which is based on extendible mobile collaboration recommendation services and SNS data and mobile sensor data to support on-demand collaborations, we have categorized the related work into different sections. The first section covers existing literature in recommender systems, the second section covers existing literature on user hybrid data filtering, and the third covers existing work in collaborations based on mobile services. We differentiate our work from them and identify their shortcomings.

6.1 Recommender System

Recommender systems have been studied widely due to the incredible increasing information in the world, especially on the Web [16]. These systems apply knowledge covery techniques to make personalized recommendations that help people to sift through the huge amount of available data [17, 18, 19, 20]. 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 [17, 21, 22, 23, 24, 25]. 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 [17]. 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 [21]. Elnaz 2012 et al. have presented an expert recommendation method that integrates content-based recommendation algorithms into a social network-based collaborative filtering [22]. Liu et al. propose SoCo, which systematically combines contextual information with social network information to improve the quality of recommendations [23]. However, these methods above don't concern the mobile sensor data that are very important to predict the mobile users' situation.

There is also some related work on collaboration contact recommenders. Min et al. have implemented a contact recommendation application on a mobile device that recommends phone numbers in a phonebook according to the user's situation based on logs stored in mobile devices [26]. But it recommends people suitable to contact, rather than suitable ways to contact specific friends. Bhargava et al. present a system and an approach for performing multi-dimensional collaboration recommendations for Who (User), What (Activity), When (Time) and Where (Location), using tensor factorization on sparse user-generated data [27]. But it is mainly for enhancing the overall experience of users for travel, not for the collaboration.

6.2 User Hybrid Data Filtering

Given the abundance of information available to mobile users, how to effectively filter this information and suggest related information to mobile users become increasingly important. The most straightforward method for keyword extraction is using TFIDF [10] to rank candidate keywords and select the top-M as keywords. After PageRank becomes popular, graph-based ranking methods like TextRank [28], 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)[29] 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.

6.3 Collaborations Based on Mobile Services

Research work that focuses on how to use smartphones to support collaborations becomes more and more popular [1, 2, 3, 30]. Duan et al. analyze different incentive mechanisms for a master to motivate the collaboration of smartphone users on both data acquisition and distributed computing applications, then propose a reward-based collaboration mechanism [1]. Wu et al. design an indoor localization system based on off-the-shelf Wi-Fi infrastructure and mobile phones in order to support crowdsourcing [2]. But these approaches above do not concern key collaboration problems such as who, how and when to collaborate. Kang et al. propose a mobile community service platform that enables a mobile phone to promote collaborations [3]. In this method, a mobile community is a set of mobile phone users who collaborate to achieve a common goal. The mobile community follows a task model to finish the task. However, this method mainly supports the defined task, and it is short of support for mobile users' on-demand collaborations. 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 [30]. 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.

Analyzing the related work above, they only concern certain issue(s) of collaborations, and seldom support key collaboration problems such as who, how and when to collaborate together. Moreover, these approaches are not scalable so that they could not combine existing services or extend new services according to mobile users' on-demand collaboration requirements. To support mobile users' on-demand collaboration, we put forward an on-demand collaboration support framework that utilizes: 1) extendible mobile collaboration support services, 2) mobile users' hybrid data including friend relationships, SNS data, mobile sensor data and calendar events etc. The framework is cloud-based, which makes the framework scalable to support mobile users' on-demand collaborations easily.

7 Conclusion And Future Work

We design and implement a scalable on-demand collaboration support framework which is easy to extend mobile services based on SNS, calendar data, and sensor data from smart phones, in order to support mobile users' ubiquitous on-demand collaborations, such as who, how, when or where to collaborate, etc. To verify the effectiveness of the approach and accuracy of collaboration recommendations, we have implemented three basic key collaboration recommendation services including collaborator recommendation service, collaboration contact recommendation service and collaboration time recommendation service on cloud, as well as their clients as an app on Android platform. The collaboration recommendation services can help mobile

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 services can be used either together or independently.

In the collaborator recommendation service, our approach not only considers collaborators' topic relevance, but also considers the similarity between collaborators. Moreover, two different but effective algorithms for statuses and blogs are designed and used respectively. In the collaboration 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. In the collaboration time recommendation service, the users' data is utilized to provide feasible time recommendations with predictions of possible future events. The experiments show that the precision of collaborator recommendation, contact recommendation and collaboration time recommendation is high respectively.

However, there are still several aspects that need to be further improved. 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.

In the future, we plan to solve the shortages above first. Moreover, we plan to extend the SNS data from other SNS platforms to support a wider range of mobile users. As well as, we intend to implement more collaboration support services according to mobile users' feedbacks.

ACKNOWLEDGMENT

This effort is sponsored by the National Basic Research Program of China (973) under Grant No. 2015CB352200, 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] Duan L., Kubo T., Sugiyama K., Huang J., Hasegawa T., Walrand J. Motivating Smartphone Collaboration in Data Acquisition and Distributed Computing. IEEE Transactions on Mobile Computing, 2014, 13(10): 2320 – 2333.

[2] C. Wu, Z. Yang, Y. Liu. Smartphones Based Crowdsourcing for Indoor Localization. IEEE transaction on Service Computting, 2015, 14(2):444-457.

[3] Kang K., Lee J., Beek K., Kim J.. A Mobile Community Service Platform Promoting Ubiquitous Collaboration. In Proceedings of 2011 Ninth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2011, 939-946.

[4] Caton S., Haas C., Chard K., Bubendorfer, K.; Rana, O.F. A Social Compute Cloud: Allocating and Sharing Infrastructure Resources via Social Networks.IEEE Transaction on Service Computing, 2014, 7(13): 359-372.

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

[6] Yanchun Sun, Xiwei Zhuang, Kui Wei, Xudong Shan, Tianyuan Jiang. Using Mobile Services Based on SNS to Recommend Who, How and When to Collaborate. In Proceedings of 2015 IEEE International Conference on Mobile Services, New York, USA, June 27-July 2, 2015, 17-24.

[7] 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.

[8] Zhang J, Kuc D, Lu S. Confucius: A tool supporting collaborative scientific workflow composition [J]. IEEE Transactions on, Service Computing, 2014, 7(1): 2-17.

[9] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative location and activity recommendations with gps history data. In Proceedings of the 19th international conference on World wide web, 2010.

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

[11] Dai L, Liu B, Xia Y, et al. Measuring semantic similarity between words using HowNet[C]. In proceedings of. In proceedings of 2008 IEEE International Conference on Computer Science and Information Technology. 2008, 601-605.

[12] HowNet. http://www.keenage.com/.

[13] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

[14] Suykens, J. A., & Vandewalle, J. (1999). Least squares support vector machine classifiers. Neural processing letters, 9(3), 293-300.

[15] Bottou, L., & Lin, C. J. (2007). Support vector machine solvers. Large scale kernel machines, 301-320.

[16] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutie´rrez, ''Recommender Systems Survey,'' Knowl.-Based Syst., vol. 46, pp. 109-

[17] 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.

[18] 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.

[19] 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.

[20] Personalized Web Service Recommendation via Normal Recovery Collaborative Filtering. Sun, Huifeng, +, TSC October-December 2013 573-579 (collabration)

[21] 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.

[22] 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.

[23] 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.

[24] Weilong Yao, Jing He, Guangyan Huang, Yanchun Zhang. So-Rank: 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.

[25] 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.

[26] 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.

[27] P. Bhargava, T. Phan, J. Zhou and J. Lee.Who, What, When, and Recommendations Using Tensor Factorization on Sparse User-Generated Data, WWW 2015, 130-140.

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

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

[30] 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.

Yanchun Sun received the PhD degree from Northeastern University, China, in 1999. She is an associate professor in the School of Electronics Engineering and Computer Science, Peking University, China. Her current research interests include service computing, cloud-based education, mobile computing, collaboration based on SNS, software architecture, etc. She has served as PC member of several IEEE conferences in software engineering and service computing fields and is an editor of IEEE SWEBOK Version 3. She is currently leading a research project funded by a grant of National Natural Science Foundation of China and a project funded by the Seeding Grant for Medicine and Information Sciences of Peking University. She has published more than 70 papers and written a textbook "Software Engineering". She is a member of the IEEE and the IEEE Computer Society.

Kui Wei received the BS degree from Beihang University in computer science. He is currently working toward the MS degree in software engineering at Peking University. He is currently working on travel time estimation based on express trajectory data. His research interests include social data mining, trajectory data mining, collaboration recommendation and mobile services. He worked on several projects including the National Natural Science Foundation of China and the Seeding Grant for Medicine and Information Sciences of Peking University. He has been recently working on cooperation project on express trajectory data management and analysis, with an strong emphasis on express travel time estimation.

Zhuang Xiwei received the BS degree in computer science and the MS degree in software engineering under the guidance of professor Hong Mei and Yanchun Sun from Peking University. His research interests include social network services (SNSs), collaborative works and mobile services. He worked on several projects including the National Natural Science Foundation of China and the Seeding Grant for Medicine and Information Sciences of Peking University.

Tianyuan Jiang is an undergraduate student of the School of Electronics Engineering and Computer Science at Peking University. He is currently with the Operating System and Middleware Lab under the Software Engineering Institute, where he works with professor Yanchun Sun. His research interests include data mining, machine learning techniques and applications in ubiquitous computing as well as software development on mobile platforms. He has recently been working on several projects on mobile health and a ubiquitous on-demand collaboration support framework.

Wenpin Jiao is a full professor of the School of Electronics Engineering and Computer Science at Peking University. He received his Ph.D. in computer science from Institute of Software, Chinese Academy of Sciences in 2000, and his M.S. and B.S. in computer science from East China University of Science and Technology in 1997 and 1991, respectively. Jiao's research interests include software engineering, autonomous component technology, and multi-agent systems.