Friendbook: A Semantic-based Friend Recommendation System for Social Networks

Zhibo Wang, *Student Member, IEEE,* Jilong Liao, Qing Cao, *Member, IEEE,* Hairong Qi,*Senior Member, IEEE,* and Zhi Wang, *Member, IEEE,*

Abstract—Existing social networking services recommend friends to users based on their social graphs, which may not be the most appropriate to reflect a user's preferences on friend selection in real life. In this paper, we present Friendbook, a novel semantic-based friend recommendation system for social networks, which recommends friends to users based on their life styles instead of social graphs. By taking advantage of sensor-rich smartphones, Friendbook discovers life styles of users from user-centric sensor data, measures the similarity of life styles between users, and recommends friends to users if their life styles have high similarity. Inspired by text mining, we model a user's daily life as *life documents*, from which his/her life styles are extracted by using the Latent Dirichlet Allocation algorithm. We further propose a similarity metric to measure the similarity of life styles between users, and calculate users' impact in terms of life styles with a *friend-matching graph*. Upon receiving a request, Friendbook returns a list of people with highest recommendation scores to the query user. Finally, Friendbook integrates a feedback mechanism to further improve the recommendation accuracy. We have implemented Friendbook on the Android-based smartphones, and evaluated its performance on both small-scale experiments and large-scale simulations. The results show that the recommendations accurately reflect the preferences of users in choosing friends.

Index Terms—Friend recommendation, mobile sensing, social networks, life style

1 Introduction

Twenty years ago, people typically made friends with others who live or work close to themselves, such as neighbors or colleagues. We call friends made through this traditional fashion as G-friends, which stands for geographical location-based friends because they are influenced by the geographical distances between each other. With the rapid advances in social networks, services such as Facebook, Twitter and Google+ have provided us revolutionary ways of making friends. According to Facebook statistics, a user has an average of 130 friends, perhaps larger than any other time in history [2].

One challenge with existing social networking services is how to recommend a good friend to a user. Most of them rely on pre-existing user relationships to pick friend candidates. For example, Facebook relies on a social link analysis among those who already share common friends and recommends symmetrical users as potential friends. Unfortunately, this approach may not be the most appropriate based on recent sociology findings [16], [27], [29], [30]. According to these studies, the rules to group people together include: 1) habits or life style; 2) attitudes; 3) tastes; 4) moral standards; 5) economic level; and 6) people they already know. Apparently, rule #3 and rule #6 are the mainstream factors considered by existing recommendation systems. Rule #1, although probably the most intuitive, is not widely used because users' life styles are difficult, if not impossible, to capture through web actions. Rather, life styles are usually closely correlated with daily routines and activities. Therefore, if we could gather information on users' daily routines and activities, we can exploit rule #1 and recommend friends to people based on their similar life styles. This recommendation mechanism can be deployed as a standalone app on smartphones or as an add-on to existing social network frameworks. In both cases, Friendbook can help mobile phone users find friends either among strangers or within a certain group as long as they share similar life styles.

In our everyday lives, we may have hundreds of activities, which form meaningful sequences that shape our lives. In this paper, we use the word *activity* to specifically refer to the actions taken in the order of seconds, such as "sitting", "walking", or "typing", while we use the phrase *life style* to refer to higher-level abstractions of daily lives, such as "office work" or "shopping". For instance, the "shopping" life style mostly consists of the "walking" activity, but may also contain the "standing" or the "sitting" activities.

To model daily lives properly, we draw an analogy between people's daily lives and documents, as shown in Figure 1. Previous research on probabilistic topic models in text mining has treated documents as mixtures of

- *Zhibo Wang is with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, USA, 37909, and the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou, P.R.China, 310027. E-mail: zwang32@utk.edu*
- *Jilong Liao, Qing Cao and Hairong Qi are with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, USA, 37909. E-mail: {jliao2, cao, hqi}@utk.edu*
- *Zhi Wang is with the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou, P.R.China, 310027. E-mail: wangzhi@iipc.zju.edu.cn*

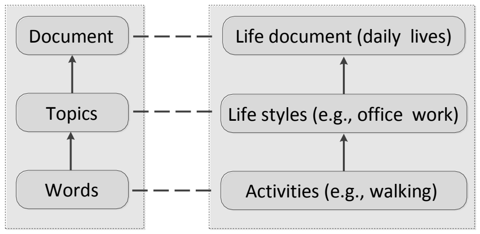

Document
Topics
Words
Life document (daily lives)
Life styles (e.g., office work)
Activities (e.g., walking)

Fig. 1: An analogy between word documents and people's daily lives.

topics, and topics as mixtures of words [10]. Inspired by this, similarly, we can treat our daily lives (or life documents) as a mixture of life styles (or topics), and each life style as a mixture of activities (or words). Observe here, essentially, we represent daily lives with "life documents", whose semantic meanings are reflected through their topics, which are life styles in our study. Just like words serve as the basis of documents, people's activities naturally serve as the primitive *vocabulary* of these life documents.

Our proposed solution is also motivated by the recent advances in smartphones, which have become more and more popular in people's lives. These smartphones (e.g., iPhone or Android-based smartphones) are equipped with a rich set of embedded sensors, such as GPS, accelerometer, microphone, gyroscope, and camera. Thus, a smartphone is no longer simply a communication device, but also a powerful and environmental reality sensing platform from which we can extract rich context and content-aware information. From this perspective, smartphones serve as the ideal platform for sensing daily routines from which people's life styles could be discovered.

In spite of the powerful sensing capabilities of smartphones, there are still multiple challenges for extracting users' life styles and recommending potential friends based on their similarities. First, how to automatically and accurately discover life styles from noisy and heterogeneous sensor data? Second, how to measure the similarity of users in terms of life styles? Third, who should be recommended to the user among all the friend candidates? To address these challenges, in this paper, we present Friendbook, a semantic-based friend recommendation system based on sensor-rich smartphones. The contributions of this work are summarized as follows:

- To the best of our knowledge, Friendbook is the first friend recommendation system exploiting a user's life style information discovered from smartphone sensors.
- Inspired by achievements in the field of text mining, we model the daily lives of users as *life documents* and use the probabilistic topic model to extract life style information of users.
- We propose a unique similarity metric to characterize the similarity of users in terms of life styles and then construct a friend-matching graph to recommend friends to users based on their life styles.
- We integrate a linear feedback mechanism that exploits the user's feedback to improve recommendation accuracy.
- We conduct both small-scale experiments and large-scale simulations to evaluate the performance of our system. Experimental results demonstrate the effectiveness of our system.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 provides the high-level overview of Friendbook. Section 4 presents activity recognition and life style modeling and extraction. In Section 5, we describe the social graph construction and user impact estimation. We elaborate on the user query and friend recommendation in Section 6. We describe the feedback mechanism in Section 7. In Section 8, we evaluate the performance of Friendbook intensively with both simulations and real experiments. Finally, we conclude the paper and present the future work in Section 9.

2 Related Work

Recommendation systems that try to suggest items (e.g., music, movie, and books) to users have become more and more popular in recent years. For instance, Amazon [1] recommends items to a user based on items the user previously visited, and items that other users are looking at. Netflix [3] and Rotten Tomatoes [4] recommend movies to a user based on the user's previous ratings and watching habits. Recently, with the advance of social networking systems, friend recommendation has received a lot of attention. Generally speaking, existing friend recommendation in social networking systems, e.g., Facebook, LinkedIn and Twitter, recommend friends to users if, according to their social relations, they share common friends.

Meanwhile, other recommendation mechanisms have also been proposed by researchers. For example, Bian and Holtzman [8] presented MatchMaker, a collaborative filtering friend recommendation system based on personality matching. Kwon and Kim [20] proposed a friend recommendation method using physical and social context. However, the authors did not explain what the physical and social context is and how to obtain the information. Yu et al. [32] recommended geographically related friends in social network by combining GPS information and social network structure. Hsu et al. [18] studied the problem of link recommendation in weblogs and similar social networks, and proposed an approach based on collaborative recommendation using the link structure of a social network and content-based recommendation using mutual declared interests. Gou et al. [17] proposed a visual system, SFViz, to support users to explore and find friends interactively under the context of interest, and reported a case study using the system to explore the recommendation of friends based on people's tagging behaviors in a music community. These existing friend recommendation systems, however, are significantly different from our work, as we exploit recent sociology findings to recommend friends based on their similar life styles instead of social relations.

Activity recognition serves as the basis for extracting high-level daily routines (in close correlation with life styles) from low-level sensor data, which has been widely studied using various types of wearable sensors. Zheng et al. [33] used GPS data to understand the transportation mode of users. Lester et al. [21] used data from wearable sensors to recognize activities based on the Hidden Markov Model (HMM). Li et al. [22] recognized static postures and dynamic transitions by using accelerometers and gyroscopes. The advance of smartphones enables activity recognition using the rich set of sensors on the smartphones. Reddy et al. [26] used the built-in GPS and the accelerometer on the smartphones to detect the transportation mode of an individual. CenceMe [24] used multiple sensors on the smartphone to capture user's activities, state, habits and surroundings. SoundSense [23] used the microphone on the smartphone to recognize general sound types (e.g., music, voice) and discover user specific sound events. EasyTracker [7] used GPS traces collected from smartphones that are installed on transit vehicles to determine routes served, locate stops, and infer schedules.

Although a lot of work has been done for activity recognition using smartphones, there is relatively little work on discovery of daily routines using smartphones. The MIT Reality Mining project [12] and Farrahi and Gatica-Perez [14] tried to discover daily location-driven routines from large-scale location data. They could infer daily routines such as leaving from home to office and eating at a restaurant. However, they could not discover the daily routines of people who are staying at the same location. For instance, when one stays at home, his/her daily routines like "eating lunch" and "watching movie" could not be discovered if only using the location information. In [13], Farrahi and Gatica-Perez took a step further and overcame the short-coming of discovering daily routines of people staying in the same location by considering combined location and physical proximity sensed by the mobile phone. Another closely related work was presented in [19], which used a topic model to extract activity patterns from sensor data. However, they used two wearable sensors, but not smartphones, to discover the daily routines. In our work, we attempt to use the probabilistic topic model to discover life styles using the smartphone. We further utilize patterns discovered from activities as a basis for friend recommendation that helps users find friends who have similar life styles. Note that the work in this paper is significantly different from our preliminary demo work of Friendbook [31] that recommended friends to users based on the similarity of pictures taken by users.

3 System Overview

In this section, we give a high-level overview of the Friendbook system. Figure 2 shows the system architecture of Friendbook which adopts a client-server mode where each client is a smartphone carried by a user and the servers are data centers or clouds.

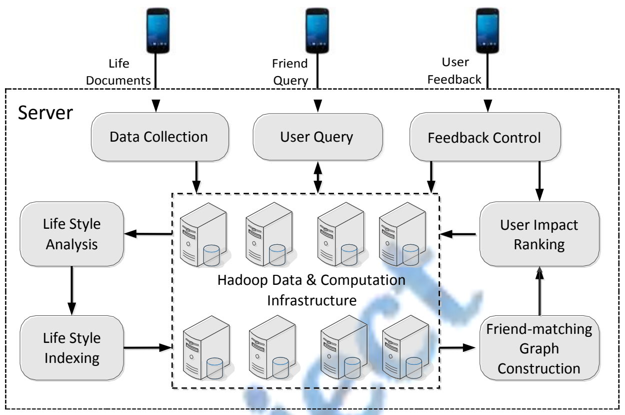


Fig. 2: System architecture of Friendbook.

On the client side, each smartphone can record data of its user, perform real-time activity recognition and report the generated life documents to the servers. It is worth noting that an offline data collection and training phase is needed to build an appropriate activity classifier for real-time activity recognition on smartphones. We spent three months on collecting raw data of 8 volunteers for building a large training data set. As each user typically generates around 50MB of raw data each day, we choose MySQL as our low level data storage platform and Hadoop MapReduce as our computation infrastructure. After the activity classifier is built, it will be distributed to each user's smartphone and then activity recognition can be performed in real-time manner. As a user continually uses Friendbook, he/she will accumulate more and more activities in his/her life documents, based on which, we can discover his/her life styles using probabilistic topic model.

On the server side, seven modules are designed to fulfill the task of friend recommendation. The *data collection* module collects life documents from users' smartphones. The life styles of users are extracted by the *life style analysis* module with the probabilistic topic model. Then the *life style indexing* module puts the life styles of users into the database in the format of (*life-style, user*) instead of (*user, life-style*). A friend-matching graph can be constructed accordingly by the *friend-matching graph construction* module to represent the similarity relationship between users' life styles. The impacts of users are then calculated based on the friend-matching graph by the *user impact ranking* module. The *user query* module takes a user's query and sends a ranked list of potential friends to the user as response. The system also allows users to give feedback of the recommendation results which can be processed by the *feedback control* module. With this module, the accuracy of friend recommendation can be improved.

In the following sections, we will elaborate on all the components of the system.

4 Life Style Extraction using Topic Model

Life Style Modeling

As stated in Section 1, *life styles* and *activities* are reflections of daily lives at two different levels where daily lives can be treated as a mixture of life styles and life styles as a mixture of activities. This is analogous to the treatment of documents as ensemble of topics and topics as ensemble of words. By taking advantage of recent developments in the field of text mining, we model the daily lives of users as *life documents*, the life styles as *topics*, and the activities as *words*.

Given "documents", the probabilistic topic model could discover the probabilities of underlying "topics". Therefore, we adopt the probabilistic topic model to discover the probabilities of hidden "life styles" from the "life documents". In probabilistic topic models, the frequency of vocabulary is particularly important, as different frequency of words denotes their information entropy variances. Following this observation, we propose the "*bag-of-activity*" model (Figure 3) to replace the original sequences of activities recognized based on the raw data with their probability distributions. Thereafter, each user has a bag-of-activity representation of his/her life document, which comprises a mixture of activity words.

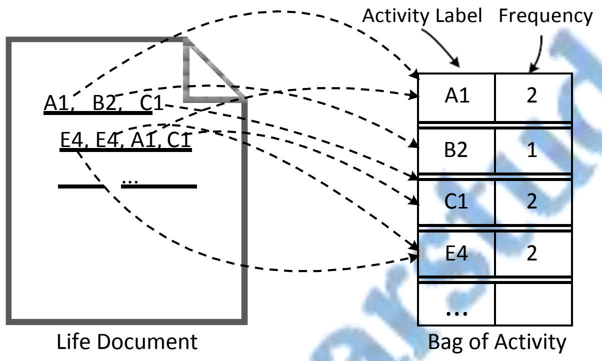


Fig. 3: Bag-of-Activity modeling for life document.

Let $\mathbf{w} = [w_1, w_2, ..., w_W]$ denote a set of activities, where w_i is the ith activity and W is the total number of activities. Let $\mathbf{z} = [z_1, z_2, ..., z_Z]$ denote a set of life styles, where z_i is the ith life style and Z is the total number of life styles. Let $\mathbf{d} = [d_1, d_2, ..., d_n]$ denote a set of life documents, where d_i is the ith life document and n is the total number of users. Let $p(w_i|d_k)$ denote the probability of the activity w_i in a certain life document d_k, $p(w_i|z_j)$ denote the probability of how much the activity w_i contributes to the life style z_j, and $p(z_j|d_k)$ denote the probability of the life style z_j embedded in the life document d_k. According to the probabilistic topic model, we have

$$p(w_i|d_k) = \sum_{j=1}^{Z} p(w_i|z_j)p(z_j|d_k) \tag{1}$$

Observe that $p(w_i|d_k)$ can be easily calculated by using the "bag-of-activity" representation for the life document d_k.

$$p(w_i|d_k) = \frac{f_k(w_i)}{\sum_{i=1}^{W} f_k(w_i)}$$

where $f_k(w_i)$ denotes the frequency of w_i in d_k.

We represent the life styles of a user using the *life style vector*, denoted by $\mathbf{L}_k = [p(z_1|d_k), p(z_2|d_k), ..., p(z_Z|d_k)]$. In this paper, our objective is to discover the life style vector for each user given the life documents of all users. However, in Eq. 1, although $p(w_i|d_k)$ can be calculated easily, $p(w_i|z_j)$ and $p(z_j|d_k)$ are difficult to solve because of the hidden feature of life styles.

In the following sections, we first present the details of activity recognition used to calculate $p(w_i|d_k)$, then show how to use the Latent Dirichlet Allocation (LDA) decomposition algorithm to solve Eq. 1 so that we can obtain the life style vector of each user.

Activity Recognition

To derive $p(w_i|d_k)$, we need to first classify or recognize the activities of users. Life styles are usually reflected as a mixture of motion activities with different occurrence probability. Therefore, two motion sensors, accelerometer and gyroscope, are used to infer users' motion activities. Generally speaking, there are two mainstream approaches: supervised learning and unsupervised learning. For both approaches, mature techniques have been developed and tested. In practice, the number of activities involved in the analysis is unpredictable and it is difficult to collect a large set of ground truth data for each activity, which makes supervised learning algorithms unsuitable for our system. Therefore, we use unsupervised learning approaches to recognize activities. Here, we adopt the popular K-means clustering algorithm [9] to group data into clusters, where each cluster represents an activity. Note that activity recognition is not the main concern of our paper. Other more complicated clustering algorithms can certainly be used. We choose K-means for its simplicity and effectiveness.

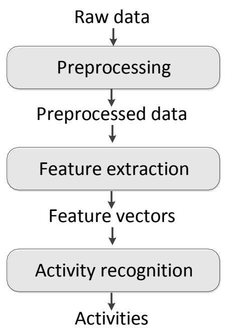


Fig. 4: The flowchart of activity recognition

Figure 4 shows the flowchart of activity recognition. Since the raw data collected on the smartphones are noisy, we first use a median filter [5] with sliding windows to filter out the outliers of the noisy data. In order to further improve recognition accuracy, features are extracted to characterize the data after preprocessing. We have tested several features, such as mean, standard deviation, correlation, and the combination of them, on the data and found that standard deviation is the most representative feature for characterizing motion activities. Therefore, a feature vector $\mathbf{f} = [t_c, acc_x, acc_y, acc_z, gyr_x, gyr_y, gyr_z]$ is used to characterize user's activities instead of the raw data, where t_c is the mean of normalized time; acc_x, acc_y and acc_z represent the standard deviation of normalized data obtained from the accelerometer in the x, y and z directions, respectively; gyr_x, gyr_y and gyr_z represent the standard deviation of normalized data obtained from the gyroscope in the x, y and z directions, respectively. We then apply the K-means clustering algorithm on the feature vectors to group them into different clusters as well as calculating the cluster centroids. Our empirical study in Section 8.1.1 indicates that $K = 15$ is a good compromise between classification accuracy and computational time. The cluster centroids are then distributed to the smartphones. Then each smartphone could independently recognize activities based on the minimum distance rule and upload the activity sequence instead of the raw data to the server.

Life Style Extraction using LDA

Given the life documents of all users, Eq. 1 can be further represented as a matrix decomposition problem.

$$p(\mathbf{w}|\mathbf{d}) = p(\mathbf{w}|\mathbf{z})p(\mathbf{z}|\mathbf{d}) \quad (3)$$

where $p(\mathbf{w}|\mathbf{d}) = [p(\mathbf{w}|d_1), p(\mathbf{w}|d_2), ..., p(\mathbf{w}|d_n)]$ is the activity-document matrix as shown in Figure 5 containing the probability of each activity over each life document, and $p(\mathbf{w}|d_k) = [p(w_1|d_k), p(w_2|d_k), ..., p(w_W|d_k)]^T$ is the kth column in the activity-document matrix representing the probabilities of activities over the life document d_k of user k; $p(\mathbf{w}|\mathbf{z}) = [p(\mathbf{w}|z_1), p(\mathbf{w}|z_2), ..., p(\mathbf{w}|z_Z)]$ is the activity-topic matrix as shown in Figure 5 representing the probability of each activity over each life style (topic), and $p(\mathbf{w}|z_k) = [p(w_1|z_k), p(w_2|z_k), ..., p(w_W|z_k)]^T$ is the kth column in the activity-topic matrix representing the probabilities of activities over the life style z_k; $p(\mathbf{z}|\mathbf{d}) = [p(\mathbf{z}|d_1), p(\mathbf{z}|d_2), ..., p(\mathbf{z}|d_n)]$ is the topic-document matrix as shown in Figure 5 containing the probability of each topic over each life document, and $p(\mathbf{z}|d_k) = [p(z_1|d_k), p(z_2|d_k), ..., p(z_Z|d_k)]^T$ is the kth column in the topic-document matrix representing the probabilities of life styles over the life document d_k of user k.

The above matrix decomposition problem is actually the Latent Dirichlet Allocation (LDA) model [10]. We use the *Expectation-Maximization* (EM) method to solve the LDA decomposition, where the E-step is used to estimate the free variational Dirichlet parameter γ [15]

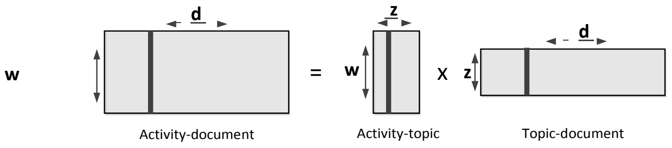


Fig. 5: Matrix decomposition for life styles analysis. (Redrawn from [19])

and multinomial parameter Φ in the standard LDA model [10] and the M-step is used to maximize the log likelihood of the activities under these parameters. After the EM algorithm converges, we are able to calculate the decomposed activity-topic matrix. Readers are referred to [10] for more details of the LDA algorithm and alternative decomposition approaches. It is worth noting that the matrix decomposition process can be implemented more efficiently through incremental iteration. That is, when a user's life document changes or a new user's life document is uploaded to the system, Friendbook can calculate the new life style vectors for each user based on previously derived life style vectors and the new life document. We did not implement the incremental iteration in the current framework by simply replying on the computing power of cloud computing. As part of our future work, we could add this implementation to the framework to make Friendbook scalable to large-scale systems.

It is also worth noting that since our system uses unsupervised learning algorithms to recognize activities and the topic model to discover life styles, the physical meanings of derived "activities" (or cluster centers from the K-means algorithm) or "topics" are unknown to us. As mentioned in [19], such meaning can be estimated via the additional step of comparing the topic activations to the actual structure of the subject's day and then identifying topics that correspond to possible daily routines. In Friendbook, since we are to only compare "similarity" in activities or topic patterns, there is no need to infer the physical meaning of each cluster center or topic. On the other hand, not revealing the actual physical meaning of activities and topics also has advantages from the perspective of preserving privacy.

5 Friend-Matching Graph and User Impact

To characterize relations among users, in this section, we propose the *friend-matching* graph to represent the similarity between their life styles and how they influence other people in the graph. In particular, we use the link weight between two users to represent the similarity of their life styles. Based on the friend-matching graph, we can obtain a user's affinity reflecting how likely this user will be chosen as another user's friend in the network.

Similarity Metric

We define a new similarity metric to measure the similarity between two life style vectors.

Let $\mathbf{L}_i = [p(z_1|d_i), p(z_2|d_i), ..., p(z_Z|d_i)]$ and $\mathbf{L}_j = [p(z_1|d_j), p(z_2|d_j), ..., p(z_Z|d_j)]$ denote the life style vectors of user i and user j, respectively.

We argue that the similarity is not only affected by their life style vectors as a whole, but also by the most important life styles, i.e., the elements within the vector with larger probability values, also known as the dominant life styles. We also argue that two users do not share much similarity if majority of their life styles are totally different. Therefore, the similarity of lifestyles between user i and user j, denoted by $S(i, j)$, is defined as follows:

$$S(i, j) = S_c(i, j) \cdot S_d(i, j) \tag{4}$$

where $S_c(i, j)$ is used to measure the similarity of the life style vectors of users as a whole, $S_d(i, j)$ is used to emphasize the similarity of users on their dominant life styles.

We adopt the commonly used *cosine* similarity metric for $S_c(i, j)$, that is,

$$S_c(i,j) = \cos(\mathbf{L}_i, \mathbf{L}_j) \tag{5}$$

In order to calculate $S_d(i, j)$, we first define the set of dominant life styles of a user.

Definition 1. ***Dominant life styles***: *The set of dominant life styles of a user i, D_i, is a subset of all the life styles satisfying the following requirements:*

1) *The total probability distribution of the set is larger than or equal to λ which is a predefined threshold.*
2) *The probability distribution of any life style in the set is larger than or equal to that of any life style not in the set.*
3) *The set should have the minimum number of life styles.*

These requirements guarantee that life styles with larger probabilities are more probably to be included in the set. To find D_i, we sort the life style vector $\mathbf{L}_i$ in the descending order with respect to the probabilities of life styles. Then we have $\tilde{\mathbf{L}}_i = [p(z_{i1}|d_i), p(z_{i2}|d_i), \cdots, p(z_{iZ}|d_i)]$ where $p(z_{is}|d_i) \geq p(z_{it}|d_i)$ if $s \leq t$. The size of dominant life style set is calculated as

$$q_i = \arg\min_q \left(\sum_{k=1}^{q} p(z_{ik}|d_i) \geq \lambda\right) \tag{6}$$

Finally, we can obtain the dominant life style set $D_i = \{z_{i1}, \cdots, z_{iq_i}\}$.

The similarity metric $S_d(i,j)$ for measuring the similarity of the dominant life style sets of two users is then defined as

$$S_d(i, j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} \tag{7}$$

The range of $S_d(i, j)$ is [0, 1]. Observe that the higher the percentage of the same life styles, the larger the similarity. When there is no overlap between D_i and D_j, the similarity is 0. When D_i and D_j are the same, the similarity is 1. Since both $S_c(i, j)$ and $S_d(i, j)$ vary between 0 and 1, we conclude that the similarity metric $S(i, j)$ varies between 0 and 1.

As an example to show the calculation of two users' life style similarity, we assume that there are two users 1 and 2 in the system, who have the life style vectors $\mathbf{L}_1$ = [0.3, 0.1, 0.2, 0.3, 0.1] and $\mathbf{L}_2$ = [0.2, 0.1, 0.4, 0, 0.3], respectively. The number of life style topics is 5. We first calculate $S_c(1, 2) = \cos(\mathbf{L}_1, \mathbf{L}_2) = 0.6708$. Given λ = 0.8, we can calculate the dominant life style sets of these two users, $D_1 = \{z_1, z_4, z_3\}$ and $D_2 = \{z_3, z_5, z_1\}$, respectively. Therefore, the dominant life style similarity is calculated as $S_d(1, 2) = \frac{2\times2}{3+3} = 0.67$. Finally, the similarity of user 1 and 2 is $S(1, 2) = S_c(1, 2) \cdot S_d(1, 2) = 0.45$.

Friend-matching Graph Construction

Based on the similarity metric, we model the relations between users in real life as a friend-matching graph.

Definition 2. ***Friend-matching graph***: *It is a weighted undirected graph $\mathcal{G} = (V, E, W)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of users and n is the number of users, $E = \{e(i, j)\}$ is the set of links between users, and $W : E \rightarrow \mathbb{R}$ is the set of weights of edges. There is an edge $e(i, j)$ linking user i and user j if and only if their similarity $S(i, j) \geq S_{thr}$, where S_{thr} is the predefined similarity threshold. The weight of that edge is represented by the similarity, that is, $\omega(i, j) = S(i, j)$.*

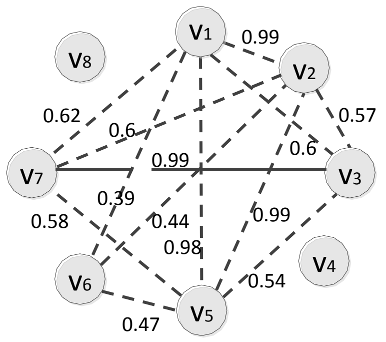


Fig. 6: An example of Friend-matching Graph for 8 users.

Figure 6 demonstrates a friend-matching graph based on the life styles of 8 users and the similarity threshold S_{thr} is set to 0.3. An edge linking two users means they have similar life styles (e.g., $e(1, 7)$), and their similarity is quantified by the weight of the edge (e.g., $\omega(1, 7)$ = 0.62). Some isolated vertices mean that they do not share enough similar life styles with others (e.g., user 4). We use the following representation to convert the graph into a matrix representation.

$$\mathbf{N} = (N_{ij})_{n\times n} = \begin{pmatrix} 0 & \omega(1,2) & \cdots & \omega(1,n) \\ \omega(2,1) & 0 & \cdots & \omega(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ \omega(n,1) & \omega(n,2) & \cdots & 0 \end{pmatrix} \tag{8}$$

Note that the values on the diagonal are all 0 because we would not recommend himself/herself to a user.

User Impact Ranking

The friend-matching graph has been constructed to reflect life style relations among users. However, we still lack a measurement to identify the impact ranking of a user quantitatively. Intuitively, the impact ranking means a user's capability to establish friendships in the network. In other words, the higher the ranking, the easier the user can be made friends with, because he/she shares broader life styles with others. Inspired by PageRank [25] which is used in web page ranking, we form the idea that a user's ranking is reflected by his neighbors in the friend-matching graph and how much his neighbors endorse the user as a friend.

Once the ranking of a user is obtained, it provides guidelines to those who receive the recommendation list on how to choose friends. The ranking itself, however, should be independent from the query user. In other words, the ranking depends only on the graph structure of the friend-matching graph, which contains two aspects: 1) how the edges are connected; 2) how much weight there is on every edge. Moreover, the ranking should be used together with the similarity scores between the query user and the potential friend candidates, so that the recommended friends are those who not only share sufficient similarity with the query user, and are also popular ones through whom the query user can increase their own impact rankings.

Let $N(i)$ denote the set of neighbors of user i. Let $\mathbf{r} = [r(1), r(2), \cdots, r(n)]^T$ denote the impact ranking vector where $r(i)$ is the impact ranking of user i in the friend-matching graph, and n is the number of users in the system. The calculation of $r(i)$ is defined as follows:

$$r(i) = \frac{\sum_{j \in N(i)} \omega(i,j) \cdot r(j)}{\sum_{j \in N(i)} \omega(i,j)} \tag{9}$$

As shown in Eq. 9, the impact ranking of a user is affected by its neighbors from two aspects: first, the similarity between itself and a neighbor; second, the impact ranking of its neighbors. Note that in friend-matching graph, $\omega(i,j) = 0$ if j is not a neighbor of i. Also $\omega(i,i) = 0$ because we would not recommend himself/herself to a user. Therefore, Eq. 9 can be rewritten as follows.

$$r(i) = \frac{\sum_j \omega(i,j) \cdot r(j)}{\sum_j \omega(i,j)} \tag{10}$$

The calculation of $r(i)$ is an iterative process because any change of its neighbors will change $r(i)$ accordingly. Therefore, we use a matrix representation to clearly show the iterative process of Eq. 10.

$$\mathbf{r}_{k+1}^T = \mathbf{r}_k^T \cdot \mathbf{H} \tag{11}$$

where k and $k+1$ indicate two subsequent iteration steps, and $\mathbf{H} = (H_{ij})_{n\times n}$ is the transitional matrix representing the similarity of neighbors. Combining Eq. 8 and Eq. 10, an element H_{ij} in $\mathbf{H}$ is calculated as follows:

$$H_{ij} = \frac{\omega(i,j)}{\sum_j \omega(i,j)} = \frac{N_{ij}}{\sum_j N_{ij}} \tag{12}$$

In practice, it is possible that people choose friends randomly rather than based on their importance. Therefore, we revise the transitional matrix by introducing a matrix with equal value. Then the transitional matrix is modified to Eq. 13.

$$\tilde{\mathbf{H}} = \phi\mathbf{H} + (1 - \phi)\frac{1}{n}\mathbf{e}\mathbf{e}^T \tag{13}$$

where $\mathbf{e}$ is the $n \times 1$ unit vector and ϕ is the damping factor used to emphasize the importance of the friend-matching graph.

Finally, the iterative process for calculating the impact rank of users is carried out as follows.

$$\mathbf{r}_{k+1}^T = \mathbf{r}_k^T \cdot \tilde{\mathbf{H}} \tag{14}$$

The process terminates when the impact ranking vector converges to a stable value. In our experiment, the process terminates when $\sum_{i=1}^{n} |\mathbf{r}_{k+1}(i) - \mathbf{r}_k(i)| \leq \varepsilon$ where ε is an arbitrary small value larger than 0. The pseudocode for the iterative process is shown in Algorithm 1. After the impact vector $\mathbf{r}$ is calculated, we can rank the users in a non-descending order so that a user with higher impact ranking is always ahead of a user with lower impact ranking.

In general, by using the Hadoop MapReduce framework, the impact ranking process can converge quickly. However, this is becoming increasing infeasible when the size of the system is becoming very large. Fortunately, accordingly to the incremental computation of PageRank [6], [11] and the distributed computation of PageRank [28], the iterative matrix-vector multiplication method in Eq. (14) can be implemented incrementally or distributively for large-scale evolving graphs. In [28], the authors presented a fast algorithm that takes $O(\sqrt{\log n/s})$ rounds in undirected graphs, where n is the network size and s is a fixed constant. Therefore, Friendbook is scalable to large-scale systems if we could implement the iterative matrix-vector multiplication method incrementally or distributively, which would be our future work.

6 Query and Friend Recommendation

Before a user initiates a request, he/she should have accumulated enough activities in his/her life documents for efficient life styles analysis. The period for collecting data usually takes at least one day. Longer time would be expected if the user wants to get more satisfied friend recommendation results. After receiving a user's request (e.g., life documents), the server would extract the user's life style vector, and based on which recommend friends to the user.

Algorithm 1 Computing users' impact ranking

```
Input: The friend-matching graph tt.
Output: Impact ranking vector r for all users.
 1: for $i = 1$ to $n$ do
 2:     $\mathbf{r}_0(i) = \frac{1}{n}$
 3: end for
 4: $\delta = \infty$
 5: $s = e^{-9}$
 6: while $\delta > s$ do
 7:     for $i = 1$ to $n$ do
 8:         $\mathbf{r}_{k+1}(i) = \sum_j \frac{1-\phi}{n}\mathbf{r}_k(j) + \phi \frac{\sum_j \omega(i,j)\cdot\mathbf{r}_k(j)}{\sum_j \omega(i,j)}$
 9:     end for
10:     $\delta = \sum_{i=1}^{n} |\mathbf{r}_{k+1}(i) - \mathbf{r}_k(i)|$
11: end while
12: return r
```

The recommendation results are highly dependent on users' preference. Some users may prefer the system to recommend users with high impact, while some users may want to know users with the most similar life styles. It is also possible that some users want the system to recommend users who have high impact and also similar life styles to them. To better characterize this requirement, we propose the following metric to facilitate the recommendation,

$$R_i(j) = \beta S(i,j) + (1-\beta)r_j\kappa \qquad (15)$$

where $R_i(j)$ is the recommendation score of user j for the query user i, $S(i,j)$ is the similarity between user i and user j, and r_j is the impact of user j. $\beta \in [0,1]$ is the recommendation coefficient characterizing users' preference. κ is introduced to make $S(i,j)$ and r_j in the same order of magnitude, which can be roughly set to $n/10$, where n is the number of users in the system. When $\beta = 1$, the recommendation is solely based on the similarity; when $\beta = 0$, the recommendation is solely based on the impact ranking.

With the metric in Eq. 15, our recommendation mechanism for finding the most appropriate friends to a query user is described as follows. For a query user i, the server calculates the recommendation scores for all the users in the system and sorts them in the descending order according to their recommendation scores. The top p users will be returned to the query user i. The parameter p is an integer and can be defined by the querying user. The complexity of our recommendation mechanism is $O(n)$ since it checks all users in the system, where n is the overall number of users in the system.

As the number of users increases, the overhead of query and recommendation increases linearly. In reality, users may have totally different life styles and it is not necessary to calculate their recommendation scores at all. Therefore, in order to speed up the query and recommendation process, we adopt the reverse index table using *(life-style, user)* pair instead of *(user, life-style)* pair in the database. Figure 7 shows the difference. With the

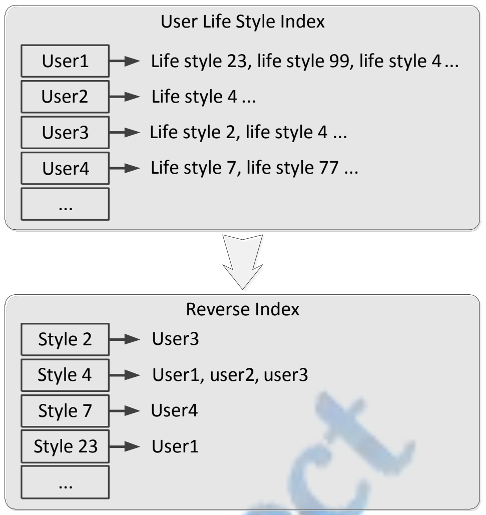


Fig. 7: Illustration of the reverse index table.

reverse index table, before calculating recommendation score for each user, the server first picks up all the users having overlapping life styles with the query user and sets the similarities of rest users to the query user to 0. The server then checks all the users to calculate their recommendation scores. Although the complexity is still $O(n)$, we can observe that the reverse index table reduces the computation overhead, the advantage of which is considerable when the system is in large-scale.

The pseudocode of the friend recommendation mechanism is shown in Algorithm 2.

Algorithm 2 Friend recommendation

```
Input: The query user $i$, the recommendation coefficient
    $\beta$ and the required number of recommended friends
    from the system $p$.
Output: Friend list $F_i$.
 1: $F_i \leftarrow \emptyset$, $Q \leftarrow \emptyset$
 2: extracts $i$'s life style vector $\mathbf{L}_i$ using the LDA algo-
    rithm.
 3: for each life style $z_k$ the probability of which in $\mathbf{L}_i$
    is not zero do
 4:     put users in the entry of $z_k$ into $Q$
 5: end for
 6: for each user $j \notin Q$ do
 7:     $S(i,j) \leftarrow 0$
 8: end for
 9: for each user $j$ in the database do
10:     $R_i(j) = \beta S(i,j) + (1-\beta)r_j\kappa$
11: end for
12: sort all users in decreasing order according to $R_i(j)$
13: put the top $p$ users in the sorted list to $F_i$
```

Friendbook also uses GPS location information to help users find friends within some distance. In order to protect the privacy of users, a region surrounding the accurate location will be uploaded to the system. When a user uses Friendbook, he/she can specify the distance of friends before recommendation. In this way, only friends having similarity with the user within the specified distance can be recommended as friends.

Privacy is very important especially for users who are sensitive to information leakage. In our design of Friendbook, we also considered the privacy issue and the existing system can provide two levels of privacy protection. First, Friendbook protects users' privacy at the data level. Instead of uploading raw data to the servers, Friendbook processes raw data and classifies them into activities in real-time. The recognized activities are labeled by integers. In this way, even if the documents containing the integers are compromised, they cannot tell the physical meaning of the documents. Second, Friendbook protects users' privacy at the life pattern level. Instead of telling the similar life styles of users, Friendbook only shows the recommendation scores of the recommended friends with the users. With the recommendation score, it is almost impossible to infer the life styles of recommended friends.

7 Feedback Control

To support performance optimization at runtime, we also integrate a feedback control mechanism into Friendbook. After the server generates a reply in response to a query, the feedback mechanism allows us to measure the satisfaction of users, by providing a user interface that allows the user to rate the friend list. Let $\acute{\mathbf{r}}$ denote the impact ranking vector calculated from the feedback of users. Here, $\acute{\mathbf{r}} = [\acute{r}(1), \acute{r}(2), \cdots, \acute{r}(n)]^T$ where n is the number of current users of the system. Let $\acute{r}(i,j)$ denote the score that user j rates user i. Then we have:

$$\acute{r}(i,j) = \begin{cases} \acute{r}(i,j) & \text{if user } j \text{ rates user } i, \\ r(i) & \text{otherwise.} \end{cases} \quad (16)$$

where the second equation in Eq. 16 means that the feedback score is equal to the original score if user j does not rate user i. This may commonly occur when the user does not know the persons being recommended especially when our system becomes very large.

Based on Eq. 16, we define the impact ranking of user i influenced by the feedback of users as follows:

$$\acute{r}(i) = \sum_j \acute{r}(i,j)/n. \quad (17)$$

which takes the feedback of all users into consideration.

Finally, the original impact ranking vector R calculated from the friend-matching graph is updated as follows:

$$\mathbf{r} = \alpha\mathbf{r} + (1-\alpha)\acute{\mathbf{r}} \quad (18)$$

where α is named as confidence factor and $0 \le \alpha \le 1$. The final impact ranking vector considers both the influence of friend-matching graph and the feedback from users. When $\alpha > 0.5$, the friend-matching graph dominates the impact ranking, however, when $\alpha < 0.5$, users' feedback will significantly affect the impact ranking. In this way, the system takes users' feedback into consideration to improve the accuracy of future recommendations.

8 Evaluation

In this section, we present the performance evaluation of Friendbook on both small-scale field experiments and large-scale simulations.

Evaluation using Real Data

We first evaluate the performance of Friendbook on small-scale experiments. Eight volunteers help contribute data and evaluate our system. Table 1 demonstrates the profession of these users. Most of them are students, while the rest include a businessman, an office worker, and a waitress. Each volunteer carries a Nexus S smartphone with Friendbook application installed in advance.

They are required to start the application after they wake up and turn it off before they go to bed. Aside from this, we do not impose any additional requirement on the usage of the smartphone. For example, we do not require them to carry the smartphone all the time during the day or attach the smartphone to some special parts of the body.

It is worth noting that some of the eight users are already friends before experiments but some of them are not. In fact, some strangers within the group become friends afterwards. However, strangers living far away from each other do not become friends although they choose each other as a friend at the friend recommendation phase. This also motivates the usage of GPS information into the system to improve the recommendation accuracy.

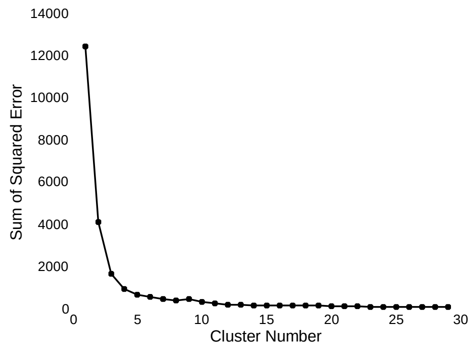
TABLE 1: Profession of users

User ID	1	2	3	4	5	6	7	8
Student	√	√	√		√		√	
Waitress				√				
Office Worker						√		
Businessman								√

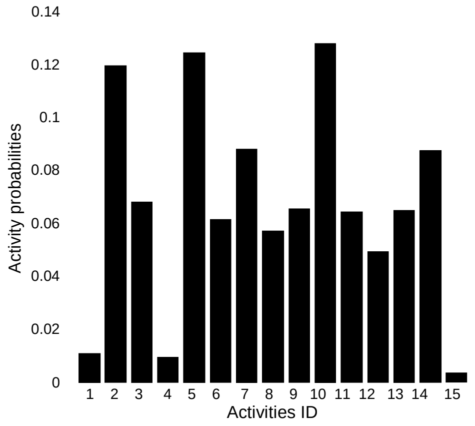
In the following, we first present the activity classification results using K-means clustering algorithm, and then show the performance evaluation on real experiments.

Activity Classification

Figure 8 shows the classification results using the K-means clustering algorithm on the data collected from the 8 users for a period of three months. Feature vectors instead of raw data are used for classification. Each feature vector consists of 7 attributes, $\mathbf{f} = [t_c, acc_x, acc_y, acc_z, gyr_x, gyr_y, gyr_z]$ (see Section 4.2), and we extract feature vectors every 60 seconds. As shown in Figure 8(a), the sum of squared error drops quickly when the cluster number K increases from 1 to 10, and



(a) Classification results



(b) Activity distribution

Fig. 8: Classification performance using the K-means clustering.

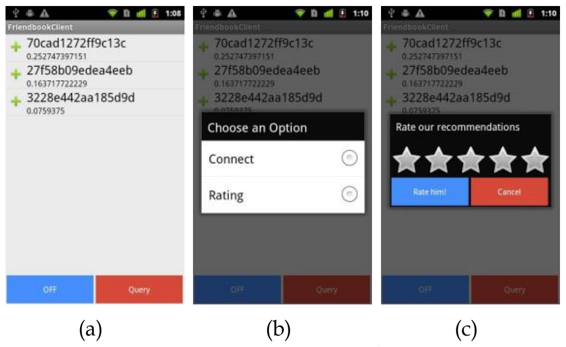


Fig. 9: User interfaces: (a) query-recommendation interface; (b) connection interface; (c) rating interface.

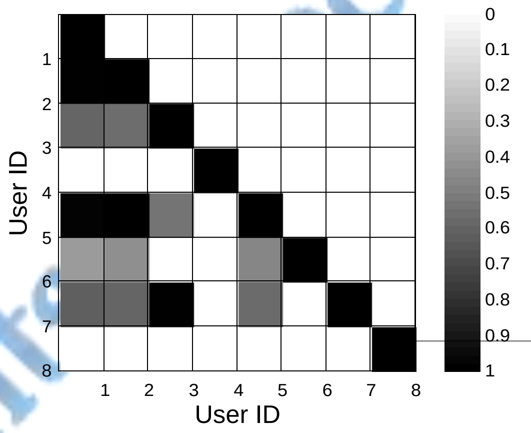


Fig. 10: The gray image representation of the eight users' similarity.

then does not change too much after 15. This implies that the daily lives of the volunteers are roughly composed of 15 different types of activities. Although we can use additional activities to characterize their daily lives at a finer scale, we find 15 an appropriate compromise as the most important activities are already involved. Other activities occur rarely and cannot considerably affect the friend recommendation results. Therefore, we use $K = 15$ as the number of activities in Friendbook.

Figure 8(b) demonstrates the distribution of 15 activities. Most of activities have the probability of about 6%, and three activities have the probability of larger than 10%, and the remaining three activities have the probability of less than 2%. Corresponding to each activity, a centroid feature vector is calculated by using the K-means clustering algorithm. These 15 centroids are distributed to each smartphone so that it can perform real-time onboard activity recognition.

Friend Recommendation Results

There are four free parameters used to generate the friend recommendation results, including the similarity threshold for friend-matching graph S_{thr} (Definition 2), the threshold λ (Eq. 6) that controls the number of dominant life styles, the damping factor ϕ that emphasizes the importance of the friend matching graph (Eq. 13), and the number of life styles. In our practical experiments, we have used the following values as default through empirical studies, i.e., the similarity threshold S_{thr} is set to 0.5, the threshold λ is set to 0.8, the damping factor ϕ is set to 0.85, and the number of life styles is set to 10.

Figure 9 shows several user interfaces of Friendbook. Figure 9(a) shows a snapshot of the query and recommendation user interface. The IDs and recommendation scores of recommended friends are shown in the list. Note that Friendbook returns the ID of users instead of their real names due to privacy concerns in our experiments. Figure 9(b) and 9(c) show the snapshots of user feedback interfaces. Users can connect to people in the recommended friend list through our system and also give a score on the recommended friends. Note that we intentionally anonymize the personal information in Figure 9 to protect the privacy of subjects. In the real system, when a user wants to use the system, he/she will be encouraged to complete his/her personal profile, e.g., name and photo. Therefore, the name and photo information as well as the similarity score of each recommended friend will be shown to the user.

Figure 10 illustrates the gray-scale image representation of the similarity matrix for 8 users. The blocks in the diagonal has the darkest color because users always have the perfect match with themselves. As shown in Figure 10, user 1 has strong relationship with user 2 and user 5, user 3 has strong relationship with user 7, user 6 has relationship with the aforementioned users but not very strong, while user 4 and user 8 have no relationship with others at all. The result is consistent with the ground truth of professions shown in Table 1 because people have the same profession usually have the same life styles.

TABLE 2: User impact ranking of 8 users.

Rank	User ID	Rank Score
1	1	0.133
2	7	0.127
3	4	0.125
4	8	0.125
5	5	0.124
6	2	0.123
7	6	0.123
8	3	0.118

The user impact rankings of the 8 users are shown in Table 2. The top ranks are users 1 and 7, followed by users 4 and 8 who seem to have high impact ranks. However, users 4 and 8 are not supposed to be higher than others because they have no connections. Indeed, because of this, they should always maintain the initial score. Since we only have 8 users in the system, each of whom uses $\frac{1}{8} = 0.125$ as its initial random impact, as described in Algorithm 1, which results in that their rankings are even higher than some of the connected users. If we have 1000 users, they should only have 0.001 for their final score, thus they would not have so high rank. Although the independent users are not avoidable, we only recommend a small portion of top p friends. The chance that an independent user ranks in top p is small when the total number of users is large. Therefore, the recommendation quality should not be affected much from independent users.

8.2 Evaluation using Simulated Data

We perform simulations to further evaluate performance of Friendbook when the scale of the system is large. Our friend recommendation method is based on life styles extracted from sensors on users' smartphones, which is quite different from existing friend recommendation methods. To the best of our knowledge, there is no real data set that can be used for a large-scale performance evaluation.

In our simulation, we randomly and independently generate the life style vectors for 1,000 users, $L_k = [p(z_1|d_k), p(z_2|d_k), \cdots, p(z_{10}|d_k)], k = 1, \cdots, 1000$. Note that since the life style vector contains the probability of each life style that sum to 1, each entry of the life style vector is randomly generated between 0 and 1, and normalized to guarantee the sum of the values in the vector is equal to 1. For each user, the similarities between itself and all the other users can be calculated based on the similarity metric in Eq. 4 and the 100 most similar users are chosen as its true friends, denoted as

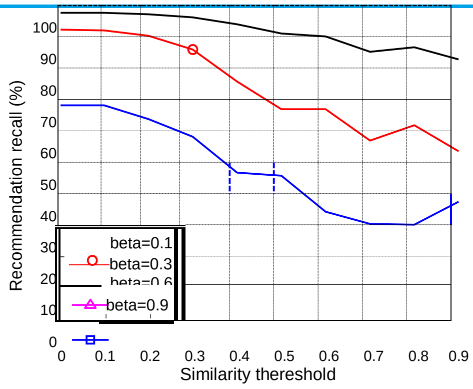


Fig. 11: Evaluation on similarity threshold. The number of returned recommended friends is 100.

tt_i for user i. After the life styles are uploaded to the system, a friend-matching graph can be constructed and each user has an impact. We then let each user query the system and obtain its friend recommendation results. Let F_i denote the set of recommended friends. The following measurement metrics are used for performance evaluation.

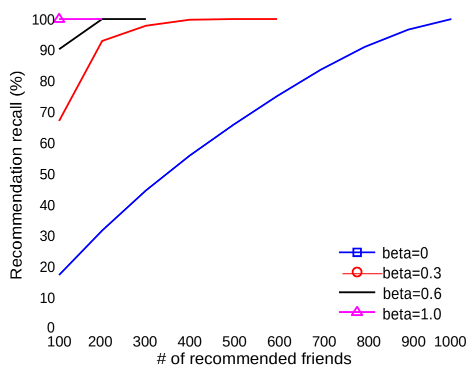
- *Recommendation precision* R_p: the average of the ratio of the number of recommended friends in the set of true friends of the query user over the total number of recommended friends.
$$R_p = \frac{\sum_i |F_i \cap tt_i|/|F_i|}{1000}$$
where $|\cdot|$ denotes the number of elements in a set·. The dominator is 1000 because R_p is the average of 1000 users in one experiment.
- *Recommendation recall* R_r: the average of the ratio of the number of recommended friends in the set of true friends of the query user over the number of the set of true friends of the query user, which is 100 in our experiments.

$$R_r = \frac{\sum_i |F_i \cap tt_i|/|tt_i|}{1000} = \frac{\sum_i |F_i \cap tt_i|/100}{1000}$$

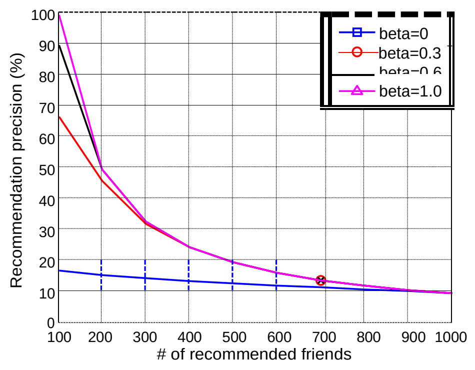
Using these performance metrics, we study the effect of a few parameters, such as the similarity threshold, S_{thr}, for friend-matching graph construction, and the recommendation coefficient, β, for balancing users' preference on impact and similarity. For all the simulation results presented in this paper, each data point is an average of 100 experiments.

8.2.1 *Effect of Similarity Threshold S_{thr}*

We first evaluate the effect of the similarity threshold S_{thr} which is important for friend-matching graph construction. When S_{thr} is too small, almost all users in the system are connected, which increases the overhead for graph construction and maintenance as well as the ranking process. While if S_{thr} is too large, the friend-matching graph cannot reflect the true relationships between users since those with high similarity may not be connected with each other. Therefore, the value of



(a) Recommendation recall



(a) Recommendation Precision

Fig. 12: Evaluation on recommendation coefficient β.

the similarity threshold for the friend-matching graph construction should be carefully selected.

Figure 11 shows the effect of the similarity threshold on the recommendation recall. We only consider the case that 100 friends are recommended. In this case, the recommendation precision equals the recommendation recall. We first observe that the recommendation recall drops when S_{thr} increases. The reason is that links with small similarities are gradually removed from the graph as S_{thr} increases. We also notice that the influence of S_{thr} is more obvious for smaller β. This is because removing links affects the impact rankings of users and meanwhile smaller β relies more on the impact rankings. In our following experiments, we choose $S_{thr} = 0.5$ when we evaluate other parameters.

8.2.2 *Effect of recommendation coefficient β*

β is introduced to characterize users' preference on impact and similarity. Figure 12(a) and 12(b) shows the evaluation results of β on recommendation recall and precision, respectively.

When $\beta = 0$, the recommendation recall is relatively low which is just a little better than random recommendation. This is because the recommendation only relies on the impact rankings of users rather than similarities between users with the query user. The recommendation recall increases as β increases because similarity is more and more emphasized. When β reaches 1, all friends in

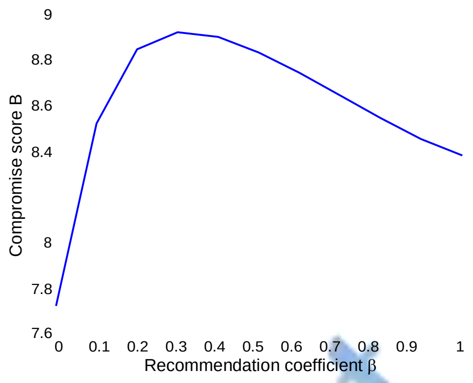


Fig. 13: The impact of β on B.

a query user's group are recommended. Note that even when β is not too large (e.g., $\beta = 0.3$), the recommendation recall is still high and reaches 100% quickly as the number of recommended friends increases. As shown in Figure 12, the recommendation precision decreases when the number of recommendation friends increases and finally reaches 10% when 1000 users are recommended. This is because we only choose 100 users as the set of true friends for each user. The recommendation precision also increases when β increases because similarity plays more and more important roles as β increases.

It is obvious that our recommendation is much better than random recommendation which is usually 10% on average. However, note that we cannot conclude that the larger β the better, as shown in Figure 12. Because users may prefer impact to similarity and our metrics cannot reflect the recommendation accuracy on impact. To better characterize the recommendation results, we define a new metric, called *compromise score*, as follows.

$$B = \sum_{j=1}^{100} S(i, i_j) r_{i_j}$$

where B is the compromise score and i_j is the jth recommended friend for the query user i. In this experiment, 100 friends are recommended for each query user. Figure 13 shows the impact of β on the metric. As we mentioned before, each data point is an average of 100 experiments. We can see that the metric achieves its maximum when β is around 0.3. Although the recommended friends have high impact when β is small, the similarity between them and the query user is low. In contrast, the similarity between the recommended friends and the query user is high, but their popularity is low. Therefore, in order to well balance the similarity and popularity, β should be carefully selected. Our system leaves the setting of β to users, so they can find friends based on their preferences.

Resource Consumption

Finally, we evaluate the energy consumption performance of the Friendbook client application. Usually, the user will not have the incentive to use the application if the battery runs out in less than 10 hours, which is

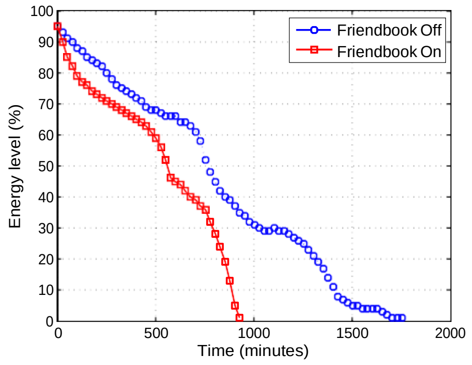


Fig. 14: Energy consumption comparison.

the typical hour of usage for a day (the user can recharge the battery during the night). Therefore, energy consumption is another important metric that has to be measured. We test the energy consumption of the same smartphone under two modes: idle mode with Friendbook off and active mode with Friendbook on. Either mode is under a user's normal use such as making phone calls, checking emails, sending SMS, etc. As shown in Figure 14, Friendbook drops the battery to 15% in about 13 hours. The evaluation shows that Friendbook achieves satisfactory results on the energy performance.

TABLE 3: Resources requirements comparison.

Aspect	Friendbook Client	Android Service	Google Maps
Size	76 KB	–	12 MB
CPU usage	<1%	13%	<1%
Data usage	<10KB	1.95MB	526KB
Runtime memory	4.2MB	33MB	6.9MB

In addition to energy, as shown in Table 3, we monitor the runtime resources that our client application consumes comparing to other Android system services and applications. It is easily observed that the client application is small in size while the runtime memory usage is comparable to some popular applications, such as Google Maps. In addition, the wireless data usage is low so that the user does not need to worry about the data plan of their mobile phone. Moreover, even when we turn on GPS, accelerometer and gyroscope sensors all the time, the total CPU usage of the client application is still very low on average.

9 Conclusion and Future Work

In this paper, we presented the design and implementation of Friendbook, a semantic-based friend recommendation system for social networks. Different from the friend recommendation mechanisms relying on social graphs in existing social networking services, Friendbook extracted life styles from user-centric data collected from sensors on the smartphone and recommended potential friends to users if they share similar life styles. We implemented Friendbook on the Android-based smartphones, and evaluated its performance on both small-scale experiments and large-scale simulations. The results showed that the recommendations accurately reflect the preferences of users in choosing friends.

Beyond the current prototype, the future work can be four-fold. First, we would like to evaluate our system on large-scale field experiments. Second, we intend to implement the life style extraction using LDA and the iterative matrix-vector multiplication method in user impact ranking incrementally, so that Friendbook would be scalable to large-scale systems. Third, the similarity threshold used for the friend-matching graph is fixed in our current prototype of Friendbook. It would be interesting to explore the adaption of the threshold for each edge and see whether it can better represent the similarity relationship on the friend-matching graph. At last, we plan to incorporate more sensors on the mobile phones into the system and also utilize the information from wearable equipments (e.g., Fitbit, iwatch, Google glass, Nike+, and Galaxy Gear) to discover more interesting and meaningful life styles. For example, we can incorporate the sensor data source from Fitbit, which extracts the user's daily fitness infograph, and the user's place of interests from GPS traces to generate an infograph of the user as a "document". From the infograph, one can easily visualize a user's life style which will make more sense on the recommendation. Actually, we expect to incorporate Friendbook into existing social services (e.g., Facebook, Twitter, LinkedIn) so that Friendbook can utilize more information for life discovery, which should improve the recommendation experience in the future.

Acknowledgements

The authors would like to thank the anonymous reviewers whose insightful comments have helped improve the presentation of this paper significantly. This work was supported in part by NSF CNS-1017156 and CNS-0953238, and in part by NSFC 61273079 and NSFC 11174316.

References

[1]Amazon. `http://www.amazon.com/`.
[2]Facebook statistics. `http://www.digitalbuzzblog.com/facebook-statistics-stats-facts-2011/`.
[3]Netfix. `https://signup.netflix.com/`.
[4]Rotten tomatoes. `http://www.rottentomatoes.com/`.
[5]G. R. Arce. Nonlinear Signal Processing: A Statistical Approach. *John Wiley & Sons*, 2005.
[6]B. Bahmani, A. Chowdhury, and A. Goel. Fast incremental and personalized pagerank. *Proc. of VLDB Endowment*, volume 4, pages 173-184, 2010.
[7]J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones. *Proc. of SenSys*, pages 68-81, 2011.
[8]L. Bian and H. Holtzman. Online friend recommendation through personality matching and collaborative filtering. *Proc. of UBICOMM*, pages 230-235, 2011.
[9]C. M. Bishop. Pattern recognition and machine learning. *Springer New York*, 2006.

[10]D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993-1022, 2003.
[11]P. Desikan, N. Pathak, J. Srivastava, and V. Kumar. Incremental page rank computation on evolving graphs. *Proc. of WWW*, pages 1094-1095, 2005.
[12]N. Eagle and A. S. Pentland. Reality Mining: Sensing Complex Cocial Systems. *Personal Ubiquitous Computing*, 10(4):255-268, March 2006.
[13]K. Farrahi and D. Gatica-Perez. Probabilistic mining of socio-geographic routines from mobile phone data. *Selected Topics in Signal Processing, IEEE Journal of*, 4(4):746-755, 2010.
[14]K. Farrahi and D. Gatica-Perez. Discovering Routines from Largescale Human Locations using Probabilistic Topic Models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(1), 2011.
[15]B. A. Frigyik, A. Kapila, and M. R. Gupta. Introduction to the dirichlet distribution and related processes. *Department of Electrical Engineering, University of Washignton, UWEETR-2010-0006*, 2010.
[16]A. Giddens. Modernity and Self-identity: Self and Society in the late Modern Age. *Stanford Univ Pr*, 1991.
[17]L. Gou, F. You, J. Guo, L. Wu, and X. L. Zhang. Sfviz: Interestbased friends exploration and recommendation in social networks. *Proc. of VINCI*, page 15, 2011.
[18]W. H. Hsu, A. King, M. Paradesi, T. Pydimarri, and T. Weninger. Collaborative and structural recommendation of friends using weblog-based social network analysis. *Proc. of AAAI Spring Symposium Series*, 2006.
[19]T. Huynh, M. Fritz, and B. Schiel. Discovery of Activity Patterns using Topic Models. *Proc. of UbiComp*, 2008.
[20]J. Kwon and S. Kim. Friend recommendation method using physical and social context. *International Journal of Computer Science and Network Security*, 10(11):116-120, 2010.
[21]J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford. A Hybrid Discriminative/Generative Approach for Modeling Human Activities. *Proc. of IJCAI*, pages 766-772, 2005.
[22]Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou. Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information. *Proc. of BSN*, pages 138-143, 2009.
[23]E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell. Darwin Phones: the Evolution of Sensing and Inference on Mobile Phones. *Proc. of MobiSys*, pages 5-20, 2010.
[24]E. Miluzzo, N. D. Lane, S. B. Eisenman, and A. T. Campbell. Cenceme-Injecting Sensing Presence into Social Networking Applications. *Proc. of EuroSSC*, pages 1-28, October 2007.
[25]L. Page, S. Brin, R. Motwani, and T. Winograd. The Pagerank Citation Ranking: Bringing Order to the Web. *Technical Report, Stanford InfoLab*, 1999.
[26]S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using Mobile Phones to Determine Transportation Modes. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):13, 2010.
[27]I. Ropke. The Dynamics of Willingness to Consume. *Ecological Economics*, 28(3):399-420, 1999.
[28]A. D. Sarma, A. R. Molla, G. Pandurangan, and E. Upfal. Fast distributed pagerank computation. *Springer Berlin Heidelberg*, pages 11-26, 2013.
[29]G. Spaargaren and B. Van Vliet. Lifestyles, Consumption and the Environment: The Ecological Modernization of Domestic Consumption. *Environmental Politics*, 9(1):50-76, 2000.
[30]M. Tomlinson. Lifestyle and Social Class. *European Sociological Review*, 19(1):97-111, 2003.
[31]Z. Wang, C. E. Taylor, Q. Cao, H. Qi, and Z. Wang. Demo: Friendbook: Privacy Preserving Friend Matching based on Shared Interests. *Proc. of ACM SenSys*, pages 397-398, 2011.
[32]X. Yu, A. Pan, L.-A. Tang, Z. Li, and J. Han. Geo-friends recommendation in gps-based cyber-physical social network. *Proc. of ASONAM*, pages 361-368, 2011.
[33]Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma. Understanding Transportation Modes Based on GPS Data for Web Applications. *ACM Transactions on the Web (TWEB)*, 4(1):1-36, 2010.

Zhibo Wang received the B.E. degree in Automation from Zhejiang University, China, in 2007. He is currently pursuing the Ph.D. degree in Electrical Engineering and Computer Science at University of Tennessee, Knoxville. His research interests include wireless sensor networks, mobile sensing systems and cyber physical systems. He is a student member of IEEE.

Jilong Liao received his B.E. degree from University of Electronic Science and Technology, China, in 2010. He received his Master of Science degree from University of Tennessee, Knoxville in 2013. His major research interests included mobile system, data analysis, distributed system and wireless sensor networks. He is currently working in Microsoft's operating system group (OSG) as a software development engineer.

Qing Cao received his Ph.D. degree from the University of Illinois in 2008, his M.S. degree from the University of Virginia, and his B.S. degree from Fudan University, China. He is currently an assistant professor in the Department of Electrical Engineering and Computer Science at the University of Tennessee. His research interests include wireless sensor networks, embedded systems, and distributed networks. He is a member of ACM, IEEE, and the IEEE Computer Society.

Hairong Qi received the B.S. and M.S. degrees in computer science from Northern JiaoTong University, Beijing, China in 1992 and 1995 respectively, and the Ph.D. degree in computer engineering from North Carolina State University, Raleigh, in 1999. She is currently a Professor with the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville. Her current research interests are in advanced imaging and collaborative processing in resource-constrained distributed environment, hyperspectral image analysis, and bioinformatics. Dr. Qi is the recipient of the NSF CAREER Award. She also received the Best Paper Award at the 18th International Conference on Pattern Recognition and the 3rd ACM/IEEE International Conference on Distributed Smart Cameras. She recently receives the Highest Impact Paper from the IEEE Geoscience and Remote Sensing Society.

Zhi Wang received the B.S. degree from Shenyang Jian Zhu University, China, in 1991, the M.S. degree from Southeast University, China, in 1997, and the Ph.D. degree from Shenyang Institute of Automation, the Chinese Academy of Sciences, China, in 2000. During 2000-2002as Post-doc, he has conducted research in Institut National Polytechique de Lorraine, France and Zhejiang University, China respectively. During 2010-2011, he has conducted research as advanced scholar at UW-Madion, USA. He is currently an Associate Professor in the Department of Control Science and Engineering of Zhejiang University. His research interests include wireless sensor networks, visual sensor networks, industrial communication and systems and networked control systems. He is a member of IEEE and ACM.