A Lightweight Privacy Preserving SMS-based Recommendation System for Mobile Users*

E.Baglioni, L. Becchetti, L. Bergamini
U. Colesanti, L.Filipponi, A.Vitaletti
Sapienza University of Rome
Via Ariosto 25
Rome, Italy
vitale@dis.uniroma1.it

G.Persiano,
University of Salerno
Via Ponte Don Melillo
Fisciano (SA), Italy
giuper@dia.unisa.it

ABSTRACT

In this paper we propose a fully decentralized approach for recommending new contacts in the social network of mobile phone users. With respect to existing solutions, our approach is characterized by some distinguishing features. In particular, the application we propose does not assume any centralized coordination: it transparently collects and processes user information that is accessible in any mobile phone, such as the *log of calls*, the *list of contacts* or the *inbox/outbox of short messages* and exchanges it with other users. This information is used to recommend new friendships to other users. Furthermore, the information needed to perform recommendation is collected and exchanged between users in a privacy preserving way. Finally, information necessary to implement the application is exchanged transparently and opportunistically, by using the residual space in standard short messages occasionally exchanged between users. As a consequence, we do not ask users to change their habits in using SMS.

1. INTRODUCTION

Mobile social networking is an emerging trend. eMarketer forecasts [1] that mobile social networking will grow from 82 million users in 2007 to over 800 million worldwide by 2012. In most mobile communities, mobile users can create their own profiles, make friends, create and participate in chat rooms, hold private conversations, share photos and videos. Major players in social networking, such as Facebook, MySpace and LinkedIn, have already deployed mobile versions of their applications.

Moreover, mobile applications can be extended to support physical presence detection and thus eventually create a link and some kind of convergence between the virtual and real world. For example, Centrl (centrl.com/mobile) is a smartphone application that lets you see which of your Facebook friends are around and Pelago (www.pelago.com) provides a similar application for Twitter users.

On the other hand, while western countries are experiencing the increasing availability of high speed connections and the diffusion of last generation smart phones with advanced interfaces to access mobile social networks, many still consider Short Messages the most convenient means for instant message exchange [1]. In any case, SMS traffic is still a consistent part of non-voice traffic. According to Lloyd's [5], overall Person-to-Person SMS traffic has been 4.5 trillion of messages in 2008. These figures seem to justify the investments of some companies in social networking applications based on Short Messages, such as Jyngle[2] [3] and Peekamo [4]. Furthermore, in large parts of the world, in particular Asia and Africa, SMS are expected to remain the primary means for data communication, at least in the near future. In 2007, nearly 1.5 trillion mobile messages were sent in the Asia-Pacific region [12].

Mobile social networks are thus raising in popularity, but along with clear benefits for users and companies, some concerns primarly related to privacy issues are arising. In the last W3C Workshop on the Future of Social Networking [6], several position papers on this issue appeared. For example, the basic operation of establishing a "friendship" in a social network, whatever the term means for the specific application, is a simple operation (e.g. a mouse-click), but it necessarily entails trust in the likely exchange of private information. As a matter of fact, privacy is one of the main concerns in mobile communities. As Jeff Chester, executive director of the Center for Digital Democracy , put it: "The fact of the matter is that the business model they have developed for mobile advertising is one where lots of user data is collected and user profiles are analyzed" and "You're talking about multiple layers of surveillance at the heart of the mobile marketing business model that raise serious privacy concerns."

Contribution of the paper. In this paper we propose an approach that uses Short Messages (SMs) and local information available on mobile phones to design a fully decentralized application for recommending new contacts in the

*Partially supported by EU STREP Project ICT-215270 FRONTS and by PRIN 2008 research project COGENT, funded by the Italian Ministry of University and Research.

RecSys2010, September 26–30, 2010, Barcelona, Spain.

[1] "If you look at instant messaging, e-mail or even social networking, they don't have the ubiquity and the reach to replace messaging" - Bill Dudley, Sybase 365's group director for product management.

[2] Jyngle closed in August 2009.

social network of mobile phone users. Recommending new contacts is a basic service provided by virtually every social network application. With respect to existing solutions, our approach is characterized by some distinguishing features:

The social networking application we propose is completely decentralized. This implies that the social network is not maintained in a centralized fashion, as usually done in nowadays social networking applications, but it is updated and managed in a fully distributed way by the collective effort of user devices. It transparently collects and processes user information that is accessible in any mobile phone, such as the *log of calls*, the *list of contacts* or the *inbox/outbox of short messages* possibly enriched by user profile information. This information is used to recommend new contacts.

The techniques we propose greatly reduce the amount of personal information that is disclosed, since it is exchanged with other users in the form of a compact summary that allows limited extraction of private data. In addition, we provide a simple and practical cryptographic protocol that can be used to ensure that the computation required by our recommendation system is performed without revealing any additional private information.

Information necessary to implement the application is exchanged transparently and opportunistically, by using the residual space in standard short messages occasionally exchanged by users. As a consequence, we do not ask users to change their habits in using SMS.

Past research has also considered decentralized systems for item recommendation, such as in [16], where the authors propose the P2P-based PocketLens architecture. We are aware of this body of work, but recommending items using statistical information about past user transactions is not the focus of our work, which is rather on the related but well distinguished "social matching" problem [19], in which we want to infer the latent structure of a social network.

Organization of the paper. The rest of the paper is organized as follows: in Section 2 we describe the approach we follow. In particular, we discuss some social networks naturally arising when analyzing the behaviour of users in a (mobile) telephone network. We then discuss the issues arising in the recommendation of new contacts in such networks, in the first place the notion of *similarity* between users. In Section 3 we review and discuss the application of sophisticated hashing techniques that allow to estimate the degree of similarity between users in a fully decentralized and privacy preserving way. In Section 4 we discuss experimental work assessing the effectiveness of our approach on real, publicly available datasets and in Section 5 we quickly introduce a working prototype of our receommendation system.

2. SOCIAL NETWORKING OVER SMS MESSAGING

In this work, a node in the social network of mobile phone users is a mobile phone subscriber generating some amount of user-to-user communication. A link connecting two nodes, represents an ongoing social relationship (e.g. nodes are friends, colleagues, classmates, etc.) between the corresponding users. In our approach, this social relationship can only be inferred estimating the users' *social profiles* similarity. Speaking in general terms, two users are similar when their social profiles are similar. In fact, the profile of a user is a general notion that depends on the information available to the system. In some cases this includes some biograhpical data, such as date of birth, sex, information about tastes, interests or activities. A profile is also completed by information that can be extracted transparently from the system, without explicit user intervention, such as the *log of calls*, the *list of contacts* or the *inbox/outbox of short messages*.

We stress that in many cases, even limited information, e.g., example the address book or the log of calls, can be used to infer possible relationships: for example, two users appearing in each other's address books are likely to be socially related, be it through a shared interest, a professional relationship, or simply because they are friends.

Mining the social network underlying telephone traffic has been considered in the past, for example in [8, 7]. Here, there is a (possibly labelled) link from A to B if A calls B at some point. The main goal in [8, 7] was to study the way in which such networks evolve over time, so as to infer and analyze probabilistic generative models [17] describing their evolution.

2.1 Recommending Social Relationships

Recommending new social relationships is one of the most basic services provided by social network applications. In our context, we are interested in strategies to recommend new contacts of potential interest to users. The challenge here is clearly to find contacts that are likely to share some common traits or, put differently, that are in some way "similar" to the user to whom the recommendation is being provided. As stated, this problem is very close to the link prediction problem studied by Liben-Nowell and Kleinberg [15], whose focus is on statistical indicators of social closeness and not on their efficient and decentralized computation.

More formally, if a node A recommends a node B to a third node C, A is suggesting a potential interest or utility for C in establishing a contact with B (unless this contact already exists). Recommendation is performed on the basis of knowledge about the social profiles $L(B)$ and $L(C)$, which are used to estimate the extent to which B and C are "similar". The underlying assumption, made more precise in Subsection 2.2, is that the more similar B and C, the more likely it is that they either have a contact, or they might benefit from establishing one.

Privacy requirements make the explicit exchange of private profile information or user contact lists unrealistic for applications. Furthermore, data must fit into the residual space of person-to-person short messages and thus they must be represented in a compact form (i.e. a *sketch*). Figure 1 outlines the general application scenario we consider. In step 1, users A and B compute the sketches $sk(L(A))$ and $sk(L(B))$ of their respective social profiles. As observed before, this is a compact representation of the user's social profile preserving her privacy. In step 2 and 3, A and B occasionally send a short message to C. The message space is partially filled with some personal text (e.g. SM Text = "shall we meet this evening?") while the residual space is exploited to deliver the sketches. Observe that users interact with the SMS as usual, while the residual space is transparently managed by a suitable application. In step 4, user C (i.e. the recommender) infer a high degree of similarity between A and B on the basis of their respective sketches. In steps 5 and 6, C eventually recommends a possible friendship to users A and B.

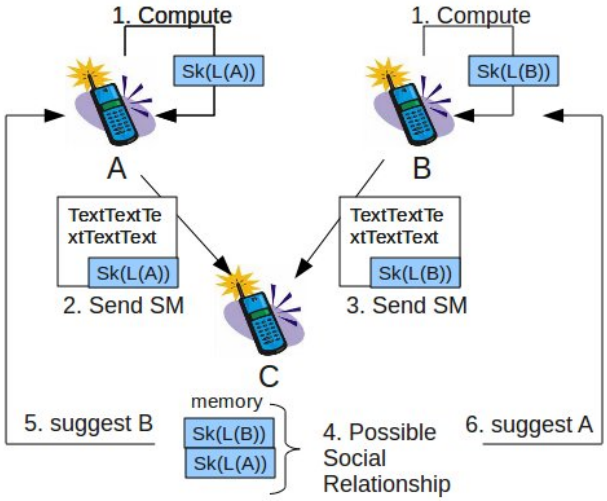

Figure 1: A scenario

2.2 Locally inferring community structure

One of the main issues in recommendation systems for social networking is predicting the potential benefit of new links between users. In the fully decentralized scenario we consider here, this amounts to answering the following question: when should a user A recommend a contact between two other users B and C she is aware of. This in turn implies a number of other issues: i) What information about B and C does A combine in order to decide whether or not she should suggest a contact between B and C if not existing already; ii) how is this information obtained, manipulated and exchanged; iii) how are computational, storage and communication constraints met; iv) how is privacy preserved.

Alike many networking applications, we recommend new contacts on the basis of similarities between users. Thus, A will recommend B and C to establish a contact if A assesses that B and C are "similar". In particular, if we view profiles as feature sets, we say that two users A and B are *similar* when their social profiles $L(A)$ and $L(B)$ overlap significantly. In this perspective, we estimate user similarity by the Jaccard coefficient $J(L(A), L(B)) = \frac{|L(A) \cap L(B)|}{|L(A) \cup L(B)|}$, a widely accpeted measure of similarity between sets. In the social networking scenario we consider, it captures the well known fact [17, 15] that social networks are densely connected at a local level or, roughly put, the folklore that two friends of the same person are significantly more likely to be friends than any two randomly chosen people.

3. EFFICIENTLY MINING THE SOCIAL NETWORK OF SMS USERS

A key aspect in the applications we consider is estimating the size of the intersection between the social profiles of two users in a fully decentralized way. More precisely, if a user C receives short messages from A and B, she should be able to estimate $J(L(A), L(B))$ from summary information about $L(A)$ and $L(B)$ piggybacked in the messages themselves. It is clear that short message size poses stringent constraints on the amount of information that can be piggybacked. This is at most 140 bytes, but recalling that we only use the residual space on the message, a variable number of those bytes will be occupied by the message body itself. We show below how to address these issues in the following way: i) we adapt a technique initially conceived for Web page similarity estimation to the scenario we consider. The adoption of this technique allows to compute *compact summaries* or *sketches* of each social profile, which in turn allows efficient estimation of the Jaccard coefficient between social profiles. The space required by the proposed sketches is in the order of a few tenths of bytes; ii) we address the issue of variable SMS size under the assumption [20] that SMS sizes are (approximately) uniformly distributed. Specifically, for those messages created in person-to-person communications, the length seems to evenly span the whole range of the allowed message sizes [20], whose maximum value depends on the encoding that is used for each message, but it is typically 140 bytes. In the following, we refer to profiles based on users' contact lists, since contact information is locally available on virtually every recent commercial device. For this reason, the terms *social profile* and *contact list* will be used interchangeably. We emphasize that the techniques described in the remainder can be extended to more general notions of user profile, as described in section 2.

3.1 Estimation of the Jaccard coefficient.

Consider the set of possible contact identifiers. Recall that, as motivated further in this section, they can be regarded as integer numbers falling in the range $[n] = \{0, \ldots, n-1\}$ for suitable n. The only assumption we need is that they are unique, a constraint that is met in practice in the applications we consider; they are users' telephone numbers or a suitable representation of them. As a consequence, considered any two users A and B, their contact lists $L(A)$ and $L(B)$ may be simply regarded as two subsets of $[n]$. Our goal is to measure their overlap using the Jaccard coefficient: $J(L(A), L(B)) = \frac{|L(A) \cap L(B)|}{|L(A) \cup L(B)|}$.

A very simple and elegant technique to estimate the Jaccard coefficient has been proposed in several equivalent forms by Broder et al. [10, 11]. Assume we are able to choose a permutation $\pi(\cdot)$ mapping $[n]$ onto itself uniformly at random. For every $X \subseteq [n]$, denote by $\pi(X)$ the set of the images of elements in X when $\pi(\cdot)$ is applied and let $\min(\pi(X))$ denote their minimum. Then it can be shown [10] that (i) considered a set $S \subseteq [n]$ and for every $a \in S$, $\mathbf{P}[a = \arg\min(\pi(S))] = 1/|S|$; (ii) for every $S_1, S_2 \subseteq [n]$: $\mathbf{P}[\min(\pi(S_1)) = \min(\pi(S_2))] = J(S_1, S_2)$. This property immediately yields a technique to estimate $J(S_1, S_2)$.

The algorithm consists in performing m independent executions of the following procedure: i) pick one permutation $\pi(\cdot)$ of [n] uniformly at random from the $n!$ possible ones; ii) in the i-th iteration, let $\min(S_1) = \min(\pi(S_1))$ and $\min(S_2) = \min(\pi(S_2))$. We increment a counter C_m whenever $\min(S_1) = \min(S_2)$. At the end of the process, our estimation of $J(S_1, S_2)$ is C_m/m. Standard tools from probability theory tell us that C_m is an increasingly (with m) accurate estimation of $J(S_1, S_2)$.

3.2 Computing and maintaining contact list sketches.

Unfortunately, generating permutations uniformly at random requires a number of truly random bits that is in the order of n [10]. Fortunately, suitable families of simple, linear hash functions perform well in practice (e.g. see [14,

UPDATE($sk(A)$, pn)

Require: Sketch $sk(A)$, number pn

1: x = hash(pn) {Hash pn to an integer in [n]}
2: **for** i: 1 ... m do
3: $M_i = h_i(x)$ {Map x according to a random permutation}
4: if $M_i < \min_i(A)$ then
5: $\min_i(A) = M_i$
6: end if
7: end for
8: return $sk(A)$

Figure 2: Update algorithm.

RECOMMEND(A, $sk(B)$, $sk(C)$, θ)

Require: Node A, Sketch $sk(A)$, $sk(C)$, threshold θ

1: Estimate $J(L(B), L(C))$ from $sk(B)$ and $sk(C)$ {Node must have both}
2: **if** $J(L(B), L(C)) > \theta$ **then**
3: A recommends B to C or viceversa
4: **end if**

Figure 3: Recommendation algorithm.

9]). In particular, we use linear permutations [9], i.e., functions of the form $h(x) = ((ax + b) \mod p) \mod n$. Here, p is large prime, while a and b are integers belonging to the intervals $[1, p-1]$ and $[0, p-1]$ respectively.

We next describe how each node A of the network maintains the local *sketch* $sk(A)$ associated to $L(A)$. As pointed out before, we assume below that every number in $L(A)$ is an integer falling in $[n]$. To this purpose, it is enough to perform a first step in which each contact identifier (e.g., a user's mobile phone number) is regarded as a string and this string is mapped onto an integer in $[n]$, using any hash function, as long as the probability of collision is sufficiently small. This is for instance the case if we hash contact identifiers to 32-bit integers using a good hash function, e.g., implemented in Java standard classes. As a second step, m hash functions are generated. The i-th hash function has the form $h_i(x) = ((a_i x + b_i) \mod p) \mod n$. The integers $\{a_1, b_1, \ldots, a_m, b_m\}$ are generated *independently* and uniformly at random, respectively in the interval $[1, p-1]$ for the a_i's and $[0, p-1]$ for the b_i's. Finally, for $i = 1, \ldots, m$, let $\min_i(A) = \min_{x \in L(A)}\{h_i(x)\}$. The sketch of $L(A)$ is the *ordered* vector $sk(A) = (\min_1(A), \ldots, \min_m(A))$. A version of this algorithm that allows dynamic updates when new numbers are added to the contact list is given in Figure 2.

The cost of algorithm UPDATE($sk(A)$, pn) is $O(m)$. The deletion of items from the contact list is more expensive, since the element removed might be the one achieving minimum value on one or more of the hash functions. Therefore, in the case of deletions $sk(A)$ has to be recomputed from scratch and the cost becomes $O(m|L(A)|)$. Note however, that m is in the order of a few tenths at most (10 in our experiments). This complexity is therefore fully compatible with standard commercial mobile phones.

In addition to $sk(A)$, A's device stores $sk(B)$, if available, for every B in her contact list. The required amount of additional memory, as discussed further in greater detail, is a few tenths of bytes for each entry in the contact list (40 in the current implementation), thus perfectly compatible with standard commercial devices.

3.3 Exchanging sketches.

In the scenario we envision, if both user A and B run the application and B sends an SMS to A, B will use the available free space of the message to send its own sketch $sk(B)$, or part of it, to A. Let's assume for the moment that there is enough residual space in the message to send the whole $sk(B)$. Note that this is likely to be often the case since, as we see later, the size of a sketch is typically a few tenths of bytes, 40 in the present implementation. Morover, we discuss how to address cases in which the SMS free space is not sufficient to contain $sk(B)$ in a further paragraph of this section. Whenever A's device receives the message, it transparently extracts $sk(B)$ from the message body. If B is one of A's contacts, then $sk(B)$ is stored in A's contact list, associated to B, possibly replacing an older copy of $sk(B)$.

3.4 Fully decentralized recommendation of contacts.

Recall that we assume that two users are *similar* to the purpose of the application whenever their contact lists overlap significantly. The algorithm in Figure 3 implements this general idea. In particular, the algorithm describes the behaviour of the generic, mobile terminal of some user A. If A has the sketches of both B's and C's contact lists, A will recommend B (C) to C (B) whenever the local estimation of $J(L(B), L(C))$ exceeds some given threshold θ. In Section 4 we study, among others, how the choice of the threshold affects the quality of recommended contacts.

3.5 Implementation issues.

We discuss in this paragraph several implementation issues.

If we consider the generic node A, the amount of memory needed to store its contact list is $\Theta(L(A))$. In our implementation, A also needs to store i) its own sketch $sk(A)$ and, in the worst case, ii) $sk(B)$, for a subset of nodes from which A received SMS messages in the past. If we assume that A stores the sketch of every contact, the required amount of memory is $O(m(|L(A)|))$. In practice, if we use $m = 10$, the additive amount of bytes required for each contact is about 40. This is in the same order of magnitude of an entry in any address book of a commercial device.

The computational cost of maintaining sketches and providing recommendations is also compatible with current commercial devices. In particular, adding a new contact to the contact list of a node A requires updating $sk(A)$ (algorithm UPDATE($\cdots$) in Figure 2) and has cost $(O(m))$. Removing a contact from $L(A)$ (typically a less frequent operation) is more expensive but it has (up to m) still linear cost, i.e., $O(m|L(A)|)$. Finally, for two nodes B and C other than A, deciding at A as to whether recommending each of them to the other requires estimating $J(L(B), L(C))$, which has cost $O(m)$. Computation is performed at user devices. Nowadays, these are typically small computers, whose computational capabilities are perfectly compatible with the computational effort required by the proposed techniques.

The number of hash functions required (i.e., m) is chosen, so that probability that the estimation of the Jaccard coefficient differs from the true value by more than a chosen constant is below a suitably small constant. We refer the

reader to specific work (e.g., [11, 10]) for technical details. In our case, experimental evidence suggests that 10 hash functions are sufficient to strike a reasonable balance between accuracy of the estimation and memory requirements.

A further constraint is that all user devices use the same set of hash functions. In practice, hash functions and the algorithms we propose will be implemented and maintained in the device's memory. This in turn requires storing, for each hash function, its coefficients and p in binary form. In our implementation, coefficients are 32-bit integers, while p is the well-known Mersenne prime $2^{32}-1$, which does not need to be stored explicitely. So, it turns out that the actual storage requirements for maintaining hash functions is around 80 bytes. The overall implementation (code, hash functions, runtime data structures) requires less than 1Kbyte space. To this, we must add the (variable) size of the user's (modified) contact list. Thus, the storage requirement of the modified contact list is in the same order of magnitude as in a standard implementation.

We observed earlier that we cannot always assume that the message body of an SMS sent from some node A to another node B has enough free space to host $sk(A)$. The most direct way to circumvent this problem is for A to send its sketch whenever the avaiable free space in the message body exceeds $|sk(A)|$. In fact, the distribution of SMS message sizes seems to be approximately uniform [20]. Assuming for the sake of simplicity that it is exactly uniform and that message sizes of different messages are independent variables, we have that half of the messages have 80 bytes available space in the average, more than 75% have at least 40 bytes available to carry sketches and so on. This means that, in the average, 1.34 message are enough for A to send its sketch to B, which means that, in practice, if A sends 2 SMS to B, the latter is very likely to receive A's sketch. [3]

3.6 Enforcing privacy

As described in the previous sections, a sketch is a representation of the contact list that, besides reducing the amount of data to be exchanged, does not fully discloses a user's contact list. As an example of the type of information that is leaked by the sketch $sk(A) = (\min_1(A), \ldots, \min_m(A))$ of contact list $L(A)$, we point out that if $h_i(x) < \min_i(A)$ then certainly $x \notin L(A)$. In this section we show how to securely compute the Jaccard coefficient of two contact lists, $L(A)$ and $L(B)$, without revealing any information except what can be deduced from the Jaccard coefficient itself.

Let us consider two parties A and B, each holding a vector of length m; with a slight abuse of notation we identify each party with his/her input vector. In our application to the computation of the Jaccard coefficient, the vectors will be the sketches of the respective contact lists. A and B wish to compute the number of positions i for which $A[i] = B[i]$ without revealing any additional information on the vectors. We will describe a protocol that uses an additively homomorphic encryption scheme $(E; D; K)$ like Paillier cryptosystem (see [18] for further information).

Homomorphic encryption scheme. Let $(E; D; K)$ be a homomorphic encryption scheme and assume that the message space for a public key pk returned by the key generator algorithm K on input security parameter m is $\mathbb{Z}_p$ for some integer p of length m. The following additive homomorphic properties hold: ii) the product of two ciphertexts is a ciphertext for the sum of the plaintexts; that is, for all messages $a; b \in \mathbb{Z}_p$ and public keys pk, we have $D(E(pk, a)\cdot E(pk, b), sk) = a + b$; ii) raising a ciphertext for message a to power r gives a ciphertext for $r\cdot a$; that is, for all $r \in \mathbb{Z}_p$ we have that $D(E(pk, a)^r, sk) = r\cdot a$.

The protocol. The protocol can be described as follows:

1. A picks a pair of public and secret key (pk, sk) for encryption scheme (E, D, K) by running the key generator algorithm K on input 1^m; for $i \in [n]$, A computes encryption $a_i = E(pk, A[i])$ of $A[i]$; A sends pk and $(a_i)_{i\in[n]}$ to B;
2. for $i \in [n]$, B computes encryption $b_i = E(pk, -B[i])$ of $-B[i]$, picks random $r_i \in \mathbb{Z}_p$ and sets $c_i = (a_i + b_i)^{r_i}$. Notice that by the homomorphic properties of (E, D, K), c_i is a ciphertext for $r_i\cdot(A[i]-B[i])$. Therefore if $A[i] = B[i]$, then c_i is an encryption of 0; otherwise c_i is an encryption of a random element of $\mathbb{Z}_p$. B randomly permutes the c_i's and sends them to A.
3. A decrypts the m ciphertexts received from A, counts the number s of ciphertexts that are an encryption of 0 and sends s to B.

Properties of the protocol. We make the following simple observations: *Correctness.*The value s computed by the protocol is the number of indices i for which $A[i] = B[i]$, with probability exponentially close to 1. *Privacy of the input.* Each of A and B gets no information on the other party's vector, besides what can be obtained from the output of the protocol. For A, this can be easily seen by exhibiting a probabilistic polynomial-time simulator S that, for all vectors A and B, on input vector A and the number s of positions in which A and B coincide (but not vector B) outputs A's view of the protocol. Similarly, we can construct a simulator for B.

Coming back to the recommendation system, we have two parties A and B, each with a private contact list, $L(A)$ and $L(B)$, that wish to compute the Jaccard coefficient $J(L(A), L(B))$. Obviously, the Jaccard coefficient can be computed by applying the above protocol to the characteristic vector of the two sets. The protocol will then run in time linear in the size of the underlying universe set. A much more efficient protocol is instead obtained by running the above protocol with each party holding as an input the sketch of his/her contact list computed using the same sequence of random (or min-wise independent) permutations.

4. EXPERIMENTAL ANALYSIS

In this section we present the results of experimental on real, publicly available data sets, in our opinion supporting the effectiveness of the approach we propose.

4.1 Experimental setting

4.1.1 Objectives

Our experimental work had the following main goals. In the first place, we wanted to understand the intrinsic effectivess of the Jaccard coefficient to infer social relationships

[3]An alternative solution is that A sends to B part of its sketch, compatibly with the available space in the SMS message body. This solution requires bookeeping both at A and B, to keep track of the portions of $sk(A)$ still missing at B. In fact, the former solution can be more easily implemented than the latter and it requires no additional data structures.

in the mobile phone user network. A further issue was to assess whether the techniques we use to approximate the Jaccard coefficient and discussed in Section 3, are compatible with the hard space constraints, imposed by Short Message size. In particular, these severely limit the number of hash functions we can use to to compute contact list sketches (we considered the use of 10 or 20 hash functions in our implementation). This in turn affects the accuracy of the estimation, especially when the value of the Jaccard coefficient is relatively small in absolute terms, as is the case for the data set we consider. Finally, we wanted to investigate the effectiveness of our overall approach in suggesting contacts to users. The problem here is that the data do not allow us to directly assess the *a posteriori* effect of recommendations. For this reason, in our experiments we considered the ability of our approach in predicting existing links as a proxy of its effectiveness in providing useful recommendations.

4.1.2 Data sets and call graph

Accessing telephone traffic data is far from trivial, since very few public datasets are available. The Reality Mining project [2, 13] represents the largest mobile phone experiment ever attempted in academia. Its dataset contains thousands hours of continuous data on daily human behavior and contains information on *call logs*, Bluetooth devices in proximity, cell tower IDs, application usage, phone status. We used call logs to build a *call graph* where nodes are mobile users characterized by unique ids (i.e., telephone numbers) and there is an edge connecting two users i and j if and only if the call log contains a number of calls between i and j that exceeds a given threshold[4]. For the purpose of this experimental analysis, we exploited the call graph to build the contact list of users in the network. To avoid problems related to data incompleteness, we restricted our experiments only to the people actually participating in the Reality Mining project (around 100 people), whose logs are complete and accurate.

Formally, the call graph $G_w(V, E)$ is built as follows:

- V is the set of users appearing in the log of calls
- for each pair$(i, j) \in V$, edge $(i, j) \in E$ if and only if at least w calls occurred between i and j.

Note that the graph G_w is *undirected*, i.e., we assume that contact lists are *symmetric*, i.e., i belongs to the contact list of j (and viceversa) if at least w calls occurred between i and j during the period of observation. Following the definition above, changing values of w can originate different graphs modeling stronger or weaker relations (i.e. a higher w can filter out occasional contacts). In our experiments we considered $G_1(V, E)$ and thus worked directly on the call graph, since the data set is relatively small and higher values of w further reduced the data set size.

4.1.3 Experimental scenario for recommendations

We assessed the quality of our technique in providing recommendations of good quality by using its ability to uncover existing relationships as a proxy. In particular, for each node v_i and for each pair $\{v_a, v_b\}$ both belonging to v_i's contact list, we ran our algorithm to predict the existence or non-existence of link (v_a, v_b). We then checked whether the link existed or not. This is synthetically described by the algorithm in Figure 4, which simulates the general recommendation algorithm described in Section 3.

[4]Note that this threshold is the number of calls above which we declare the existence of a link between the involed users. It is different from the threshold θ in Figure 3 of Section 3, i.e., the value of the estimated Jaccard coefficient above which a contact is recommended.

Require: $G_w(V, E)$
1: **for** $v_1 \dots v_n \in V$ `do`
2: `for each` v_a, v_b `among the contacts of` v_i `do`
3: `retrieve` $sk(A)$, $sk(B)$ `by emulating an SMS reception`
4: `RECOMMEND(A,` $sk(A)$, $sk(B)$, θ`)`
5: `end for`
6: `end for`

Figure 4: Simulation algorithm.

4.1.4 Performance indices

The error of the recommendation strategy we propose is potentially affected by two factors: i) inaccuracy in the estimation of the actual value of the Jaccard coefficient; ii) error in the recommendation itself, i.e., the contact we recommend is not interesting to the user. These two aspects are clearly interrelated in complex ways. We treat them separately, which corresponds to the worst-case assumptions that the effects of the two sources of error sum up. As for the former aspect, assessing the accuracy of our approximation of the actual Jaccard coefficient poses some issues. In the first place, our data show that even values of the Jaccard coefficient related to a significant degree of social relationship can be low in absolute terms. This makes an accurate estimation harder to attain given the stringent constraints we have to comply with. In particular, if we use m hash functions, we only have m possible values for our estimation of the Jaccard coefficient. When $m = 10$ as we assume, this provides very little granularity. Namely, possible values of the estimated Jaccard coefficient are $0.1j$, with $j = 0, \dots, 10$, whereas values of the true Jaccard coefficient corresponding to a significant degree of social interaction are around $[0.05, 0.1]$ in the dataset collection we consider. On the other hand, our algorithm is threshold-based: it recommends a contact between two nodes A and B whenever $J(L(A), L(B))$ is above the threshold. For this reasons, we consider the *Jaccard-Estimation Performance* (JEP), defined as the fraction of times that our algorithm gives the same recommendation as it would give if it knew the exact values of the Jaccard coefficient. We call the two versions of the algorithm *apxJacc* and *exactJacc* in the definitions that follows. Formally, for every node i, let C_i denote the number of times that *apxJacc* and *exactJacc* take the same recommendation decision for pairs of nodes beloning to i's contact list.

The *Jaccard-Estimation Performance* (JEP) is formally defined as:

$$JEP = \frac{\sum_{1 \le i \le |V|} C_i}{t}$$

where V is the vertex set, i.e., the overall number of users and t is the overall number of node pairs evaluated.

To assess the quality of our recommendations, we checked to which extent the contacts that are recommended corre-

spond to actual links evaluating *precision* (i.e., the fraction of existing links that have been recommended over the total number of given recommendations) and *recall* (i.e., fraction of all existing links that have been recommended over the total number of actual links) of our recommendation algorithm.

4.2 Experimental results

In this section, we provide experimental results that address the following issues: i) whether or not the Jaccard coefficient is a good indicator of social ties in the datasets we consider and which are reasonable threshold values for the recommendation heuristic we propose; ii) how good is our estimation of the Jaccard coefficient, at least in the sense made precise in the previous subsection; iii) how good are the recommendations we provide, which we indirectly answer by to which extent we are able to infer existing contacts between node pairs. Since our algorithms contain a probabilistic part in the selection and use of the hash functions used to estimate the Jaccard coefficient, all results reported below refer to averages taken over 5 indipendent runs of the algorithm.

4.2.1 Jaccard coefficient and social ties

Figure 5 synthetically describes the correlation existing between values of the Jaccard coefficient and existence of links between node pairs. More in detail, the x-axis is divided into interval of width 0.05 each, starting at 0.0 and ending at 0.2. For the j-th interval ($j = 0, 1, 2, 3$), the ordinate represents the fraction of pairs (A, B) of users such that i) $J(L(A), L(B))$ falls in the interval $[0.05j, 0.05(j+1)]$ and ii) A and B are contacts, i.e., they are in each other's contact lists. The x-intervals stops at the value 0.2, since we observed too few pairs with Jaccard coefficient beyond this interval, to be statistically meaningful. This picture clearly shows that the Jaccard coefficient is a good indicator of social ties in mobile user networks. Furthermore, at least in the datasets we considered, the Jaccard coefficient allows to identify a sharp transition around the value 0.05, from a region characterized by sporadic ties to one characterized by frequent social relationships. In light of these observations, we chose the value 0.05 as a threshold in our recommendation algorithm.

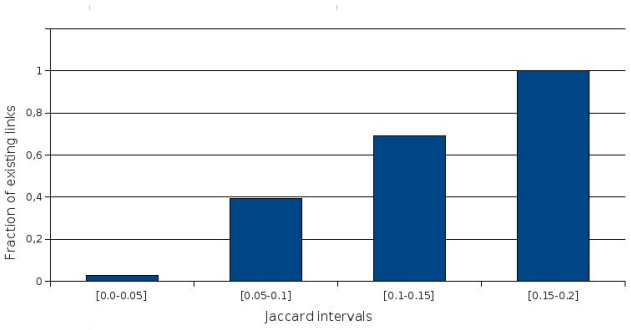

Figure 5: Existing linked pairs over total pairs

4.2.2 Jaccard estimation performance

Figure 6 shows the behaviour of the Jaccard-Estimation Performance, as defined in the previous subsection, as a function of the threshold, both when 10 and 20 hash functions are used to estimate the Jaccard coefficient. In particular, the function has been computed in 5 points for 20 hash functions. Each point represents an indipendent run of the algorithm. More precisely, for $j = 1, \ldots, 5$, the j-th run is executed with threshold value $0.05j$. For 10 hash functions we only considered two points, since for values of the threshold above 0.1 the distance between the two curves becomes smaller and smaller. Results show that the algorithm that estimates the Jaccard coefficient using hash functions takes the same decisions as the one knowing the exact value of the Jaccard coefficient in most cases. This means that, even under the stringent constraints for sketch sizes, we are able to follow the ideal algorithm pretty close, as far as the recommendation decision is concerned.

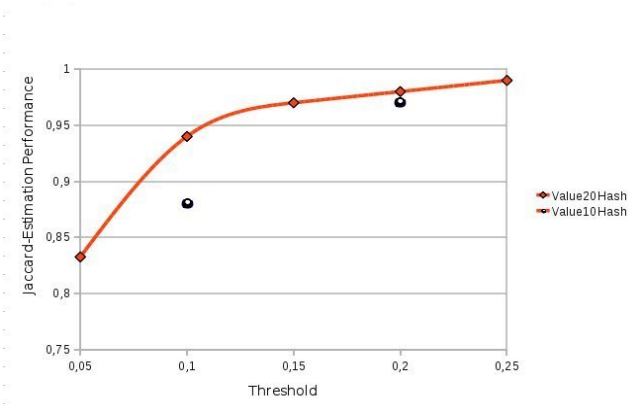

Figure 6: Jaccard-Estimation Performance

4.2.3 Quality of recommendation

Figure 7 shows the effectiveness of our algorithms in predicting the existence of contacts in the social network of mobile users. In particular, Figure 7 is a scatter-plot showing the trade-off between precision and recall as the threshold and number of hash functions used vary[5]. For a better reading, values with *precision* < 0.2 or *recall* < 0.2 have been filtered out[6]. The following remarks are in order: i) the best trade-off between precision and recall is struck near the interval $[0.05, 0.1]$ of the threshold, both for the algorithm using exact Jaccard coefficient and for our heuristics; ii) For higher values precision increases and recall decreases, meaning that on one hand, a similarity beyond the threshold implies a contact with increasing probability, but we omit to recommend many contacts that fall below the threshold; iii) the values of precision/recall we obtain for the best choice of the threshold fall in the interval $[0.4, 0.6]$. Such values are indeed relatively high, since they refer to the prediction of really existing links; if we were only recommending links that already exist, there would be no point in providing recommendations. These results in our opinion provide an indication that our fully decentralized strategies might prove effective in providing recommendations of good quality.

[5]The threshold is represented as a label over each point in the scatterplot.

[6]This is the reason why only one point of the 10 hash algorithm is represented on the scatterplot.

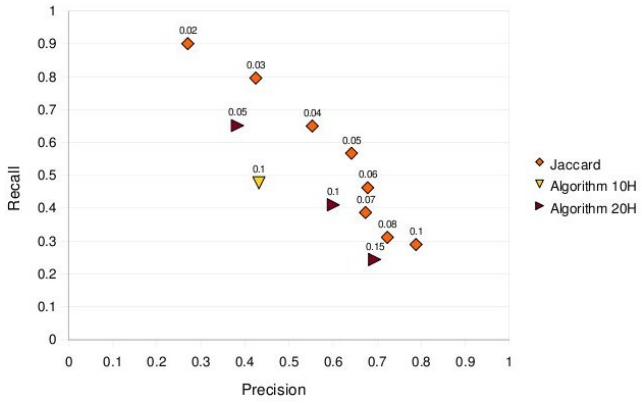

Figure 7: Precision vs recall.

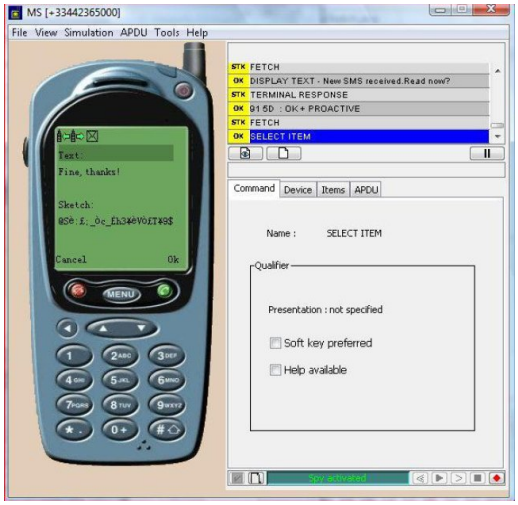

Figure 8: A screenshot of the Proof-of-concept implementation in Gemalto Developer suite

5. PROOF-OF-CONCEPT

We actually implemented a proof of concept of our solution as a Java Card ™Applet exploiting the Sim Application Toolkit on the Gemalto Developer Suite. The Java Card™technology allows applets written in the Java ™language to be executed on a smart card. It defines a Java Card™Runtime Environment (JCRE) and provides classes and methods to help developers create applets. The Gemalto Develper suite also features Card, Server and Mobile Simulators, together with a rich developement environment. The Applet we developed, shown in figure 8 implements the application we envised in the previous sections. It transparently uses the residual space in an SMS and fills it with the user Sketch. Figure 8 shows the screenshot of the Mobile Simulator during the reception of a message containing the sender's sketch. This sketch is then exploited from our recommendation algorithm to find similarities with other contacts.

6. REFERENCES

[1] emarketer forecasts online. http://technokitten.blogspot.com/2008/05/big-growth-predicted-for-mobile-social.html.

[2] Reality mining project. http://reality.media.mit.edu/.

[3] Jyngle web site. http://www.jyngle.com/, 2008.

[4] Peekamo web site: http://peekamo.com/, 2008.

[5] Market intelligence on messaging and marketing. http://www.lloydsmiu.com/pdf/Jun-2005/16/mma_sample_issue_june.pdf, 2009.

[6] W3c workshop on the future of social networking. http://planb.nicecupoftea.org/2009/01/14/w3c-workshop-on-the-future-of-social-networking/, 2009.

[7] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 171–180. ACM, 2000.

[8] W. Aiello, F. Chung, and L. Lu. Random evolution in massive graphs. In B. Werner, editor, *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 510–521, 2001.

[9] T. Bohman, C. Cooper, and A. M. Frieze. Min-wise independent linear permutations. *Electr. J. Comb*, 7, 2000.

[10] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *ACM Symposium on the Theory of Computing*, New York, NY, USA, 1998.

[11] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. In *Proc. of the World Wide Web Conference*, 1997.

[12] G. Consulting. Gartner says mobile messages to surpass 2 trillion messages in major markets in 2008, 2008.

[13] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[14] P. Indyk. A small approximately min-wise independent family of hash functions. In *SODA*, 1999.

[15] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58(7):1019–1031, 2007.

[16] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, 2004.

[17] M. E. J. Newman. The structure and function of complex networks, March 2003.

[18] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. pages 223–238. Springer-Verlag, 1999.

[19] L. Terveen and D. W. McDonald. Social matching: A framework and research agenda. *ACM Trans. Comput.-Hum. Interact.*, 12(3):401–434, 2005.

[20] P. Zerfos, X. Meng, S. H. Wong, V. Samanta, and S. Lu. A study of the short message service of a nationwide cellular network. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 263–268, New York, NY, USA, 2006. ACM.