FriendSensing: Recommending Friends Using Mobile Phones

Daniele Quercia[§‡] Licia Capra[‡]
[§]MIT SENSEable City Laboratory, Cambridge, USA
[‡]Dept. of Computer Science, University College London, UK
quercia@mit.edu, l.capra@cs.ucl.ac.uk

ABSTRACT

We propose *FriendSensing*, a framework that automatically suggests friends to mobile social-networking users. Using short-range technologies (e.g., Bluetooth) on her mobile phone, a social-networking user "senses" and keeps track of other phones in her proximity. *FriendSensing* processes proximity records using a variety of algorithms that are based on social network theories of geographical proximity and of link prediction. It then returns a personalized and automatically generated list of people the user may know. We evaluate the extent to which *FriendSensing* helps users find people they know against real mobility and social network data.

Categories and Subject Descriptors

H.3.3 [**Online Information Services**]: Web-based services.

General Terms

Algorithms

Keywords

Social matching systems, recommender systems, Web 2.0

1. INTRODUCTION

Finding and confirming friends on social-networking websites is a tedious and time-consuming task. To automate the process, different ways of recommending friends have been proposed and scrutinized. These are based on either social-networking profiles (e.g., they recommend people with shared interests) or audio recordings from collar devices (e.g., they recommend people with whom one has had lengthy face-to-face contacts). The former requires users to create fairly detailed profiles, and is thus nonetheless tedious than finding friends in the first place. The latter has had so far very limited applicability, as it requires the usage of invasive technology (i.e., collar devices) for data collection.

RecSys'09, October 23–25, 2009, New York, New York, USA.

Interestingly, a less invasive and more widely available form of data collection exists. This is to simply have mobile phones keep their Bluetooth on to track other phones in proximity. Since people usually carry their mobile phones [12], this way of collecting data is appealing for its simplicity and consequently begs an important research question: can proximity data from Bluetooth be used to recommend friends? We demonstrate that the answer is 'Yes', and we do so by making two main contributions:

- A framework called *FriendSensing* that automatically recommends friends by logging and analyzing colocation data. More precisely, using short-range connections (e.g., Bluetooth), mobile phones "sense" and record which other mobile devices are in proximity. FriendSensing then processes these records and suggests to users people they may know. It does so by using social network theories of "geographical proximity" and of "link prediction" (Section 2).
- An evaluation of the effectiveness of FriendSensing on real mobility and social network data from the Reality Mining Project [4] (Section 3).

2. OUR PROPOSAL: FriendSensing

To enable (new) members of social-networking websites automatically discover their friends, we have designed the *FriendSensing* framework. *FriendSensing* automatically creates personalized recommendations of people a user may know, and it does so in two steps:

Step 1. Logging Encounters - using short-range radio technologies ready available on almost all modern mobile phones (e.g., Bluetooth), each user transparently records encounters with colocated people. More precisely, each phone A keeps track of how many times it has met another phone B and how much time it has spent been colocated with B. We make here the assumption that a mobile phone is a *personal* device, and that it is not shared among people. Moreover, we assume it is possible to link devices (e.g., phone's Bluetooth ID number) to users' identities in social-networking websites (as pioneered by the Cityware project [9]).

Step 2. Recommending Friends - colocation records are processed to elicit relevant encounters and to arrange them into a weighted social network; this network is then traversed to compute personalized lists of people each user may know. FriendSensing does not prescribe

where the processing of proximity records and the navigation of the inferred social network should occur: both can be performed either by the social-networking website (after these records have been uploaded) or by the mobile device itself (if such records are considered sensitive and should thus be maintained private).

We now present algorithms for proximity processing and for network navigation in general terms, and defer a discussion about the implications of different architectural deployments to our evaluation (Section 3).

2.1 Processing Encounters

Once colocation logs have been collected, FriendSensing must filter out irrelevant encounters from *relevant* ones; that is, for each user A, it must identify which of A's encounters are likely to be A's friends. FriendSensing does so by computing the probabilities of A befriending other individuals (A's friendship probabilities) from proximity data.

Researchers have already suggested ways of computing these probabilities from *geographical proximity*, based on the intuition that friendship probability increases with geographic proximity - the closer two individuals are, the likelier they are to be friends. Kleinberg [7, 8], for example, modeled the probability of A and B being friends as $p(A \rightarrow B) \propto dist(A,B)^{-r}$. That is, the probability of being friends with a person at a distance d decays as d^{-r} for some power of r (typically $r = 2$). As later demonstrated by Liben-Nowell *et al.* [11], the absolute value of geographic distance alone is insufficient to model friendship. To see how, consider that A and B live 500 meter apart: at the very same distance, A and B would likely be next-door neighbors in the countryside, while complete strangers in central London. This suggests that one also needs to consider population *density*. Libel-Nowell *et al.* [11] did so in a simple way - they replaced the absolute distance $dist(A,B)$ with a *ranked distance*: $p(A \rightarrow B) \propto 1/(rankDist_A(B)+1)$. The denominator is A's rank of B, which is the number of people who are closer to A than B is, and it is expressed as:

$$rankDist_A(B) = |\{C : dist(A,C) < dist(A,B)\}| + 1.$$

In other words, the probability of A befriending B depends on the number of people within distance $dist(A,B)$. The more dense the population between A and B, the lower B ranks. Consequently, at the same distance, B is more likely to befriend A in the countryside than in central London. This model was successfully evaluated on half a million profiles collected from the LiveJournal blogging website, suggesting that geography is a good predictor of friendship. However, geographical information is not widely available on mobile phones; should localization technology like GPS become a commodity, it would still fail to capture indoor encounters (e.g., at home, in the office, on the tube, in the pub). We thus need to reformulate the problem based on "mobile phone proximity".

Using mobile phones, we keep track of: how many times a user A has met (e.g., it has been within Bluetooth range of) user B (frequency $freq(A,B)$), and how much time it has spent with B (duration $dur(A,B)$). So we now need to express the friendship probability as a function of frequency or duration. One plausible way of doing so is to consider that the probability of A befriending B increases with $freq(A,B)$ and with $dur(A,B)$ respectively. However, as with geographical information, we cannot consider frequency or duration alone to compute friendship probabilities, because both of them are non-uniformly distributed. Indeed, individuals do have skewed mobility patterns; this has been shown not only for college students [4] (against whose movements we will run our evaluation), but also for conference attendees [2], and for hundreds of thousands of mobile users [5]. Rather than using absolute frequency and duration values, we have thus taken their rank. From *frequency*, the friendship probability becomes:

$$p(A \rightarrow B) \propto \frac{1}{rankFreq_A(B)+1}, \quad (1)$$
$$rankFreq_A(B) = |\{C : freq(A,C) > freq(A,B)\}| + 1$$

Consequently, the probability of A befriending B depends on the number of people who have met A more frequently than B has done.

Similarly, by replacing frequency with *duration*, the friendship probability becomes:

$$p(A \rightarrow B) \propto \frac{1}{rankDur_A(B)+1}, \quad (2)$$
$$rankDur_A(B) = |\{C : dur(A,C) > dur(A,B)\}| + 1$$

Again, the probability of A befriending B depends not on $freq(A,B)$ itself but on the number of people who have met A for longer than B has done.

From the proximity logs, the above friendship probabilities can be computed and used to infer a *weighted social network of encounters*: each mobile device is represented as a node, and a link is added between any pair of individuals who have met at least twice (this is to remove encounters caused by chance). Each link $A \rightarrow B$ is then weighted using either *friendship probability* $p(A \rightarrow B)$ or *friendship ranking* (i.e., A's ranking of B, which is computed from the friendship probability itself). We explain when to opt for probabilities and when for ranks next.

2.2 Computing Recommendation Lists

Once the network of encounters has been computed, FriendSensing processes it to compute personalized lists of people each user may know, that is, to predict which of A's encounters are likely to be A's friends. In the literature of social networks, this problem is called "link prediction" and different methods have been proposed to tackle it [10]. These methods assigns a $score(A,B)$ to a pair of nodes (A,B) following one of two possible strategies:

Shortest Path - The score between a pair of nodes A and B is the weighted length of the shortest path between them [13]. The intuition behind it is that social networks are "small worlds" (individuals are connected by short chains [13]) and, as such, if there are short paths between A and B, then A and B are likely to befriend each other. The shortest path algorithm accepts weights on the network that represent capacity constraints - in our case, weights that reflect how unlikely it is for two nodes to befriend each other. Since rankings reflect just that (the higher $rankDur_A(B)$ or $rankFreq_A(B)$, the less likely A befriends B), we adopt *rankings* as link weights in the social network

of encounters. The path length is then weighted in the sense that it is the sum of the weights along the shortest path.

Markov Chain Algorithms - For this class of algorithms, the score between a pair of nodes A and B is computed as the fraction of time spent at B by a random walk in the network originating in A [14]. In this type of network, weights should reflect connection strength between pairs of nodes. Therefore, we adopt friendship *probabilities* $prob(A \rightarrow B)$ as link weights instead. Then, algorithmically, to compute scores, algorithms in this class all convert the network in a first-order Markov chain (hence their common name). The idea is that, after starting at node A (which is called prior node), the walk may unfold in different ways depending on which of the following algorithms is deployed:

- *PageRank with prior.* At each node, the walk either iteratively moves through one of the node's outgoing links (whose weights are transition probabilities) or jumps back to the prior node A [6].
- *K-MarkovChain.* It is similar to *PageRank with prior.* The difference is that the walk has now fixed length K [14].
- *HITS with prior.* At each node, the walk either moves through one of the node's *incoming* or outgoing links or jumps back to the prior A [14].

Once scores for a walk originating in A have been computed, they are then used to build A's personalized recommendation list.

2.3 Summary of FriendSensing Strategies

The *FriendSensing* framework thus offers eight strategies for recommending friends, derived from combining a strategy for processing proximity data into friendship probabilities (either *frequency* or *duration*), with one of the link-prediction algorithms (*shortest path*, *PageRank*, *HITS*, and *KMarkovChain*).

3. EVALUATION

3.1 Simulation Setup

The goal of FriendSensing is to recommend to its users people they may know. To ascertain the effectiveness of FriendSensing at meeting this goal, we set up a simulation driven by real data collected as part of the Reality Mining project at MIT [4]. The MIT traces contain colocation information from 96 subjects (staff and students) at the MIT campus over the course of the 2004-2005 academic year, to whom Bluetooth-enabled Nokia 6600 phones were given; colocation information (roughly 10 meters range) was collected via frequent (5 minute) Bluetooth device discoveries. Beside providing mobility traces, the MIT dataset also implicitly includes information about the users' social network. In fact, it logs both the text messages sent, and the phone calls made by each phone in the study. Using this information, we have extracted a social network whereby a link between user A and user B is created if A sent a text message or made a phone call to B.

In our simulations, we used the MIT mobility traces to log encounters; using these logs, we ran FriendSensing and computed friends' recommendations. We then compared these recommendations with the MIT actual social network (largest connected component) and computed the fraction of the social network's ties correctly predicted by FriendSensing. We refer to this fraction as "*good* recommendations" g, and we study how g varies while we increase the percentage r of people recommended to each user from 0 to 100%.

3.2 Results

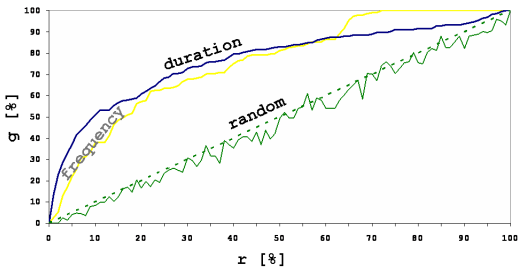
To study the effect of the colocation processing strategy separately from the link prediction strategy, we performed two sets of experiments.

(1) Frequency vs. Duration. In the first set of experiments, we aimed to compare the effectiveness of *frequency* as a colocation processing strategy, as opposed to *duration*. We did so by disabling any link propagation strategy, and by using the ranking produced by the frequency / duration colocation processing strategies locally. This is equivalent to running FriendSensing on people's mobile devices, without reporting their proximity logs to the social-networking website (where the full FriendSensing approach, including link propagation, could be executed). Figure 1(a) plots g (good recommendations) versus r (recommended people) for these strategies with respect to a *random* selection of people to recommend. For the random strategy, g increases linearly with r - the random strategy fluctuates around a straight line (dashed in the figure). That is because the more people are recommended, the likelier to get some of them right. At the extreme of $r = 100\%$ (all users have been recommended to each user), g reaches 100% (for all strategies). As for the two remaining strategies, they both perform significantly better than random. Note that *duration* discovers friends faster than *frequency*. To see now which strategy performs better over another, we compare *frequency* and *duration* against the random one. We do so by defining the *gain factor* over *random* as:

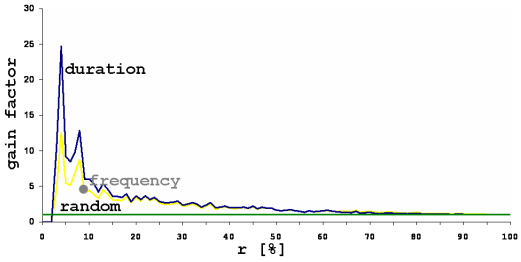
$$gain_{strategy} = \frac{g_{strategy}}{g_{random}}$$

where $g_{strategy}$ is the fraction of good recommendations for $strategy = duration \mid frequency$, and g_{random} is that for *random*. A gain factor of one means the strategy performs no better than random (no gain); a factor of two means that the strategy performs twice as better as random. Figure 1(b) shows that *duration* gains more than *frequency* - especially so for the first 20% of people recommended. As one expects, frequency and duration die off up to a point where both of them flatten toward *random* (no gain). That is because, after recommending most friends, any strategy has left only few friends to recommend, and those are hard to predict.

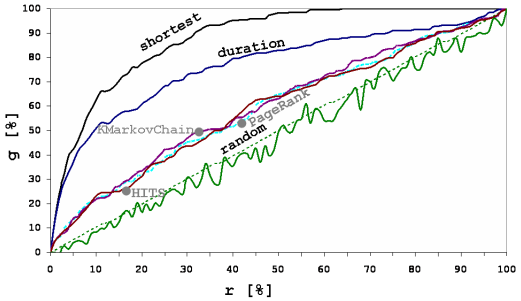
(2) Duration and "Link Prediction". The second set of experiments compared the four different link prediction strategies presented in Section 2.2. We did experiments whereby these strategies were executed on a social network of encounters built using *duration* information and *frequency* information. Since results obtained with *duration* were consistently better than those obtained with *frequency*, we report results for the former case only. Figure 1(c) plots g versus r for all the four strategies. We also plot the results obtained with our baseline *random* strategy, as well as when using *duration* without propagation, to highlight what privacy-conscious users would miss by not sharing their colocation information for propagation processing. *PageRank*,



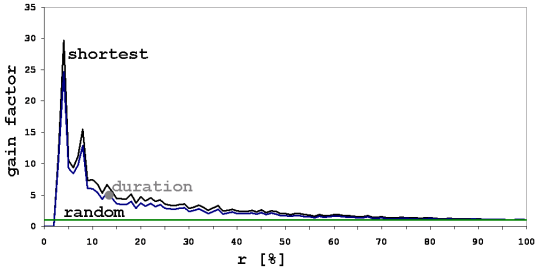
(a) Predicted ties g vs. recommended people r.



(b) Gain factor vs. recommended people r.



(c) Predicted ties g vs. recommended people r.



(d) Gain factor vs. recommended people r.

Figure 1: Evaluation Results.

HITS, and *KMarkovChain* perform equally and only show small differences due to confidence on the results. Those results are similar and come from the common use of Markov chains by the three algorithms. Also, one would be better off using only *duration* rather than combining it with those three algorithms. That is not necessarily bad news as it suggests that, by relying only on her own proximity information, a user both gets quality recommendations and, while doing so, she retains control of her own data. In line with the literature, *shortest path* performs best. Indeed, Figure 1(d) shows that it gains more than *duration*, and it does so consistently. That is because, unlike *duration*, *shortest path* is able to suggest to a user A also those friends who belong to the A's social circle but have not been met by A yet.

4. CONCLUSION AND FUTURE WORK

We have presented *FriendSensing*, a framework that exploits human co-location information to automate the process of finding friends on social-networking websites.

To go from inferring friends to accurately inferring social networks, FriendSensing needs to be refined. First, the combination of duration and frequency of colocation could be investigated. Non-geographic information should then be considered, as research has shown that friendship does not only depend on geographic factors, but also on whether individuals have similar occupation, cultural backgrounds, or roles within a company [1, 3]. We thus aim to reason not only on how long people have been co-located but also, for example, *where* and *when* they have been so.

Unlike existing friends-of-friends approaches that infer social relationships by exposing sensitive information, FriendSensing suits privacy conscious individuals better: in fact, they could run FriendSensing with the *duration* strategy that, as shown experimentally, produces quality recommendations while relying on proximity information collected by their own device only.

5. REFERENCES

[1] L. A. Adamic and E. Adar. How to search a social network. *Social Networks*, 2005.

[2] A. Chaintreau, P. Hui, C. Diot, R. Gass, and J. Scott. Impact of Human Mobility on Opportunistic Forwarding Algorithms. *IEEE Trans. on Mobile Computing*, 2007.

[3] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 2008.

[4] N. Eagle and A. S. Pentland. Reality mining: sensing complex social systems. *Personal Ubiquitous Computing*, 2006.

[5] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 2008.

[6] T. H. Haveliwala. Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. *IEEE Trans. on Knowledge and Data Engineering*, 2003.

[7] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. of ACM STOC*, 2000.

[8] J. Kleinberg. The convergence of social and technological networks. *Communications of the ACM*, 2008.

[9] V. Kostakosand and E. O'Neill. Cityware: Urban computing to bridge online and real-world social networks. *Handbook of Research on Urban Informatics*, 2008.

[10] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of American Society for Information Science and Technology*, 2007.

[11] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic Routing in Social Networks. *Journal of the National Academy of Sciences*, 2005.

[12] P. J. Ludford, D. Frankowski, K. Reily, K. Wilms, and L. Terveen. Because I carry my cell phone anyway: functional location-based reminder applications. In *Proceedings of ACM CHI*, 2006.

[13] M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 2003.

[14] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *Proc. of the* 9^{th} *ACM SIGKDD*, pages 266–275, 2003.